2018

Pengamanan File Teks Menggunakan Algoritma Skipjack dan Kompresi Algoritma Taboo

Suanda, Rizky

Universitas Sumatera Utara

PENGAMANAN FILE TEKS MENGGUNAKAN ALGORITMA SKIPJACK DAN KOMPRESI ALGORITMA TABOO

SKRIPSI

RIZKY SUANDA
121401037

PROGRAM STUDI S-1 ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2018

PENGAMANAN FILE TEKS MENGGUNAKAN ALGORITMA SKIPJACK DAN KOMPRESI ALGORITMA TABOO

SKRIPSI

Diajukan untuk melengkapi tugas akhir dan memenuhi syarat mencapai gelar Sarjana Komputer

RIZKY SUANDA
121401037

PROGRAM STUDI S-1 ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2017

PERSETUJUAN

Judul	:	PENGAMANAN FILE TEKS MENGGUNAKAN ALGORITMA SKIPJACK DAN KOMPRESI ALGORITMA TABOO
Kategori	:	SKRIPSI
Nama	:	RIZKY SUANDA
Nomor Induk Mahasiswa	:	121401037
Program Studi	:	S1 ILMU KOMPUTER
Fakultas	:	ILMU KOMPUTER DAN TEKNOLOGI INFORMASI UNIVERSITAS SUMATERA UTARA

Diluluskan di

Medan, Desember 2017

Komisi Pembimbing :

Pembimbing 2	Pembimbing 1
Amalia, S.T, M.T	Dr. Poltak Sihombing, M.Kom
NIP. 197812212014042001	NIP. 196203171991031001

Diketahui/Disetujui oleh

Program Studi S1 Ilmu Komputer

Ketua,

Dr. Poltak Sihombing, M.Kom

NIP. 196203171991031001

PERNYATAAN

PENGAMANAN FILE TEKS MENGGUNAKAN ALGORITMA SKIPJACK DAN KOMPRESI ALGORITMA TABOO

SKRIPSI

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

Medan, Desember 2017

Rizky Suanda
121401037

PENGHARGAAN

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

Pada pengerjaan skripsi dengan judul Pengamanan File Teks Menggunakan Algoritma *Skipjack* dan Kompresi Algoritma *Taboo*, penulis menyadari bahwa banyak pihak yang turut membantu, baik dari pihak keluarga, sahabat dan orang-orang terkasih yang memotivasi dalam pengerjaannya. Dalam kesempatan ini, penulis mengucapkan terima kasih kepada:

1. Bapak Prof. Dr. Runtung Sitepu, SH, M.Hum selaku Rektor Universitas Sumatera Utara.
2. Bapak Prof. Dr. Opim Salim Sitompul selaku Dekan Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
3. Bapak Dr. Poltak Sihombing, M.Kom selaku Dosen Pembimbing I dan Ketua Program Studi S1 Ilmu Komputer Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
4. Ibu Amalia, S.T, M.T selaku Dosen Pembimbing II yang telah memberikan arahan, kritik dan saran serta motivasi kepada penulis dalam pengerjaan skripsi ini.
5. Ayahanda Sunarto, Ibunda Sujarwati, Ibu Kasiani, Bapak Buyung, Adinda Setia Ulfa Adinda , Adinda Heru Wahyudi yang mana telah banyak memberikan kasih sayang, semangat, serta dorongan kepada penulis.
6. Seluruh tenaga pengajar dan pegawai di Fakultas Ilmu Komputer dan Teknologi Informasi USU.
7. Teman-teman seperjuangan menggapai gelar sarjana yang selama ini telah menjadi keluarga kedua penulis selama mengikuti perkuliahan, tempat berbagi suka dan duka dalam pengerjaan skripsi ini, kepada Rio Dat Permana, Ibnu

Sandiansyah Hutabarat, Ahmad Hasan Pohan, Dicky Hamdani, Edo Aditya Chandra, Parlin Saputra H Tanjung, Joko Kurniawan, Jabbar Muhammad Lubis, Ilham Saputra, Bobby Putra Johan, Aulia Rahman Saragih, Abdurasyid Aulia, Miftahul Huda, Fachri Irfandi, Josua Sihotang, Naszri Adlani, T.Triyandi.

8. Stambuk 2012 yang tidak dapat disebut satu-persatu, yang telah banyak membantu dalam selesainya pengerjaan skripsi ini.
9. Sahabat dan teman – teman di desa tempat penulis tinggal, kepada Arif Sanjaya, Maia Ulfa Aldri, Teguh Hermawan, Rizal Hariono, Deni, Dicky Syahputra, Sugeng, Josadi Hakim, Fachri Amrullah, Lili Indriyani, Risa Deva Aldri, Evi Chairah, Fauziah.
10. Semua pihak yang terlibat langsung maupun tidak langsung yang tidak dapat penulis ucapkan satu demi satu yang telah membantu penyelesaian laporan ini.

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

Medan, Desember 2017

Penulis,

Rizky Suanda

PENGAMANAN FILE TEKS MENGGUNAKAN ALGORITMA SKIPJACK DAN KOMPRESI ALGORITMA TABOO

ABSTRAK

Keamanan merupakan suatu kebutuhan penting dalam proses pertukaran data. Teknik kriptografi adalah untuk menjaga keamanan dan kerahasiaan data dalam pengiriman dan bertukar informasi. Suatu algoritma kriptografi dapat ditingkatkan keamanannya dengan cara digabungkan dengan algoritma kriptografi lainnya. Salah satu solusi dengan Algoritma Skipjack adalah algoritma kunci simetri. Selain melakukan pengamanan data, juga diperlukan proses pengecilan suatu data untuk efisiensi ruang penyimpanan. Algoritma Taboo adalah algoritma dengan pendekatan terdapat panjangnya suatu variable. Hasil pengujian kompresi pada ukuran *file* teks dengan jenis *file* *.txt tetap seperti dari ukuran *file* sebelumnya. Pada kompresi jenis *file* *.doc ukuran hasil kompresi mengecil lebih banyak dari *file* sebelumnya, sehingga ketika ukuran *file* asli 28 Kb maka ukuran hasil kompresi menjadi sekitar 3 Kb. Pengujian kompresi pada *file* teks dengan jenis *file* *.txt atau jenis *file* *.doc yang berisi 3072 karakter di kompresi menjadi 2205 karakter. Pada proses enkripsi *file* teks dienkripsi menjadi *ciphertext* yang memiliki panjang 848 karakter. Hasil pengujian berupa waktu eksekusi kompresi pesan *file* *.txt adalah 0,1305 detik, sedangkan pesan *file* *.doc adalah 0,0989 detik. Selanjutnya, waktu eksekusi dekompresi pesan *file* *.txt adalah 0,1672 detik, sedangkan pesan *file* *.doc adalah 0,1525 detik. Sedangkan rata-rata *Ratio of Compression* (RC) sebesar 71,88 %, *Compression Ratio* (CR) sebesar 1,39 % dan *Redundancy* (RD) sebesar 28,22 %. Dari hasil pengujian berupa waktu eksekusi enkripsi pesan *file* *.txt adalah 0,1468 detik, sedangkan pesan *file* *.doc adalah 0,111 detik. Selanjutnya waktu eksekusi dekripsi pesan *file* *.txt adalah 0,1523 detik, sedangkan pesan *file* *.doc adalah 0,1156 detik. Oleh karena itu dapat disimpulkan bahwa waktu enkripsi pesan lebih lama daripada waktu dekripsi pada pesan.

Kata Kunci : Kritografi, Skipjack, Taboo, *Ratio of Compression* (RC), *Compression Ratio* (CR), *Redundancy* (RD)

TEXT FILE SECURITY USING SKIPJACK ALGORITHM AND TABOO ALGORITHM COMPRESSION

ABSTRACT

Security is an important requirement in the process of data exchange. The cryptographic technique is to maintain the security and confidentiality of data in sending and exchanging information. A cryptographic algorithm can be increased in security by being combined with other cryptographic algorithms. One of the solution is Skipjack Algorithm with symmetry key algorithm. In addition to data security, it is also necessary to reduce the data for storage space efficiency. Taboo algorithm is an algorithm with approach to the length of a variable. The result of compression testing on text file size with file type * .txt remain like from previous file size. In the file type compression * .doc the size of the compression results smaller than the previous file, so when the original file size 28 Kb then the compression result size to about 3 Kb. Test compression on a text file with * .txt file type or * .doc file type containing 3072 characters in compression to 2205 characters. In the encryption process the text file is encrypted into a ciphertext that has a length of 848 characters. The test result of the execution time of message file compression * .txt is 0.1305 seconds, while the message file * .doc is 0.0989 seconds. Furthermore, the execution time of decompressing the * .txt file is 0.1672 seconds, while the * .doc file is 0.1525 seconds. While the average Ratio of Compression (RC) of 71.88%, Compression Ratio (CR) of 1.39% and Redundancy (RD) of 28.22%. From the test result of execution time of file encryption message * .txt is 0,1468 second, while message file * .doc is 0,111 second. Next time the executable decryption message * .txt file is 0.1523 seconds, while the message file * .doc is 0.1156 seconds. Therefore it can be concluded that the encryption time of the message is longer than the decryption time in the message.

Kata Kunci : Cryptography, Skipjack, Taboo, *Ratio of Compression* (RC), *Compression Ratio* (CR), *Redundancy* (RD)

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1
PENDAHULUAN

Pada bab ini penulis akan menguraikan tentang latar belakang, rumusan masalah, batasan masalah dari penelitian, tujuan dan manfaat dari penelitian, metodologi yang dipakai dalam melakukan penelitian ini serta sistematika penulisannya.

1.1 Latar Belakang

Akibat pengembangan teknologi yang begitu pesat, manusia dapat berkomunikasi dan saling bertukar informasi/data secara jarak jauh. Tetapi tidak semua pengiriman informasi/data terjamin keamanan dan kerahasiaannya. Oleh karena itu, dikembangkanlah berbagai macam teknik untuk melindungi kerahasiaan dan keamanan pesan. Salah satu teknik yang paling sering digunakan adalah teknik kriptografi.

Kriptografi adalah ilmu dan seni untuk menjaga kerahasiaan pesan dengan cara menyandikannya ke dalam bentuk yang tidak dapat dimengerti lagi maknanya (Munir, 2006).

Salah satu algoritma kriptografi untuk pengamanan informasi/data adalah Algoritma Skipjack. Dengan Algoritma Skipjack, data akan aman karena algoritma ini yang aman dengan 32 kali putaran algoritma dibandingkan dengan algoritma yang lain sehingga data sulit untuk dibuka kerahasiannya tanpa didekripsi (Supriato, 2007). Pengamanan (Enkripsi) dimaksudkan untuk melindungi informasi agar tidak terlihat oleh orang atau pihak yang tidak berhak. Enkripsi dibentuk berdasarkan suatu algoritma yang akan mengacak suatu informasi menjadi bentuk yang tidak bisa dibaca atau tidak bisa dilihat. Dekripsi proses dengan algoritma yang sama untuk mengembalikan informasi teracak menjadi bentuk aslinya.

Selain melakukan pengamanan data, juga diperlukan proses pengecilan suatu data untuk efisiensi ruang penyimpanan atau biasa disebut kompresi data.

Kompresi data merupakan proses mengkodekan informasi menggunakan bit atau information-bearing unit yang lain yang lebih rendah daripada representasi data yang tidak terkodekan dengan suatu system encoding tertentu.

Salah satu algoritma kriptografi untuk proses pengecilan suatu data untuk efisiensi ruang penyimpanan adalah Algoritma Taboo. Algoritma Taboo memiliki pendekatan terhadapat panjangnya suatu variable. Penggagas Algoritma Taboo adalah Steven Pigeon. Prinsip dari Algoritma Taboo yaitu untuk memilih bilangan positif n dan membalikkan pola n untuk mengindikasikan akhir dari kode tersebut. (Salomon, 2007).

Tujuan dengan dilakukan kompresi adalah untuk mereduksi ukuran data atau file. File merupakan tempat penyimpanan data dalam bentuk digital. Jumlah data yang disimpan pada file berbanding lurus dengan ukuran file itu sendiri. Ukuran file yang terlalu besar akan menjadi masalah bila file tersebut akan ditransfer atau dipertukarkan. Sebagai contoh file tidak muat pada media penyimpan seperti disket ataupun memperlambat proses download atau upload pada jaringan internet. Solusi untuk masalah ini biasanya file tersebut dibagi (split) menjadi ukuran yang lebih kecil. Cara lain adalah file tersebut dimampatkan sehingga ukurannya menjadi lebih kecil dari ukuran semula sehingga mempersingkat waktu ketika ditransmisi.

Dengan melakukan kompresi atau pemadatan data maka ukuran file atau data akan lebih kecil sehingga dapat mengurangi waktu transmisi sewaktu data dikirim dan tidak banyak menghabiskan ruang media penyimpan, sedangkan dekompresi data merupakan kebalikan dari kompresi data yaitu mengembalikan data ke bentuk dan ukurannya semula. Proses dekompresi secara harfiah merupakan proses yang dilakukan bila data hasil kompresi ingin dikembalikan ke ukuran dan bentuknya semula. Tujuan dekompresi data karena data yang telah dikompresi tidak dapat dipakai secara langsung melainkan harus melalui suatu proses untuk mengembalikan ke ukuran semula.

1.2 Rumusan Masalah

Rumusan masalah yang akan dibahas dalam penelitian ini adalah bagaimana mengamankan data menggunakan algoritma *Skipjack* dan mengecilkan ukuran

chiperteks menggunakan algoritma *Taboo* yang dimana melalui proses pengenkripsian data.

1.3 Batasan Masalah

Adapun batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Jenis data yang akan dipakai adalah string dalam file *.txt atau *.doc.
2. Penelitian ini hanya membahas Algoritma Kriptografi Skipjack dan Algoritma Kompresi Taboo.
3. Komponen lain seperti gambar yang ada di dalam file text yang digunakan akan diabaikan.
4. Parameter kompresi yang diukur adalah *Time Process* (waktu kompresi), *Completeness* (kelengkapan data), dan *Compression Ratio* (ukuran data setelah kompresi).
5. Menggunakan bahasa pemrograman Visual Studio.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk menganalisis hasil kompresi file teks *.txt atau *.doc menggunakan Algoritma Taboo yang telah diamankan dengan Algoritma Skipjack.

1.5 Manfaat Penelitian

Manfaat yang diharapkan dapat dihasilkan dari penelitian ini adalah untuk mengamankan *file* teks (*.txt, atau *.doc) agar menjadi lebih aman dan mengubah ukuran *cipher* teks menjadi lebih kecil sehingga kapasitas *memory* yang digunakan lebih efisien.

1.6 Metodologi Penelitian

Penelitian ini menerapkan beberapa metode penelitian sebagai berikut:

1. Studi Literatur

   Pada tahap ini dilakukan pengumpulan referensi yang diperlukan dalam penelitian. Hal ini dilakukan untuk memperoleh informasi dan data yang

diperlukan untuk penulisan skripsi ini. Referensi yang digunakan dapat berupa buku, jurnal, artikel, situs internet yang berkaitan dengan penelitian ini.

2. Pengumpulan dan Analisis Data

   Pada tahap ini dilakukan pengumpulan dan analisa data yang berhubungan dengan penelitian ini seperti fungsi algoritma *Taboo*, untuk mengetahui ukuran *file* teks setelah dikompresi dan dekompres, fungsi enkripsi / dekripsi algoritma *Skipjack.*

3. Perancangan Sistem

   Pada tahap ini masalah akan dianalisis untuk mengetahui apa saja hal yang dibutuhkan dalam penelitian ini, dan kemudian sistem dirancang dengan membuat gambaran sistem menggunakan *flowchart*, *Unified Modeling Language (UML),* dan *user interface*.

4. Implementasi Sistem

   Pada tahap ini perancangan diimplementasikan dengan menggunakan bahasa pemrograman Visual Studio.

5. Pengujian Sistem

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

6. Dokumentasi Sistem

   Pada tahap ini seluruh kegiatan dalam pembuatan sistem didokumentasikan kedalam bentuk tulisan berupa laporan penelitian tugas akhir.

1.7 Sistematika Penulisan

Setelah uraian di bab satu ini, penyajian selanjutnya disampaikan dengan sistematika berikut :

BAB 1 : PENDAHULUAN

Menguraikan Latar Belakang, Rumusan Masalah, Tujuan, Batasan Masalah dan Sistematika Penulisan Skripsi.

BAB 2 : TINJAUAN PUSTAKA

Berisi rangkuman informasi yang dihimpun dari pustaka yang relevan dengan topik yang menjadi objek kajian, untuk memperluas basis informasi dalam melakukan kajian dan / atau akan digunakan sebagai basis argumentasi di dalam mengemukakan pandangan.

BAB 3 : ANALISIS DAN PERANCANGAN

Menguraikan proses yang dilaksanakan dalam kajian untuk menyelesaikan persoalan yang dipertanyakan dan bagaimana pelaksanaan kajian tersebut.

BAB 4 : HASIL DAN PEMBAHASAN

Menguraikan analisis dan hasil yang diperoleh di akhir pelaksanaan Skripsi dan pokok-pokok kesimpulan yang dapat dikemukakan sehubungan dengan pencapaian hasil kajian kompresi Ternary Comma dan Levenstein.

BAB 5 : KESIMPULAN DAN SARAN

Menguraikan kesimpulan yang didapat dari hasil kajian dan hal-hal apa saja yang selayaknya dilakukan agar kajian yang diperoleh lebih sempurna, atau agar dapat dilakukan pengayaan dari pengetahuan yang diperoleh dari kajian.

BAB 2

TINJAUAN PUSTAKA

Pada bab ini penulis memaparkan teori-teori ilmiah yang didapat dari metode pencarian fakta yang digunakan untuk mendukung penulisan skripsi ini dan sebagai dasar pengembangan sistem sehingga dapat diimplementasikan dengan baik dan benar.

2.1 Kompresi

2.1.1 Pengertian Kompresi data

Kompresi berarti memampatkan / mengecilkan ukuran. Kompresi data adalah proses mengkodekan informasi menggunakan bit atau information-bearing unit yang lain yang lebih rendah daripada representasi data yang tidak terkodekan dengan suatu system encoding tertentu. Kompresi data teks dan citra digital adalah proses untuk meminimalisasi jumlah bit yang merepresentasikan suatu data baik berupa teks maupun citra / gambar sehingga ukuran data menjadi lebih kecil (Sirait, 2013).

Tujuan dari kompresi data adalah untuk merepresentasikan suatu data digital dengan sesedikit mungkin bit, tetapi tetap mempertahankan kebutuhan minimum untuk membentuk kembali data aslinya (Suhastra, 2014).

2.1.2 Penggolongan Kompresi

Secara umum kompresi dibagi menjadi 2 golongan, yaitu:

1. Kompresi *Lossy*

   Algoritma kompresi *Lossy* tidak memungkinkan untuk merekonstruksi yang asli seperti dari versi kompresi. Ada beberapa rincian yang tidak penting yang mungkin hilang selama proses kompresi. Kompresi *Lossy* disebut kompresi *irreversible* karena tidak mungkin untuk memulihkan data asli seperti dengan dekompresi. Rekontruksi atau pengembalian data seperti semula mungkin diinginkan karena dapat menyebabkan kompresi lebih efektif. Namun, sering

membutuhkan keseimbangan yang baik antara kualitas visual dan kompleksitas perhitungan (Mengyi, 2006).

2. Kompresi *Lossless*

Algoritma kompresi *Lossless* memungkinkan untuk mengembalikan data asli dari data yang telah terkompresi. Tidak ada kehilangan informasi apapun selama proses kompresi. Kompresi *Lossless* disebut kompresi *reversible* karena data asli dapat dipulihkan dengan sempurna oleh dekompresi. teknik kompresi *Lossless* digunakan ketika data asli yang sangat penting dari sumber, tidak kehilangan rincian apapun (Mengyi, 2006).

2.1.3 Parameter Analisis Kompresi

Pada suatu teknik yang digunakan dalam proses kompresi data terdapat beberapa faktor atau variabel yang biasa digunakan untuk menganalisa kualitas dari suatu teknik kompresi data tersebut, yaitu :

1. *Ratio of Compression (RC)*

   Ratio of Compression (RC) adalah nilai perbandingan antara ukuran bit data sebelum dikompresi dengan ukuran bit data yang telah dikompresi.

   $$\boldsymbol{Rc = \frac{ukuran\ data\ sebelum\ dikompresi}{ukuran\ data\ setelah\ dikompresi}}$$ (Salomon & Motta, 2010)

2. *Compression Ratio (CR)*

   Compression Ratio (CR) adalah persentase perbandingan antara data yang sudah dikompresi dengan data yang belum dikompresi.

   $$\boldsymbol{Cr = \frac{ukuran\ data\ setelah\ dikompresi}{ukuran\ data\ sebelum\ dikompresi}\ x\ 100\%}$$ (Salomom & Motta, 2010)

3. *Redudancy (Rd)*

   Redudancy (Rd) adalah kelebihan yang terdapat di dalam data sebelum dikompresi. Jadi setelah dikompresi dapat dihitung Redudancy data yaitu persentasi dari hasil selisih antara ukuran data sebelum dikompresi dengan data setelah dikompresi.

   $$\boldsymbol{Rd = 100\% - Cr}$$ (Salomon & Motta, 2010)

4. *Space Savings (SS)*

   Space Savings (SS) adalah persentase selisih antara data yang belum dikompresi dengan besar data yang dikompresi.

   $S_s = 1 - C_R$ (Salomon & Motta, 2010)

5. Waktu kompresi dan dekompresi

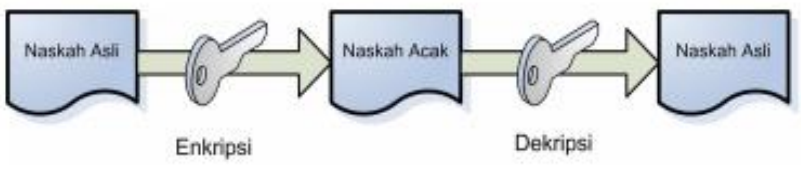
   Waktu kompresi dan dekompresi adalah waktu yang dibutuhkan untuk melakukan proses kompresi dan dekompresi. Semakin kecil waktu yang diperoleh maka semakin efisien metode yang digunakan dalam proses kompresi dan dekompresi itu.

2.2 Kriptografi

2.2.1. Definisi Kriptografi

Kriptografi merupakan ilmu yang mempelajari teknik matematika yang berhubungan dengan aspek keamanan informasi, seperti kerahasiaan data, keabsahan data, dan integritas data serta autentifikasi data (Menezes, 1996). Kriptografi adalah ilmu mengenai teknik enkripsi dimana data diacak menggunakan suatu kunci enkripsi menjadi sesuatu yang sulit dibaca oleh seseorang yang tidak memiliki kunci dekripsi (Kromodimoeljo, 2010)..

Dekripsi menggunakan kunci dekripsi mendapatkan kembali data asli. Proses enkripsi dilakukan menggunakan suatu algoritma dengan beberapa parameter. Biasanya algoritma tidak dirahasiakan, bahkan enkripsi yang mengandalkan kerahasiaan algoritma dianggap sesuatu yang tidak baik. Rahasia terletak di beberapa parameter yang digunakan, jadi kunci ditentukan oleh parameter. Parameter yang menentukan kunci dekripsi itulah yang harus dirahasiakan (parameter menjadi ekuivalen dengan kunci), (Kromodimoeljo, 2010).



Gambar 2.1 Proses Enkripsi dan Dekripsi (Kromodimoeljo, 2010)

Dapat dilihat pada gambar 1 merupakan proses dari enkripsi dan dekripsi suatu file teks. Pesan asli disebut plaintext, dan pesan menyamar disebut ciphertext. Pesan terakhir, dikemas dan dikirim, disebut kriptogram. Proses transformasi plaintext menjadi ciphertext disebut enkripsi atau enciphering. Proses kebalikan dari mengubah ciphertext menjadi plaintext, yang dicapai oleh penerima yang memiliki pengetahuan untuk menghapus menyamar, disebut dekripsi atau mengartikan. Siapapun yang terlibat dalam kriptografi disebut kriptografer (Mollin, 2007).

2.2.2. Sistem Kriptografi

Secara umum, istilah kriptografi yang sering digunakan adalah (Munir, 2006):

1. **Pesan**, yaitu data atau informasi yang dapat dibaca dan dimengerti maknanya, pesan sering juga disebut dengan *plaintext* atau teks jelas (*cleartext)*. *Plaintext* merupakan suatu pesan bermakna yang akan diproses menggunakan algoritma kriptografi.
2. ***Ciphertext****,* yaitu pesan yang telah tersandi. Pesan dalam bentuk *ciphertext* tidak dapat dibaca karena berisi karakter-karakter yang tidak memiliki makna setelah melalui proses enkripsi.
3. **Enkripsi**, yaitu proses penyandian *plaintext* menjadi *ciphertext* atau disebut *enciphering*. Enkripsi dilakukan dengan tujuan agar *plaintext* tersebut tidak dapat dibaca oleh pihak yang tidak memiliki otoritas (wewenang).
4. **Dekripsi**, yaitu proses pengembalian *ciphertext* menjadi *plaintext* semula atau disebut *deciphering*. Dekripsi dilakukan ketika pesan telah sampai kepada pihak yang dituju.
5. **Kunci (*key)*** adalah parameter yang digunakan untuk transformasi enkripsi dan dekripsi. Kunci dapat juga berupa *string* atau deretan bilangan. Keamanan suatu algoritma kriptografi biasanya bergantung kepada kerahasiaan penyebaran kunci (*key)*.
6. **Kriptosistem (*cryptosystem)***, yaitu perangkat keras atau implementasi perangkat lunak kriptografi yang diperlukan atau mentransformasi sebuah pesan asli menjadi *ciphertext* atau juga sebaliknya.

2.2.3. Tujuan Kriptografi

Menurut (Munir,2006) tujuan kriptografi dalam memberi layanan keamanan adalah sebagai berikut :

a. *Confidentiality*

Confidentiality merupakan tujuan sistem kriptografi dalam memberikan kerahasiaan pesan dan menyimpan data dengan menyembunyikan informasi lewat teknik-teknik enkripsi.

b. *Data Integrity*

Data integrity merupakan tujuan sistem kriptografi dalam memberikan jaminan bahwa pesan tidak akan mengalami perubahan selama proses pengiriman. Integritas data merupakan layanan yang bertujuan untuk mencegah terjadinya pengubahan informasi oleh pihak-pihak yang tidak berwenang.

c. *Non-repudiation*

Non-repudiation merupakan tujuan sistem kriptografi dalam memberikan cara untuk membuktikan bahwa suatu dokumen datang dari seseorang apabila ia mencoba menyangkal memiliki dokumen tersebut. Non-repudiation adalah layanan yang berfungsi untuk mencegah terjadinya penyangkalan terhadap suatu aksi yang dilakukan oleh pelaku sistem informasi.

d. *Authentication*

Authentication merupakan tujuan sistem kriptografi dalam mengidentifikasikan keaslian suatu pesan dan memberikan jaminan keotentikannya, dan menguji identitas seseorang apabila ia akan memasuki sebuah sistem. Penerima pesan dapat memastikan keaslian pengirimnya.

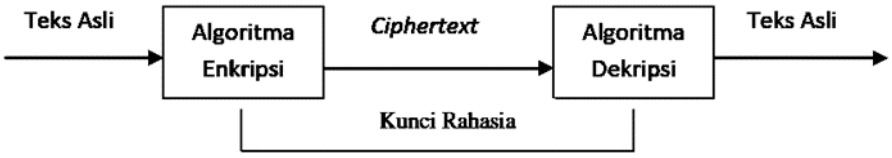
2.2.4. Jenis – jenis Algoritma Kriptografi

Berdasarkan jenis kunci, algoritma kriptografi dikelompokan menjadi dua bagian, yaitu :

1. Algoritma Kriptografi Simetri (Konvensional) / *Symetric Cryptosystem*
2. Algoritma Kriptografi Asimetri (Kunci Publik) / *Assymetric Cryptosystem.*

1. Algoritma Kriptografi Simetris (Konvensional)

Algoritma simetris adalah algoritma dimana kunci enkripsi dapat dihitung dari kunci dekripsi dan sebaliknya. Algoritma ini juga disebut algoritma kunci-rahasia, algoritma sama-kunci, algoritma satu-kunci, yang mengharuskan pengirim dan penerima menyetujui suatu kunci sebelum mereka dapat berkomunikasi dengan aman.

