APLIKASI METODE ENKRIPSI CIPHER BLOCK DAN STREAM CIPHER MENGGUNAKAN COLLISION RESISTANT HASH FUNCTION UNTUK PENGAMANAN FILE RAHASIA

Disusun dan Diajukan untuk Memenuhi Persyaratan Ujian Akhir Memperoleh
Gelar Sarjana Komputer pada Fakultas Sains dan Teknologi
Universitas Pembangunan Panca Budi
Medan

SKRIPSI

OLEH

NAMA	: PRAPANGASTA R. DINANTAKA
NPM	: 1514370292
PROGRAM STUDI	: SISTEM KOMPUTER

FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS PEMBANGUNAN PANCA BUDI
MEDAN
2020

ABSTRAK

PRAPANGASTA RAMPUANDI DINANTAKA

APLIKASI METODE ENKRIPSI *CIPHER* BLOCK DAN STREAM CIPHER MENGGUNAKAN COLLISION RESISTANT HASH FUNCTION UNTUK PENGAMANAN FILE RAHASIA

Akhir-akhir ini, beberapa algoritma enkripsi kunci simetris telah berhasil dipecahkan. Algoritma RC4, triple DES and Blowfish telah dapat dipecahkan. Untuk meningkatkan keamanan data, maka perlu dikembangkan sebuah teknik kriptografi baru yang menggabungkan teknik *block cipher* dan *stream cipher*.

Dengan menggunakan *block cipher*, pesan akan dipecahkan menjadi blok dengan ukuran tertentu dan setiap blok akan dienkripsi dengan menggunakan kunci enkripsi. Dengan menggunakan *stream cipher*, *plaintext* akan dikombinasikan dengan sebuah deretan bit *pseudorandom cipher* dan setiap bit dari *plaintext* akan dikombinasikan dengan stream bit cipher dengan sebuah operasi XOR. Sementara itu, untuk meningkatkan keamanan dari metode enkripsi simetris, maka dapat digunakan *collision resistant hash functions* (CRHF). CRHF merupakan sebuah cara yang sangat efektif untuk menghasilkan *pseudorandom block key* yang dapat digunakan dalam proses operasi XOR dengan *plaintext*.

Perangkat lunak ini dikembangkan dengan menggunakan Microsoft Visual Basic 2010.Perangkat lunak dapat digunakan untuk mengamankan berbagai tipe file yang terdapat pada sistem operasi Windows.

Kata Kunci : kriptografi, algoritma enkripsi simetris, *block cipher*, *stream cipher*, *collision resistant hash functions*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I

PENDAHULUAN

1.1. Latar Belakang Masalah

Akhir-akhir ini, beberapa algoritma enkripsi kunci simetris telah berhasil dipecahkan. Algoritma *stream cipher* RC4 dapat dipecahkan dalam waktu kurang dari satu menit pada tahun 2007 (Tews, et. al., 2007). Pada tahun 2016, keamanan dari algoritma *block cipher* triple DES and Blowfish telah dibuktikan tidak aman dan sangat rentan terhadap penyerangan SWEET32 (Bhargavan dan Leurent, 2016).

Untuk meningkatkan keamanan dari algoritma *stream cipher* dan *block cipher* tersebut, maka kedua algoritma tersebut dapat dikombinasikan untuk menutupi kelemahannya satu sama lain. Selain itu, untuk lebih meningkatkan keamanan dari penerapan metode kriptografi tersebut, maka dapat diterapkan fungsi *hash*.

collision resistant hash functions (CRHF) merupakan sebuah cara yang sangat efektif untuk menghasilkan *pseudorandom block key* yang dapat digunakan dalam proses operasi XOR dengan *plaintext*. Teknik CRHF ini diklaim sangat efektif dalam meningkatkan keamanan dari metode enkripsi simetris. pada tahun 2017 diperkenalkan sebuah metode yang diberi nama '*A cipher Block and Stream Cipher*' yang mengkombinasikan metode enkripsi *block cipher* dan *stream cipher* serta menerapkan sebuah teknik *Collision Resistant Hash Functions* untuk

meningkatkan keamanan kuncinya. Fungsi *one way hash* yang akan digunakan adalah metode SHA-1.

Berdasarkan uraian di atas, maka penulis tertarik untuk menerapkan metode Enkripsi *cipher Block* dan *Stream Cipher* menggunakan *Collision Resistant Hash Functions* untuk mengamankan *file* rahasia dan meningkatkan keamanan dari *file* rahasia tersebut. Oleh karena itu, diperlukan sebuah algoritma enkripsi baru yang aman dari serangan. Penulis mengambil skripsi dengan judul "**Aplikasi Metode Enkripsi *Cipher Block* Dan *Stream Cipher Menggunakan Collision Resistant Hash Function* Untuk Pengamanan *File* Rahasia**".

1.2. Rumusan Masalah

Berdasarkan latar belakang pemilihan judul, maka yang menjadi permasalahan adalah

1. Bagaimana menerapkan metode enkripsi *Cipher Block* dan *Stream Cipher* menggunakan *Collision Resistant Hash Functions* untuk meningkatkan keamanan dari *file* rahasia ?

1.3. Batasan Masalah

Batasan masalah yang akan dibahas dalam skripsi ini mencakup :

1. Perangkat lunak dapat menampilkan detail perhitungan dari metode Enkripsi *Cipher Block* dan *Stream Cipher* menggunakan *Collision Resistant Hash Functions*.

2. Aplikasi akan dibuat dengan menggunakan bahasa pemrograman Microsoft Visual C# 2013.
3. Perangkat lunak hanya dapat mengamankan file berupa teks, tidak untuk gambar maupun video.

1.4. Tujuan penelitian

Tujuan penyusunan skripsi ini adalah untuk merancang suatu perangkat lunak untuk mengamankan *file* rahasia dengan menerapkan metode enkripsi *Cipher Block* dan *Stream Cipher* mengunakan *Collision Resistant Hash Functions*.

1.5. Manfaat Penelitian

Berdasarkan dari pemilihan judul, Manfaat dari penyusunan skripsi ini, adalah.

1. Mempelajari penerapan metode enkripsi *Cipher Block* dan *Stream Cipher* menggunakan *Collision Resistant Hash Functions* dalam mengamankan file rahasia.
2. Sebagai bahan referensi mahasiswa yang akan melanjutkan penelitian metode enkripsi *Cipher Block* dan *Stream Cipher*.
3. Seandainya aplikasinya diterapkan akan membantu pihak-pihak yang terkait.

BAB II

LANDASAN TEORI

2.1 Pengenalan Kriptografi

Kriptografi adalah sebuah teknik untuk menyembunyikan pesan. Tetapi pada definisi kriptografi modern adalah ilmu yang berdasarkan pada teknik matematika untuk berurusan dengan keamanan informasi yang terdiri dari Kerahasiaan, keutuhan data, dan otentikasi pesan. (Asaziduhu Gea, *et, al.* 2019).