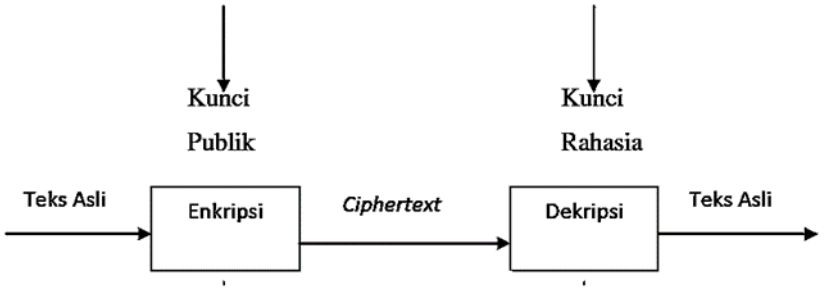


Gambar 2.2 Skema Kriptografi Simetris (Fauzana, 2013)

Sebuah *cryptosystem* disebut *symmetric-key* (juga disebut *single-key* dan konvensional) jika untuk setiap pasangan kunci (e, d), kunci d adalah "komputasi mudah" untuk menentukan d harus mengetahui e dan juga untuk menentukan e harus mengetahui d. Biasanya e = d dengan praktek *symmetric-key ciphers*, dengan demikian membenarkan penggunaan istilah kunci simetris (Mollin, 2007).

2. Algoritma Kriptografi Asimetris (Kunci Publik)

Algoritma asimetris disebut juga dengan kriptografi kunci publik karena algoritma ini memiliki kunci yang berbeda untuk enkripsi dan dekripsi, dimana enkripsi menggunakan *public-key* dan untuk dekripsinya, menggunakan *private-key*. *Public-key* dan *private-key* harus saling berpasangan secara matematis. Dengan memberikan *public key*, pembuat kunci berhak memberikan dan mendapatkan *public-key* agar pesan aman dan hanya bisa dibaca oleh si pembuat kunci.



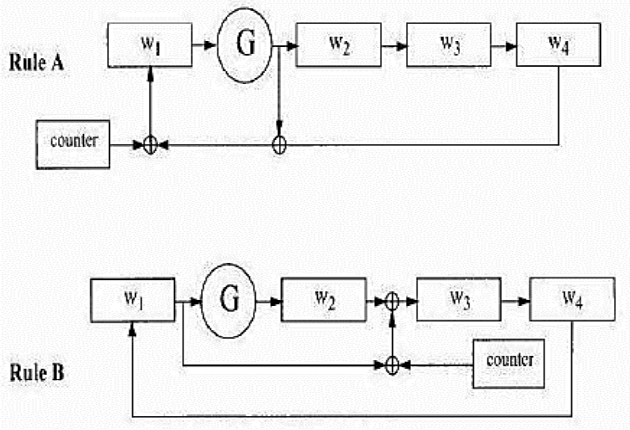
Gambar 2.3 Skema Enkripsi dan Dekripsi Kriptografi Asimetris (Suprianto, 2007)

2.3. Algoritma *Skipjack*

Skipjack adalah salah satu algoritma kriptografi blok cipher yang menggunakan 64 bit blok data dan 80 bit kunci. Dalam hal kemanan Skipjack lebih aman dibandingkan DES karena pada Skipjack digunakan 80 bit kunci dan dilakukan proses iterasi sebanyak 32 langkah untuk menghasilkan cipherteks. (Asri, 2009).

Skipjack merupakan iterated blok cipher dengan 32 putaran yang memiliki dua tipe, yaitu Rule A dan Rule B. Setiap putaran dibuat dalam bentuk *feedback linear shift register* dengan tambahan permutasi G non-linear. Rule B pada dasarnya adalah kebalikan dari Rule A dengan beberapa perbedaan pada pengaturan posisi. Skipjack melakukan 8 putaran terhadap Rule A, setelah itu melakukan 8 putaran terhadap Rule B, setelah itu melakukan 8 putaran lain dari Rule A dan 8 putaran lain dari Rule B. (Herdiantio, 2008).

Proses enkripsi dalam metode Skipjack terhadap suatu blok data dilakukan dengan menggunakan dua buah rule secara bergantian yakni rule A dan rule B. Sedangkan pada proses dekripsi, rule yang digunakan merupakan kebalikan *(inverse)* dari rule A dan rule B yakni *rule* A^{-1} dan *rule* B^{-1}. Proses *Rule* A dan *Rule* B dapat dilihat pada gambar 2.4



Gambar 2.4 Skema aturan Skipjack / *Rule* A dan *Rule* B

Fungsi permutasi pada metode Skipjack disebut dengan permutasi G yang merupakan 4 *round* dari struktur *Feistel*. Fungsi *round* tersebut merupakan tabel subtitusi *byte* yang *fixed*, yang dinamakan *F-Table* (gambar 3). Masing-masing *round* dari permutasi G juga memasukkan sebuah *cryptovariable*. Permutasi G dilakukan pada proses enkripsi di awal setiap rule yakni *Rule A* dan *Rule B.* Sedangkan pada proses dekripsi, permutasi dilakukan merupakan kebalikan (*inverse)* dari permutasi G yang disebut dengan G^{-1}.

Operasi matematik kriptografi yang digunakan di dalam *rule A, rule B, rule* A^{-1}*, dan rule* B^{-1} tersebut adalah XOR dan permutasi. Operasi permutasi dilakukan dengan menggunakan sebuah tabel substitusi yang disebut dengan F-Table dan kunci rahasia. Nilai – nilai dalam F-Table diberikan dalam nilai Heksadesimal. Baris pertama pada tabel (x0…xF) menunjukkan kolom sedangkan kolom pertama pada tabel (0x…Fx) menunjukkan baris. Sebagai contoh F(7A) = D6. Nilai – nilai F-Table dapat dilihat pada Gambar 2.5.

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	A3	d7	09	83	f8	48	f6	f4	b3	21	15	78	99	b1	af	f9
1x	e7	2d	4d	8a	ce	4c	ca	2e	52	95	D9	1e	4e	38	44	28
2x	0a	df	02	a0	17	f1	60	68	12	b7	7a	c3	e9	fa	3d	53
3x	96	84	6b	ba	f2	63	9a	19	7c	ae	e5	f5	f7	16	6a	a2
4x	39	b6	7b	0f	c1	93	81	1b	ee	b4	1a	ea	d0	91	2f	b8
5x	55	b9	da	85	3f	41	bf	e0	5a	58	80	5f	66	0b	d8	90
6x	35	d5	c0	a7	33	06	65	69	45	00	94	56	6d	98	9b	76
7x	97	fc	b2	c2	b0	fe	db	20	e1	eb	D6	e4	dd	47	4a	1d
8x	42	ed	9e	6e	49	3c	cd	43	27	d2	07	d4	de	c7	67	18
9x	89	cb	30	1f	8d	c6	8f	aa	c8	74	dc	c9	5d	5c	31	a4
Ax	70	88	61	2c	9f	0d	2b	87	50	82	54	64	26	7d	03	40
Bx	34	4b	1c	73	d1	c4	fd	3b	cc	fb	7f	ab	e6	3e	5b	a5
Cx	Ad	04	23	9c	14	51	22	f0	29	79	71	7e	ff	8c	0e	e2
Dx	0c	ef	bc	72	75	6f	37	a1	ec	d3	8e	62	8b	86	10	e8
Ex	08	77	11	be	92	4f	24	c5	32	36	9d	cf	f3	a6	bb	ac
Fx	5e	6c	a9	13	57	25	b5	e3	bd	a8	3a	01	05	59	2a	46

Gambar 2.5 ***F-Table*** **(Asri, 2009)**

Proses enkripsi Skipjack :

Saat proses dimulai, counter bernilai 1. Algoritma melakukan Rule A sebanyak 8 kali, lalu berganti melakukan Rule B sebanyak 8 kali. Berganti melakukan Rule A sebanyak 8 kali lagi dan menyelesaikan enkripsi dengan 8 putaran Rule B. Counter bertambah setiap menyelesaikan satu langkah

Proses dekripsi Skipjack :

Algoritma dimulai dengan nilai counter 32. Algoritma melakukan Rule B ^ -1 untuk 8 langkah, Rule A ^ -1 untuk 8 langkah, Rule B ^ -1 selama 8 langkah lagi dan akhirnya Rule A ^ -1 untuk 8 langkah lain. Counter berkurang satu setiap langkah.

2.4 Algoritma *Taboo*

Algoritma Taboo memiliki pendekatan terhadapat panjangnya suatu variable. Penggagas Algoritma Taboo adalah Steven Pigeon. Prinsip dari Algoritma Taboo yaitu untuk memilih bilangan positif n dan membalikkan pola n untuk mengindikasikan akhir dari kode tersebut. (Salomon, 2007)

Terdapat dua tipe yang yang ada dalam Algoritma Taboo diantaranya:

1. Tipe pertama dari Algoritma Taboo yaitu *block-based* dan memiliki panjang berupa kelipatan dari n. *Block based* Algoritma Taboo dari integer adalah string dari n- bit *blocks*, dimana nilai n dipilih oleh user dan blok terakhir memiliki sedikit pola taboo yang tidak dapat mucul pada blok lainnya. n-bit blok memilik nilai 2^n , jika salah satu nilai dicadangkan, masing-masing sisa kode blok dapat memiliki salah satu yang ada pada pola bit 2^n-1.

2. Tipe kedua dari Algoritma Taboo yaitu panjang total kode ini tidak terbatas pada kelipatann. Tipe ini dikatakan *unconstrained* dan ditunjukkan bahwa tipe ini memiliki relasi terhadap nomor fibonaci ke-n. (Salomon, 2007)

Rumus dari Algoritma Taboo codes yaitu:

$$g_n(k) = \sum_{i=1}^{k}(2^n - 1)^i = \frac{[(2^n-1)^k-1](2^n-1)}{2^n-2} \quad \text{.....(1)}$$

Gn(k) : jumlah total kode dalam rentang k

n : jumlah blok bit

Untuk tipe pertama kasus jika n=1 tidak dapat dilakukan. Sebuah blok 1-bit seperti 0 atau 1, jika kita cadangkan 0 sebagai pola tabu, maka semua blok lain dari kode tidak dapat berisi angka 0 dan hasrus semua 1. Hasilnya adalah himpunan tak terbatas seperti code berikut 10,110,1110,11110,.....

Tipe pertama dapat dilakukan jika n=2, maka blok 2-bit seperti 00 atau 11 maka dapat dibentuk pola taboo yang bisa dibuat tidak seperti blok taboo tersebut sehingga terbentuk pola seperti berikut:

Tabel 2.1 Pola Algoritma *Taboo*

M	Code	M	Code	m	Code	M	Code
0	01 00	4	01 10 00	8	10 11 00	12	01 01 01 00
1	10 00	5	01 11 00	9	11 01 00	13	01 01 10 00
2	11 00	6	10 01 00	10	11 10 00	14	01 01 11 00
3	01 01 00	7	10 10 00	11	11 11 00	..	

2.5 Penelitian yang Relevan

Berikut ini beberapa penelitian yang terkait dengan kriptografi Algoritma Skipjack dan Algoritma Taboo :

1. Yessi Asri (2009) dalam jurnal yang berjudul Perancangan Aplikasi Enkripsi dan Dekripsi Data dengan menggunakan Algoritma Skipjack. Di dalam jurnal tersebut dinyatakan bahwa Skipjack tidak diusulkan menjadi kandidat untuk menjadi AES antara lain disebabkan karena hanya digunakan oleh badan pertahanan Amerika Serikat. Skipjack adalah salah satu algoritma kriptografi blok cipher yang menggunakan 64 bit blok data dan 80 bit kunci. Dalam hal kemanan Skipjack lebih aman dibandingkan DES karena pada Skipjack digunakan 80 bit kunci dan dilakukan proses iterasi sebanyak 32 langkah untuk menghasilkan cipherteks.

2. Salomon (2007) Algoritma Taboo memiliki pendekatan terhadapat panjangnya suatu variable. Penggagas Algoritma Taboo adalah Steven Pigeon. Prinsip dari Algoritma Taboo yaitu untuk memilih bilangan positif n dan membalikkan pola n untuk mengindikasikan akhir dari kode tersebut. Di dalam jurnal tersebut dinyatakan bahwa sebagian besar metode kompresi data yang didasarkan pada kode variable-panjang, ada sejumlah besar kode yang kurang dikenal yang memiliki sifat yang berguna seperti yang mengandung pola bit tertentu.

3. Mirza Amir, Herry (2006) Dalam era informasi, pemanfaatan informasi menjadi suatu hal yang sangat vital. Nilai suatu keputusan organisasi maupun individu

sangatlah bergantung pada kekayaan informasi yang dimilikinya. Saat ini informasi telah menjadi basis pengetahuan dalam pengambilan keputusan, tentunya hal ini merupakan kekuatan bagi individu maupun organisasi yang memilikinya. Disatu sisi, hal tersebut mendorong timbulnyabentuk-bentuk kriminalitas baru, yaitu segala usaha untuk mendapatkan informasi secara tidak sah. Salah satu cara untuk mengamankan data dan menjaga kerahasiaan data adalah dengan menggunakan kriptografi yaitu dengan menyandikan isi informasi (Plaintext) menjadi isi yang tidak dapat dipahami melalui proses enkripsi dan untuk memperoleh kembali informasi yang asli dilakukan proses dekripsi. Skipjack adalah salah satu algoritma kriptografi block cipher yang menggunakan 64 bit blok data dan 80 bit kunci serta dilakukan iterasi sebanyak 32 langkah untuk menghasilkan data sandian (ciphertext) sehingga dalam hal keamanan skipjack lebih aman dibanding algoritma kriptografi lain.

4. Cui T., Jin C., Zhang G.: Observations of Skipjack-like Structure with SP/SPS Round Function (2013), ciphers modern biasanya mengadopsi data 128 bit (atau lebih lama). Jika kita membuat 128 bit block cipher dengan struktur Feistel, kita perlu mencari fungsi putaran 64-bit. Namun, untuk membangun fungsi round 64-bit tidak seperti fungsi putaran 32-bit, dan fungsi putaran 64-bit akan membawa tambahan Biaya dalam implementasi juga. Dalam [Nyberg (1996)], Nyberg diperkenalkan secara umum Struktur rompi. Struktur Feistel yang umum adalah bentuk umum dari struktur Feistel klasik. Struktur ini menyimpan beberapa keuntungan klasik Feistel cipher seperti fleksibilitas dalam perancangan fungsi putaran, dan bisa diimplementasikan dengan mudah dengan mengadopsi fungsi bulat kecil.

5. Skipjack [Biham et al. (1999)], CAST256 [Adams (1999)], MARS Burwick (1998)], CLEFIA [Shirai et al. (2007)] dll Seri besar dari ciphers seperti menggunakan struktur ini sebagai struktur mereka arsitektur. Diantaranya, blok cipher Skipjack menggunakan dua jenis putaran, disebut Rule A dan Rule B. Di dalam the Skipjack cipher, blok data terbagimenjadi empat subblok, dan delapan putaran Peraturan A dan delapan ronde Aturan B diaplikasikan secara alternatif sampai putaran penuh 32 tercapai. Untuk mengukur tingkat keamanan Skipjack, orang sering menganggap Rule A atau Rule B

BAB 3
ANALISIS DAN PERANCANGAN SISTEM

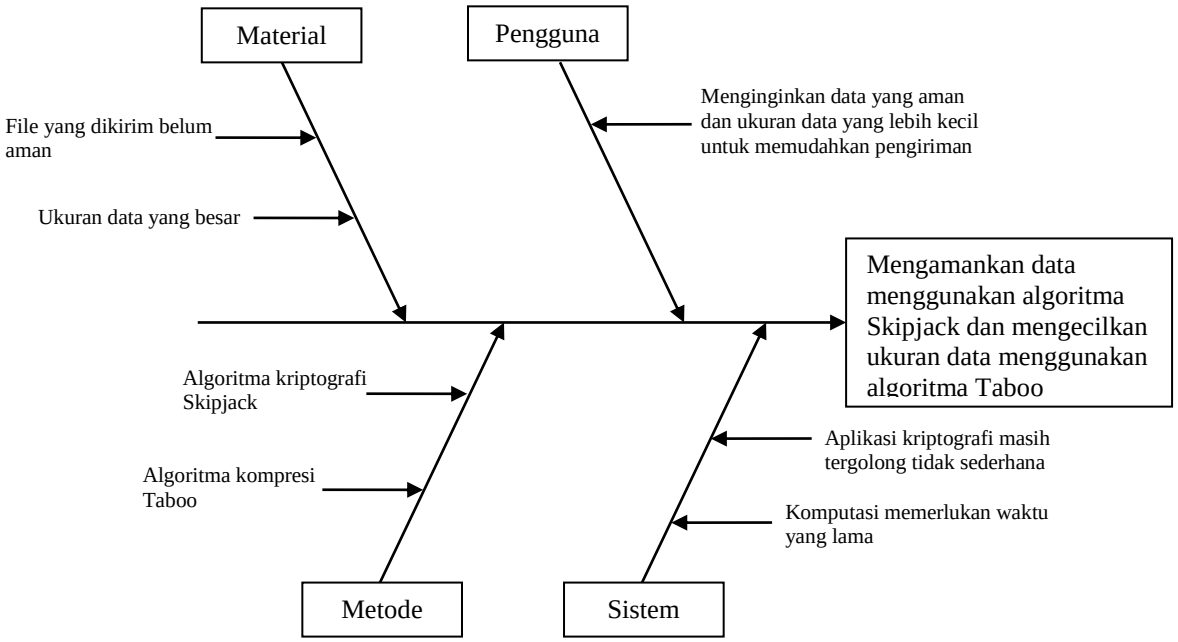
3.1 Analisis Sistem

Analisis sistem adalah suatu teknik pemecahan masalah dengan cara menguraikan, mengidentifikasi komponen dari sistem agar sistem dapat berjalan dengan baik (Whitten & Bentley, 2007). Untuk menganalisis sebuah sistem ada beberapa tahap yang dapat dilakukan yaitu :

3.1.1. Analisis masalah

Untuk mengatasi masalah keamanan data, penyimpanan dan kecepatan akses suatu proses pengiriman data adalah dengan kriptografi serta melakukan kompresi data. Algoritma kriptografi data sangat membantu pengguna untuk mengamankan data dari pihak yang tidak diinginkan untuk melihat data serta algoritma kompresi data sangat membantu pengguna untuk memperkecil ukuran data dari data aslinya agar ruang penyimpanan lebih kecil serta memudahkan proses transmisi data. Algoritma Taboo adalah salah satu algoritma kompresi dan algoritma Skipjack adalah salah satu algoritma kriptografi. Kedua algoritma ini akan digabungkan yang bertujuan agar data menjadi aman serta memperoleh ukuran data yang kecil sehingga memudahkan proses transmisi data.

Diagram *Ishikawa* adalah diagram yang dapat menggambarkan hubungan sebab akibat dalam sebuah sistem. Analisa masalah lebih jelas melalui diagram *Ishikawa* yang dapat dilihat pada Gambar 3.1.



Gambar 3.1 Diagram *Ishikawa*

Pada Gambar 3.1 dapat dilihat bahwa terdapat empat kategori penyebab masalah yang digambarkan dengan tanah panah yang mengarah ke tulang utama, yaitu berkaitan dengan Material, Pengguna, Metode, dan Sistem. Setiap detail penyebab masalah tersebut digambarkan dengan tanda panah yang mengarah pada masing-masing kategori.

3.1.2. Analisis kebutuhan

Analisis ini bertujuan untuk mengidentifikasi dan menyatakan apa saja yang akan dibutuhkan oleh sistem agar dapat berjalan sesuai dengan yang diharapkan. Analisis persyaratan dibagi menjadi dua bagian yaitu persyaratan fungsional (*functional requirement)* dan persyaratan nonfungsional (*nonfunctional requirement).*

a. Kebutuhan Fungsional

Kebutuhan fungsional berhubungan langsung dengan proses sistem, dimana membahas tentang layanan sistem yang disediakan dan dapat dilakukan oleh sistem (Dennis, et al. 2012). Berikut adalah layanan yang dapat dilakukan oleh sistem:

1. Sistem harus mampu membaca *string* yang berada dalam *file* teks.
2. Sistem harus mampu melakukan kompresi data terhadap *file* teks dengan ekstensi file *.txt dan *.doc dengan algoritma Taboo.

3. Sistem harus mampu melakukan enkripsi terhadap *file* yang telah dikompresi dengan ekstensi *file* *.tb dengan algoritma Skipjack.
4. Sistem harus mampu melakukan dekripsi terhadap hasil enkripsi dengan algoritma Skipjack ke *file* aslinya yaitu *file* kompresi dengan ekstensi *file* *tb.
5. Sistem harus mampu melakukan dekompresi data terhadap hasil dekripsi dengan algoritma Taboo ke *file* aslinya yaitu *file* teks dengan ekstensi *.doc dan *.txt.

b. Persyaratan Non-Fungsional

Persyaratan non-fungsional adalah kebutuhan yang harus dipenuhi agar aplikasi yang dirancang mendapat umpan-balik yang baik dari pengguna aplikasi. Kebutuhan non-fungsional yang harus dipenuhi aplikasi yang dirancang adalah sebagai berikut:

1. Per*form*a

   Sistem harus mampu melakukan proses kompresi, dekompresi, enkripsi dan dekripsi dengan waktu yang relative singkat sesuai dengan ukuran data *input* yang diberikan.
2. In*form*asi

   Sistem harus mampu menyediakan in*form*asi tentang data-data yang akan digunakan pada sistem.
3. Ekonomi

   Sistem harus dapat bekerja dengan baik tanpa harus mengeluarkan biaya tambahan dalam penggunaan perangkat keras maupun perangkat lunak.
6. Kontrol

   Sistem yang telah dibangun harus tetap dikontrol setelah selesai dirancang agar fungsi dan kinerja sistem tetap terjaga dan dapat memberikan hasil yang sesuai dengan keinginan pengguna.
7. Efisiensi

   Sistem harus dirancang sesederhana mungkin agar memudahkan pengguna dalam menggunakan atau menjalankan aplikasi tersebut.

6. Pelayanan

   Sistem yang telah dirancang bisa dikembangkan ke tingkat yang lebih kompleks lagi bagi pihak-pihak yang ingin mengembangkan sistem tersebut.

3.1.3. Analisis Proses

Analisis proses merupakan suatu tahapan untuk menyelesaikan suatu masalah tertentu dengan cara memahami dan menguraikan proses secara spesifik. Analisis proses dilakukan pada tahap pemodelan sistem.

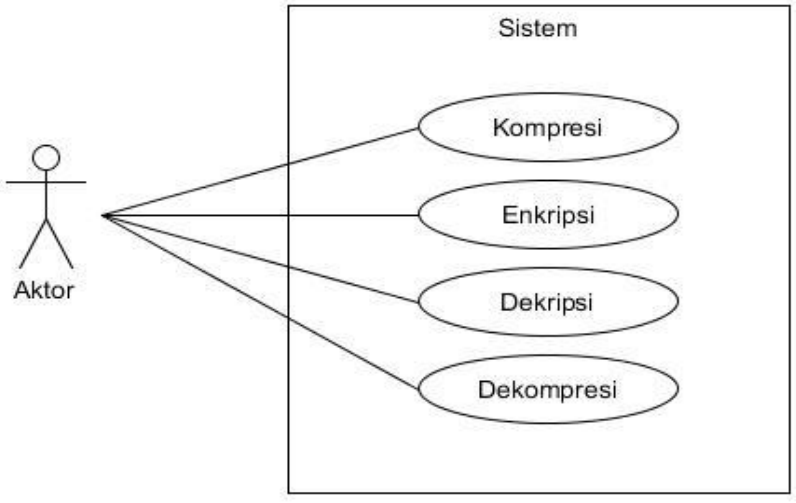
Dalam sistem ini terdapat empat proses utama yaitu proses kompresi, enkripsi, dekripsi dan dekompresi dengan menggunakan algoritma Taboo untuk proses kompresi dan dekompresi serta algoritma Skipjack untuk enkripsi dan dekripsi.

3.2 Pemodelan Sistem

Pemodelan sistem dilakukan untuk memperoleh gambaran yang lebih jelas tentang objek apa saja yang akan berinteraksi dengan sistem serta hal-hal apa saja yang harus dilakukan oleh sebuah sistem. Pemodelan ini menggunakan UML (*Unified Modelling Languange*).

3.2.1 Use case diagram

Use Case Diagram merupakan bentuk pemodelan dari sistem yang menggambarkan *functional requirements* dari sebuah sistem. *Use case diagram* digunakan untuk menggambarkan hubungan atau interaksi antara sistem dengan pengguna. *Use case* diagram untuk sistem dalam penelitian ini dapat dilihat pada Gambar 3.2.



Gambar 3.2 *Use Case Diagram* pada sistem

Use case diagram pada gambar 3.2 dapat mendeskripsikan narasi *use case* yang memiliki 1 aktor yang dinamakan pengguna (*user*). Aktor *user* dapat melakukan :

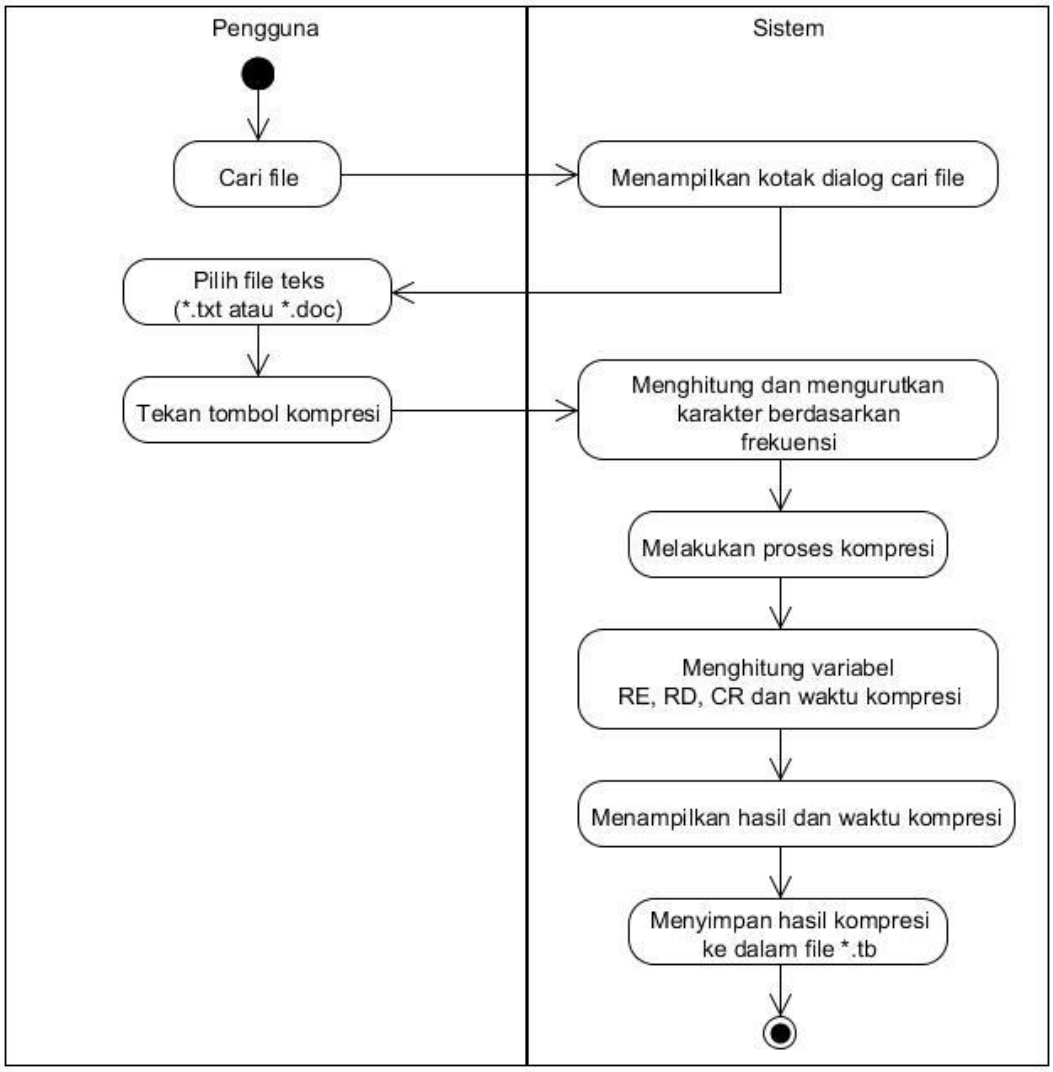
1. User memilih *file* teks yang akan di kompresi. Sistem akan mengkompresi *file* menggunakan algoritma Taboo dan menampilkan hasil kompresi. Selanjutnya *user* menyimpan *file* hasil kompresi.
2. User memasukkan key (*.key) dan file hasil kompresi yang akan di enkripsi. Sistem melakukan proses enkripsi menggunakan algortima Skipjack dan menampilkan hasil enkripsi. Kemudian user menyimpan *file* hasil enkripsi.
3. User memilih key (*.key) dan file hasil enkripsi yang akan di dekripsi. Sistem melakukan proses dekripsi dan menampilkan hasil dekripsi. Kemudian user menyimpan *file* hasil dekripsi.
4. User memilih file hasil kompresi yang akan di dekompresi. Sistem melakukan proses dekompresi dan menampilkan hasil dekompresi. Kemudian user menyimpan *file* hasil enkripsi.