Jika anda mengirim pesan (misalnya dalam bentuk surat) kepada orang lain, tentu anda ingin pesan yang dikirim sampai ke pihak yang dituju dengan aman. Pengertian aman di sini sangat luas. Aman bisa berarti bahwa selama pengiriman pesan tentu anda berharap pesan tersebut tidak dibaca oleh orang yang tidak berhak. Sebab, mungkin saja pesan tersebut berisi sesuatu yang rahasia sehingga jika pesan rahasia dibaca oleh pihak lawan atau pihak yang tidak berkepentingan, maka bocorlah kerahasiaan pesan yang anda kirim. Ini adalah masalah keamanan pesan yang dinamakan kerahasiaan (*confidentiality* atau *privacy*). (Rinaldi Munir, Kriptografi, 2019, 3)

Aman bisa juga berarti bahwa pesan yang dikirim sampai dengan utuh ke tangan penerima, artinya isi pesan tidak diubah atau dimanipulasi selama pengiriman oleh pihak ketiga. Di sisi penerima pesan, ia tentu ingin memastikan bahwa pesan yang ia terima adalah pesan yang masih asli, bukan pesan yang sudah ditambah-tambah atau dikurangi. Ini adalah masalah keamanan pesan yang disebut integritas data (*data integrity*). Selain itu, penerima yakin bahwa pesan

tersebut memang benar berasal dari anda, bukan dari orang lain yang menyamar seperti anda, dan anda pun yakin bahwa orang yang anda kirimi pesan adalah orang yang sesungguhnya. Ini adalah masalah keamanan pesan yang dinamakan otentikasi (*authentication*). (Rinaldi Munir, Kriptografi, 2019, 3)

Jika anda sebagai penerima pesan, anda pun tidak ingin kelak pengirim pesan membantah pernah mengirim pesan kepada anda. Ini adalah masalah keamanan yang disebut penyangkalan (*repudiation*). Zaman sekarang, banyak orang yang membantah telah mengirim atau menerima pesan. Padahal anda yakin bahwa anda memang menerima pesan dari orang tersebut. Jika pengirim membantah telah mengirim pesan, maka anda perlu membuktikan ketidakbenaran penyangkalan tersebut (*non-repudiation*). (Rinaldi Munir, Kriptografi, 2019, 4)

Keempat masalah keamanan yang disebutkan di atas, yaitu kerahasiaan, integritas data, otentikasi dan penyangkalan dapat diselesaikan dengan menggunakan kriptografi. Kriptografi tidak hanya menyediakan alat untuk keamanan pesan, tetapi juga sekumpulan teknik yang berguna.

Kriptografi (*cryptography*) berasal dari Bahasa Yunani: "*cryptos*" artinya "*secret*" (rahasia), sedangkan "*graphein*" artinya "*writing*" (tulisan). Jadi, kriptografi berarti "*secret writing*" (tulisan rahasia). Ada beberapa definisi kriptografi yang telah dikemukakan di dalam berbagai literatur, definisi lama yang dipakai di dalam buku-buku sebelum tahun 1980-an menyatakan bahwa :

1. Kriptografi adalah ilmu dan seni untuk menjaga kerahasiaan pesan dengan cara menyandikan nya ke dalam bentuk yang tidak dapat dipahami lagi maknanya yang dikemukakan oleh Meyer, 1992.

2. Kriptografi adalah ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan, integritas data, serta otentikasi yang dikemukakan oleh Menez, 1996. (Rinaldi Munir, Kriptografi, 2019, 5)

2.1.1 Terminologi

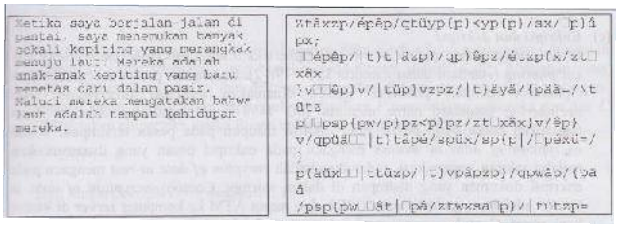
Di dalam kriptografi kita akan sering menemukan berbagai istilah atau terminologi. Beberapa istilah yang penting untuk diketahui diberikan di bawah ini.

1. Pesan, Plainteks dan Cipherteks.

Pesan (*message*) adalah sebuah data atau informasi yang dapat dibaca, diperepsi dan dimengerti artinya. Nama lain untuk pesan adalah plainteks (*plaintext*) atau teks-jelas (*cleartext*). Pesan dapat berupa data atau informasi yang dikirim (melalui kurir, saluran telekomunikasi) atau yang disimpan di dalam media perekaman (kertas, *storage*). Pesan yang tersimpan tidak hanya berupa teks, tetapi juga dapat berbentuk citra (*image*), suara/bunyi (*audio*) dan video atau berkas biner lainnya.

Agar pesan tidak dapat dimengerti maknanya oleh pihak lain, maka pesan perlu disandikan ke bentuk lain yang tidak dapat dipahami. Bentuk pesan yang tersandi disebut cipherteks (*ciphertext*) atau kriptogram (*cryptogram*). Cipherteks harus dapat ditransformasikan kembali menjadi plainteks semula agar pesan yang dierima bisa dibaca. Gambar 2.1 dan 2.2 memperlihatkan contoh dari dua buah plainteks, masing-masing berupa teks dan gambar, serta cipherteks yang berkoresponden. (Rinaldi Munir, Kriptografi, 2019, 6)

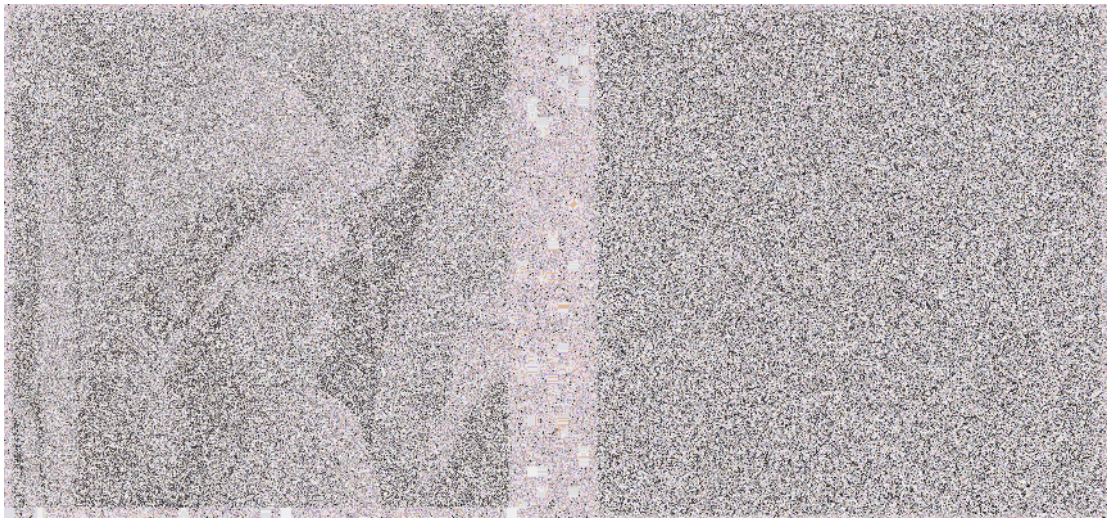
Salah satu mekanisme dalam meningkatkan suatu keamanan yaitu dengan menggunakan teknologi enkripsi. Data-data yang dikirimkan akan diubah sedemikian rupa sehingga tidak mudah di serang atau disadap. Jadi enkripsi adalah proses yang dilakukan untuk mengamankan sebuah pesan (yang disebut plaintext) menjadi pesan yang tersembunyi (disebut ciphertext) adalah enkripsi. Ciphertext yaitu pesan yang sudah tidak dapat dibaca dengan mudah. Terminologi yang lebih tepat digunakan adalah "encipher" proses sebaliknya, untuk mengubah ciphertext menjadi plaintext, disebut dekripsi. Terminologi yang lebih tepat untuk proses ini adalah "decipher". (Rifkie Primartha, 2011)



(a) Plainteks (b) Cipherteks

Gambar 2.1 Plainteks berupa Teks dan Cipherteksnya

Sumber : Rinaldi Munir, Kriptografi, 2019, 7



(a) Plainteks (b) Cipherteks

Gambar 2.2 Plainteks berupa Gambar dan Cipherteksnya

Sumber : Rinaldi Munir, Kriptografi, 2019, 7

2. Pengirim dan penerima

Komunikasi data melibatkan pertukaran pesan antara dua entitas. Pengirim (*sender*) adalah entitas yang mengirim pesan kepada entitas lainnya. Penerima (*receiver*) adalah entitas yang menerima pesan. Entitas di sini dapat berupa orang, mesin (komputer), kartu kredit, dan sebagainya. Jadi, orang bisa bertukar pesan dengan orang lainnya (contoh: Alice berkomunikasi dengan Bob), sedangkan di dalam jaringan komputer, mesin (komputer) berkomunikasi dengan mesin (contoh: mesin ATM berkomunikasi dengan komputer *server* di bank).

Pengirim tentu menginginkan pesan dapat dikirim secara aman, yaitu ia yakin bahwa pihak lain tidak dapat membaca isi pesan yang ia kirim. Solusinya adalah dengan cara menyandikan pesan menjadi cipherteks. (Rinaldi Munir, Kriptografi, 2019, 9)

3. Enkripsi dan dekripsi

Proses menyandikan plainteks menjadi cipherteks disebut enkripsi (*encryption*) atau *enciphering*. Sedangkan, proses mengembalikan cipherteks menjadi plainteks semula dinamakan dekripsi (*decryption*) atau *deciphering*. Enkripsi dan dekripsi dapat diterapkan baik pada pesan yang dikirim maupun pada pesan tersimpan. Istilah *encryption of data in motion* mengacu pada enkripsi pesan yang ditransmisikan melalui saluran komunikasi, sedangkan istilah *encryption of data at-rest* mengacu pada enkripsi dokumen yang disimpan di dalam *storage*. Contoh *encryption of data in motion* adalah pengiriman nomor PIN dari mesin ATM ke komputer *server* di kantor bank pusat. Contoh *encryption of data at-rest* adalah enkripsi *file* basis data di dalam *hard disk*. Gambar 2.3 memperlihatkan enkripsi *file* basis data, di mana enkripsi hanya dilakukan terhadap *field-field* tertentu saja (Nama, Tinggi dan Berat). (Rinaldi Munir, Kriptografi, 2019, 10)

Plainteks (siswa.dbf):

NIM	Nama	Tinggi	Berat
000001	Elin Jamilah	160	50
000002	Fariz RM	157	49
000003	Taufik Hidayat	176	65
000004	Siti Nurhaliza	172	67
000005	Oma Irama	171	60
000006	Aziz Burhan	181	54
000007	Santi Nursanti	167	59
000008	Cut Yanti	169	61
000009	Ina Sabarina	171	62

Cipherteks (siswa2.dbf):

NIM	Nama	Tinggi	Berat
000001	tüp`vzpz/|t}äyä/{aâ	|äzp}	épêp
000002	□□|t`tâpé/spux/sp	péxü=	ztwxsä□
000003	□□ât|□pê/ztwxsa□p}/|	}/|cü	spüx/
000004	épêp/|t}t|äzp}/qpêpz	qp}épz	wxsä
000005	étzp{x/zt□xâx}v□□êp}	paâ/psp	étzp{
000006	spüx/sp{p|/□péxü=/]	xâx}v	ttüzp/|
000007	Ztäxzp/épêp/qtüypp}<	äzp}	}äyä/{
000008	qpwàp/{päâ/psp{pw□	ztwxs	xâx}v□□
000009	}t|äzp}/qp}êpz/ép{	qp}êp	äzp}/qp

Gambar 2.3 Enkripsi Data tertentu di dalam Arsip Basisdata

Sumber : Rinaldi Munir, Kriptografi, 2019, 10

4. *Cipher* dan kunci

Algoritma kriptografi disebut juga *cipher* yaitu aturan untuk *enciphering* dan *deciphering*, atau fungsi matematika yang digunakan untuk enkripsi dan dekripsi. Beberapa *cipher* memerlukan algoritma yang berbeda untuk *enciphering* dan *deciphering*.

Konsep matematis yang mendasari algoritma kriptografi adalah relasi antara dua buah himpunan yaitu himpunan yang berisi elemen-elemen plainteks dan himpunan yang berisi cipherteks. Enkripsi dan dekripsi merupakan fungsi yang memetakan elemen-elemen antara kedua himpunan tersebut. Misalkan P menyatakan plainteks dan C menyatakan cipherteks, maka fungsi enkripsi E memetakan P ke C,

$$E(P) = C$$

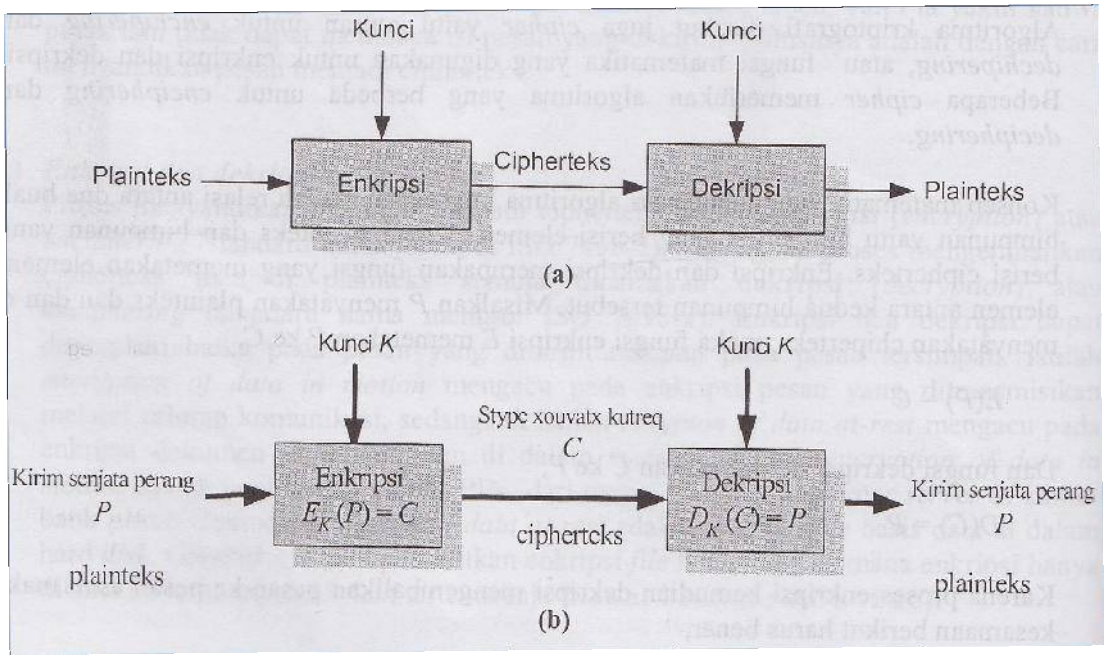
Dan fungsi dekripsi D memetakan C ke P,

$$D(C) = P$$

Karena proses enkripsi kemudian dekripsi mengembalikan pesan ke pesan asal, maka kesamaan berikut harus benar,

$$D(E(P))= P$$

(Rinaldi Munir, Kriptografi, 2019, 11)



Gambar 2.4(a) Skema Enkripsi dan Dekripsi,

(b) Contoh Ilustrasi Enkripsi dan Dekripsi Pesan

Sumber : Rinaldi Munir, Kriptografi, 2019, 10

2.2 Aplikasi

Pengertian aplikasi adalah suatu bagian dari perangkat lunak yang dapat digunakan untuk menyeleaikan masalah-masalah khusus yang akan dihadapi oleh pengguna atau *user* dengan menggunakan kemampuan komputer. (Fergian Listianto, et, al. 2017).

Secara istilah pengertian dari aplikasi adalah suatu program yang siap untuk digunakan yang sengaja dibuat untuk melaksanakan atau melakukan suatu fugsi bagi pengguna jasa aplikasi serta aplikasi lain yang dapat digunakan oleh suatu sasaran yang akan dituju. Menurut dari kamus *computer* eksekutif, aplikasi mempunyai arti yaitu pemecahan sebuah masalah yang menggunakan salah satu teknik pemrosesan data aplikasi yang biasanya berpacu pada sebuah komputansi yang diinginkan atau diharapkan maupun pemrosesan data yang diharapkan. (Andi Juansyah, 2015).