3.2.2 Activity Diagram

Activity diagram adalah bentuk pemodelan dari sistem yang menggambarkan alur dari proses yang terjadi pada sebuah *use case* dan untuk menggambarkan logika dari suatu sistem. *Activity diagram* dibuat berdasarkan *use case* yang telah ditentukan sebelumnya pada proses *requirement analysis*.

a. *Activity Diagram* pada proses kompresi

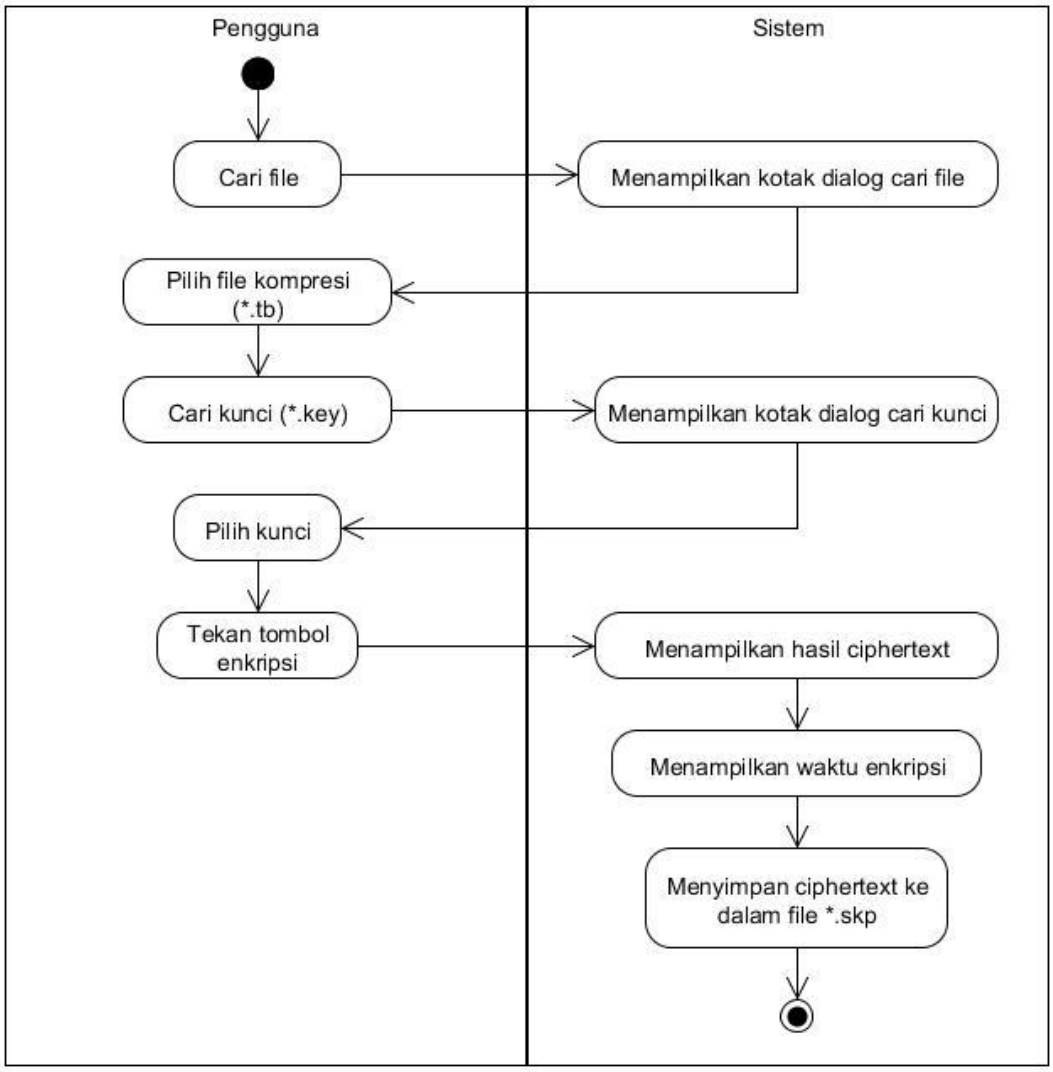
Pada proses kompresi, aktivitas yang berlangsung dapat digambarkan pada *Activity Diagram* gambar 3.3 sebagai berikut.



Gambar 3.3 *Activity Diagram* proses kompresi

b. *Activity Diagram* pada proses enkripsi

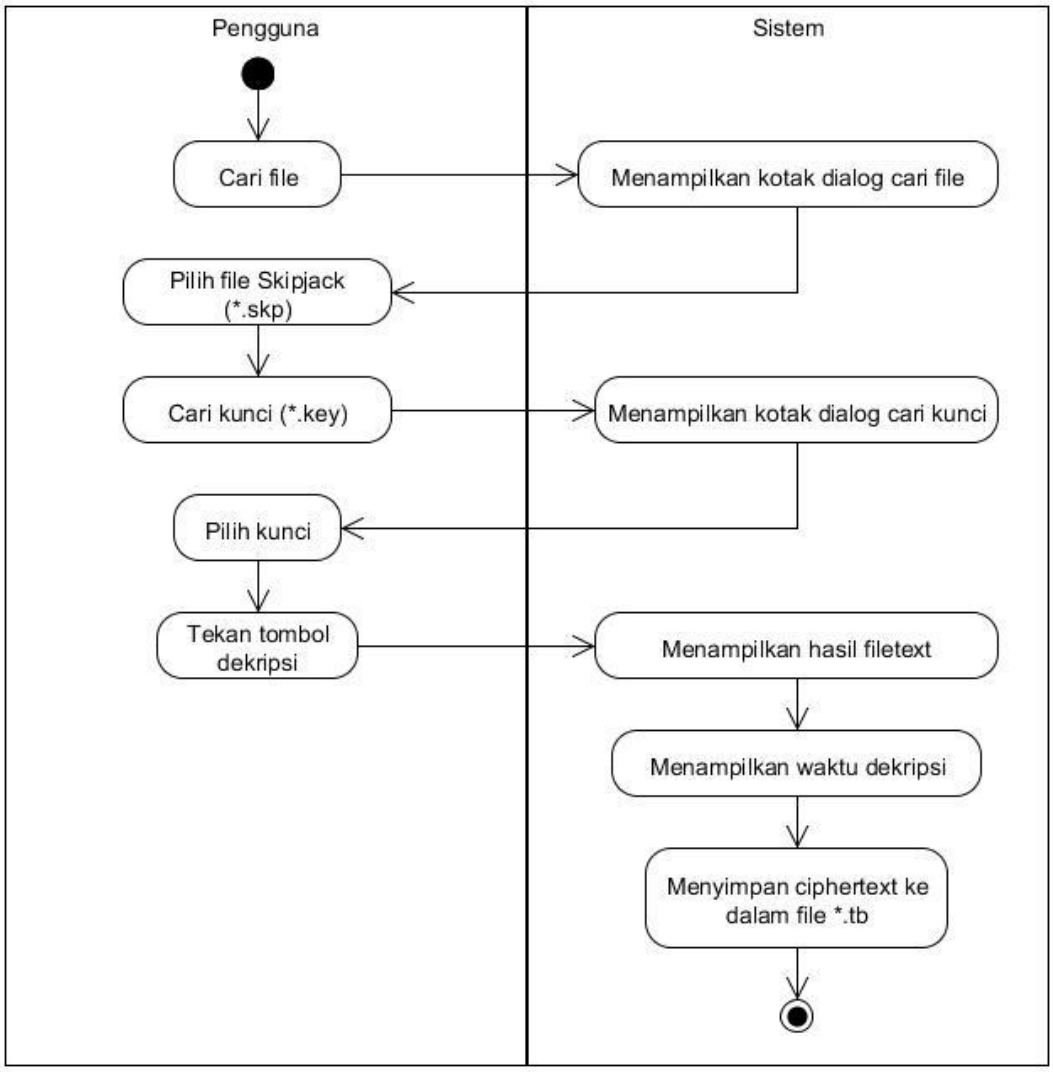
Pada proses enkripsi, aktivitas yang berlangsung dapat digambarkan pada *Activity Diagram* gambar 3.4 sebagai berikut.



Gambar 3.4 *Activity Diagram* proses enkripsi

c. *Activity Diagram* pada proses dekripsi

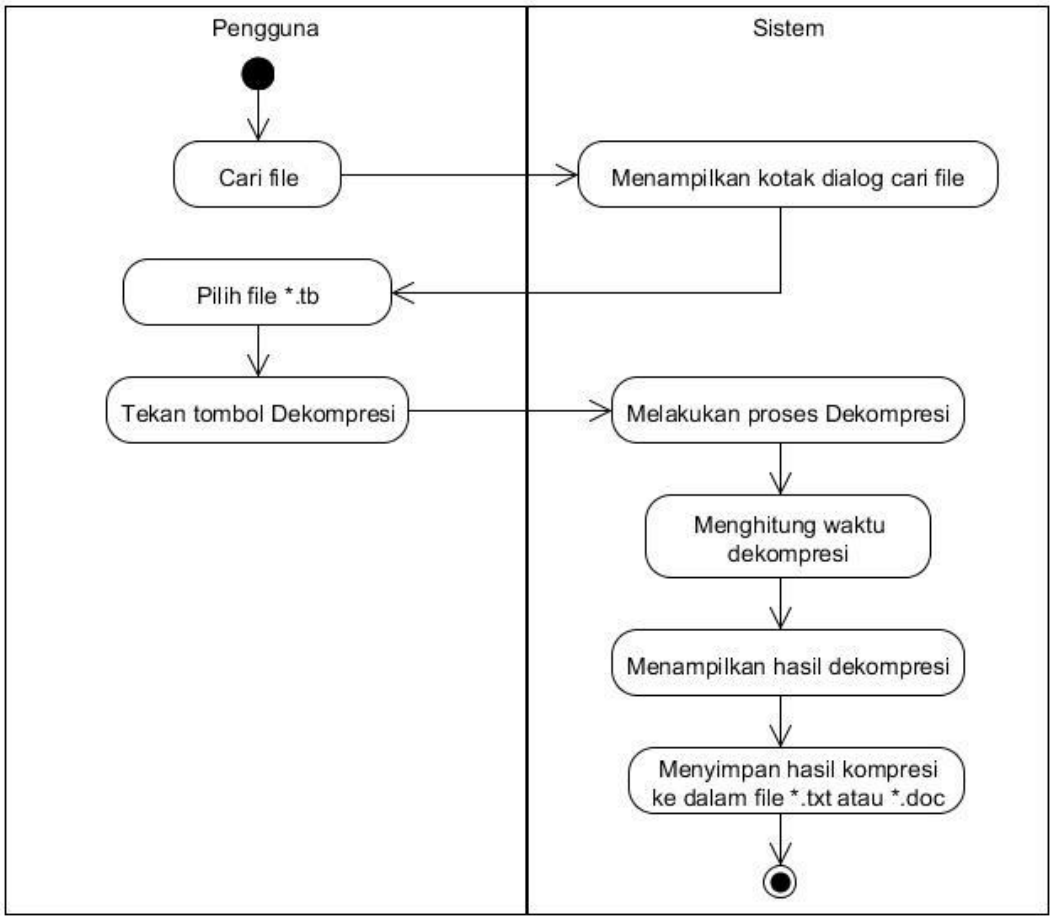
Pada proses dekripsi, aktivitas-aktivitas yang berlangsung dapat digambarkan pada *Activity Diagram* gambar 3.5 sebagai berikut.



Gambar 3.5 *Activity Diagram* Proses Dekripsi

d. *Activity Diagram* pada proses dekompresi

Pada proses dekompresi, aktivitas-aktivitas yang berlangsung dapat digambarkan pada *Activity Diagram* gambar 3.6 sebagai berikut.



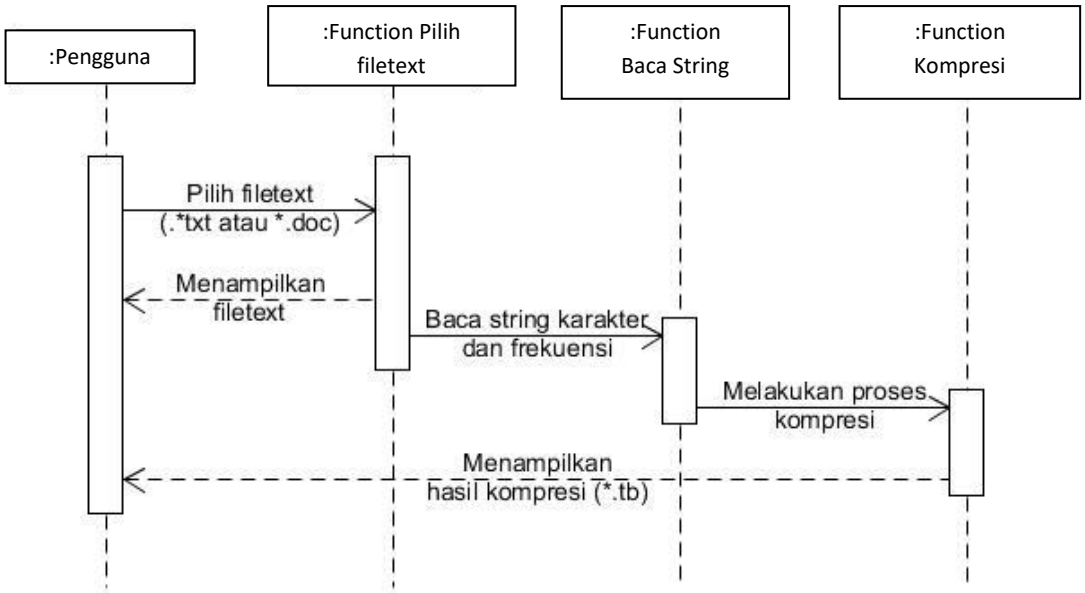
Gambar 3.6 *Activity Diagram* Proses dekompresi

3.2.3 Sequence Diagram

Sequence diagram adalah suatu diagram yang memperlihatkan atau menampilkan interaksi-interaksi antar objek di dalam sistem yang disusun pada sebuah urutan atau rangkaian waktu. *Sequence Diagram* untuk sistem yang dirancang pada penelitian ini adalah sebagai berikut :

a. *Sequence Diagram* pada proses kompresi

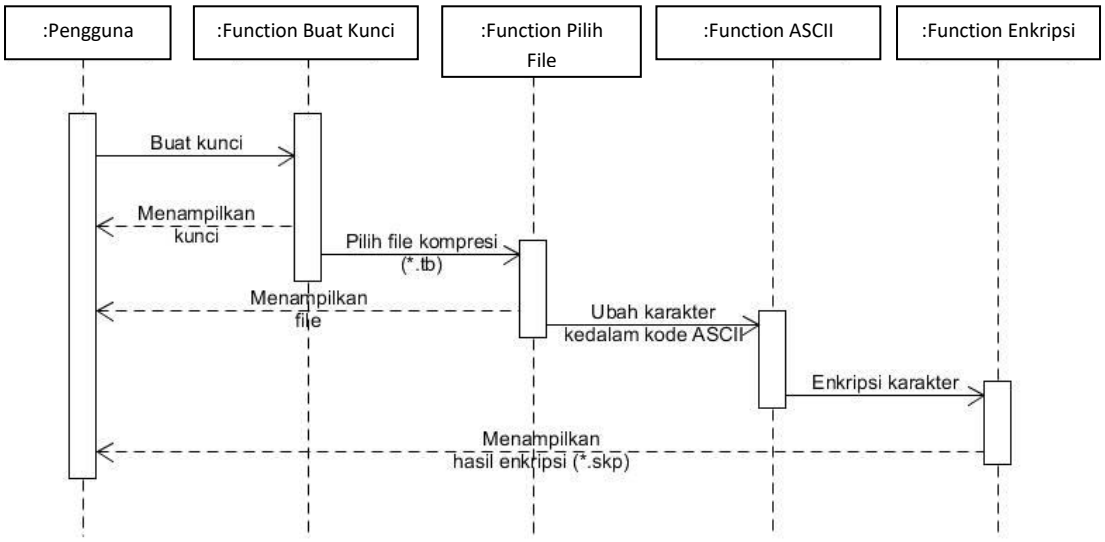
   Sequence Diagram untuk proses kompresi dapat dilihat pada Gambar 3.7



Gambar 3.7 *Sequence Diagram* Pada Proses Kompresi

b. *Sequence Diagram* pada proses enkripsi

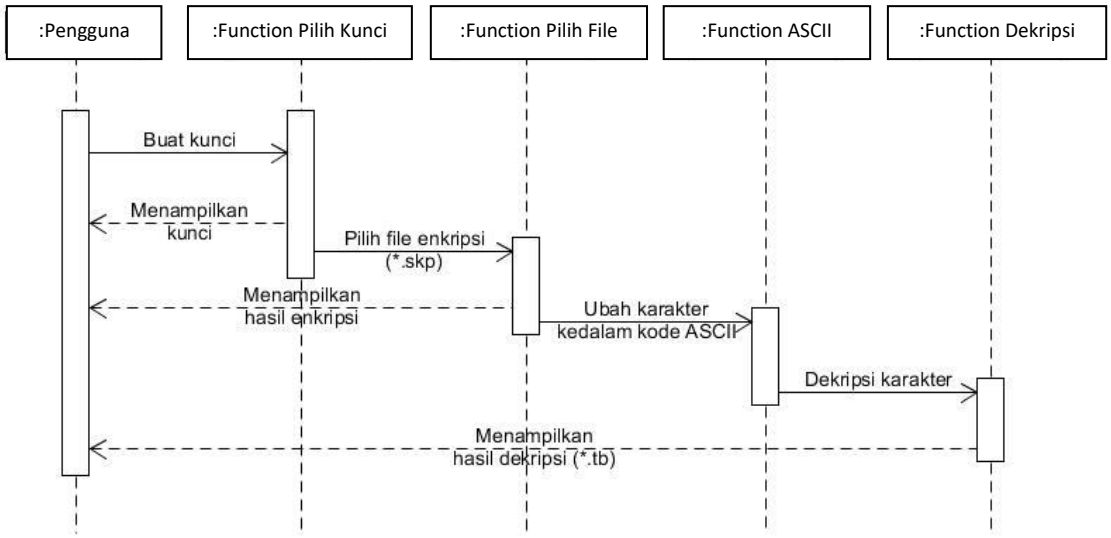
Sequence Diagram untuk proses enkripsi dapat dilihat pada Gambar 3.8



Gambar 3.8 *Sequence Diagram* Pada Proses Enkripsi

c. *Sequence Diagram* pada proses dekripsi

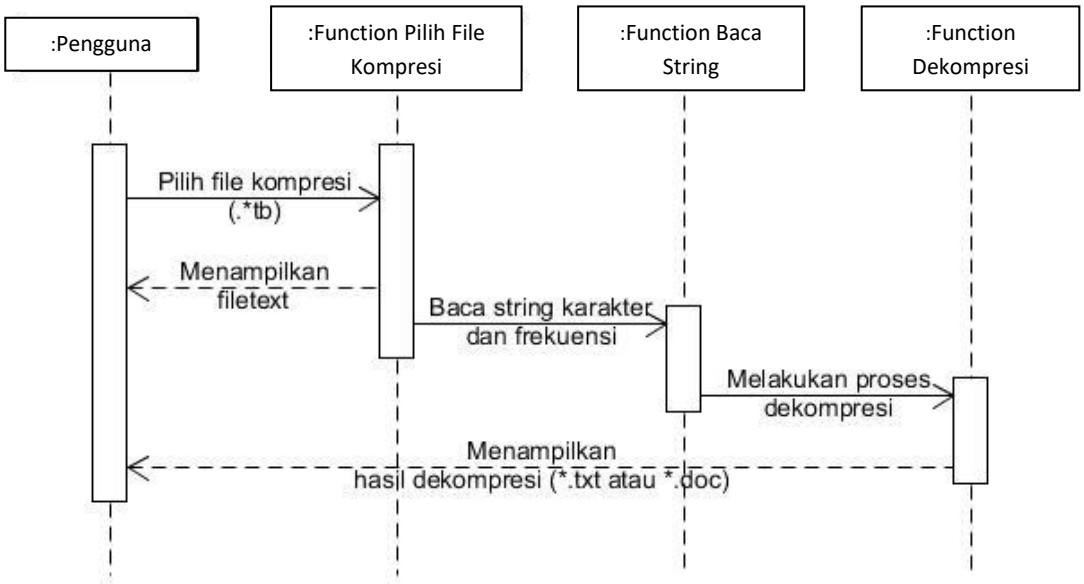
Sequence Diagram untuk proses dekripsi dapat dilihat pada Gambar 3.9



Gambar 3.9 *Sequence Diagram* Pada Proses Dekripsi

d. *Sequence Diagram* pada proses dekompresi.

Sequence Diagram untuk proses dekompresi dapat dilihat pada Gambar 3.10.

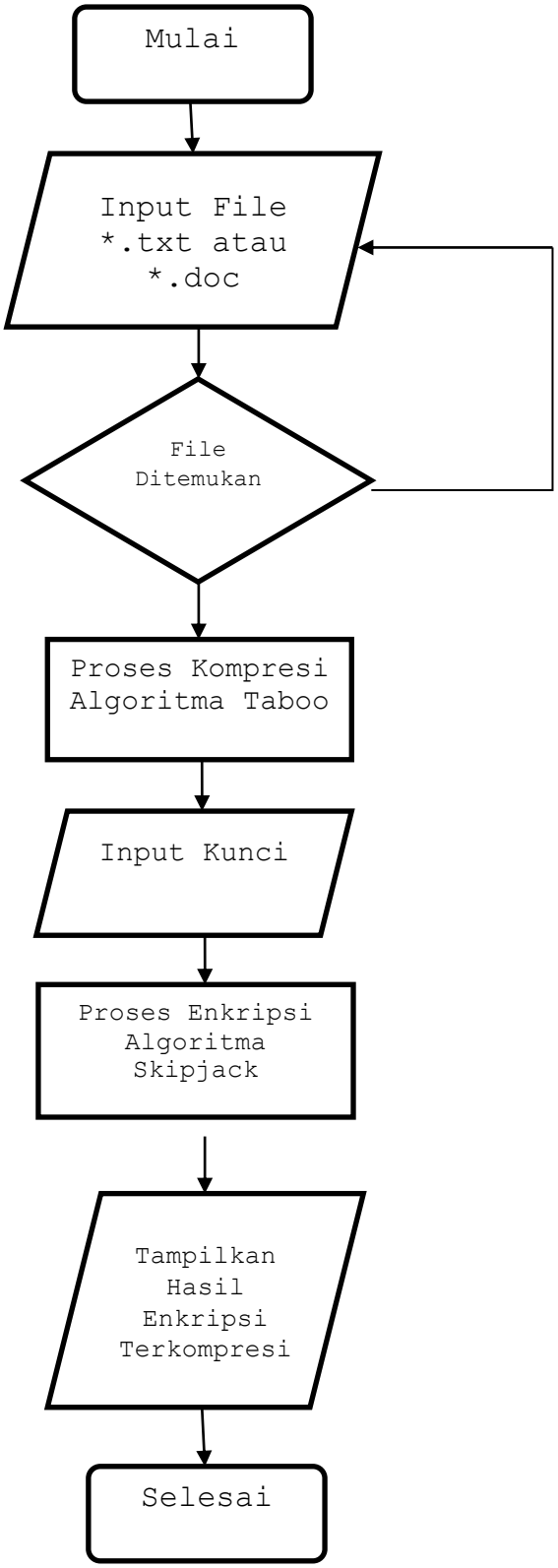


Gambar 3.10 *Sequence Diagram* Pada Proses Dekompresi

3.2.4. Flowchart dan *General Architecture*

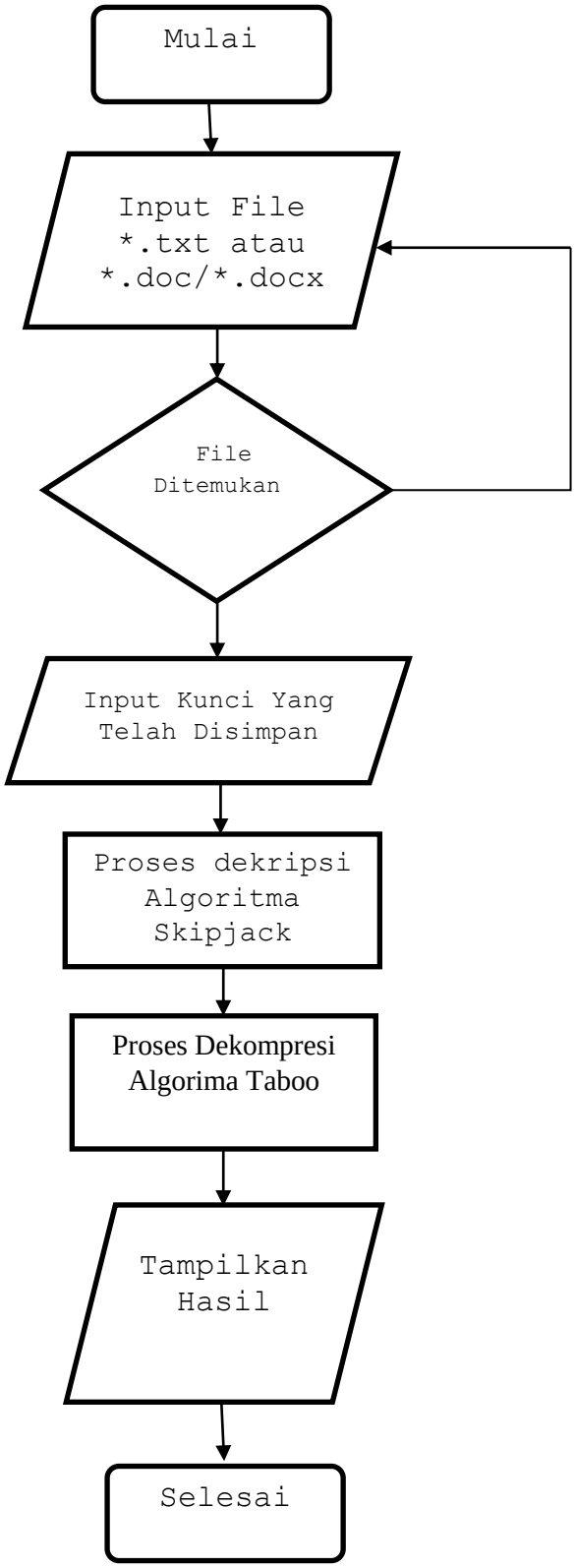
3.2.4.1 Flowchart sistem secara umum

Perancangan sistem secara umum digambarkan dengan *flowchart* seperti pada Gambar 3.11 dan 3.12



Gambar 3.11 *Flowchart* Sistem Secara Umum Untuk Proses Kompresi dan Enkripsi

Dari gambar 3.11 fungsi dari sistem adalah dengan menginputkan *file* .*txt atau *file*.*doc. Jika *file* tidak ditemukan maka akan kembali ke proses input file, tetapi jika sudah ditemukan maka sistem akan memproses dengan mengkompresi *file* .*txt atau *file*.*doc menggunakan algoritma *taboo.* Setelah proses kompresi berhasil maka akan masuk ke eksekusi enkripsi algoritma *skipjack* yang dimana menginputkan terlebih dahulu kunci rahasia sebagai pengamanan *file* yang sudah di kompresi. Kemudian setelah proses input kunci selesai, hasil dari file yang sudah terkompresi akan dienkripsi dan proses enkripsi selesai.