Adapula pengertian lain dari aplikasi yang dikutip dari Jogianto, aplikasi adalah software yang dibuat oleh suatu perusahaan komputer untuk mengerjakan tugas-tugas tertentu, misalnya *Microsoft Word*, *Microsoft Excel*, *Microoft Office.* Yang menggabungkan suatu aplikasi pengolah kataa, lembar kerja, serta beberapa aplikasi lainnya. (Meti Atiroh, et, al. 2014).

2.3 Keamanan

Keamanan adalah keadaan bebas dari sebuah bahaya atau serangan, Istilah ini dapat digunakan dengan hubungan yang ditujukan kepada kejahatan, dan segala sesuatu bentuk kecelakaan. Keamanan merupakan topik yang sangat luas

termasuk keamanan nasional terhadap teroris, dan keamanan terhadap jaringan komputer yang dilakukan oleh cracker. (Ruri zain, 2012)

Keamanan telah menjadi aspek yang sangat penting dari suatu sistem informasi. Sebuah informasi umunya hanya ditujukan bagi segolongan tertentu. Oleh karena itu sangat penting untuk mencegahnya jatuh kepada pihak-pihak lain yang tidak berkepentingan. untuk melaksanakan tujuan tersebutlah dirancang suatu sistem keamanan yang berfungsi melindungi sistem informasi. (Henri Pandiangan, Salomo Sijabat, 2016).

Salah satu upaya pengamanan sistem informasi yang dapat dilakukan adalah kriptografi, kriptografi sesungguhnya merupakan studi terhadap teknik matematis yang terkait dengan aspek keamanan suatu sistem informasi, antara lain seperti kerahasiaan (confidentiality), integritas data (data integriry), otentikasi (authentication), dan ketiadaan penyangkalan (non-repudation). keempat aspek tersebut merupakan tujuan dari suatu sistem keamanan kriptografi. (Henri Pandiangan, Salomo Sijabat, 2016).

2.4 File

Pada saat kita memakai komputer, maka kita pasti akan berhubungan dengan kumpulan data. Data yang dikumpulkan menjadi satu, dan membentuk sebuah informasi, kemudian informasi-informasi tersebut akan disimpan oleh komputer ke dalam suatu wadah yang disebut dengan sebuah *file*. Secara umum format *file* teks dibagi menjadi dua bagian yaitu. Teks sederhana (*plaintext*) dan teks terformat (*formatted text*). (Sastra ramadanu, 2019).

Adapun pengertian lain dari *file* yaitu, *file* merupakan sekumpulan *record* atau rekaman yang saling berhubungan dan diperlakukan sebagai satuan. Beberapa istilah yang penting dalam *file* yakni. *Character* (karakter) dasar utama dalam pembentukan data yang berupa huruf, angka, dan simbol khusus, *Field* yaitu sekumpulan karakter yang membentuk satu untaian arti, *Record* (rekaman) kumpulan dari *field* ketika bekerja dengan sistem aplikasi, program akan menyimpan data yang kita buat kedalam suatu *file*. Struktur *file* data yang akan dibentuk akan bervariasi sesuai program aplikasi yang sedang dipergunakan. (Sudi Suryadi, 2014).

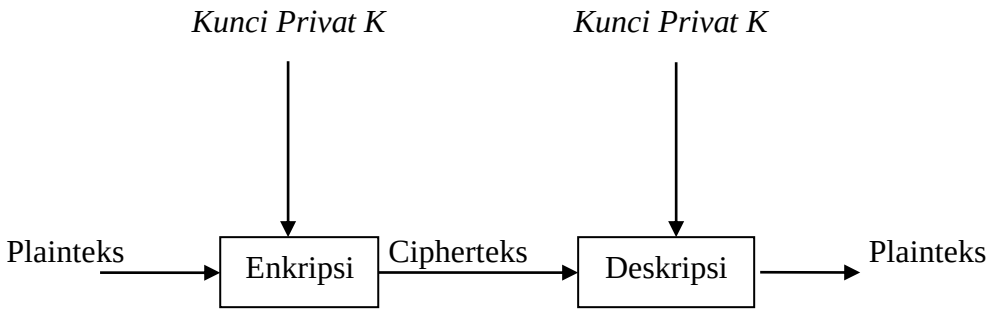
2.5 Sistem Kriptografi

Kriptografi membentuk sebuah sistem yang dinamakan sistem kriptografi. Sistem kriptografi (*cryptosystem*) adalah kumpulan yang terdiri dari algoritma kriptografi, semua plainteks dan cipherteks yang mungkin dan kunci. Di dalam sistem kriptografi, *cipher* hanyalah salah satu komponen saja. (Rinaldi Munir, Kriptografi, 2019, 13)

Berdasarkan kunci yang digunakan untuk enkripsi dan dekripsi, kriptografi dapat dibedakan menjadi kriptografi kunci-simetri (*symmetric-key cryptography*) dan kriptografi kunci-nirsimetri (*asymmetric-key cryptography*). (Rinaldi Munir, Kriptografi, 2019, 14)

2.5.1 Kriptografi Kunci Simetris

Algoritma simetris adalah algoritma yang menggunakan kunci yang sama untuk proses enkripsi dan proses deskripsi. algoritma kriptografi simteris dibagi menjadi 2 kategori yaitu algoritma aliran (*Sream Cipher*) dan algoritma block (*Block Cipher*), dimana pada algoritma aliran, proses penyandiannya akan beriorientasi pada satu bit data. sedangkan pada algoritma block, proses penyandian beriorientasi pada sekumpulan bit data per block atau karakter. (Irham Arrijal, *et, al,* 2016).



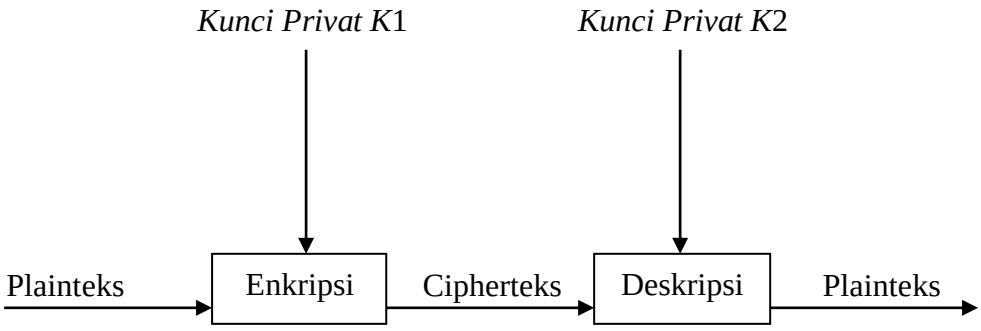
Gambar 2.5 Skema Algoritma kunci simetris.

Sumber. (Novi Natahsia, Anang Wicaksono, 2011).

2.5.2 Kriptografi Kunci Asimetris

Pada kriptografi ini, kunci untuk enkripsi tidak rahasia dan dapat diketahui oleh siaapapun, sementara kunci untuk deksipsi hanya diketahui oleh penerima pesan. setiap orang yang berkomunikasi mempunyai sepasang kunci, yaitu kunci privat dan kunci publik. pengirim mengenkripsi pesan dengan menggunakan kunci publik si penerima pesan. Hanya penerima pesan yang dapat mengenkripsi

pesan karena hanya ia yang mengetahui kunci privatnya sendiri. (Novi Natahsia, Anang Wicaksono, 2011).



Gambar 2.6 Skema Algoritma Kunci Asimetris

Sumber : (Novi Natahsia, Anang Wicaksono, 2011).

2.6 Landasan Matematika Kriptografi

2.6.1 Teori Bilangan

Teori bilangan (*number theory*) adalah teori yang mendasar dalam memahami kriptografi, khususnya sistem kriptografi kunci-publik. Bilangan yang dimaksudkan di sini hanyalah bilangan bulat (*integer*). Bilangan bulat adalah bilangan yang tidak mempunyai pecahan desimal, misalnya 8, 21, 8675, -34, 0. Berlawanan dengan bilangan bulat adalah bilangan riil yang mempunyai titik desimal, seperti 8.0, 34.25, 0.02. (Rinaldi Munir, Kriptografi, 2019, 59)

2.6.2 Algoritma *Euclidean*

Algoritma *Euclidean* adalah algoritma untuk mencari PBB dari dua buah bilangan bulat. *Euclid*, penemu algoritma *Euclidean*, adalah seorang matematikawan Yunani yang menuliskan algoritmanya tersebut dalam bukunya

yang terkenal, *Element*. Diberikan dua buah bilangan bulat tak-negatif a dan b ($a \geq b$). Algoritma Euclidean berikut mencari pembagi bersama terbesar dari m dan n.

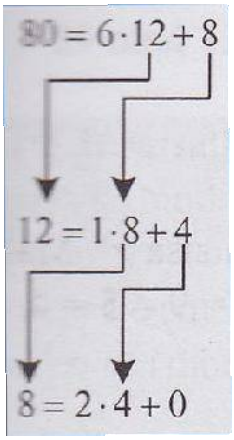
Algoritma Euclidean:

1. Jika $b = 0$, maka a adalah PBB(a, b); stop.

   Kalau tidak (yaitu $b \neq 0$) lanjutkan ke langkah 2.
2. Bagilah a dengan b dan misalkan r adalah sisanya.
3. Ganti nilai a dengan nilai b dan nilai b dengan nilai r, lalu ulang kembali ke langkah 1.

Contoh:

Misalkan a = 80, b = 12 dan dipenuhi syarat $a \geq b$, maka PBB(80, 12) dihitung dengan algoritma Euclidean sebagai berikut:



Sisa pembagian terakhir sebelum 0 adalah 4, maka PBB(80, 12) = 4.

(Rinaldi Munir, Kriptografi, 2019, 61)

2.6.3 Relatif Prima

Dua buah bilangan bulat a dan b dikatakan relatif prima jika PBB(a, b) = 1. dalam bentuk kombinasi lanjar, Jika a dan b relatif prima, maka terdapat bilangan bulat m dan n sedemikian sehingga:

$$ma + nb = 1$$

Contoh :

- Bilangan 20 dan 3 adalah relatif prima karena PBB(20,3) = 1, atau dapat ditulis:

  $$2 . 20 + (–13) . 3 = 1$$

  dengan m = 2 dan n = –13. Tetapi 20 dan 5 tidak relatif prima karena PBB(20, 5) = 5 ≠ 1 sehingga 20 dan 5 tidak dapat dinyatakan dalam m . 20 + n . 5 = 1.

(Rinaldi Munir, Kriptografi, 2019, 63)

2.6.4 Aritmetika Modulo

Misalkan a adalah bilangan bulat dan m adalah bilangan bulat > 0. Operasi a mod m (dibaca ”a modulo m”) memberikan sisa apabila a dibagi dengan m. Bilangan m disebut modulus atau modulo, dan hasil aritmetika modulo m terletak di dalam himpunan {0, 1, 2, ..., m – 1}. Notasi : a mod m = r sedemikian sehingga a = mq + r, dengan 0 ≤ r < m.

Contoh:

Beberapa contoh operasi dengan operator modulo:

a. 23 mod 5 = 3 (23 = 5 . 4 + 3)

b. 27 mod 3 = 0 (27 = 3 . 9 + 0)

c. 6 mod 8 = 6 (6 = 8 . 0 + 6)

d. 0 mod 12 = 0 (0 = 12 . 0 + 0)

e. -41 mod 9 = 4 (-41 = 9 . (-5) + 4)

f. -39 mod 13 = 0 (-39 = 13 . (-3) + 0)

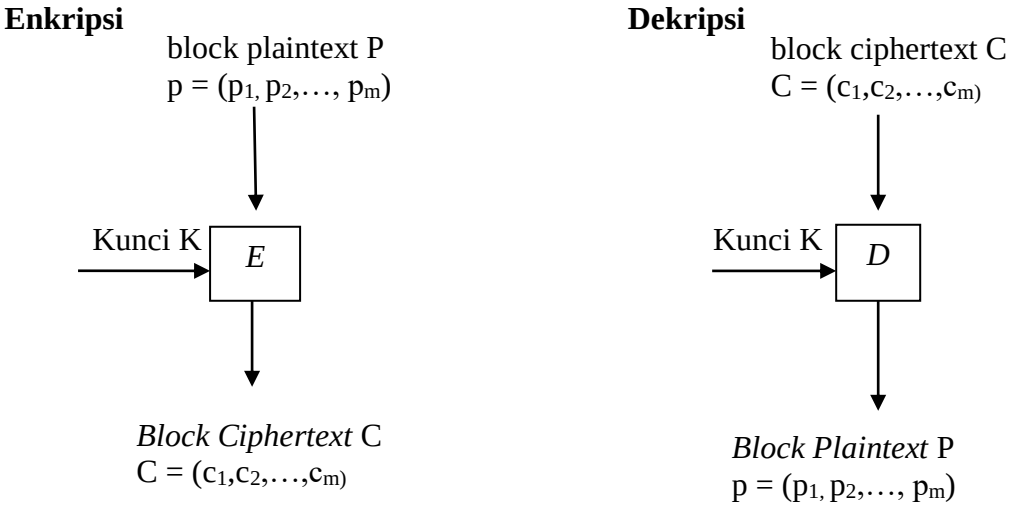
Penjelasan (e): Karena a negatif, bagi |a| dengan m mendapatkan sisa r'. Maka a mod m = m – r' bila r' ≠ 0. Jadi |-41| mod 9 = 5, sehingga -41 mod 9 = 9 – 5 = 4. (Rinaldi Munir, Kriptografi, 2019, 63)

2.7 *Cipher Block*

Kriptografi cipher block menerima masukan dalam bentuk sekumpulan blok bit dengan panjang tertentu. Dengan algoritma ini, blok plainteks yang sama akan di enkripsi menjadi blok cipherteks yang sama bila menggunakan kunci yang sama juga. (Paulus Irawan, 2015).

Adapun pengertian lain dari *block cipher* yaitu mengkodekan data dengan cara membagi *plaintext* menjadi per block dengan ukuran yang sama dan tetap. Kemudian setiap bloknya dienkripsi atau didekripsi sekaligus. (Ruri Zain, 2012).

Block cipher adalah bentuk algoritma sandi yang akan membagi text atau karakter yang akan dikirimkan dengan ukuran tertentu (disebut blok), dengan panjang kunci enkripsi dan setiap blok akan dienkripsi dengan kunci yang sama. Pada umumnya, *block cipher* memproses *text* atau karakter dengan blok relatif panjang lebih dari 64 bit, untuk mempersulit penggunaan pola-pola serangan yang ada untuk membongkar kunci. (Nuniek Fahriani, *et. al*. 2016).



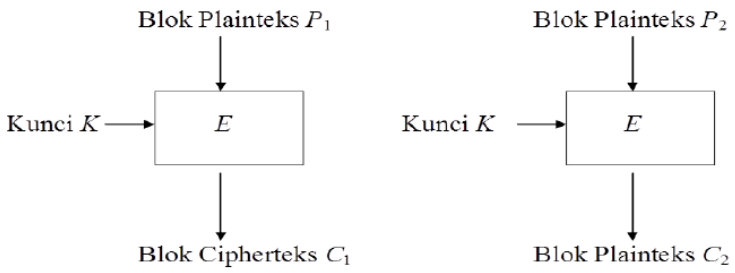
Gambar 2.7 Skema Enkripsi *Block Cipher*

Sumber : (Nuniek Fahriani, *et. al*. 2016).