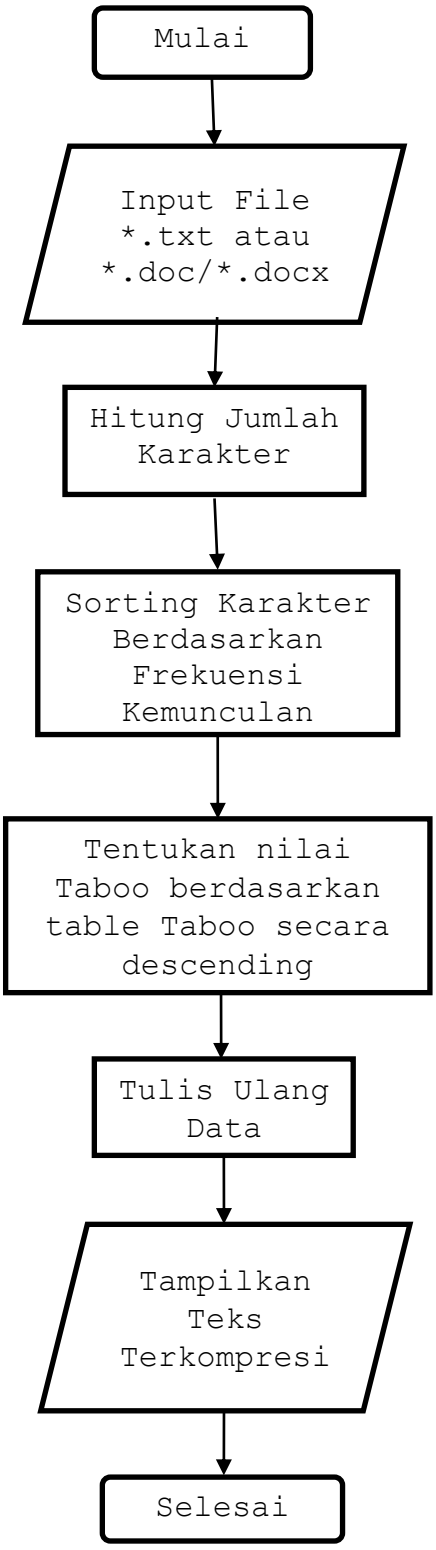


Gambar 3.12 *Flowchart* Sistem Secara Umum Untuk Proses Dekripsi dan Dekompresi

Dari gambar 3.12 fungsi dari sistem adalah dengan menginputkan *file* .*txt atau *file*.*doc. Jika *file* tidak ditemukan maka akan kembali ke proses input *file*, tetapi jika sudah ditemukan maka sistem akan memproses dengan meminta untuk menginputkan kunci yang sudah disimpan. Jika sudah diinputkan kemudian sistem akan memproses hasil *file* enkripsi yang telah disimpan dan akan melakukan proses dekripsi *file* dengan algoritma *skipjack*. Setelah proses dekripsi selesai maka hasil dekripsi akan didekompresi dengan algortima *taboo* dan akan mendapatkan *file* yang sudah didekompresi dan proses selesai

3.2.4.2 Flowchart Algoritma *Taboo*

Flowchart algoritma Taboo dapat dilihat pada gambar 3.13 dibawah ini :

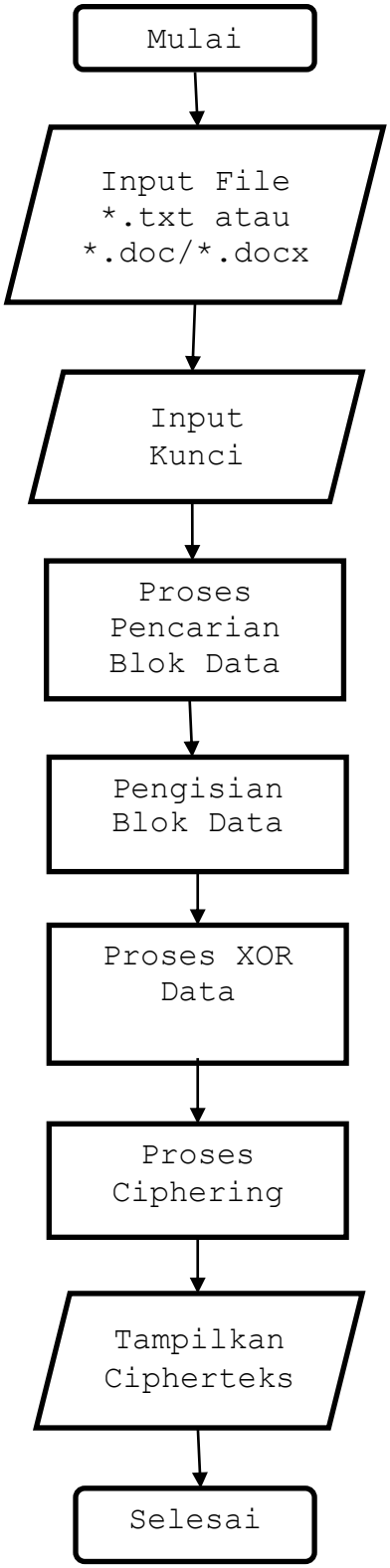


Gambar 3.13 *Flowchart* Algoritma Taboo

Dari gambar 3.13 fungsi dari sistem adalah menginputkan *file* .*txt atau *file*.*doc. Jika *file* tidak ditemukan maka akan kembali ke proses input *file*, jika ditemukan maka sistem akan memproses dengan algoritma *taboo* dengan menghitung jumlah karakter yang ada di dalam *file* .*txt atau *file*.*doc. Setelah selesai menghitung jumlah karakter maka akan melakukan proses sorting karakter yang berdasarkan frekuensi kemunculan. Setelah selesai melakukan sorting karakter akan ditentukan nilai taboo berdasarkan table taboo secara descending. Kemudian dari hasil penentuan nilai maka akan melakukan proses menulis ulang data yang sudah ditentukan nilai taboo. Setelah selesai tulis ulang data maka akan di dapat teks yang sudah terkompresi dan proses selesai.

3.2.4.3 Flowchart Algoritma *Skipjack*

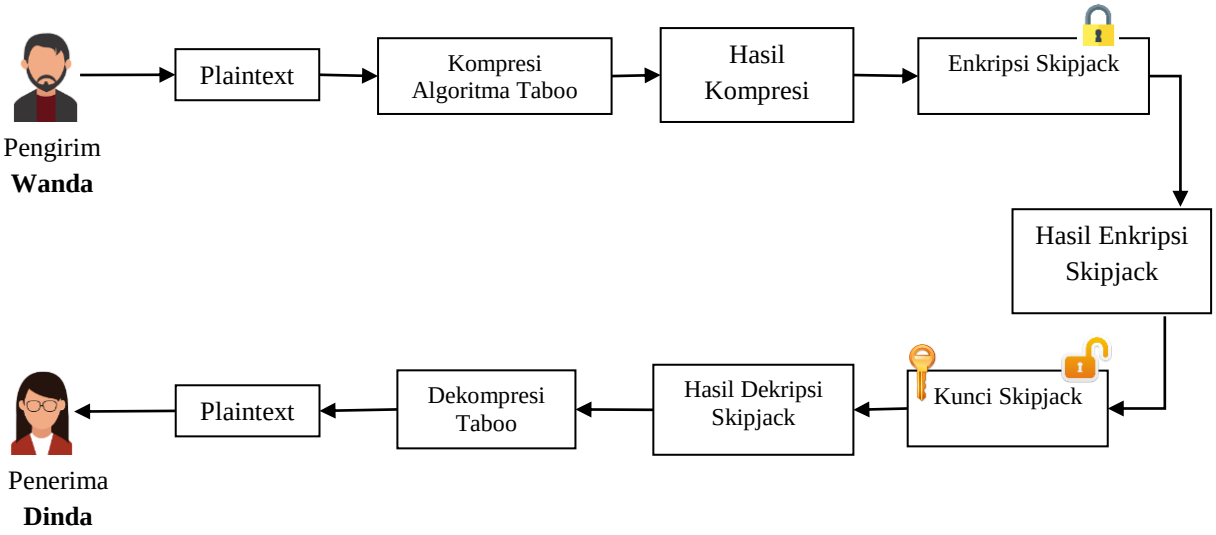
Flowchart proses enkripsi algoritma Skipjack dapat dilihat pada gambar 3.14 dibawah ini :



Gambar 3.14 *Flowchart* Algoritma Skipjack

Dari gambar 3.14 fungsi dari sistem adalah menginputkan *file* .*txt atau *file*.*doc. Jika *file* tidak ditemukan maka akan kembali ke proses input *file*, jika *file* ditemukan maka sistem akan memproses dengan algoritma *skipjack* dengan menginputkan kunci rahasia *skipjack*. Setelah selesai menginputkan kunci maka akan melakukan proses pencarian blok data dan proses pengisian blok data dengan melakukan proses iterasi blok *chiper* sebanyak 32 langkah untuk menghasilkan *cipherteks*. Hasil dari pengisian data kemudian akan diproses XOR data dan dilakukan proses ciphering untuk menghasilkan *cipherteks* hasil enkripsi *skripjack* dan proses selesai.

3.2.4.4 General Architecture



Gambar 3.15 *General Architecture System*

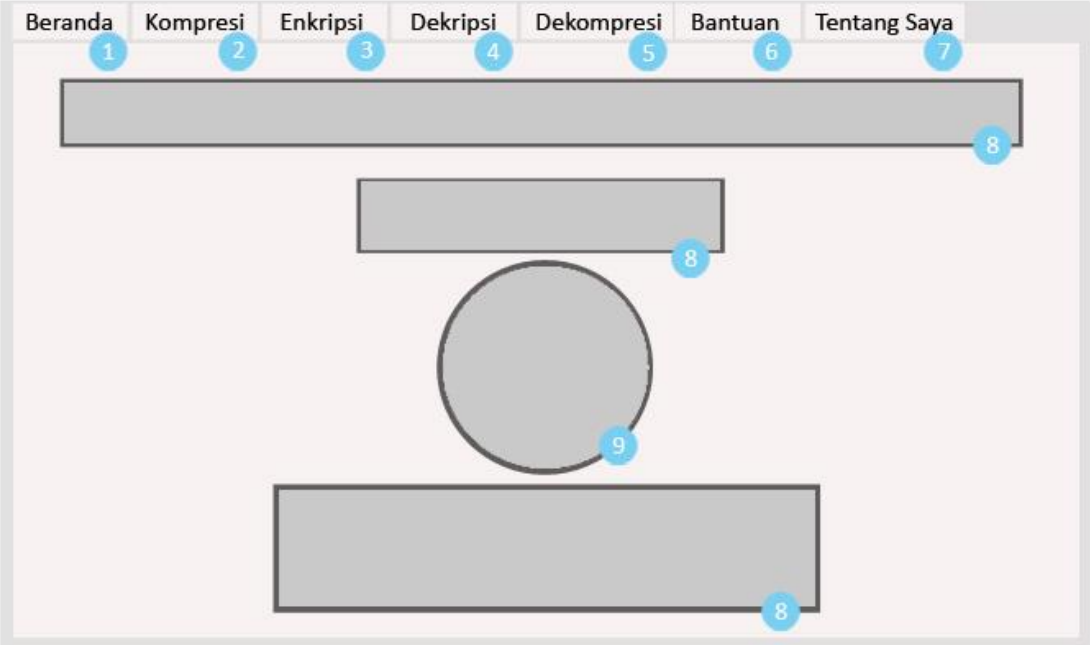
Pada gambar , menjelaskan bahwa sistem pertama kali melakukan *input* pesan teks *(plaintext)*. Pesan teks tersebut dikompresi dengan Algoritma *Taboo* dan menghasilkan output yaitu hasil kompresi. Kemudian melakukan *input* kunci rahasia enkripsi *skipjack* (*ChiperKey*). Setelah itu *plaintext* dienkripsi maka akan menghasilkan *ciphertext*. *Ciphertext* didekripsi menggunakan kunci Skipjack untuk menghasilkan file teks asli (plaintext). Kemudian didekompresi dengan Algoritma *Taboo* Untuk mengembalikan file asli input hasil kompresi.

3.3 Perancangan Interface

Pada perancangan sistem terdapat pembuatan *Graphic User Interface* (GUI) yang akan mempermudah user dalam menggunakan sistem yang ada. Pada sistem, terdapat 7 buah *Form* yang akan ditampilkan antara lain *Form Beranda*, *Form* Kompresi, *Form* Enkripsi, *Form* Dekripsi, *Form* Dekompresi, *Form* Bantuan dan *Form Tentang Saya*.

a. *Form Beranda*

Pada halaman *Beranda* terdapat keterangan mengenai sistem yaitu judul dari skripsi yang dibuat, logo universitas dan juga nama. *Form Beranda* pada aplikasi dapat dilihat pada Gambar 3.16.



Gambar 3.16 *Form Beranda* pada Sistem

Keterangan :

1. Tab Beranda

Berguna untuk menampilkan menu *Beranda* pada sistem yang ada.

2. Tab Kompresi

Berguna untuk menampilkan menu Kompresi pada sistem.

3. Tab Enkripsi

Berguna untuk menampilkan menu Enkripsi pada sistem.

4. Tab Dekripsi

Berguna untuk menampilkan menu Dekripsi pada sistem.

5. *Tab* Dekompresi

   Berguna untuk menampilkan menu Dekompresi pada sistem.

6. *Tab* Bantuan

   Berguna untuk menampilkan menu Bantuan yang ada pada sistem

7. *Tab Tentang Saya*

   Berguna untuk menampilkan menu *Tentang Saya* pada sistem.

8. *Text Box*

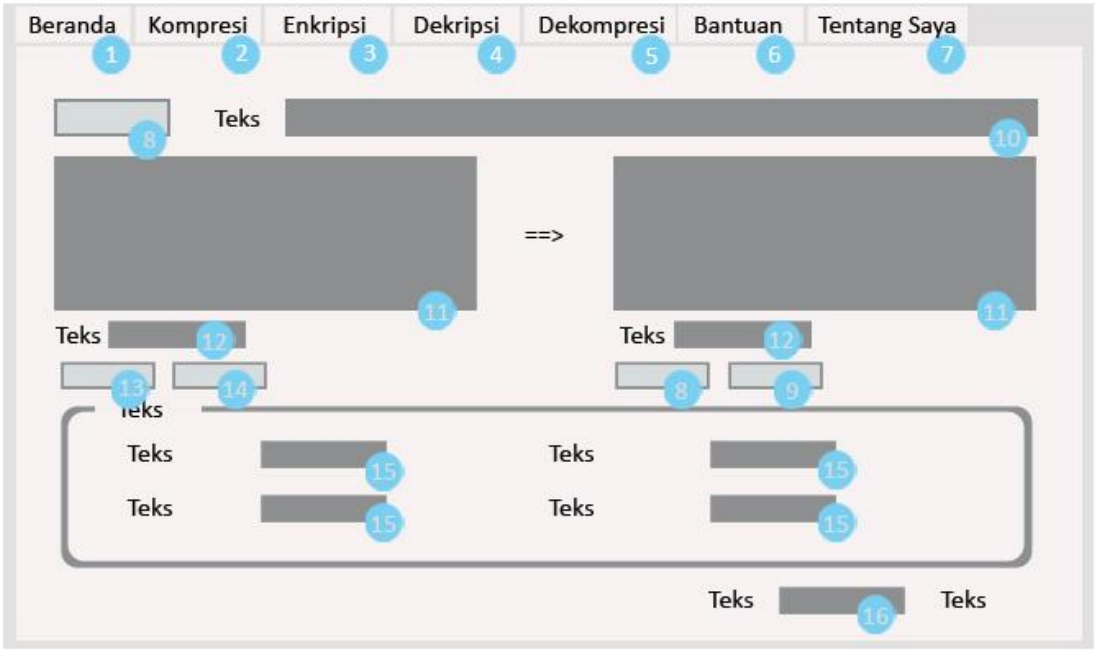
   Berisi keterangan mengenai sistem.

9. *Picture Box*

   Berisi gambar logo berformat .png sebagai keterangan dari sistem.

b. *Form* Kompresi

Pada *Form* Kompresi terdapat *button* yang berguna dalam mengeksekusi dari *file* yang akan dikompresi. *Form* Kompresi pada aplikasi dapat dilihat pada Gambar 3.17.



Gambar 3.17 *Form* Kompresi pada Sistem

Keterangan :

1. *Tab Beranda*

   Berguna untuk menampilkan menu *Beranda* pada sistem yang ada.

2. *Tab* Kompresi

   Berguna untuk menampilkan menu Kompresi pada sistem.

3. *Tab* Enkripsi

   Berguna untuk menampilkan menu Enkripsi pada sistem.

4. *Tab* Dekripsi

   Berguna untuk menampilkan menu Dekripsi pada sistem.

5. *Tab* Dekompresi

   Berguna untuk menampilkan menu Dekompresi pada sistem.

6. *Tab* Bantuan

   Berguna untuk menampilkan menu Bantuan yang ada pada sistem

7. *Tab Tentang Saya*

   Berguna untuk menampilkan menu *Tentang Saya* pada sistem.

8. *Button Open dan Button Simpan*

   Berguna untuk membuka *file* teks (*.txt atau *.doc) dan menyimpan file hasil kompresi.

9. *Button Keluar*

   Berguna untuk keluar aplikasi.

10. *Text Box*

    Berguna untuk menampilkan direktori file yang sedang dibuka.

11. *Rich Text Box*

    Berguna untuk menampilkan isi *file* teks sebelum dan sesudah dikompresi.

12. *Text Box*

    Berguna untuk menampilkan jumlah karakter isi *file* teks sebelum dan sesudah dikompresi.

13. *Button Kompresi*

    Berguna untuk melakukan proses kompresi *file* teks.

14. *Button Clear*

    Berguna untuk menghapus data yang telah diambil dan menjadikannya *default*.

15. *Text Box*

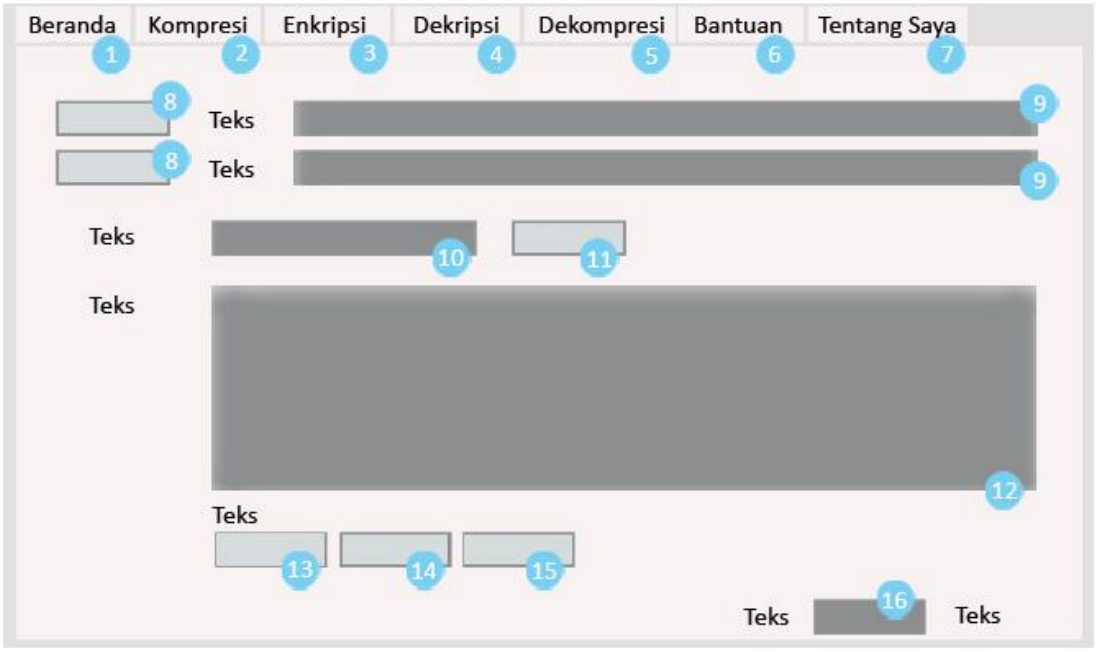
    Berguna untuk menampilkan informasi hasil kompresi (Rc, Cr, Rd, waktu).

16. *Text Box*

    Berguna untuk menampilkan informasi waktu proses kompresi pada sistem.

c. *Form* Enkripsi

Pada *Form* Enkripsi terdapat *button* yang berguna dalam mengeksekusi dari *file* yang telah dienkripsi. *Form* Enkripsi pada aplikasi dapat dilihat pada Gambar 3.18.



Gambar 3.18 *Form* Enkripsi pada Sistem

Keterangan :

1. *Tab Beranda*

   Berguna untuk menampilkan menu *Beranda* pada sistem yang ada.

2. *Tab* Kompresi

   Berguna untuk menampilkan menu Kompresi pada sistem.

3. *Tab* Enkripsi

   Berguna untuk menampilkan menu Enkripsi pada sistem.

4. *Tab* Dekripsi

   Berguna untuk menampilkan menu Dekripsi pada sistem.

5. *Tab* Dekompresi

   Berguna untuk menampilkan menu Dekompresi pada sistem.

6. *Tab* Bantuan

   Berguna untuk menampilkan menu Bantuan yang ada pada sistem

7. *Tab Tentang Saya*

   Berguna untuk menampilkan menu *Tentang Saya* pada sistem.

8. *Button Open dan Save*

   Berguna untuk membuka file yang telah dikompresi (*.tb) dan menyimpan *ciphertext* hasil enkripsi.

9. *Text Box*

   Berguna untuk menampilkan direktori file yang dibuka dan disimpan.

10. *Text Box*

    Berguna untuk memasukkan kata kunci atau *password*.

11. *Button Simpan Kunci*

    Berguna untuk menyimpan kunci yang telah dibuat.

12. *Rich Text Box*

    Berguna untuk menampilkan direktori *file* yang sedang dibuka.

13. *Button Enkripsi*

    Berguna untuk mengenkripsi *file* kompresi menjadi *ciphertext*.

14. *Button Clear*

    Berguna untuk menghapus data yang telah diambil dan menjadikannya *default.*

15. *Button Keluar*

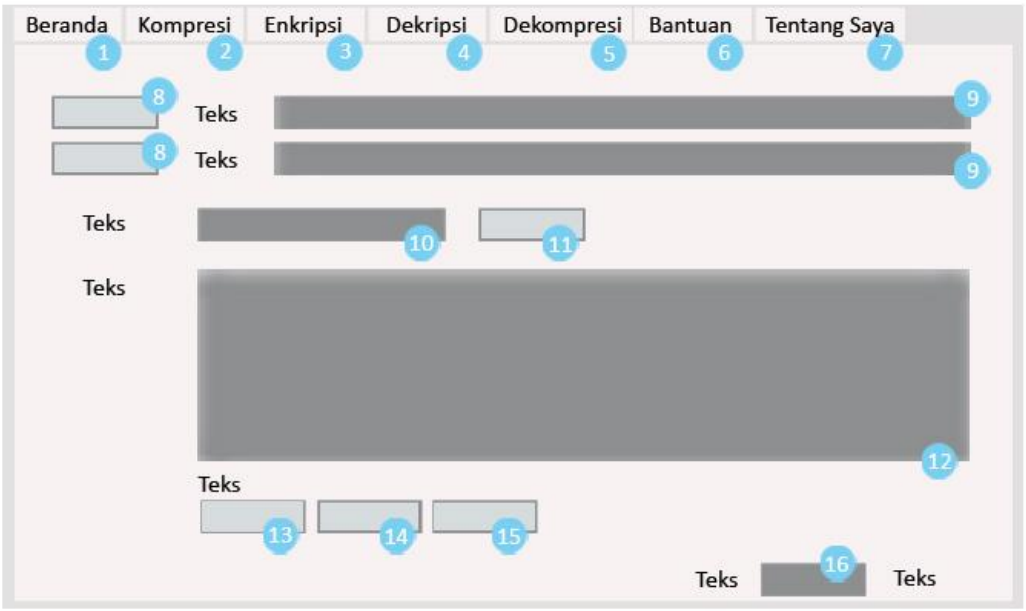
    Berguna untuk keluar dari aplikasi.

16. *Text Box*

    Berguna untuk menampilkan informasi waktu proses enkripsi pada sistem.

d. *Form* Dekripsi

Pada *Form* Dekripsi terdapat *button* yang berguna dalam mengeksekusi dari *file* yang telah didekompresi. *Form* Dekripsi pada aplikasi dapat dilihat pada Gambar 3.19.



Gambar 3.19 *Form* Dekripsi pada Sistem

Keterangan :

1. *Tab Beranda*

   Berguna untuk menampilkan menu *Beranda* pada sistem yang ada. .

2. *Tab* Kompresi

   Berguna untuk menampilkan menu Kompresi pada sistem.

3. *Tab* Enkripsi

   Berguna untuk menampilkan menu Enkripsi pada sistem.

4. *Tab* Dekripsi

   Berguna untuk menampilkan menu Dekripsi pada sistem.

5. *Tab* Dekompresi

   Berguna untuk menampilkan menu Dekompresi pada sistem.

6. *Tab* Bantuan

   Berguna untuk menampilkan menu Bantuan yang ada pada sistem

7. *Tab Tentang Saya*

   Berguna untuk menampilkan menu *Tentang Saya* pada sistem.

8. *Button Open dan Save*

   Berguna untuk membuka file yang telah dienkripsi (*.skp) dan menyimpan file *kompresi* hasil enkripsi.

9. *Text Box*

   Berguna untuk menampilkan direktori file yang dibuka dan disimpan.

10. *Text Box*

    Berguna untuk menampilkan kata kunci atau *password*.

11. *Button Ambil Kunci*

    Berguna untuk mengambil kunci atau *password* yang telah dibuat.

12. *Rich Text Box*

    Berguna untuk menampilkan direktori *file* yang sedang dibuka.

13. *Button Dekripsi*

    Berguna untuk mendekripsi *file* enkripsi menjadi *file kompresi*.

14. *Button Clear*

    Berguna untuk menghapus data yang telah diambil dan menjadikannya *default.*

15. *Button Keluar*

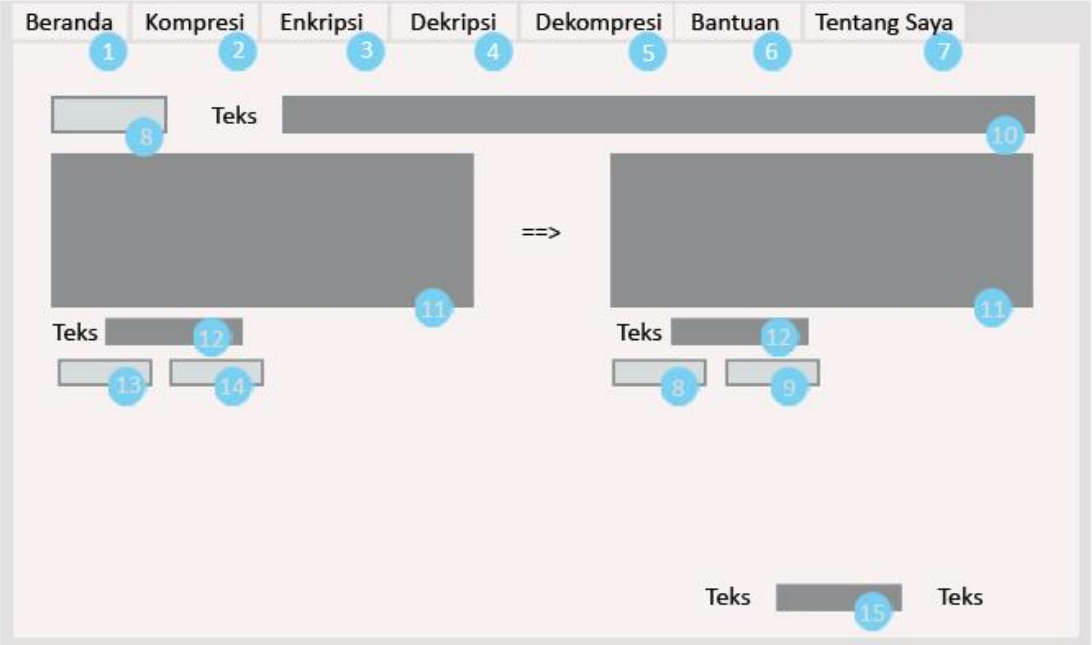
    Berguna untuk keluar dari aplikasi.

16. *Text Box*

Berguna untuk menampilkan informasi waktu proses dekripsi pada sistem

e. *Form* Dekompresi

Pada *Form* Dekompresi terdapat *button* yang berguna dalam mengeksekusi dari *file* yang telah dikompresi. *Form* Dekompresi pada aplikasi dapat dilihat pada Gambar 3.20.



Gambar 3.20 *Form* Dekompresi pada Sistem

Keterangan :

1. Tab Beranda

Berguna untuk menampilkan menu *Beranda* pada sistem yang ada.

2. Tab Kompresi

Berguna untuk menampilkan menu Kompresi pada sistem.

3. Tab Enkripsi

Berguna untuk menampilkan menu Enkripsi pada sistem.

4. Tab Dekripsi

Berguna untuk menampilkan menu Dekripsi pada sistem.

5. Tab Dekompresi

Berguna untuk menampilkan menu Dekompresi pada sistem.

6. Tab Bantuan

Berguna untuk menampilkan menu Bantuan yang ada pada sistem

7. Tab Tentang Saya

Berguna untuk menampilkan menu *Tentang Saya* pada sistem.

8. Button Open dan Button Simpan

Berguna untuk membuka *file* teks (*.tb) dan menyimpan file hasil dekompresi.

9. *Button Keluar*

Berguna untuk keluar aplikasi.

10. Text Box

Berguna untuk menampilkan direktori file yang sedang dibuka.

11. Rich Text Box

Berguna untuk menampilkan isi *file* teks sebelum dan sesudah dikompresi.

12. *Text Box*

Berguna untuk menampilkan jumlah karakter isi *file* teks sebelum dan sesudah didekompresi.

13. Button Dekompresi

Berguna untuk melakukan proses dekompresi *file*.