2.7.1 Empat Macam Mode Operasi *Block Cipher*

1. *Electronic Code Book* (ECB)

Setiap blok *plaintext* P_1, dieenkripsi secara individual dan independen menjadi blok *ciphertext* C_1 mainng-maisng blok *plaintext* dan *ciphertext* ke-*i* (Nuniek Fahriani, *et. al*. 2016).

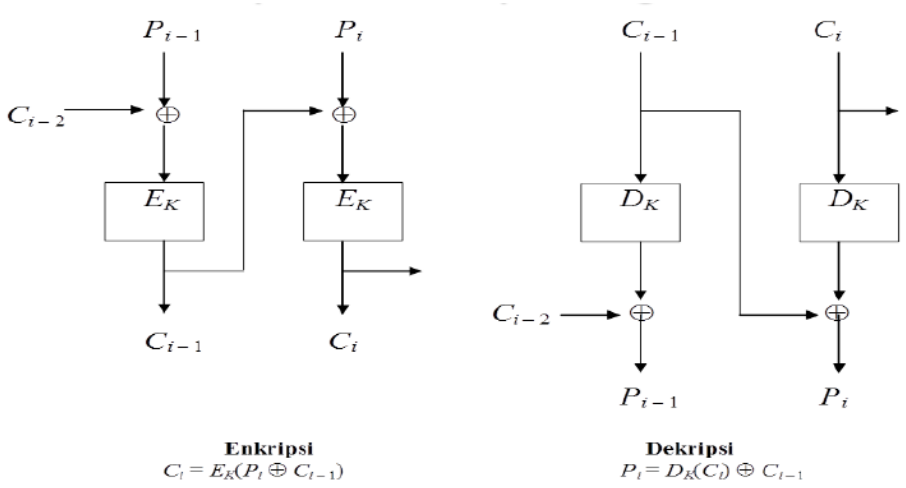


Gambar 2.8 Skema Enkripsi Dan Dekripsi Mode ECB

Sumber : (Nuniek Fahriani, *et. al*. 2016).

2. *Cipher Block Chaining* (CBC)

Tujuan dari CBC adalah membuat ketergantungan antar blok. Dari setiap blok *ciphertext* bergantung tidak hanya pada blok plaintextnya tetapi juga pada seluruh blok *plaintext* sebelumnya. Hasil enkripssi blok sebelumnya diumpan balik kedalam enkripsi blok yang *current*. (Nuniek Fahriani, *et. al*. 2016).

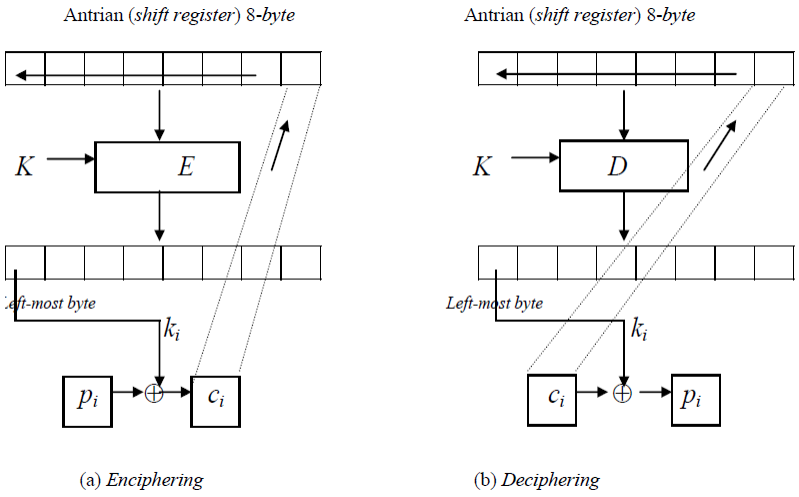


Gambar 2.9 Skema Enkripsi Dan Dekripsi Mode CBC

Sumber : (Nuniek Fahriani, *et. al*. 2016).

3. *Cipher Feedback* (CFB)

Pada mode CFB data dienkripsi dalam unit yang lebih kecil daripada ukuran blok, unit yang akan dienkripsi dapat berupa bit per bit, 2 bit, 3 bit dan seterunya. Bila unit yang akan dienkripsikan satu karakter setiap kalinya, maka mode CFB nya disebut CFB 8 bit. Secara umum CFB n-bit mengenkripikan *plaintext* sebanyak n bit setiap kalinya, yang mana n lebih kecil dari m (m = ukuran blok). (Nuniek Fahriani, *et. al*. 2016).



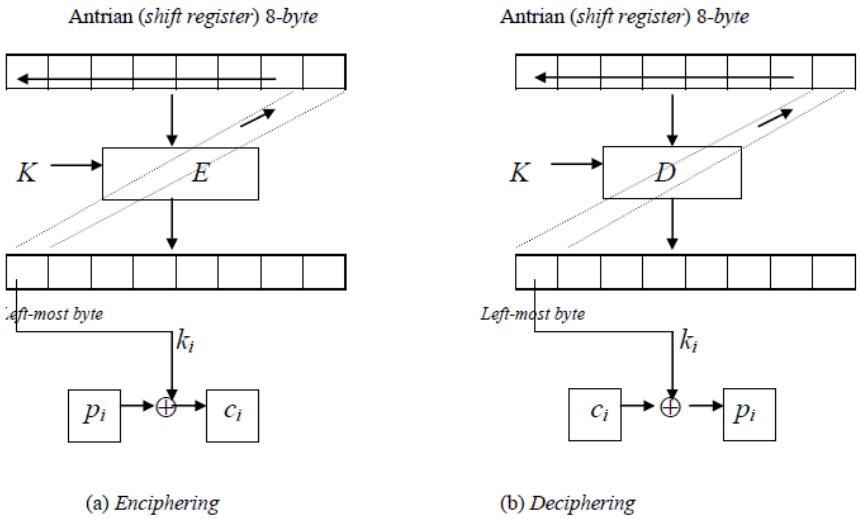
Gambar 2.10 Skema Mode CFB Pada 8 Bit

Sumber : (Nuniek Fahriani, *et. al*. 2016).

4. Output Feedback (OFB)

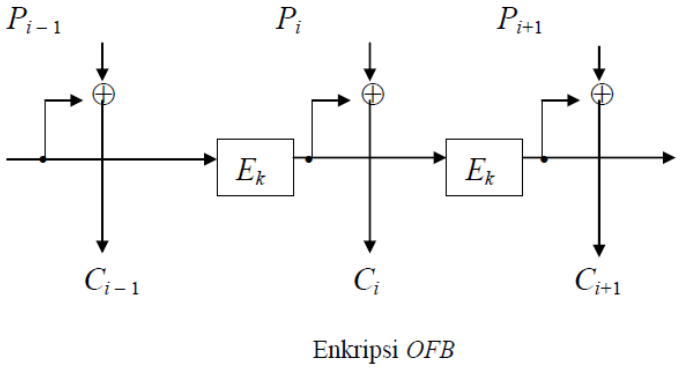
Pada mode OFB ini data dienkripsikan dalam unit yang lebih kecil daripada ukuran blok, unit yang dienkripsikan dapat berupa bit per bit, 2 bit, 3 bit dan seterusnya. Bila unit yang dienkripikan satu karakter setiap kalinya, maka mode OFB nya disebut OFB 8 bit. secara umum OFBB n-bit mengenkripsi *plaintext* sebanyak n bit setiap kalinya, yang mana n lebih kecil dari m (m = ukuran blok). mode OFB ini memiliki sebuah antrian (*queue*) yang berukuran sama dengan ukuran blok masukan yaitu antrian di isi dengan *initialization vektoc*, dan enkripsi antrian kunci K, n bit paling kiri dari hasil enkripsi dimasukkan kedalam antrian. dan m – n bit lainnya didalam antrian digeser kekiri menggantikan n bit pertama yang sudah digunakan. n bit paling kiri dari hasil enkripsi juga berlaku sebagai

keystream (K_i) yang kemudian di XOR kan dengan bit pertama dari *ciphertext.* (Nuniek Fahriani, *et. al*. 2016).



Gambar 2.11 Skema Mode OFB Pada Bit

Sumber : (Nuniek Fahriani, *et. al*. 2016).



Gambar 2.12 Mode *Enkripsi* OFB n-bit Untuk Blok n-Bit

Sumber : (Nuniek Fahriani, *et. al*. 2016).

2.8 *Stream Cipher*

Kriptografi *stream cipher* atau aliran menerima parameter masukan dalam bentuk aliran bit. proses enkripsi dan dekripsinya dilakukan pada aliran bit dengan memproses bit satu per satu. Secara umum *stream cipher* membangkitkan aliran kunci dari kunci yang dimasukkan oleh pengguna. Ketika aliran kunci telah dibangkitkan, maka proses enkripsi dan dekripsi dilakukan melalui operasi XOR antara bit aliran kunci dengan bit *plaintext* atau *ciphertext*. Keamanan kriptografi *stream cipher* sangat bergantung pada proses pembangkitan aliran kunci, aliran kunci yang dihasilkan bernilai nol pada keseluruhan bitnya, maka *cipher teks* yang dihasilkan akan sama seperti *plainteks*, yang berakibat pada proses enkripsi menjadi sia-sia. semakin acak keluaran yang dihasilkan dari proses pembangkit aliran kunci, maka cipher text yang dihasilkan juga akan semakin sulit dipecahkan. (Paulus Irawan, 2015).

Adapun pengertian lain dari stream cipher adalah suatu sistem yang dimana setiap proses enkripsi dan dekripsinya dilakukan dengan cara bit perbitnya. pada sistem ini aliran bit kuncinya akan dihasilkan oleh suatu pembangkit bit acak. (Ruri Zain, 2012).

Stream cipher merupakan cipher yang hampir sama dengan Caesar cipher (cipher yang menggeser urutan alphabet sehingga urutan afabetnya berubah), tetapi cipher ini mempunyai kunci yang unik, yaitu menggunakan karakter sebelumnya sebagai kunci.

fungsi matematika nya terdiri dari :

C = (**P+K**) *mod n*

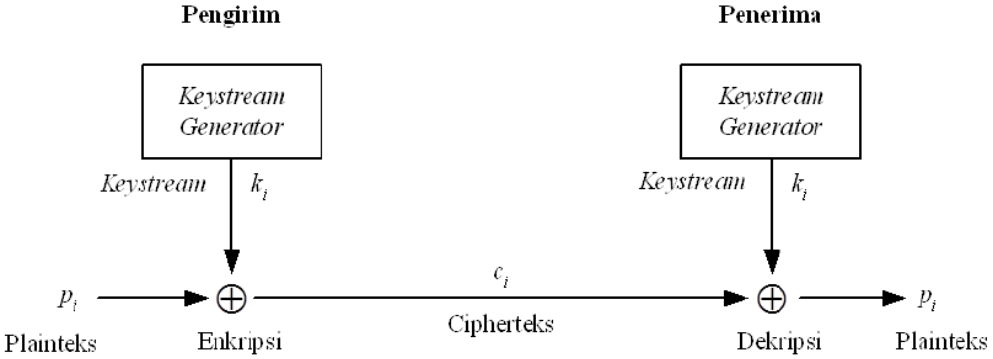
P = (**C+K**) *mod n*

C = *Plaintex*t **K** = kunci

N = Jumlah Karakter

(Wiwiek Nuryanti, Indra Yatini, 2013)

Stream cipher adalah algoritma sandi yang mengenkripsikan data persatuan data, seperti bit, byte, nible atau perlima bit (saat data yang dienkripsi berupa data boundout). Setiap mengenkripsi satu kesatuan data digunakan kunci yang merupakan hasil pembangkitan dari kunci sebelumnya. (Nuniek Fahriani, *et. al*. 2016).



Gambar 2.13 Skema *Stream Cipher*

Sumber : (Nuniek Fahriani, *et. al*. 2016).

Stream cipher adalah jenis algoritma enkripsi simetri. *Stream ciphers* dapat dibuat sangat cepat sekali, jauh lebih cepat dibandingkan dengan algoritma *block cipher* yang manapun. Sementara algoritma *block cipher* secara umum digunakan untuk unit *plaintext* yang besar sedangkan *stream cipher* digunakan untuk blok

data yang lebih kecil, biasanya ukuran bit. Proses enkripsi terhadap *plaintext* tertentu dengan algoritma *block cipher* akan menghasilkan *ciphertext* yang sama jika kunci yang sama digunakan. Dengan *stream cipher*, transformasi dari unit *plaintext* yang lebih kecil ini berbeda antara satu dengan lainnya, tergantung pada kapan unit tersebut ditemukan selama proses enkripsi.

Satu *stream cipher* menghasilkan apa yang disebut suatu *keystream* (suatu barisan bit yang digunakan sebagai kunci). Proses enkripsi dicapai dengan menggabungkan *keystream* dengan *plaintext* biasanya dengan operasi bitwise XOR. Pembentukan *keystream* dapat dibuat independen terhadap *plaintext* dan *ciphertext*, menghasilkan apa disebut dengan *synchronous stream cipher*, atau dapat dibuat tergantung pada data dan enkripsinya, dalam hal mana *stream cipher* disebut sebagai *self-synchronizin*g. Kebanyakan bentuk *stream cipher* adalah *synchronous stream cipher*s.

Sampai saat ini belum ada *stream cipher* sebagai standard secara *de facto*. Metode *stream cipher* yang umum digunakan adalah RC4. Satu hal yang menarik bahwa mode operasi tertentu dari suatu *block cipher* dapat men-transformasikan secara efektif hasil operasi tersebut ke dalam satu *keystream generator* dan dalam hal ini, *block cipher* apa saja dapat digunakan sebagai suatu *stream cipher*; seperti dalam DES, CFB atau OFB. Akan tetapi, *stream ciphers* dengan desain khusus biasanya jauh lebih cepat.

2.9 Collision Resistant Hash Function (Fungsi hash SHA-1)

SHA dikembangkan oleh *National Institute Of Standards And Technology (NIST)* dan dipublikasikan sebagai *Federal Information Processing Standart (FIPS 180)* pada tahun 1993. *Secure Hash Standard (SHS)* mempesifikasikan SHA-1 untuk menghitung nilai hash dari sebuah pesan atau file. SHA-1 memiliki panjang pesan maksimal 2^{64} bit dan memiliki keluaran sebesar 160 bit yang dinamakan *message digest* atau *hash code. message digest* tersebut dapat digunakan sebagai masukan untuk *digital signature algorithm* (*DSA),* yang digunakan untuk menghasilkan s*ignature* untuk memverifikasi pesan tersebut.(Firlhi Kurniawan, *et, al*, 2017).