14. Button Clear

Berguna untuk menghapus data yang telah diambil dan menjadikannya *default*.

15. *Text Box*

Berguna untuk menampilkan informasi waktu proses dekompresi pada sistem.

f. *Form* Bantuan

Pada *Form* Bantuan berisi tentang keterangan yang dapat membantu user dalam mengoperasikan sistem yang ada. *Form* Bantuan pada aplikasi dapat dilihat pada Gambar 3.21.

Gambar 3.21 *Form* Bantuan pada Sistem

Keterangan :

1. Tab Bantuan

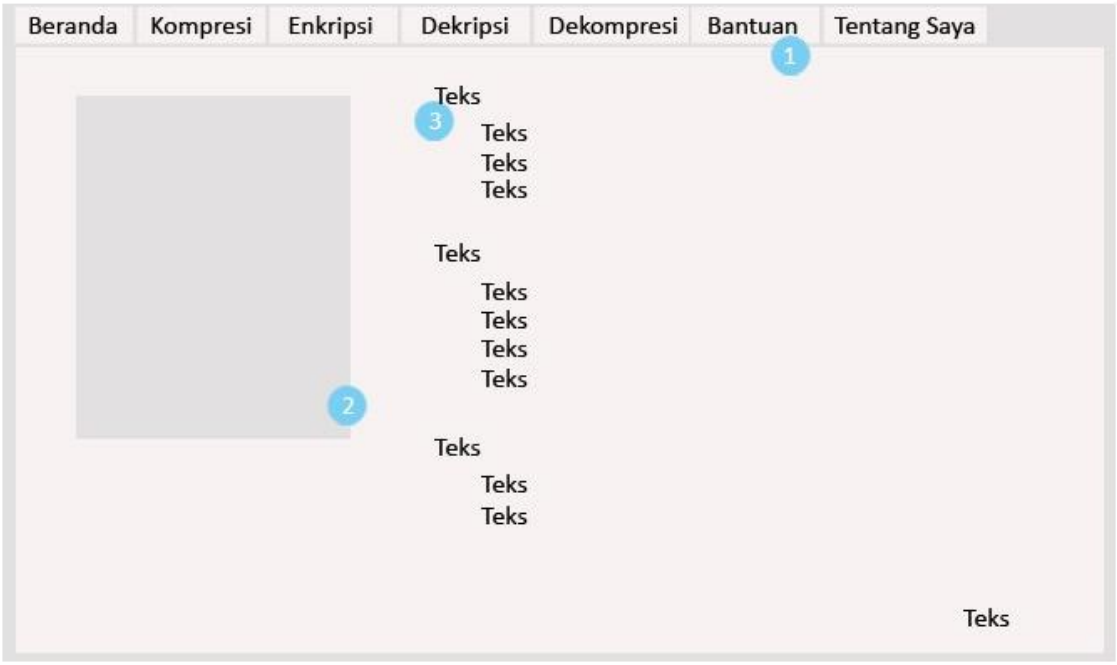
Berguna untuk menampilkan menu Bantuan yang ada pada sistem

2. Text Box

Berisi penjelasan menggunakan menu Kompresi, Enkripsi, Dekripsi, dan Dekompresi.

g. Form Tentang Saya

Pada *Form Tentang Saya* berisi tentang informasi mengenai si pembuat sistem. *Form Tentang Saya* pada aplikasi dapat dilihat pada Gambar 3.22.



Gambar 3.22 *Form* Tentang Saya pada Sistem

Keterangan :

1. Tab Tentang Saya

Berguna untuk menampilkan menu *Tentang Saya* pada sistem.

2. Picture Box

Berisi gambar berformat .png.

3. Text Box

Berisi tentang informasi mengenai si pembuat sistem.

BAB 4
IMPLEMENTASI DAN PENGUJIAN SISTEM

4.1. Implementasi Algoritma

4.1.1. Proses kompresi

Berikut ini adalah contoh proses kompresi file teks dengan menggunakan algoritma Taboo. Terdapat file teks yang berisikan *string* "ilmu komputer usu". Penjelasan *string* yang belum dikompresi dapat dilihat pada tabel 4.1.

Tabel 4.1 Penjelasan string yang belum dikompresi

Karakter	Frequensi	ASCII Code (Binary)	Bit	Freq*Bit
i	1	01101001	8	8
l	1	01101100	8	8
m	2	01101101	8	16
u	4	01110101	8	32
spasi	2	00100000	8	16
k	1	01101011	8	8
o	1	01101111	8	8
p	1	01110000	8	8
t	1	01110100	8	8
e	1	01100101	8	8
r	1	01110010	8	8
s	1	01110011	8	8
Total Bit				**136**

Berdasarkan kode ASCII, satu karakter bernilai delapan bit bilangan biner. Sehingga 17 karakter pada *String* mempunyai nilai biner sebanyak 144 bit. Algoritma Taboo bekerja dengan memulai *sorting* jumlah kemunculan tiap karakter yang akan di kompresi seperti dapat dilihat pada tabel 4.1 di atas. String bit sebelum dikompresi dari "ilmu komputer usu" adalah 01101001 01101100 01101101 01110101 00100000 01101011 01101111 01101101 01110000 01110101 01110100 01100101 01110010

00100000 01110101 01110011 01110101. Pada kode ASCII, sebuah karakter akan bernilai delapan bit dalam bilangan biner. Sehingga 12 karakter pada string akan memiliki nilai biner sebanyak 136 bit. Pola algoritma Taboo dapat dilihat pada Table 4.2.

Tabel 4.2. Tabel Pola Algoritma Taboo

M	Code	M	Code	m	Code	M	Code
0	01 00	4	01 10 00	8	10 11 00	12	01 01 01 00
1	10 00	5	01 11 00	9	11 01 00	13	01 01 10 00
2	11 00	6	10 01 00	10	11 10 00	14	01 01 11 00
3	01 01 00	7	10 10 00	11	11 11 00	..	

Jumlah karakter terbanyak dimasukkan pada kolom pertama pada table di atas.

Tabel 4.3. Tabel Karakter Taboo

n	char	Biner	Bit	Frek	Frek sebelum kompresi	Bit x Frek
1	u	0100	4	4	32	16
2	m	1000	4	2	16	8
3	spasi	1100	4	2	16	8
4	i	010100	6	1	8	6
5	l	011000	6	1	8	6
6	k	011100	6	1	8	6
7	o	100100	6	1	8	6
8	p	101000	6	1	8	6
9	t	101100	6	1	8	6
10	e	110100	6	1	8	6
11	r	111000	6	1	8	6
12	s	111100	6	1	8	6
Total Bit					**136**	**86**

Pada tabel 4.3 dapat dilihat bahwa jumlah karakter hasil kompresi yaitu 86 bit, dan dapat dibuat dalam string bit “ilmu komputer usu” setelah dikompresi yaitu :

“ 010100 011000 1000 0100 1100 011100 100100 1000 101000 0100 101100 110100 111000 1100 0100 111100 0100 “

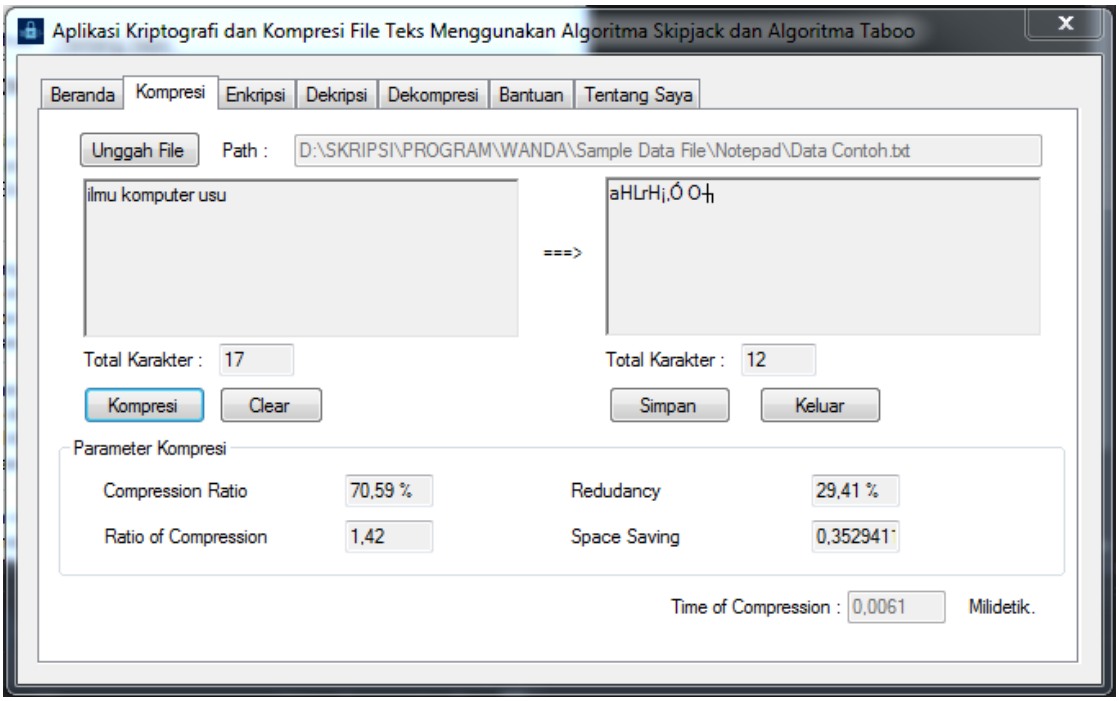
Hasil string bit dijadikan kedalam ASCII dengan masing masing string per karakter yaitu 8 bit, maka string bitnya yaitu :

“ 01010001 10001000 01001100 01110010 01001000 10100001 00101100 11010011 10001100 01001111 000100 “

Karena jumlah *string* bit 86 tidak habis dibagi delapan dan sisanya adalah 2. Maka dapat dibuat *padding* “00” dan *flagnya* adalah “00000011”. Maka diperoleh bit sebagai berikut :

“ 01010001 10001000 01001100 01110010 01001000 10100001 00101100 11010011 10001100 01001111 00000100 00000011 “

Total bit seluruhnya setelah penambahan *padding* dan *flag* adalah 96 bit. Sehingga jumlah karakter setelah di kompresi yaitu 96 dibagi dengan 8 yaitu “12” karakter. Contoh tampilan proses kompresi teks menggunakan algoritma Taboo Codes dapat dilihat pada gambar 4.1.



Gambar 4.1. Hasil Kompresi Algoritma Taboo

Pada gambar 4.1. dapat dilihat bahwa karakter yang diinputkan ke dalam sistem sebanyak 17 karakter. Setelah dikompresi maka didapat banyaknya karakter berkurang sebesar 5 karakter. Setelah didapatkan hasil kompresi maka data disimpan dalam bentuk *.tb.

Hasil Kompresi yaitu “ aHLrHi,ÓŒO__ “

Dari hasil kompresi menggunakan *Taboo* diatas dapat dihitung kinerja kompresinya :

a. *Ratio of Compression* (Rc)

$$\boldsymbol{Rc} = \frac{Ukuran\ data\ sebelum\ dikompres}{Ukuran\ data\ setelah\ dikompres}$$

$$\boldsymbol{Rc} = \frac{136}{86}$$

$$\boldsymbol{Rc} = \mathbf{1.42}$$

b. *Compression Ratio* (Cr)

$$\boldsymbol{Cr} = \frac{\textbf{Ukuran data setelah dikompresi}}{\textbf{Ukuran data sebelum dikompresi}} \times \mathbf{100\%}$$

$$\boldsymbol{Cr} = \frac{86}{136} \times \mathbf{100\%}$$

$$\boldsymbol{Cr} = \mathbf{70{,}59\%}$$

c. *Redundancy* (Rd)

$$\boldsymbol{Rd} = \mathbf{100\% - Cr}$$

$$\boldsymbol{Rd} = \mathbf{100\% - 70{,}59\%}$$

$$\boldsymbol{Rd} = \mathbf{29{,}41\%}$$

Proses kompresi pada sistem memerlukan waktu 0.0061 detik.

4.1.2. Proses Enkripsi

Setelah didapat *hasil kompresi* maka dapat dilakukan enkripsi pesan pesan “ilmu komputer usu” yang telah dikompresi menjadi “ aHLrH¡,ÓODLESTX “.

Kemudian karakter ASCII tersebut dapat dibuat kedalam bentuk heksadesimal menjadi `51 88 4C 72 48 A1 2C D3 8C 4F 04 03.`

Karena *plaintext* melebihi batas 64 bit, maka *plaintext* dilakukan *padding* sebanyak 32 bit atau “ 0000 “, lalu *plaintext* dibagi menjadi 2 bagian saat melakukan enkripsi.

Perhitungan manual dari proses enkripsi adalah sebagai berikut :

Plainteks : aHLrH¡,ÓODLESTX atau 51884C7248A12CD3

Kunci : skripsiaku

Ubah kunci ke dalam bentuk heksadesimal : 736B7269707369616B75

Bagi *key* menjadi 10 bagian (*cryptovariable)* sebagai berikut :

cv[0] : 73

cv[1] : 6B

cv[2] : 72

cv[3] : 69

cv[4] : 70

cv[5] : 73

cv[6] : 69

cv[7] : 61

cv[8] : 6B

cv[9] : 75

Ubah *plaintext* kedalam bentuk heksadesimal : 51884C7248A12CD3.

Bagi *plaintext* menjadi 4-bagian (W^1, W^2, W^3, dan W^4) sebagai berikut:

W^1_0 : 5188 ; W^2_0 : 4C72 ; W^3_0 : 48A1 ; W^4_0 : 2CD3 .

Putaran-1 (Rule A, k=0, counter=1)

W^1_0 = 5188

g1 = 51

g2 = 88

g3 = F(g2 ^ cv[(4*k) mod 10]) ^ g1

= F(88 ^ cv[0]) ^ 51

= F(FB) ^ 51

= A5 ^ 51 = F4

g4 = F(g3 ^ cv[((4*k)+1) mod 10]) ^ g2

= F(F4^ cv[1]) ^ 88

= F(9F) ^ 88

= A8 ^ 88 = 20

g5 = F(g4 ^ cv[((4*k)+2) mod 10]) ^ g3

= F(20 ^ cv[2]) ^ F4

= F(52) ^ F4

= F1 ^ F4 = 05

g6 = F(g5 ^ cv[((4*k)+3) mod 10]) ^ g4

= F(05 ^ cv[3]) ^ 20

= F(6C) ^ 20

= 22 ^ 20 = 02

g = g5 + g6 = 0502

W^1_1 = g ^ W^4_0 ^ counter = 0502 ^ 2CD3 ^ 1 = 29D0

W^2_1 = g = 0502

W^3_1 = W^2_0 = 4C72

W^4_1 = W^3_0 = 48A1

Putaran-2 (Rule A, k=1, counter=2)

W^1_1 = 29D0

g1 = 29

g2 = D0

g3 = F(g2 ^ cv[(4*k) mod 10]) ^ g1

= F(D0 ^ cv[4]) ^ 29

= F(A0) ^ 29

= 15 ^ 29 = 3C

g4 = F(g3 ^ cv[((4*k)+1) mod 10]) ^ g2

= F(3C^ cv[1]) ^ D0

= F(57) ^ D0

= FE ^ D0 = 2E

g5 = F(g4 ^ cv[((4*k)+2) mod 10]) ^ g3

= F(2E ^ cv[2]) ^ 3C

= F(5C) ^ 3C

= 51 ^ 3C = 6D

g6 = F(g5 ^ cv[((4*k)+3) mod 10]) ^ g4

= F(6D ^ cv[3]) ^ 2E

= F(04) ^ 2E

= 39 ^ 2E = 17

g = g5 + g6 = 6D17

W^1_2 = g ^ W^4_1 ^ counter = 6D17 ^ 48A1 ^ 2 = 25B4

W^2_2 = g = 6D17

W^3_2 = W^2_1 = 0502

W^4_2 = W^3_1 = 4C72

Putaran-3 (Rule A, k=2, counter=3)

W^1_2 = 25B4

g1 = 25

g2 = B4

g3 = F(g2 ^ cv[(4*k) mod 10]) ^ g1
= F(B4 ^ cv[8]) ^ 25
= F(DF) ^ 25
= 59 ^ 25 = 7C

g4 = F(g3 ^ cv[((4*k)+1) mod 10]) ^ g2
= F(7C^ cv[9]) ^ B4
= F(09) ^ B4
= 89 ^ B4 = 3D

g5 = F(g4 ^ cv[((4*k)+2) mod 10]) ^ g3
= F(3D ^ cv[0]) ^ 7C
= F(4E) ^ 7C
= 92 ^ 7C = EE

g6 = F(g5 ^ cv[((4*k)+3) mod 10]) ^ g4
= F(EE ^ cv[1]) ^ 3D
= F(85) ^ 3D
= 5A ^ 3D = 67

g = g5 + g6 = EE67

W^1_3 = g ^ W^4_2 ^ counter = EE67 ^ 4C72 ^ 3 = A216

W^2_3 = g = EE67

W^3_3 = W^2_2 = 6D17

W^4_3 = W^3_2 = 0502

Putaran-4 (Rule A, k=3, counter=4)

W^1_3 = A216

g1 = A2

g2 = 16

g3 = F(g2 ^ cv[(4*k) mod 10]) ^ g1
= F(16 ^ cv[2]) ^ A2
= F(64) ^ A2
= 81 ^ A2 = 23

g4 = F(g3 ^ cv[((4*k)+1) mod 10]) ^ g2
= F(23^ cv[3]) ^ 16
= F(4A) ^ 16
= 9F ^ 16 = 89

g5 = F(g4 ^ cv[((4*k)+2) mod 10]) ^ g3

= F(89 ^ cv[4]) ^ 23

= F(F9) ^ 23

= A4 ^ 23 = 87

g6 = F(g5 ^ cv[((4*k)+3) mod 10]) ^ g4

= F(87 ^ cv[5]) ^ 89

= F(F4) ^ 89

= B8 ^ 89 = 31

g = g5 + g6 = 8731

W^1_4 = g ^ W^4_3 ^ counter = 8731 ^ 0502 ^ 4 = 8237

W^2_4 = g = 8731

W^3_4 = W^2_3 = EE67

W^4_4 = W^3_3 = 6D17

Maka, hasil setelah melalui 32 putaran adalah É ˜\Ô^$DC1m9|tODLESTX

4.1.3. Proses Dekripsi

Untuk mengembalikan *ciphertext* menjadi *plaintext*, maka dilakukan dekripsi dengan *ciphertext* sebagai berikut :

Ciphertext = É ˜\Ô^$DC1m9|tODLESTX

Kunci = skripsiaku

Ubah kunci ke dalam bentuk heksadesimal : 736B7269707369616B75

Bagi *key* menjadi 10 bagian (*cryptovariable)* sebagai berikut :

cv[0] : 73

cv[1] : 6B

cv[2] : 72

cv[3] : 69

cv[4] : 70

cv[5] : 73

cv[6] : 69

cv[7] : 61

cv[8] : 6B

cv[9] : 75

Ubah *plaintext* kedalam bentuk heksadesimal : 5CD45E24116D397C.

Bagi *plaintext* menjadi 4-bagian (W^1, W^2, W^3, dan W^{34}) sebagai berikut:

W^1_{32} : 5CD4 ; W^2_{32} : 5E24 ; W^3_{32} : 116D ; W^4_{32} : 397C .

Putaran-1 (Rule B^{-1}, k=32, counter=32)

W^2_{32} = 5E24

g5 = 5E

g6 = 24

g4 = F(g5 ^ cv[(4*k-1)+3 mod 10]) ^ g6

= F(5E ^ cv[7]) ^ 24

= F(3F) ^ 24

= 13 ^ 24 = 37

g3 = F(g4 ^ cv[(4*k-1)+2 mod 10]) ^ g5

= F(37 ^ cv[6]) ^ 5E

= F(4D) ^ 5E

= 75 ^ 5E = 2B

g2 = F(g3 ^ cv[(4*k-1)+1 mod 10]) ^ g4

= F(2B ^ cv[5]) ^ 37

= F(58) ^ 37

= 3C ^ 37 = 0B

g1 = F(g2 ^ cv[(4*k-1) mod 10]) ^ g3

= F(0B ^ cv[4]) ^ 2B

= F(7B) ^ 2B

= 3B ^ 2B = 10

g’ = g1 + g2 = 100B

W^1_{31} = g’ = 100B

W^2_{31} = g’ ^ W^3_{32} ^ counter = 100B ^ 116D ^ 32 = 0146

W^3_{31} = W^4_{32} = 397C

W^4_{31} = W^1_{32} = 5CD4

Putaran-2 (Rule B^{-1}, k=31, counter=31)

W^2_{31} = 0146

g5 = 01

g6 = 46

g4 = F(g5 ^ cv[(4*k-1)+3 mod 10]) ^ g6

= F(01 ^ cv[3]) ^ 46

= F(68) ^ 46

= CD ^ 46 = 8D

g3 = F(g4 ^ cv[(4*k-1)+2 mod 10]) ^ g5

= F(8D ^ cv[2]) ^ 01

= F(F9) ^ 01

= A4 ^ 01 = A5

g2 = F(g3 ^ cv[(4*k-1)+1 mod 10]) ^ g4

= F(A5 ^ cv[1]) ^ 8D

= F(CE) ^ 8D

= F3 ^ 8D = 7E

g1 = F(g2 ^ cv[(4*k-1) mod 10]) ^ g3

= F(7E ^ cv[0]) ^ A5

= F(0D) ^ A5

= 0C ^ A5 = A9

g' = g1 + g2 = A97E

W^1_{30} = g' = A97E

W^2_{30} = g' ^ W^3_{31} ^ counter = A97E ^ 397C ^ 31 = 901D

W^3_{30} = W^4_{31} = 5CD4

W^4_{30} = W^1_{31} = 100B

Putaran-3 (Rule B^{-1}, k=30, counter=30)

W^2_{30} = 901D

g5 = 90

g6 = 1D

g4 = F(g5 ^ cv[(4*k-1)+3 mod 10]) ^ g6

= F(90 ^ cv[9]) ^ 1D

= F(E5) ^ 1D

= D8 ^ 1D = C5

g3 = F(g4 ^ cv[(4*k-1)+2 mod 10]) ^ g5

= F(C5^ cv[8]) ^ 90

= F(AE) ^ 90

= 9D ^ 90 = 0D

g2 = F(g3 ^ cv[(4*k-1)+1 mod 10]) ^ g4

= F(0D ^ cv[7]) ^ C5

= F(6C) ^ C5

= 22 ^ C5 = E7

g1 = F(g2 ^ cv[(4*k-1) mod 10]) ^ g3

= F(E7 ^ cv[6]) ^ 0D

= F(8E) ^ 0D

= 32 ^ 0D = 3F

g' = g1 + g2 = 3FE7

W^1_{29} = g' = 3FE7

W^2_{29} = g' ^ W^3_{30} ^ counter = 3FE7 ^ 5CD4 ^ 30 = 632D

W^3_{29} = W^4_{30} = 100B

W^4_{29} = W^1_{30} = A97E

Putaran-4 (Rule B^{-1}, k=29, counter=29)

W^2_{29} = 632

g5 = 63

g6 = 2D

g4 = F(g5 ^ cv[(4*k-1)+3 mod 10]) ^ g6

= F(63 ^ cv[5]) ^ 2D

= F(10) ^ 2D

= D7 ^ 2D = FA

g3 = F(g4 ^ cv[(4*k-1)+2 mod 10]) ^ g5

= F(FA ^ cv[4]) ^ 63

= F(8A) ^ 63

= 50 ^ 63 = 33

g2 = F(g3 ^ cv[(4*k-1)+1 mod 10]) ^ g4

= F(33 ^ cv[3]) ^ FA

= F(5A) ^ FA

= 0D ^ FA = F7

g1 = F(g2 ^ cv[(4*k-1) mod 10]) ^ g3

= F(F7 ^ cv[2]) ^ 33

= F(85) ^ 33

= 5A ^ 33 = 69

g' = g1 + g2 = 69F7

W^1_{28} = g' = 69F7

W^2_{28} = g' ^ W^3_{29} ^ counter = 69F7 ^ 100B ^ 29 = 79E1

W^3_{28} = W^4_{29} = A97E

W^4_{28} = W^1_{29} = 3FE7

Putaran-5 (Rule B^{-1}, k=28, counter=28)

W^2_{28} = 79E1

g5 = 79

g6 = E1

g4 = F(g5 ^ cv[(4*k-1)+3 mod 10]) ^ g6

= F(79 ^ cv[1]) ^ E1

= F(12) ^ E1

= DF ^ E1 = 3E

g3 = F(g4 ^ cv[(4*k-1)+2 mod 10]) ^ g5

= F(3E ^ cv[0]) ^ 79

= F(4D) ^ 79

= 75 ^ 79 = 0C

g2 = F(g3 ^ cv[(4*k-1)+1 mod 10]) ^ g4

= F(0C ^ cv[9]) ^ 3E

= F(79) ^ 3E

= AA ^ 3E = 94

g1 = F(g2 ^ cv[(4*k-1) mod 10]) ^ g3

= F(94 ^ cv[8]) ^ 0C

= F(FF) ^ 0C

= 46 ^ 0C = 4A

g' = g1 + g2 = 4A94

W^1_{27} = g' = 4A94

W^2_{27} = g' ^ W^3_{28} ^ counter = 4A94 ^ A97E ^ 28 = E3F6

W^3_{27} = W^4_{28} = 3FE7

W^4_{27} = W^1_{28} = 69F7

Maka, hasil setelah melalui 32 putaran adalah aHLrH¡,ÓO

3.3.4 Proses Dekompresi

Setelah file berhasil di dekripsi maka akan menghasilkan file baru yang berisi informasi serta *string* bit hasil proses kompresi. Hasil dekripsi “ilmu kompuer usu” dengan menggunakan algoritma Taboo yaitu :

“ 01010001 10001000 01001100 01110010 01001000 10100001 00101100 11010011 10001100 01001111 00000100 00000011 “

Proses dekompresi didapatkan dari tabel kompresi Taboo 4.1 diatas yang menghasilkan bit sebagai berikut :

“ 01010001 10001000 01001100 01110010 01001000 10100001 00101100 11010011 10001100 01001111 00000100 00000011 ”

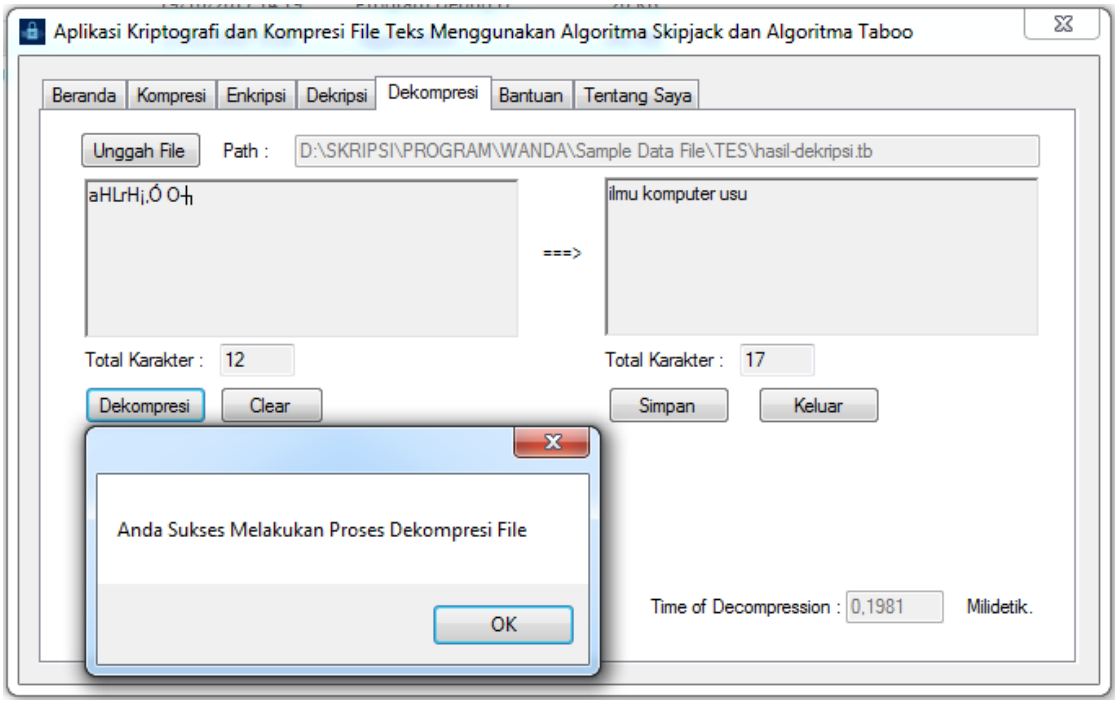
Dari hasil diatas maka hasil dekompresi diatas dihilangkan padding dan flag yang ditambahkan pada saat proses kompresi, padding “00” dan flag “00000011”. Maka hasil dari penghilangan padding dan flag dapat dilihat sebagai berikut:

“ 010100 011000 1000 0100 1100 011100 100100 1000 101000 0100 101100 110100 111000 1100 0100 111100 0100 “

Berdasarkan tabel kompresi Taboo 4.1 maka dari string bit didapatkan hasil “ilmu kompuer usu”. Lalu hasil dekompresi dijadikan kedalam kode ASCII yaitu:

“ 01101001 01101100 01101101 01110101 00100000 01101011 01101111 01101101 01110000 01110101 01110100 01100101 01110010 00100000 01110101 01110011 01110101 “

Pada string bit diatas maka didapatkan hasil 144 bit dengan total 17 karakter. Contoh tampilan proses dekompresi teks menggunakan algoritma Taboo Codes dapat dilihat pada gambar 4.2.



Gambar 4.2. Hasil Dekompresi Algoritma *Taboo Codes*

Proses dekompresi pada sistem memerlukan waktu 0.1981 detik.

4.2. Implementasi Sistem

Sistem ini dibangun dengan menggunakan bahasa pemrograman Visual Studio. Pada proses implementasi yang dirancang pada sistem ini dibagi menjadi 7 *form* yaitu *form* beranda, *form* enkripsi, *form* kompresi, *form* dekompresi, *form* dekripsi, *form* bantuan dan *form* tentang saya, yang dapat dilihat sebagai berikut :

4.2.1. Form Beranda

Form Beranda merupakan *form* yang pertama kali muncul pada saat aplikasi dijalankan.

4.2.2. Form Kompresi

Form Kompresi merupakan *form* yang digunakan untuk melakukan proses kompresi. Pada *tab* ini disediakan *interface* untuk *input file* hasil enkripsi, *textbox* untuk menampilkan isi *file* teks dan hasil kompresi, menentukan direktori *output file* hasil kompresi, *group box* yang berisi beberapa *text box* informasi hasil kompresi.

4.2.3. Form Enkripsi

Form Enkripsi merupakan *form* yang digunakan untuk melakukan proses enkripsi. Pada *tab* ini disediakan *interface* untuk menginput *file* hasil kompresi, textbox untuk menampilkan isi *file,* menentukan direktori ciphertext yang dihasilkan, *textbox* untuk menampilkan waktu proses enkripsi.

4.2.4. Form Dekripsi

Form Dekripsi merupakan *form* yang digunakan untuk melakukan proses dekripsi. Pada *form* ini disediakan *interface* untuk menginput *file* hasil enkripsi, textbox untuk menampilkan isi *ciphertext,* menentukan direktori file hasil dekripsi, *textbox* untuk menampilkan waktu proses dekripsi.

4.2.5. Form Dekompresi

Form Dekompresi merupakan *form* yang digunakan untuk melakukan proses dekompresi. Pada *form* ini disediakan *interface* untuk *input file* hasil kompresi, *textbox* untuk menampilkan isi file kompresi dan hasil dekompresi, menentukan direktori *file* hasil dekompresi, *text box* untuk menampilkan waktu proses dekompresi.

4.2.6. Form Bantuan

Form Bantuan merupakan *Form* yang digunakan untuk menampilkan panduan singkat tentang cara mengoperasikan sistem yang dibuat.

4.2.7. Form Tentang Saya

Form Tentang Saya merupakan *form* yang menampilkan informasi tentang penulis.

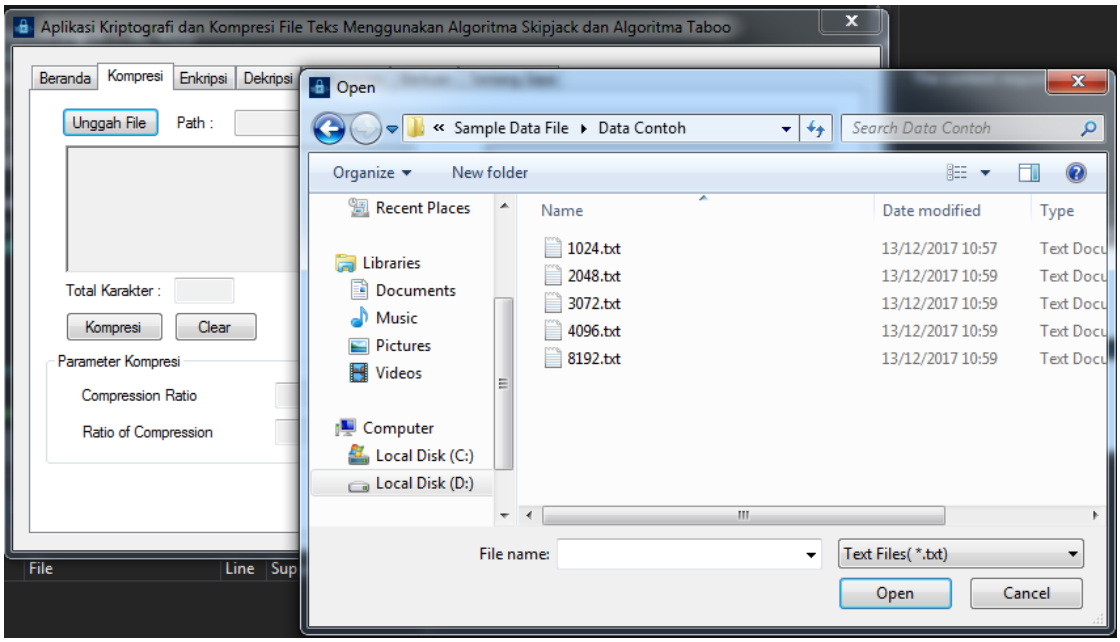
4.3. Pengujian Sistem

Pengujian sistem merupakan tahap mengidentifikasi hasil dari implementasi sistem apakah sistem telah berjalan sesuai dengan fungsi-fungsi yang sebelumnya ditentukan pada tahap analisis dan perancangan sistem. Pengujian sistem ini dilakukan pada *File* teks yang berekstensi *.doc dan *.txt. Pengujian sistem yang dilakukan pada penelitian ini dibagi dalam 4 proses utama yaitu pengujian proses kompresi, pengujian enkripsi, pengujian dekripsi, dan pengujian proses dekompresi.

4.3.1. Pengujian proses kompresi

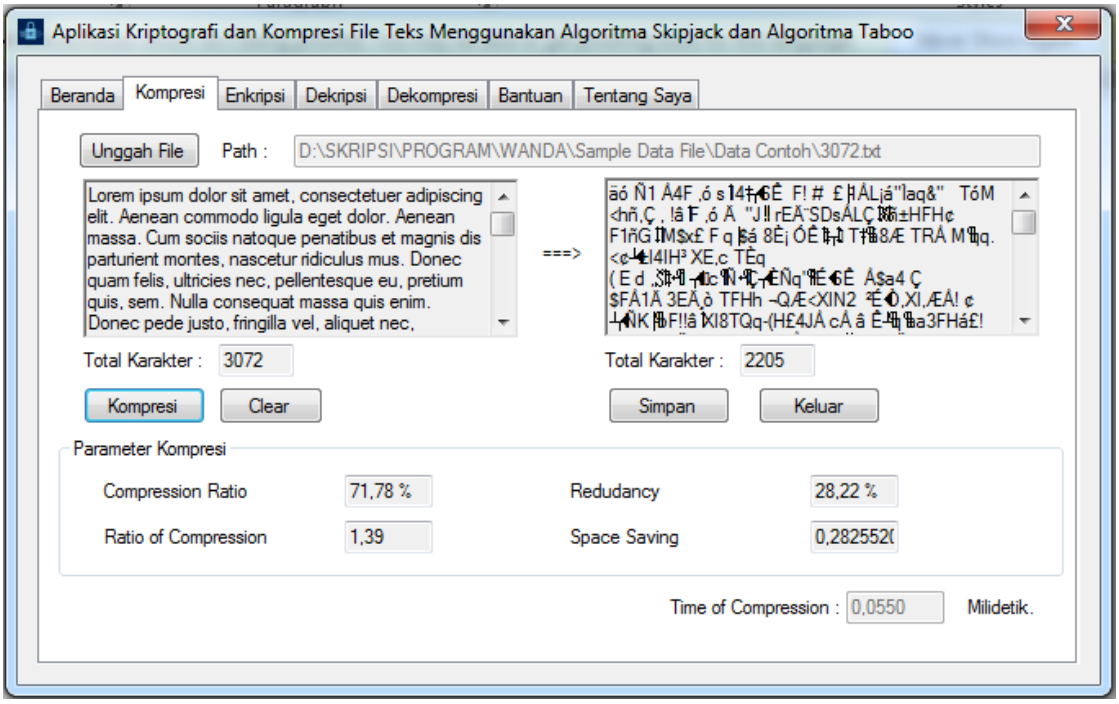
Untuk melakukan proses kompresi, tahap awal yang dilakukan adalah memilih *tab* Kompresi. Setelah tampilan *form* Kompresi muncul maka lakukan langkah-langkah berikut ini untuk melakukan proses kompresi.

1. Menekan tombol *Unggah File* untuk membuka *Open File Dialog* kemudian pilih *File* teks (*.txt atau *.doc) sebagai *Input File*, dapat dilihat pada gambar 4.3.



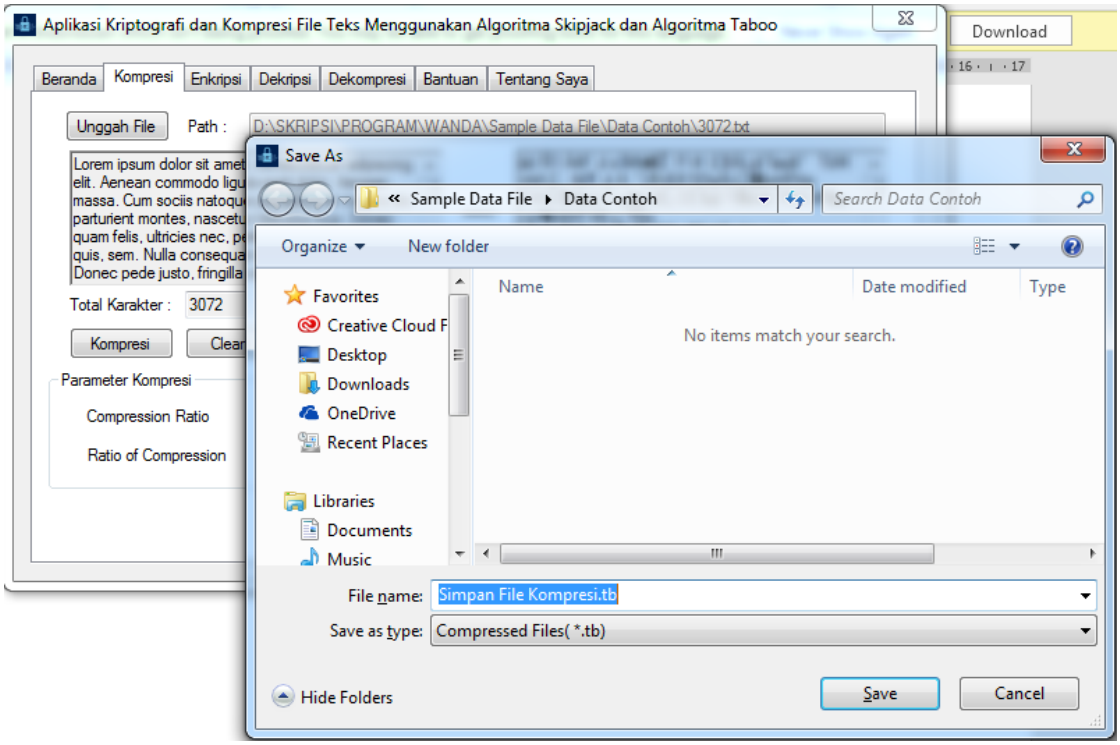
Gambar 4.3 Open File Teks

2. Menekan tombol Kompresi untuk mengeksekusi file yang akan dikompresi. Hasil kompresi pada sistem dapat dilihat pada gambar 4.4.



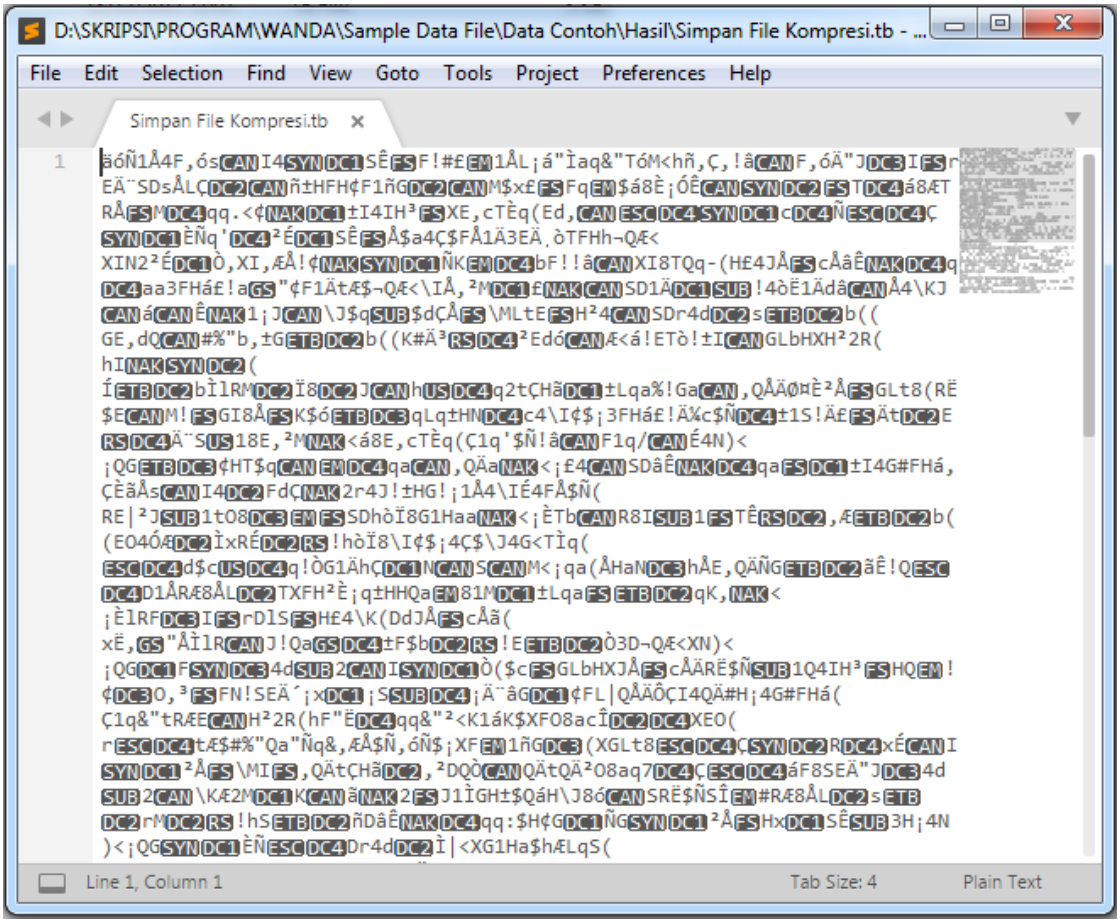
Gambar 4.4 Hasil Kompresi

3. Menekan tombol *Simpan* untuk membuka *Save File Dialog* dan tentukan direktori untuk menyimpan *file* hasil kompresi (*.tb), dapat dilihat pada gambar 4.5.



Gambar 4.5 Save File Hasil Kompresi

Proses kompresi menghasilkan sebuah *File Output* yaitu *file* dengan ekstensi *.tb (Taboo) sebagai *file* yang menyimpan informasi karakter hasil kompresi. *File* hasil kompresi ini dapat dilihat menggunakan *text* editor seperti pada Gambar 4.6 berikut.

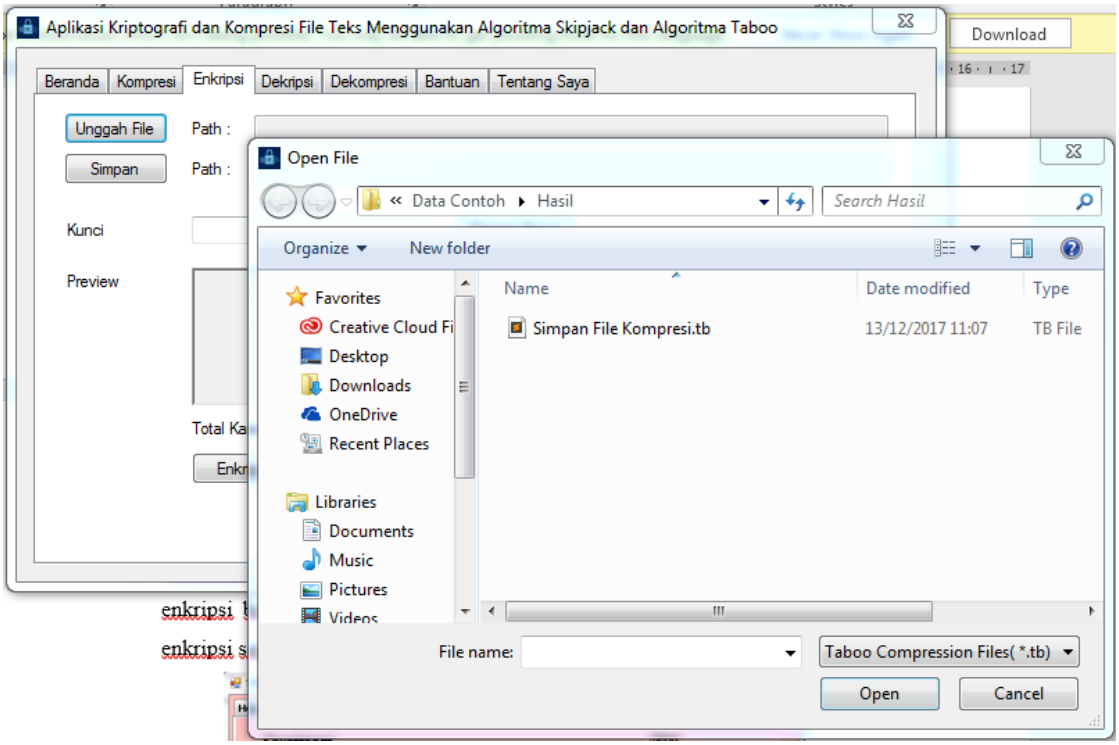


Gambar 4.6 *File* Taboo Hasil Kompresi

4.3.2. Pengujian proses enkripsi

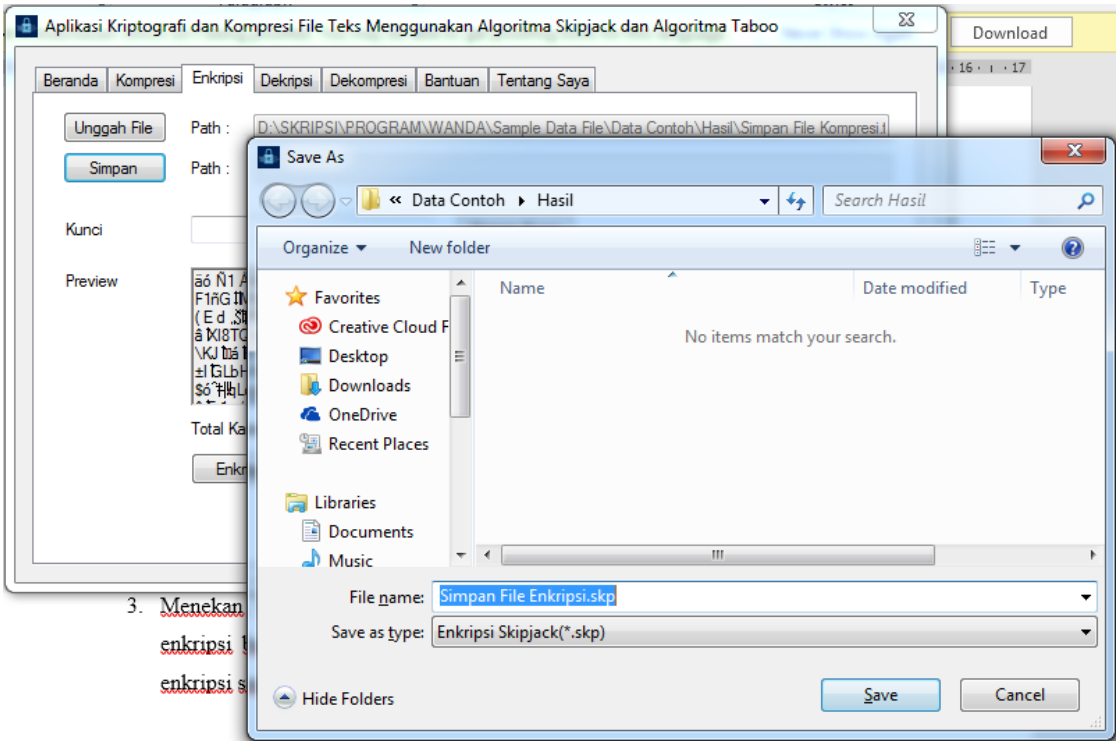
Untuk melakukan proses enkripsi, tahap awal yang dilakukan adalah memilih *tab* Enkripsi. Setelah tampilan *form* Enkripsi muncul maka lakukan langkah-langkah berikut ini untuk melakukan proses enkripsi.

1. Menekan tombol *Unggah File* untuk membuka *Open File Dialog*, kemudian pilih file yang yang telah dikompresi (*.tb) seperti pada Gambar 4.7 berikut.



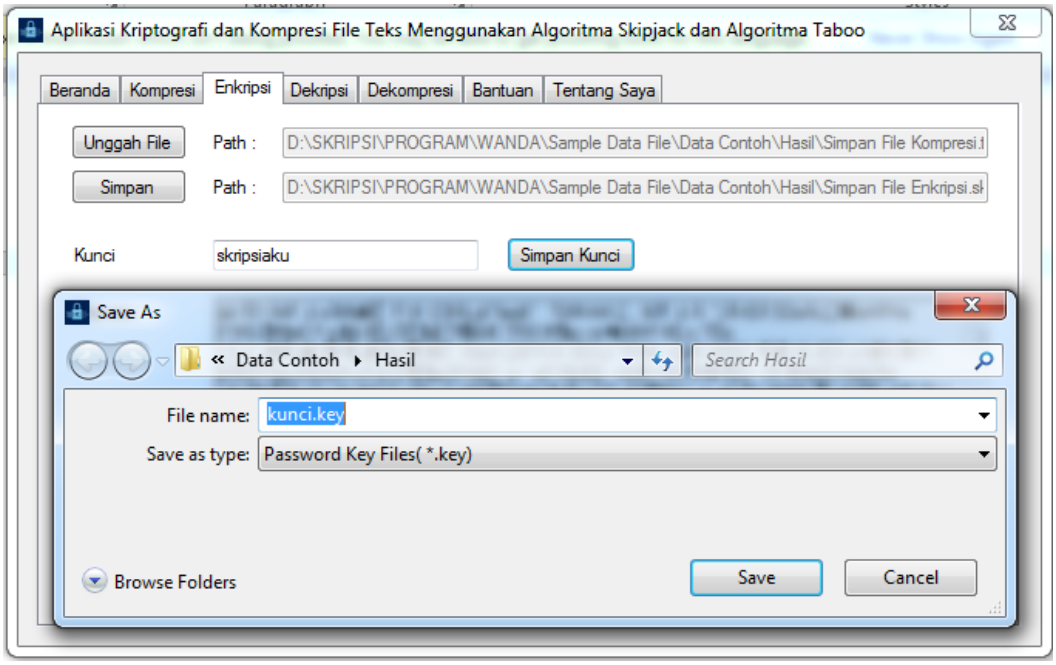
Gambar 4.7 Memilih file yang akan di enkripsi

2. Setelah itu, user memilih path direktori untuk menyimpan file seperti pada Gambar 4.8.



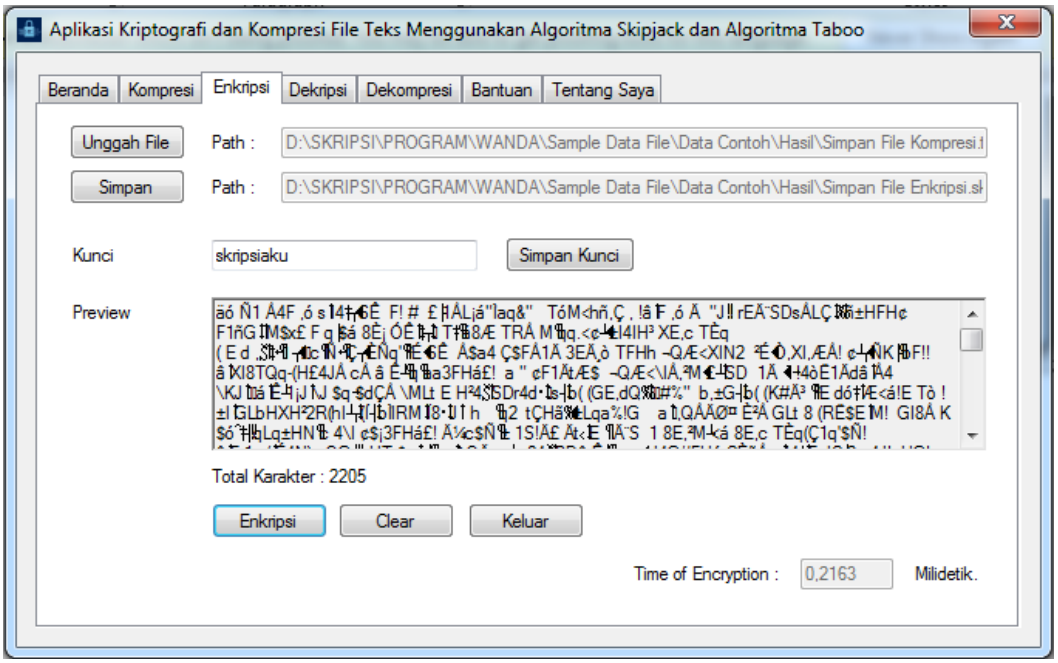
Gambar 4.8 Memilih path direktori penyimpanan file enkripsi

3. User menginput password atau kata kunci yang akan digunakan lalu tekan tombol Simpan Kunci untuk menyimpan kunci seperti pada Gambar 4.9.



Gambar 4.9 Membuat kunci dan menyimpan kunci proses enkripsi

4. Menekan tombol Enkripsi untuk melakukan proses enkripsi. Setelah proses enkripsi berhasil sistem akan menampilkan hasil enkripsi beserta waktu proses enkripsi seperti pada Gambar 4.10.



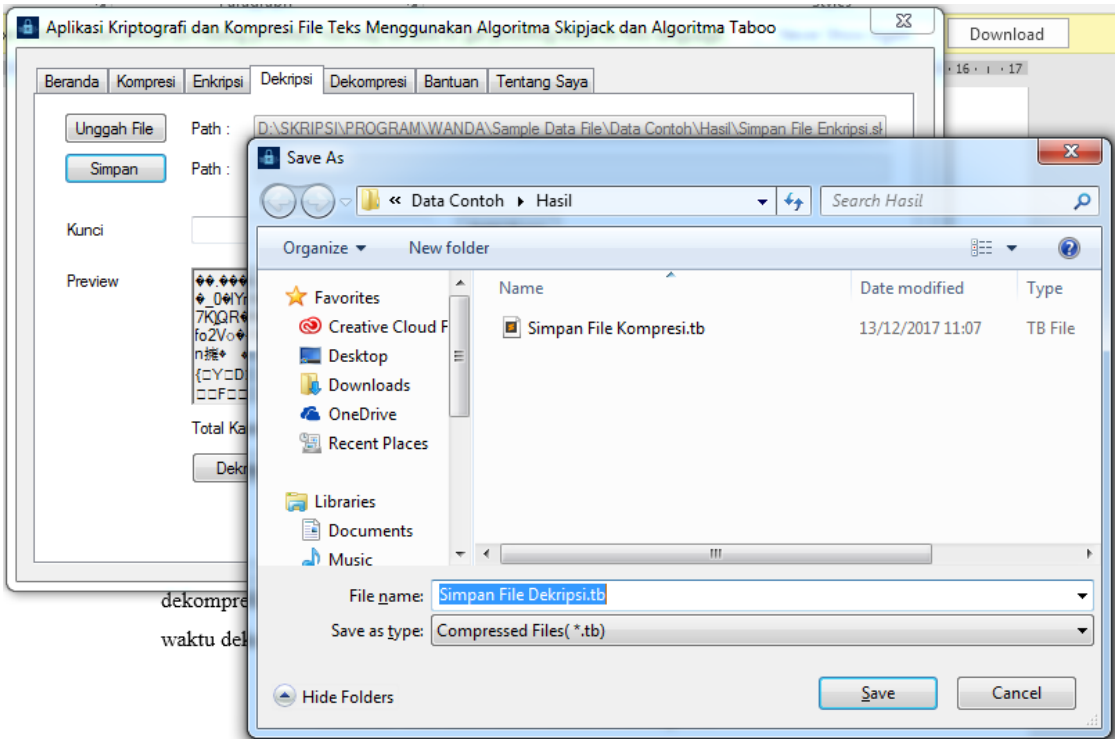
Gambar 4.10 Sistem setelah tombol Enkripsi diproses

Proses enkripsi menghasilkan sebuah *File Output* yaitu *file* dengan ekstensi *.skp sebagai *ciphertext* yang merupakan pesan rahasia dari hasil enkripsi.

4.3.3. Pengujian proses dekripsi

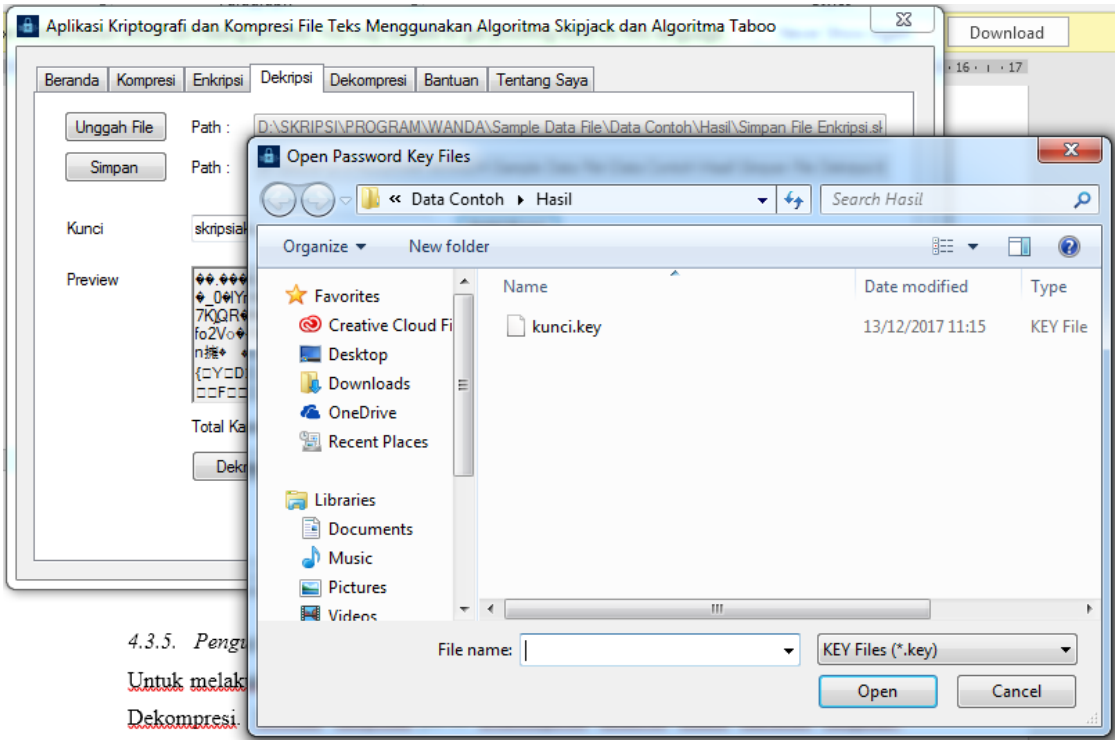
Untuk melakukan proses dekripsi, tahap awal yang dilakukan adalah memilih *tab* Dekripsi. Setelah tampilan *form* Dekripsi muncul maka lakukan langkah-langkah berikut ini untuk melakukan proses dekripsi.

1. Menekan tombol *Unggah File* untuk membuka *Open File Dialog*, kemudian pilih *file* hasil enkripsi.
2. Menekan tombol Simpan untuk memilih path direktori untuk menyimpan file proses hasil dekripsi seperti pada Gambar 4.11.



Gambar 4.11 Memilih path direktori untuk menyimpan file hasil dekripsi

3. Menekan tombol Ambil Kunci untuk menghasilkan kembali kata kunci yang telah disimpan seperti pada Gambar 4.12.



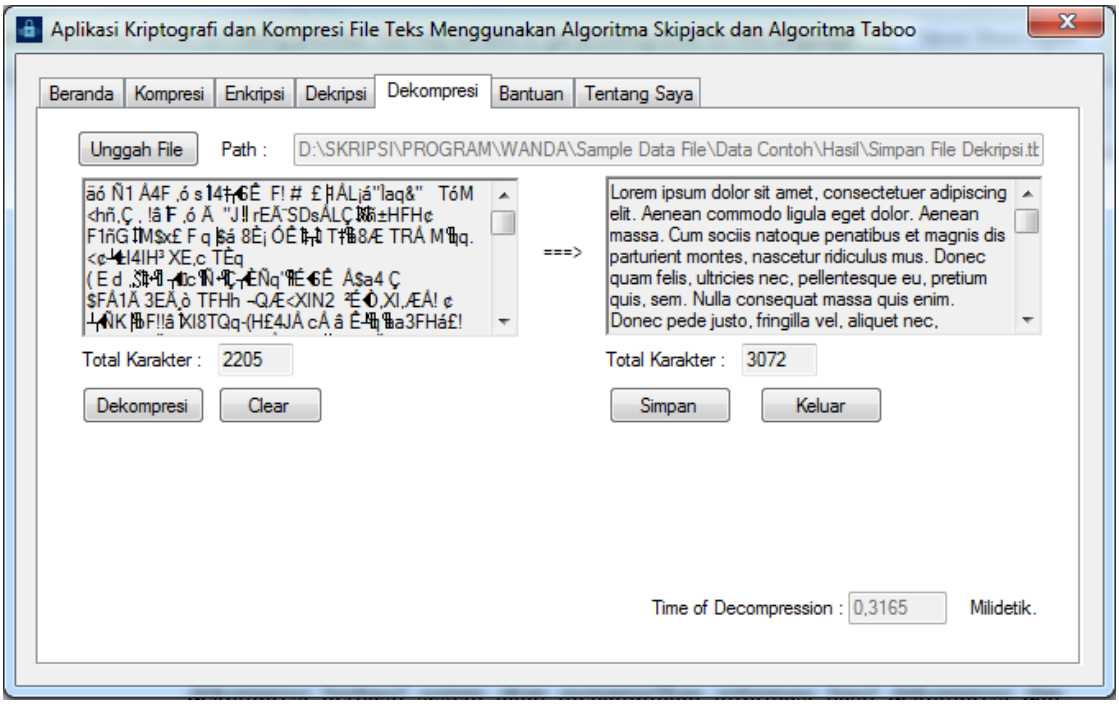
Gambar 4.12 Memilih path direktori untuk menyimpan file hasil dekripsi

4. Menekan tombol Dekripsi untuk menghasilkan proses dekripsi file berekstensi (*.tb).

4.3.4. Pengujian proses dekompresi

Untuk melakukan proses dekompresi, tahap awal yang dilakukan adalah memilih *tab* Dekompresi. Setelah tampilan *form* Dekompresi muncul maka lakukan langkah-langkah berikut ini.

1. Menekan tombol *Unggah File* untuk membuka *Open File Dialog*, dan pilih *file* hasil kompresi (*.tb) sebagai *Input File*.
2. Menekan tombol Dekompresi untuk melakukan proses dekompresi. Setelah proses dekompresi berhasil sistem akan menampilkan informasi hasil dekompresi dan waktu dekompresi pada *text box-text box* seperti pada Gambar 4.13.



Gambar 4.13 Informasi Hasil Dekompresi

3. Menekan tombol pada *Save* untuk membuka *Save File Dialog* dan tentukan direktori untuk menyimpan *file* hasil dekompresi (*.txt atau *.doc).

4.4. Hasil Pengujian

Pengujian dilakukan terhadap algoritma Kompresi Taboo dan algoritma Kriptografi Skipjack untuk mengukur keberhasilan sistem dalam melakukan proses kompresi, enkripsi, dekripsi, dan dekompresi, serta membandingkan waktu yang diperlukan untuk pengujian dua bentuk *file* teks *.txt dan *.doc.

Hasil pengujian pada *file* *.txt dapat dilihat pada tabel 4.4 dan pada *file* *.doc pada tabel 4.5

Tabel 4.4 Hasil Pengujian *File* *.txt

No.	Panjang Karakter Awal	Panjang Karakter Setelah Kompresi	Selisih Panjang Karakter	Waktu Kompresi	Waktu Enkripsi	Waktu Dekripsi	Waktu Dekompresi
				(dalam detik)			
1	1024 (Karakter)	736 (Karakter)	288 (Karakter)	0,0248	0,0695	0,0727	0,1511
				0,038	0,1461	0,1488	0,1719
				0,1832	0,185	0,1833	0,2047
	Rata – rata waktu yang didapat			0,082	0,1335	0,1349	0,1759
2	2048 (Karakter)	1469 (Karakter)	579 (Karakter)	0,0334	0,0657	0,0688	0,1027
				0,119	0,1214	0,1237	0,143
				0,1735	0,1762	0,1796	0,2063

	Rata – rata waktu yang didapat			0,1086	0,1211	0,124	0,1506
3	3072 (Karakter)	2205 (Karakter)	867 (Karakter)	0,0396	0,0768	0,0817	0,1135
				0,1277	0,1344	0,141	0,1818
				0,2243	0,2294	0,2343	0,2063
	Rata – rata waktu yang didapat			0,1305	0,1468	0,1523	0,1672
4	4096 (Karakter)	2938 (Karakter)	1158 (Karakter)	0,0225	0,0592	0,0639	0,101
				0,1324	0,1368	0,141	0,1758
				0,3899	0,3979	0,4025	0,4475
	Rata – rata waktu yang didapat			0,1816	0,1979	0,2024	0,2414
5	8192 (Karakter)	5876 (Karakter)	2316 (Karakter)	0,0343	0,0735	0,0813	0,1484
				0,172	0,1886	0,1983	0,2822
				0,4725	0,4814	0,4915	0,5721
	Rata – rata waktu yang didapat			0,2262	0,2478	0,257	0,3342
		Total Rata – rata		**0,1458**	**0,1694**	**0,1741**	**0,2138**

Tabel 4.5 Hasil Pengujian *File* *.doc

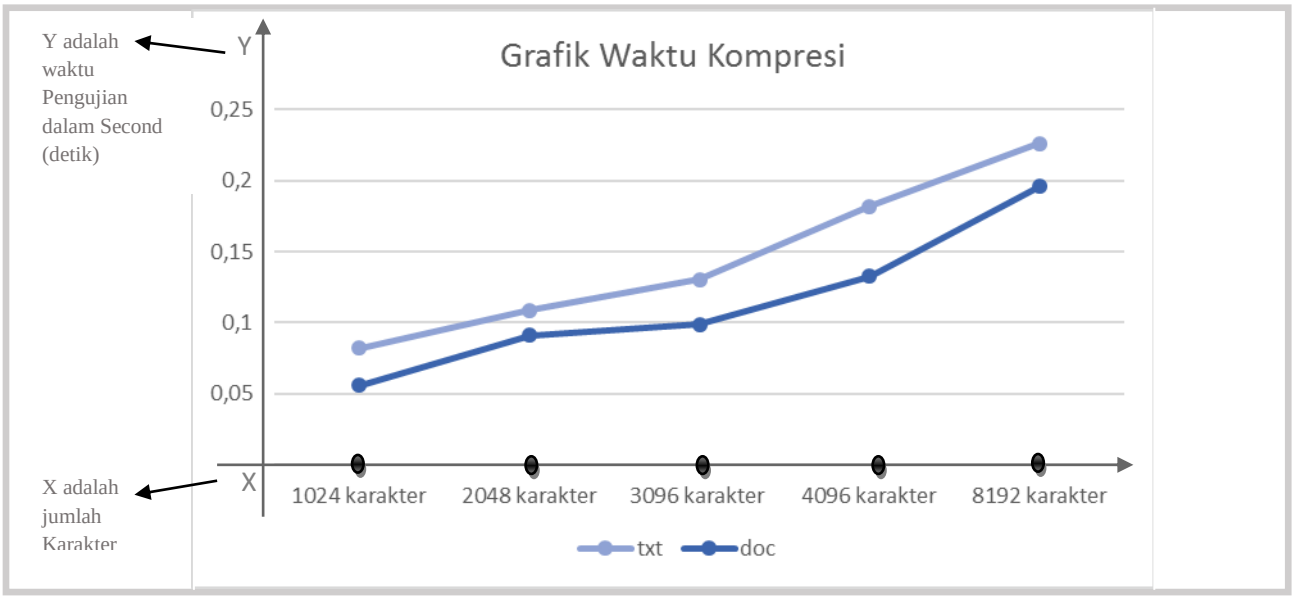
No.	Panjang Karakter Awal	Panjang Karakter Setelah Kompresi	Selisih Panjang Karakter	Waktu Kompresi	Waktu Enkripsi	Waktu Dekripsi	Waktu Dekompresi
				(dalam detik)			
1	1024 (Karakter)	736 (Karakter)	288 (Karakter)	0,0191	0,0478	0,0542	0,0787
				0,0498	0,0658	0,0689	0,0867
				0,0984	0,1	0,1015	0,1218
	Rata – rata waktu yang didapat			0,0557	0,0712	0,0748	0,0957
2	2048 (Karakter)	1469 (Karakter)	579 (Karakter)	0,0183	0,0922	0,0914	0,1394
				0,0646	0,1023	0,1021	0,1382
				0,1909	0,0924	0,113	0,1436
	Rata – rata waktu yang didapat			0,0912	0,0956	0,1021	0,1404
3	3072 (Karakter)	2205 (Karakter)	867 (Karakter)	0,0238	0,0489	0,0529	0,085
				0,1128	0,1199	0,123	0,1515
				0,1601	0,1642	0,1711	0,221
	Rata – rata waktu yang didapat			0,0989	0,111	0,1156	0,1525
4	4096 (Karakter)	2938 (Karakter)	1158 (Karakter)	0,0239	0,0517	0,0571	0,1081
				0,1286	0,1335	0,1374	0,1849
				0,2454	0,2543	0,2628	0,3215
	Rata – rata waktu yang didapat			0,1326	0,1465	0,1524	0,2048
5	8192 (Karakter)	5876 (Karakter)	2316 (Karakter)	0,0419	0,0701	0,079	0,1551
				0,183	0,1927	0,2007	0,2674
				0,3628	0,381	0,3916	0,4564
	Rata – rata waktu yang didapat			0,1959	0,2146	0,2237	0,2929
		Rata – rata		**0,1148**	**0,1391**	**0,1445**	**0,1846**

Pada pengujian kompresi terdapat parameter pembanding lain yaitu Comperation Ratio(Cr), Ratio Compression dan Redudancy(Rd), hasil pengujian dapat dilihat pada tabel 4.6.

Tabel 4.6 Hasil Pengujian Parameter Pembanding Kompresi

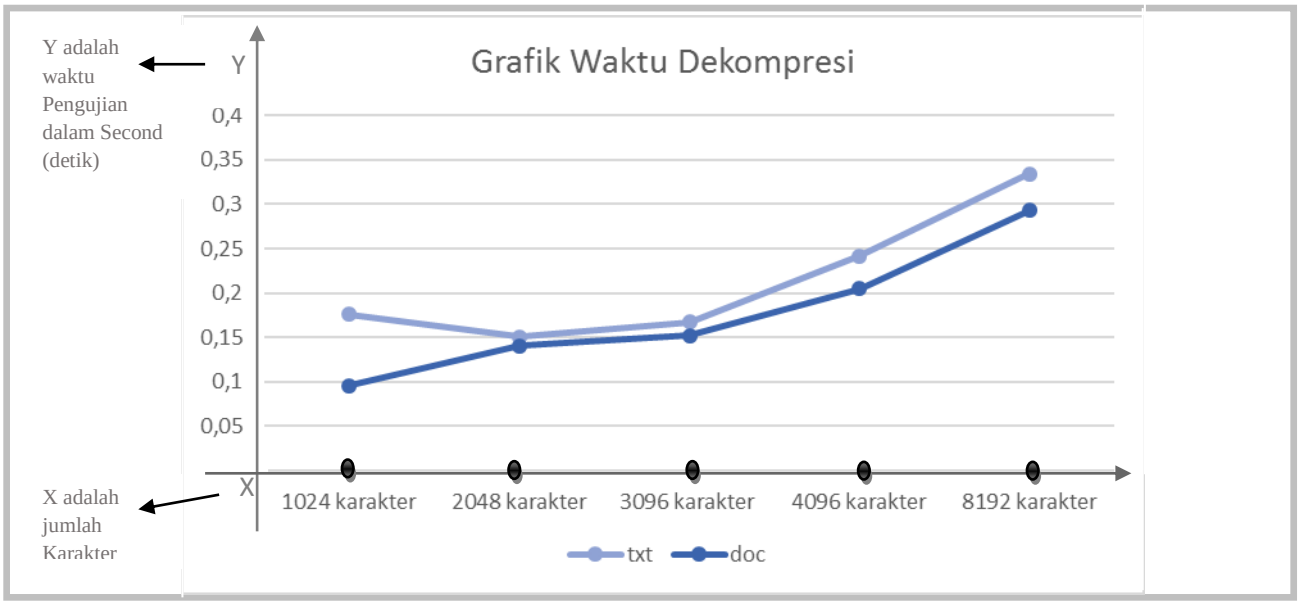
Jenis File	Ukuran Awal	Ukuran File terkompresi	Ratio Compression (%)	Compression of Ratio	Redudancy (%)
Txt	1024 (Karakter)	736 (Karakter)	71,88	1,39	28,12
	2048 (Karakter)	1469 (Karakter)	71,88	1,39	28,22
	3072 (Karakter)	2205 (Karakter)	71,88	1,39	28,22
	4096 (Karakter)	2938 (Karakter)	71,75	1,39	28,25
	8192 (Karakter)	5876 (Karakter)	71,74	1,39	28,26
Doc	1024 (Karakter)	736 (Karakter)	71,88	1,39	28,12
	2048 (Karakter)	1469 (Karakter)	71,88	1,39	28,22
	3072 (Karakter)	2205 (Karakter)	71,88	1,39	28,22
	4096 (Karakter)	2938 (Karakter)	71,75	1,39	28,25
	8192 (Karakter)	5876 (Karakter)	71,74	1,39	28,26

Dari Tabel 4.4 dan Tabel 4.5 dapat dibuat grafik perbandingan hasil pengujian *file* teks *.txt dan *.doc berdasarkan variable waktu kompresi, waktu dekompresi, waktu enkripsi, dan waktu dekripsi seperti pada Gambar 4.14, Gambar 4.15, Gambar 4.16, dan Gambar 4.17.



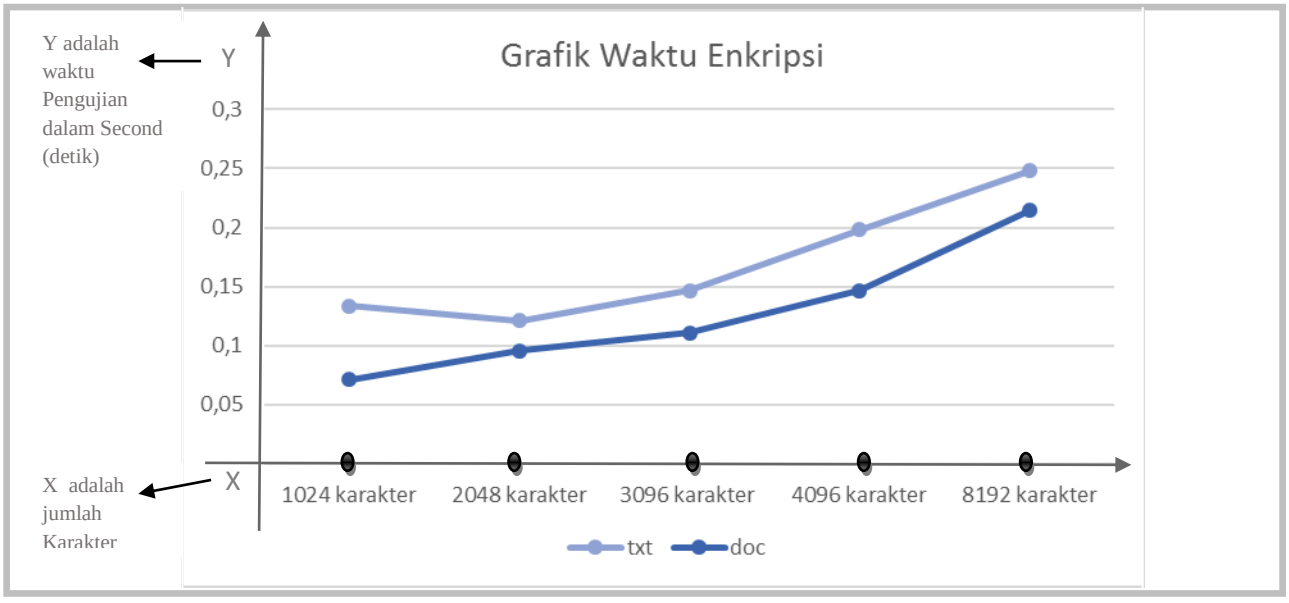
Gambar 4.14 Grafik waktu kompresi terhadap ukuran *file*

Dari grafik pada Gambar 4.14 dapat diambil kesimpulan bahwa waktu proses kompresi untuk file *.txt lebih lama dibanding untuk *.doc.



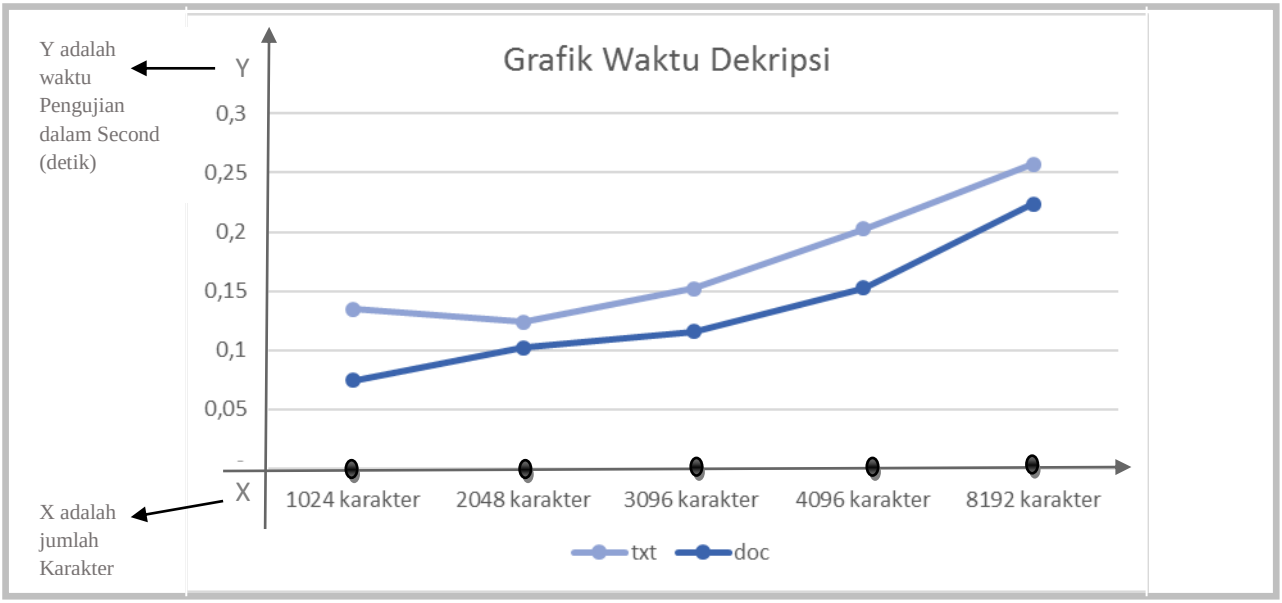
Gambar 4.15 Grafik waktu dekompresi terhadap ukuran *file*

Dari grafik pada Gambar 4.15 dapat diambil kesimpulan bahwa waktu proses dekompresi untuk file *.doc lebih lama dibanding untuk *.txt.



Gambar 4.16 Grafik Waktu Enkripsi terhadap ukuran *file*

Dari grafik pada Gambar 4.16 dapat diambil kesimpulan bahwa waktu proses enkripsi untuk file *.txt lebih lama dibanding untuk *.doc.



Gambar 4.17 Grafik Waktu Dekripsi terhadap ukuran *file*

Dari grafik pada Gambar 4.17 dapat diambil kesimpulan bahwa waktu proses dekripsi untuk file *.txt lebih lama dibanding untuk *.doc

F ormat data teks (*.txt) merupakan format teks yang digunakan untuk menyimpan huruf, angka, karakter kontrol (tabulasi, pindah baris, dan sebagainya) atau simbol-simbol lain yang biasa digunakan dalam tulisan seperti titik, koma, tanda petik, dan sebagainya. Satu huruf, angka, karakter kontrol atau simbol pada arsip teks memakan tempat satu byte. Berbeda dengan jenis teks terformat yang satu huruf saja dapat memakan tempat beberapa byte untuk menyimpan format dari huruf tersebut seperti font, ukuran, tebal atau tidak dan sebagainya. Kelebihan dari format data teks ini adalah ukuran datanya yang kecil karena tidak adanya fitur untuk memformat tampilan teks.

Sedangkan Format data dokumen (*.doc) merupakan ekstensi arsip dokumen perangkat lunak Microsoft Word yang paling banyak digunakan dalam menulis laporan, makalah dan sebagainya. Doc merupakan jenis teks terformat yang tidak hanya dapat mengatur tampilan teks seperti styles (font, ukuran huruf, dan sebagainya), namun juga dapat menyisipkan gambar. Kekurangan format teks dokumen ini terletak pada ukuran datanya yang besar.

BAB 5
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Kesimpulan yang dapat diambil dari penelitian ini adalah:

1. Algoritma Skipjack berfungsi untuk mengamankan informasi pesan yang akan dienkripsi sementara algoritma Taboo berfungsi untuk memperkecil ukuran file dan kapasitas memory yang dipakai.
2. Aplikasi yang dirancang dalam penelitian telah mampu melakukan proses kompresi, enkripsi, dekripsi, dan dekompresi.
3. Pesan yang bisa diinput untuk enkripsi dan dikompres pesan berupa *.txt dan *doc.
4. Hasil pengujian kompresi pada ukuran *file* teks dengan jenis *file* *.txt tetap seperti dari ukuran *file* sebelumnya. Pada kompresi jenis *file* *.doc ukuran hasil kompresi mengecil lebih banyak dari *file* sebelumnya, sehingga ketika ukuran *file* asli 28 Kb maka ukuran hasil kompresi menjadi sekitar 3 Kb.
5. Hasil pengujian kompresi pada *file* teks dengan jenis *file* *.txt atau jenis *file* *.doc yang berisi 3072 karakter di kompresi menjadi 2205 karakter. Pada proses enkripsi *file* teks dienkripsi menjadi *ciphertext* yang memiliki panjang 848 karakter.
6. Hasil pengujian kompresi dan dekompresi *file* memliki waktu kompresi waktu rata-rata sebesar 0,1148 detik dan waktu dekompresi rata-rata sebesar 0,1846 detik serta rata-rata *Ratio of Compression* (RC) sebesar 71,88, *Compression Ratio* (CR) sebesar 1,39 % dan *Redundancy* (RD) sebesar 28,22 %.
7. Hasil pengujian enkripsi dan dekripsi *file* memliki waktu enkripsi rata-rata 0,1391 detik dan dekripsi rata-rata sebesar 0,1445 detik.

5.2. Saran

Saran yang dapat diberikan pada penulis untuk pengembangan dan perbaikan sistem lebih lanjut adalah:

1. Untuk penelitian selanjutnya diharapkan dapat mengkompresi dan mengenkripsi *input file* teks dari dokumen yang berekstensi *.docx berupa gambar, grafik, tabel dan komponen lainnya.
2. Untuk penelitian selanjutnya diharapkan agar dapat membangun aplikasi yang dapat diterapkan pada perangkat Android, iOS atau lainnya.
3. Penelitian selanjutnya diharapkan agar dapat membangun aplikasi sistem menggunakan bahasa pemrograman yang lain.

DAFTAR PUSTAKA

Kromodimoeldjo, S. 2010. Teori dan Aplikasi Kroptografi. SPK IT Konsulting.

Taofik, M.C., Sholeh. & Erni, Y. 2015. Kompresi Teks Menggunakan Algoritma Huffman dan Md5 pada *Instant Messaging* Smartphone Android. Jurnal EECCIS 9(1): 103-108.

Mollin, R. A. 2005. *Codes: The Guide To Secrecy From Ancient To Modern Times*. Florida: Chapman & Hall/CRC.

Munir, R. 2006. Kriptografi. Informatika: Bandung.

Schneier, Bruce. 1996. *Applied Cryptography: Protocol, Algorithm, and Source Code in C. Second Edition*. John Wiley and Son, inc: New Jersey.

Mollin, R. A. 2007. *An Introduction to Cryptography*. Edisi ke-2. Florida: Chapman & Hall/CRC

Mengyi, Pu. 2006. *Fundamental Data Compression*. United Kingdom: Oxford University.

Solihin, M. 2014. Perancangan Sistem Pengamanan Dan Kompresi Data Teks Dengan Fibonacci Encoding dan Algoritma Shannon-Fano Serta Algoritma Deflate. Skripsi. Universitas Sumatera. Utara.

Taofik, M.C., Sholeh. & Erni, Y. 2015. Kompresi Teks Menggunakan Algoritma Huffman dan Md5 pada *Instant Messaging* Smartphone Android. Jurnal EECCIS 9(1): 103-108

Suprianto. 2007. Sistem pengkodean data pada file teks pada keamanan informasi dengan menggunakan metode skipjack. Jurnal. Bandung.

Asri, Y. 2009. Perancangan Aplikasi Enkripsi dan Dekripsi Data Dengan Menggunakan Algoritma Skipjack. (Online) http://portal.kopertis3.or.id (14 Januari 2017).

Salomon, D. 2007. *Variable-Length Codes for Data Compression*. London: Springer.

Cui T., Jin C., Zhang G. 2013. *Observations of Skipjack-like Structure with SP/SPS Round Function. Journal of Universal Computer Science* 19(16): 2454-2471.

Cormen, T.H., Leiserson, C.E., Rivest, R.L. & Stein, C. 2001. *Introduction to Algorithms*. Second Edition. The MIT Press: London.

Kuppuswamy, P. & Al-Khalidi Q.Y. Saeed. 2014. *Hybrid Encyption/Decryption Technique Using New Public Key and Simmetric Key Algorithm*. pp. 1-13.

Schneier, Bruce. 1996. *Applied Cryptography: Protocol, Algorithm, and Source Code in C. Second Edition*. John Wiley and Son, inc: New Jersey.

Panggabean, A. 2015. Implementasi *Hybrid Cryptosystem* dengan Algoritma One Time Pad dan Algoritma *Rabin Cryptosystem.* Skripsi. Universitas Sumatera Utara.

Sirait, L. M. 2013. *Perancangan Aplikasi Kompresi Data File Teks Digital Dengan Menggunakan Algoritma Lemple Ziv Storer Symanski (LZSS)*. Jurnal : Pelita Informatika Budi Darma, Vol : V, No. 3, pp. 54-57.

Suhastra, F 2014. *Implementasi Algoritma Kompresi Lampel Ziv Welch (LZW) Pada Berkas Digital*. Jurnal : Pelita Informatika Budi Darma, Vol : VI, pp. 71-75.

LISTING PROGRAM

a. Sourcecode Form1.cs

```
using System;
using Eleven41.Skip32;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Diagnostics;
using System.Windows.Forms;

namespace Skipjack___Taboo
{
    public partial class Form1 : Form
    {
        Stopwatch _stopWatch = new Stopwatch();
        int charCount = 0;
        public String cs;
        public string metode, kalimat;
        public String code;
        public String[] ataboo;
        private const string KEY_FILENAME = "kunci.key";
        private const string PUBLIC_KEY_FILENAME = "untitled.key";
        public Form1()
        {
            InitializeComponent();
        }

        private byte[] MakeKeyMD5(string password) {
            byte[] encodedPassword = new
UTF8Encoding().GetBytes(password);
            byte[] hash =
((System.Security.Cryptography.HashAlgorithm)System.Security.Cryptogr
aphy.CryptoConfig.CreateFromName("MD5")).ComputeHash(encodedPassword)
;
            return hash;
        }

        private void Encrypt(string sourceFile, string destFile,
string password)
        {
            //password
            Skip32Cipher skip32 = new
Skip32Cipher(MakeKeyMD5(password));
            //fileteks
            var inFileSteam = new System.IO.FileStream(sourceFile,
System.IO.FileMode.Open);
            byte[] buffer = new byte[Skip32Cipher.BlockSize];
            int bytesRead = inFileSteam.Read(buffer, 0,
buffer.Length);
            //hasil
```

```
            System.IO.BinaryWriter outFileStream = new
System.IO.BinaryWriter(new System.IO.FileStream(destFile,
System.IO.FileMode.Create));
            while (bytesRead > 0)
            {
                if (bytesRead == buffer.Length)
                {
                    byte[] encrypted = skip32.Encrypt(buffer);
                    outFileStream.Write(encrypted, 0, bytesRead);
                }
                else
                {
                    outFileStream.Write(buffer, 0, bytesRead);
                }

                bytesRead = inFileSteam.Read(buffer, 0,
buffer.Length);
            }

            outFileStream.Close();
            inFileSteam.Close();
        }

        private void Decrypt(string sourceFile, string destFile,
string password)
        {
            Skip32Cipher skip32 = new
Skip32Cipher(MakeKeyMD5(password));
            var inFileSteam = new System.IO.FileStream(sourceFile,
System.IO.FileMode.Open);
            byte[] buffer = new byte[Skip32Cipher.BlockSize];
            int bytesRead = inFileSteam.Read(buffer, 0,
buffer.Length);
            System.IO.BinaryWriter outFileStream = new
System.IO.BinaryWriter(new System.IO.FileStream(destFile,
System.IO.FileMode.Create));
            while (bytesRead > 0)
            {
                if (bytesRead == buffer.Length)
                {
                    byte[] decrypted = skip32.Decrypt(buffer);
                    outFileStream.Write(decrypted, 0, bytesRead);
                }
                else
                {
                    outFileStream.Write(buffer, 0, bytesRead);
                }

                bytesRead = inFileSteam.Read(buffer, 0,
buffer.Length);
            }

            outFileStream.Close();
            inFileSteam.Close();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            int asd;
            Stopwatch watch = new Stopwatch();
            watch.Start();
            _stopWatch.Start();
```

```
            String kalimat = isifile.Text;
            classtaboo.taboo(kalimat);
            String stb = classtaboo.StringToStb(kalimat,
classtaboo.cs, classtaboo.ataboo);
            String code = classtaboo.Encode(stb);
            watch.Stop();
            hasilfile.Text = code;
            _stopWatch.Stop();
            asd = Convert.ToInt16(hasilfile.Text.Length) - 1;
            double CrR =  (kalimat.Length * 8 * 1.0) / stb.Length;
            //textBox7.Text = asd.ToString(); //
            textBox7.Text = hasilfile.Text.Length.ToString();
            RatioofCompress.Text = Math.Round(CrR, 2).ToString();
            double RrC = (stb.Length / (kalimat.Length * 8 * 1.0)) *
100;
            CompressRatio.Text = Math.Round(RrC, 2).ToString() + "
%";
            double redu = 100 - RrC;
            Redudancy.Text = Math.Round(redu, 2).ToString() + " %";
            SpaceSaving.Text = (1 - (Convert.ToDouble(asd) /
Convert.ToDouble(textBox6.Text.ToString()))).ToString();
            // Waktu Proses belum dimasukin
            txtWaktu.Text =
Math.Round(Convert.ToDecimal(_stopWatch.Elapsed.TotalMilliseconds) /
1000, 4).ToString();
            MessageBox.Show("Anda Sukses Melakukan Proses Kompresi
File");

//Math.Round(Convert.ToDecimal(watch.ElapsedMilliseconds),
4).ToString() + " ms";
        }

        private void button2_Click(object sender, EventArgs e)
        {
            SaveFileDialog simpan = new SaveFileDialog();
            simpan.Filter = "Compressed Files( *.tb)|*.tb";
            simpan.FileName = "*.tb";
            if (simpan.ShowDialog() == DialogResult.OK) {
                FileStream fstream = new FileStream(simpan.FileName,
FileMode.Create);
                StreamWriter sw = new StreamWriter(fstream);
                SeekOrigin seekorigin = new SeekOrigin();
                sw.BaseStream.Seek(0, seekorigin);
                sw.Write(hasilfile.Text);
                sw.Flush();
                sw.Close();
                fstream.Close();
                MessageBox.Show("File telah berhasil disimpan.");
            }
        }

        private void EncryptPassword(string publicKeyFileName, string
privateKeyFileName, string encryptedPasswordFileName)
        {

        }

        private void PjgKarakterAwal_TextChanged(object sender,
EventArgs e)
        {

        }
```

```
        private void isifile_TextChanged(object sender, EventArgs e)
        {

        }

        private void button4_Click(object sender, EventArgs e)
        {
            try {
                OpenFileDialog open = new OpenFileDialog();
                open.Filter = "Compressed Files( *.tb)|*.tb";
                if (open.ShowDialog() == DialogResult.OK) {
                    txtPathKompres.Text =
Path.GetFullPath(open.FileName);
                    string nama = open.FileName.Substring(0,
open.FileName.Length);
                    FileStream fstream = new FileStream(nama,
FileMode.Open, FileAccess.ReadWrite);
                    StreamReader sreader = new StreamReader(fstream);
                    sreader.BaseStream.Seek(0, SeekOrigin.Begin);
                    code = sreader.ReadToEnd();
                    sreader.Close();
                    string namaa = open.FileName.Substring(0,
open.FileName.Length - 3) + "v.dat";
                    FileStream fstreamm = new FileStream(namaa,
FileMode.Open, FileAccess.ReadWrite);
                    StreamReader sreaderr = new
StreamReader(fstreamm);
                    sreaderr.BaseStream.Seek(0, SeekOrigin.Begin);
                    string keterangan = sreaderr.ReadToEnd();
                    string[] infochar = new string[1];
                    infochar = keterangan.Split('~');
                    cs = infochar[0];
                    string[] info = new string[cs.Length + 1];
                    ataboo = new string[cs.Length];
                    info = keterangan.Split('~');
                    for (int i = 1; i <= cs.Length; i++)
                        ataboo[i - 1] = info[i];
                    sreader.Close();
                    isifiledekom.Text = code;
                    textBox3.Text =
isifiledekom.Text.Length.ToString();
                }
            }

            catch (Exception ex) {
                MessageBox.Show(ex.ToString());
            }
        }

        private void button6_Click(object sender, EventArgs e)
        {
            _stopWatch.Start();
            String dc = classtaboo.Decode(isifiledekom.Text);
            String ds = classtaboo.Decompress(dc, cs, ataboo);
            hasilfiledekom.Text = ds.Substring(0, ds.Length);
            textBox4.Text = hasilfiledekom.Text.Length.ToString();
            _stopWatch.Stop();
            txtWaktuDekom.Text =
Math.Round(Convert.ToDecimal(_stopWatch.Elapsed.TotalMilliseconds) /
1000, 4).ToString();
```

```
            MessageBox.Show("Anda Sukses Melakukan Proses Dekompresi
File");
        }

        private void button7_Click(object sender, EventArgs e)
        {
            Stream myStream;
            OpenFileDialog theDialogOpen = new OpenFileDialog();
            theDialogOpen.Title = "Open File";
            theDialogOpen.Filter = "Taboo Compression Files(
*.tb)|*.tb|Text Files( *.txt)|*.txt|Word 97-2003( *.doc)|*.doc|Word
Document( *.docx)|*.docx";
            theDialogOpen.InitialDirectory = @".\";
            if (theDialogOpen.ShowDialog() == DialogResult.OK) {
                if ((myStream = theDialogOpen.OpenFile()) != null) {
                    txtPathEnkrip.Text = theDialogOpen.FileName;
                    string strfilename = theDialogOpen.FileName;
                    string filetext = File.ReadAllText(strfilename);
                    previewEnkrip.Text = filetext;
                }
                myStream.Close();
            }
        }

        private void button8_Click(object sender, EventArgs e)
        {
            SaveFileDialog theDialogSave = new SaveFileDialog();
            theDialogSave.Title = "Save As";
            theDialogSave.Filter = "Enkripsi Skipjack(*.skp) |
*.skp";
            theDialogSave.FileName = "untitled";
            theDialogSave.InitialDirectory = @".\";
            if (theDialogSave.ShowDialog() == DialogResult.OK) {
                txtSimpanEnkrip.Text = theDialogSave.FileName;
            }
        }

        private void button9_Click(object sender, EventArgs e)
        {
            _stopWatch.Start();
            if (File.Exists(txtPathEnkrip.Text)) {
                Encrypt(txtPathEnkrip.Text, txtSimpanEnkrip.Text,
kunci.Text);
            }
            _stopWatch.Stop();
            txtWaktuEnkrip.Text =
Math.Round(Convert.ToDecimal(_stopWatch.Elapsed.TotalMilliseconds) /
1000, 4).ToString();
            MessageBox.Show("Anda Sukses Melakukan Proses Enkripsi
File");
        }

        private void button10_Click(object sender, EventArgs e)
        {
            SaveFileDialog simpankunci = new SaveFileDialog();
            simpankunci.Filter = "Password Key Files( *.key)|*.key";
            simpankunci.FileName = "*.key";
            if (simpankunci.ShowDialog() == DialogResult.OK)
            {
                FileStream fstream = new
FileStream(simpankunci.FileName, FileMode.Create);
                StreamWriter sw = new StreamWriter(fstream);
```

```
                SeekOrigin seekorigin = new SeekOrigin();
                sw.BaseStream.Seek(0, seekorigin);
                sw.Write(kunci.Text);
                sw.Flush();
                sw.Close();
                fstream.Close();
            }
        }

        private void button11_Click(object sender, EventArgs e)
        {
            txtWaktuDekrip.Text = "";
            Stream myStream1;
            OpenFileDialog theDialogOpen = new OpenFileDialog();
            theDialogOpen.Title = "Open File";
            theDialogOpen.Filter = "Enkripsi Skipjack(*.skp) |
*.skp";
            theDialogOpen.InitialDirectory = @".\";
            if (theDialogOpen.ShowDialog() == DialogResult.OK)
            {
                if ((myStream1 = theDialogOpen.OpenFile()) != null)
                {
                    txtPathDekrip.Text = theDialogOpen.FileName;
                    string strfilename1 = theDialogOpen.FileName;
                    string filetext1 =
File.ReadAllText(strfilename1);
                    previewDekrip.Text = filetext1;
                }
                myStream1.Close();
            }
        }

        private void button12_Click(object sender, EventArgs e)
        {
            SaveFileDialog theDialogSave = new SaveFileDialog();
            theDialogSave.Title = "Save As";
            theDialogSave.Filter = "Compressed Files( *.tb)|
*.tb|Text Files( *.txt)|*.txt|Word 97-2003( *.doc)|*.doc|Word
Document( *.docx)|*.docx";
            theDialogSave.FileName = "untitled";
            theDialogSave.InitialDirectory = @".\";
            if (theDialogSave.ShowDialog() == DialogResult.OK)
            {
                string filenamee =
theDialogSave.FileName.Substring(0, theDialogSave.FileName.Length -
3) + "v.dat";
                FileStream fstreamm = new FileStream(filenamee,
FileMode.Create);
                StreamWriter sww = new StreamWriter(fstreamm);
                SeekOrigin seekoriginn = new SeekOrigin();
                sww.BaseStream.Seek(0, seekoriginn);
                sww.Write(classtaboo.cs + "~");
                for (int n = 0; n < classtaboo.cs.Length; n++)
                    sww.Write(classtaboo.ataboo[n] + "~");
                sww.Flush();
                sww.Close();
                fstreamm.Close();

                txtSimpanDekrip.Text = theDialogSave.FileName;
                FileStream fstream = new
FileStream(theDialogSave.FileName, FileMode.Create);
                StreamWriter sw = new StreamWriter(fstream);
```

```
                SeekOrigin seekorigin = new SeekOrigin();
                sw.BaseStream.Seek(0, seekorigin);
                sw.Flush();
                sw.Close();
                fstream.Close();

            }
        }

        private void button13_Click(object sender, EventArgs e)
        {
            OpenFileDialog theDiaglogOpenKey = new OpenFileDialog();
            theDiaglogOpenKey.Title = "Open Password Key Files";
            theDiaglogOpenKey.Filter = "KEY Files|*.key";
            theDiaglogOpenKey.InitialDirectory = @".\";
            try {
                if (theDiaglogOpenKey.ShowDialog() ==
DialogResult.OK)
                {
                    kuncidekrip.Text =
File.ReadAllText(theDiaglogOpenKey.FileName);
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message, "Form1",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }

        }

        private void button14_Click(object sender, EventArgs e)
        {
            _stopWatch.Start();
            if (File.Exists(txtPathDekrip.Text))
            {
                //lakukan dekripsi.
                Decrypt(txtPathDekrip.Text, txtSimpanDekrip.Text,
kuncidekrip.Text);
            }
            _stopWatch.Stop();
            txtWaktuDekrip.Text =
Math.Round(Convert.ToDecimal(_stopWatch.Elapsed.TotalMilliseconds) /
1000, 4).ToString();
            MessageBox.Show("Anda Sukses Melakukan Proses Dekripsi
File");
        }

        private void button15_Click(object sender, EventArgs e)
        {
            txtPathImport.Text = "";
            isifile.Text = "";
            hasilfile.Text = "";
            textBox6.Text = "";
            textBox7.Text = "";
            CompressRatio.Text = "";
            RatioofCompress.Text = "";
            Redudancy.Text = "";
            SpaceSaving.Text = "";
            txtWaktu.Text = "";
        }
```

```
        private void button16_Click(object sender, EventArgs e)
        {
            txtPathEnkrip.Text = "";
            txtSimpanEnkrip.Text = "";
            kunci.Text = "";
            previewEnkrip.Text = "";
            txtWaktuEnkrip.Text = "";
        }

        private void button17_Click(object sender, EventArgs e)
        {
            txtPathDekrip.Text = "";
            txtSimpanDekrip.Text = "";
            kuncidekrip.Text = "";
            previewDekrip.Text = "";
            txtWaktuDekrip.Text = "";
        }

        private void button18_Click(object sender, EventArgs e)
        {
            txtPathKompres.Text = "";
            isifiledekom.Text = "";
            hasilfiledekom.Text = "";
            textBox3.Text = "";
            textBox4.Text = "";
            txtWaktuDekom.Text = "";
        }

        private void previewEnkrip_TextChanged(object sender,
EventArgs e)
        {
            charCount = previewEnkrip.Text.Length;
            label21.Text = "Total Karakter : " +
charCount.ToString();
        }

        private void previewDekrip_TextChanged(object sender,
EventArgs e)
        {
            charCount = previewDekrip.Text.Length;
            label28.Text = "Total Karakter : " +
charCount.ToString();
        }

        private void button5_Click(object sender, EventArgs e)
        {
            SaveFileDialog simpanhasil = new SaveFileDialog();
            simpanhasil.Filter = "Text Files( *.txt)|*.txt|Word 97-
2003( *.doc)|*.doc";
            simpanhasil.FileName = "untitled";
            simpanhasil.InitialDirectory = @".\";
            if (simpanhasil.ShowDialog() == DialogResult.OK) {
                Stream fileStream = simpanhasil.OpenFile();
                StreamWriter sw = new StreamWriter(fileStream);

                sw.Write(hasilfiledekom.Text);
                sw.Close();
```

```
                fileStream.Close();
            }
            MessageBox.Show("File telah berhasil disimpan.");
        }

        private void tabPage1_Click(object sender, EventArgs e)
        {

        }

        private void button19_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show("Apakah Anda Yakin Ingin Keluar
Aplikasi ?", "Keluar Aplikasi", MessageBoxButtons.YesNo) ==
DialogResult.Yes)
            {
                Application.Exit();

            }
        }

        private void button20_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show("Apakah Anda Yakin Ingin Keluar
Aplikasi ?", "Keluar Aplikasi", MessageBoxButtons.YesNo) ==
DialogResult.Yes)
            {
                Application.Exit();

            }
        }

        private void button22_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show("Apakah Anda Yakin Ingin Keluar
Aplikasi ?", "Keluar Aplikasi", MessageBoxButtons.YesNo) ==
DialogResult.Yes)
            {
                Application.Exit();

            }
        }

        private void button21_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show("Apakah Anda Yakin Ingin Keluar
Aplikasi ?", "Keluar Aplikasi", MessageBoxButtons.YesNo) ==
DialogResult.Yes)
            {
                Application.Exit();

            }
        }

        private void label35_Click(object sender, EventArgs e)
        {

        }

        private void label37_Click(object sender, EventArgs e)
        {
```

```
        }

        private void label40_Click(object sender, EventArgs e)
        {


      }

        private void label41_Click(object sender, EventArgs e)
        {

        }

        private void label41_Click_1(object sender, EventArgs e)
        {

        }

        private void pictureBox2_Click(object sender, EventArgs e)
        {

        }

        private void listBox1_SelectedIndexChanged(object sender,
EventArgs e)
        {

        }

        private void label66_Click(object sender, EventArgs e)
        {

        }

        private void label60_Click(object sender, EventArgs e)
        {

        }

        private void label59_Click(object sender, EventArgs e)
        {

        }

        private void groupBox2_Enter(object sender, EventArgs e)
        {

        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }

        private void label69_Click(object sender, EventArgs e)
        {

        }

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

```
            try {
                OpenFileDialog open = new OpenFileDialog();
                open.Filter = "Text Files( *.txt)|*.txt|Word Document
97-2003( *.doc)|*.doc|Skipjack Encryption Files( *.skp)|*.skp";
                if (open.ShowDialog() == DialogResult.OK) {
                    metode =
open.FileName.Substring(open.FileName.Length - 3, 3);
                    if (metode == "txt")
                    {
                        txtPathImport.Text =
Path.GetFullPath(open.FileName);
                        string nama = open.FileName.Substring(0,
open.FileName.Length);
                        FileStream fstream = new FileStream(nama,
FileMode.Open, FileAccess.ReadWrite);
                        StreamReader sreader = new
StreamReader(fstream);
                        sreader.BaseStream.Seek(0, SeekOrigin.Begin);
                        isifile.Text = sreader.ReadToEnd();
                        sreader.Close();
                        textBox6.Text =
isifile.Text.Length.ToString();
                    }
                    if (metode == "skp")
                    {
                        txtPathImport.Text =
Path.GetFullPath(open.FileName);
                        string nama = open.FileName.Substring(0,
open.FileName.Length);
                        FileStream fstream = new FileStream(nama,
FileMode.Open, FileAccess.ReadWrite);
                        StreamReader sreader = new
StreamReader(fstream);
                        sreader.BaseStream.Seek(0, SeekOrigin.Begin);
                        isifile.Text = sreader.ReadToEnd();
                        sreader.Close();
                        textBox6.Text =
isifile.Text.Length.ToString();
                    }
                    if (metode == "doc")
                    {
                        txtPathImport.Text =
Path.GetFullPath(open.FileName);
                        string nama = open.FileName.Substring(0,
open.FileName.Length);
                        Microsoft.Office.Interop.Word.Application
word = new Microsoft.Office.Interop.Word.Application();
                        object miss =
System.Reflection.Missing.Value;
                        object path = nama;
                        object readOnly = false;
                        Microsoft.Office.Interop.Word.Document docus
= word.Documents.Open(ref path, ref miss, ref readOnly, ref miss, ref
miss, ref miss, ref miss, ref miss, ref miss, ref miss, ref miss, ref
miss, ref miss, ref miss, ref miss, ref miss);
                        docus.ActiveWindow.Selection.WholeStory();
```

```
                        docus.ActiveWindow.Selection.Copy();
                        IDataObject data = Clipboard.GetDataObject();
                        isifile.Text =
data.GetData(DataFormats.Text).ToString();
                        docus.Close(ref miss, ref miss, ref miss);
                        textBox6.Text =
isifile.Text.Length.ToString();
                    }
                }
            }

            catch (Exception ex) {
                MessageBox.Show(ex.ToString());
            }

        }
    }
}
```

CURRICULUM VITAE

Nama	: Rizky Suanda
Tempat, Tgl Lahir	: Pematang Kasih, 05 Februari 1994
Jenis Kelamin	: Laki-Laki
Agama	: Islam
Kewarganegaraan	: Indonesia
Status	: Belum Kawin
Alamat Sekarang	: Pematang Kasih, Kec.Pantai Cermin
	Kab.Serdang Bedagai, Sumatera Utara
Telephone	: 082273377626
Email	: rizkysuanda11@gmail.com

PENDIDIKAN

- 2000 – 2006 : SD Negeri 104274 Pematang Kasih
- 2006 – 2009 : SMP Negeri 3 Perbaungan
- 2009 – 2012 : SMA Negeri 1 Pantai Cermin
- 2012 – Sekarang : S1 Ilmu Komputer Universitas Sumatera Utara

KEMAMPUAN

- Bahasa : Indonesia
- Multimedia : Adobe Flash, Adobe Photoshop, Adobe Illustrator
- Perkantoran : Microsoft Office

PENGALAMAN ORGANISASI DAN KEPANITIAAN

- 2013 : Anggota Acara Disnatalis IMILKOM
- 2014 : Anggota Humas PMB ILKOM 2014
- 2014 : Anggota Peralatan Porseni IMILKOM
- 2015 : Anggota Seni & Olahraga IMILKOM
- 2016 : Ketua Departemen Seni & Olahraga IMILKOM

PENGALAMAN KERJA

- 2015 : Praktik Kerja Lapangan di PT. Telekomunikasi Indonseia Medan (TELKOM)
- 2016 – Sekarang : Staff TU / Operator Sekolah SD Negeri 101933 Perbaungan

SEMINAR

- 2014 : Seminar Literasi Dan Teknologi Informasi (SENARAI)

SURAT KEPUTUSAN

No. : 150 /UN5.2.1.14.2.1/TPM/2017

Tentang

SUSUNAN PERSONALIA DOSEN PEMBIMBING SKRIPSI SEORANG MAHASISWA DEPARTEMEN / PROGRAM STUDI S-1 ILMU KOMPUTER FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI (Fasilkom-TI) UNIVERSITAS SUMATERA UTARA (USU) MEDAN

KETUA DEPARTEMEN/PROGRAM STUDI S-1 ILMU KOMPUTER
Fasilkom-TI UNIVERSITAS SUMATERA UTARA MEDAN

MENIMBANG : Bahwa skripsi merupakan karya tulis ilmiah yang memiliki SKS, memiliki nilai ujian, merupakan akhir kristalisasi sikap, kemampuan dan keterampilan mahasiswa(i) dalam upaya berpikir skeptis, analitis dan kritis, serta skripsi sebagai salah satu syarat untuk mencapai penyelesaian program studi jenjang S-1 Ilmu Komputer Fasilkom-TI USU, maka dipandang perlu untuk menetapkan Dosen Pembimbing Penyusunan Skripsi mahasiswa (i) yang bersangkutan.

MENGINGAT : 1. Surat Keputusan Rektor No. 402/PT05.H/SK/Q/1992, Tgl 8 Juli 1992 Penyelesaian Teknis Pelaksanaan Skripsi di Departemen / Program Studi S-Ilmu Komputer Fasilkom-TI USU dengan BAB IV SK Kept.Rektor USU tersebut.

MEMBACA : 2. Hasil pemeriksaan berkas permohonan penyusunan skripsi mahasiswa yang bersangkutan oleh Departemen/Program Studi S-1 Ilmu Komputer Fasilkom-TI USU Tanggal 16 Januari 2017 Dan Proposal/TOR/Rancangan Skripsi yang berjudul :

Perngamanan File Teks Menggunakan Algoritma Skipjack dan Kompresi Algoritma Taboo

MEMUTUSKAN :

MENETAPKAN : Susunan Personalia Pembimbing Skripsi Seorang Mahasiswa Jurusan Departemen Program Studi S-1 Ilmu Komputer Fasilkom-TI USU Medan sebagai berikut :

1. Mahasiswa terbimbing adalah :
   Nama : Rizky Suanda
   NIM : 121401037
   Program Studi : S-1 Ilmu Komputer Fasilkom-TI USU
2. Dosen Pembimbing Penyusun Skripsi:

Dosen Pembimbing	Nama Pembimbing	N I P	Pangkat
Pertama	Dr. Poltak Sihombing, M.Kom	19620317 199103 1 001	IV/a
Kedua	Amalia, ST, MT	19781221 201404 2 001	III/b

3. Keputusan ini mulai berlaku sejak tanggal ditetapkan dan paling lama selama 6 (enam) bulan untuk dapat dilaksanakan sebaik-baiknya sesuai dengan ketentuan akan dirubah dan disempurnakan kembali apabila pada peninjauan dikemudian hari ternyata terdapat kekeliruan dalam keputusan ini.

DITETAPKAN DI : MEDAN
PADA TANGGAL : 27 SEP 2017

Ketua

Dr. Poltak Sihombing,M.Kom
NIP. 196203171991031001

Tembusan :

1. Yth. Dekan Fasilkom-TI USU
2. Yth. Dosen Pembimbing
3. Mahasiswa (i) ybs
4. Arsip