Metode yang diperkenalkan oleh Marcio Ricardo Rosemberg, Daniel Schwabe dan Marcus Poggi pada tahun 2017 ini dapat dirincikan sebagai berikut:

1. Proses Enkripsi
    a. Bangkitkan sebuah kunci *share* K antara pengirim dan penerima. Kunci ini dapat memiliki panjang maksimal setengah dari ukuran fungsi *hash* H.
    b. Pecahkan pesan M menjadi b buah blok dengan ukuran sebesar panjang H, sedemikian sehingga $M = m_1 \parallel m_2 \parallel ... \parallel m_n$, dimana $\parallel$ berarti penggabungan *string*.
    c. Untuk setiap blok, bangkitkan bilangan acak semu sepanjang 32 bit integer (S_i).
    d. Hasilkan blok *ciphertext* dengan menggunakan rumusan berikut:

$$c_i = S_i \parallel m_i \oplus H(K, S_i, i)$$

e. Hitung MAC (*Message Authentication Code*) = H(K, M, S_{MAC}) dan gabungkan ke dalam *ciphertext* sehingga pesan ciphertext yang dihasilkan adalah:

$$C = S_{MAC} \parallel MAC \parallel c_1 \parallel c_2 \parallel ... \parallel c_n$$

2. Proses Dekripsi

a. Pecahkan S_{MAC} dan MAC dari pesan *ciphertext*.

b. Panjang blok *ciphertext* adalah *l* = ukuran dari H + 32 bit.

c. Blok pesan asli diperoleh dengan menggunakan rumusan berikut:

$$m_i = \Phi_i \oplus H(K, S_i, i)$$

d. Susun ulang pesan $M = m_1 \parallel m_2 \parallel ... \parallel m_n$

e. Hitung MAC pesan MAC = H(K, M, S_{MAC})

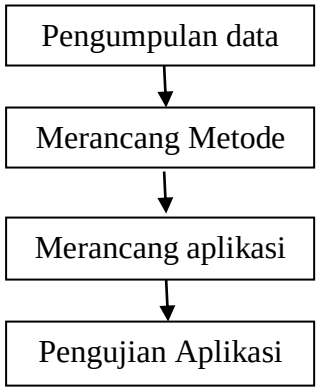
f. Cek apakah data MAC yang diterima sama dengan MAC yang dihitung. Jika sama, maka pesan yang diterima otentik dan tidak mengalami perubahan. (Marcio Rosemberg, *et, al*, 2017)

BAB III

METODE PENELITIAN

3.1 Tahapan Penelitian

Adapun tahapan dan langkah-langkah pengembangan perangkat lunak ini terdiri dari beberapa proses antara lain seperti gambar dibawah ini.



Gambar 3.1 Rancangan tahapan penelitian

1. Pengumpulan Data

Di tahap pertama, penulis mengumpulkan bahan-bahan yang diperlukan dalam penyusunan skripsi. Bahan tersebut dikumpulkan dari buku dan sumber-sumber lainnya seperti jurnal resmi di internet.

2. Merancang Metode

Tahap berikutnya ialah menganalisis dalam merancang kebutuhan-kebutuhan sistem. Sekali lagi, perangkat dan teknik-teknik tertentu akan membantu penganalisis menentukan kebutuhan. Perangkat yang dimaksud ialah penggunaan diagram alir data untuk menyusun daftar *input*, proses, dan *output* fungsi bisnis dalam bentuk grafik terstruktur.

3. Merancang Aplikasi

Dalam tahap ini dari siklus hidup pengembangan sistem, penganalisis bekerja bersama-sama dengan pemrogram untuk mengembangkan suatu perangkat lunak awal yang diperlukan. Teknik terstruktur yang digunakan untuk merancang dan mendokumentasikan perangkat lunak meliputi adalah *flowchart*. Penganalisis sistem menggunakan perangkat ini untuk menjabarkan apa yang perlu diprogram.

Pada tahap ini dilakukan perancangan tampilan antarmuka pada sistem (*user interface*) dengan aplikasi Microsoft Visual Basic.NET. Perancangan alur kerja sistem (*flow chart*) dan *use case diagram*akan dilakukan dengan bantuan perangkat lunak seperti Microsoft Visio 2007.

4. Pengujian

Sebelum sistem informasi dapat digunakan, maka harus dilakukan pengujian terlebih dulu. Akan bisa menghemat biaya bila dapat menangkap adanya masalah sebelum sistem tersebut ditetapkan. Sebagian pengujian dilakukan oleh pemrogram sendiri, dan lainnya dilakukan oleh penganalisis sistem.

3.2 Metode Pengumpulan Data

Di tahap pertama, penulis mengumpulkan bahan-bahan yang diperlukan dalam penyusunan skripsi. Bahan tersebut dikumpulkan dari buku dan sumber-sumber lainnya di *internet*.

Proses dimulai dengan mengumpulkan data-data yang diperlukan dalam penelitian. Adapun metode pengumpulan data dalam penelitian dilakukan melalui:

1. Penelitian Kepustakaan (*library research*), dimana penulis mengumpulkan data-data melalui berbagai referensi seperti *internet* dan buku-buku yang relevan yang berhubungan dengan metode enkripsi *Hybrid Block* dan *Stream Cipher* berdasarkan pada *Collision Resistant Hash Functions*.
2. Observasi, yaitu penulis akan mengamati dan mempelajari contoh pencarian solusi dari metode enkripsi *Hybrid Block* dan *Stream Cipher* berdasarkan pada *Collision Resistant Hash Functions* yang diperoleh dari *internet* ataupun buku teks.

3.3 Analisis Sistem Sedang Berjalan

Dalam tahapan ini penulis akan menjelaskan beberapa kelemahan dan kelebihan pada sistem dahulu dan sekarang ini. Pada tahun 2007, aplikasi yang digunakan untuk merahasiakan pesan yaitu algoritma *Stream Cipher* RC4 dapat dipecahkan dalam waktu kurang 1 menit, tentu ini sangat tidak efektif dalam merahasiakan suatu informasi ataupun data.

Algoritma RC4 memiliki kelemahan antara lain, tingginya banyak peluang untuk menghasilkan *array* S yang sama ataupun berulang dan *Bit-Flipping Attack*. dan kelemahan algoritma ini adalah mudahnya diserang oleh para kriptanalis dengan jenis *know-pain attack* maupun *cipher-only attack* karena proses enkripi dan dekripsinya yang sangat sederhana. Namun kelebihan dari algoritma RC4 ini adalah algoritma ini membangkitkan deretan kunci yang sangat acak sehingga juga tergolong sangat sulit untuk dipecahkan oleh pihak lain.

Maka dari itu penulis tertarik untuk melakukan penelitian dalam meningkatkan proses merahasiakan file dengan efektif menggunakan metode algoritma *block cipher* dan *stream cipher*, karena dengan menggunakan *block cipher*, pesan akan dipecahkan menjadi blok dengan ukuran tertentu dan setiap blok akan dienkripsi dengan menggunakan kunci enkripsi. Dengan menggunakan *stream cipher*, *plaintext* akan dikombinasikan dengan sebuah deretan bit *pseudorandom cipher* dan setiap bit dari *plaintext* akan dikombinasikan dengan stream bit cipher dengan sebuah operasi xor, maka kedua algoritma tersebut yaitu *block cipher* dan *stream cipher* akan dikombinasikan dengan *collision resistant hash function* (CRHF) untuk menutupi kekurangannya satu sama lain. CRHF merupakan sebuah cara yang efektif dalam meningkatkan keamanan dari metode enkripsi simetris.

3.4 Rancangan Penelitian

Metode Enkripsi *Cipher Block* dan *Stream Cipher* Berdasarkan pada *Collision Resistant Hash Functions* merupakan metode enkripsi simetris yang menggabungkan konsep *block cipher* dan *stream cipher*. Metode enkripsi *Cipher Block* dan *Stream Cipher* dapat digunakan untuk melakukan proses pengamanan data sekaligus melakukan proses otentikasi terhadap data rahasia tersebut. Selain itu, metode enkripsi *Cipher Block* dan *Stream Cipher* ini juga menggunakan fungsi *one way hash function* yang berfungsi untuk menghasilkan nilai *Message Authentication Code* (MAC) dari data rahasia yang dimasukkan. Dalam penelitian

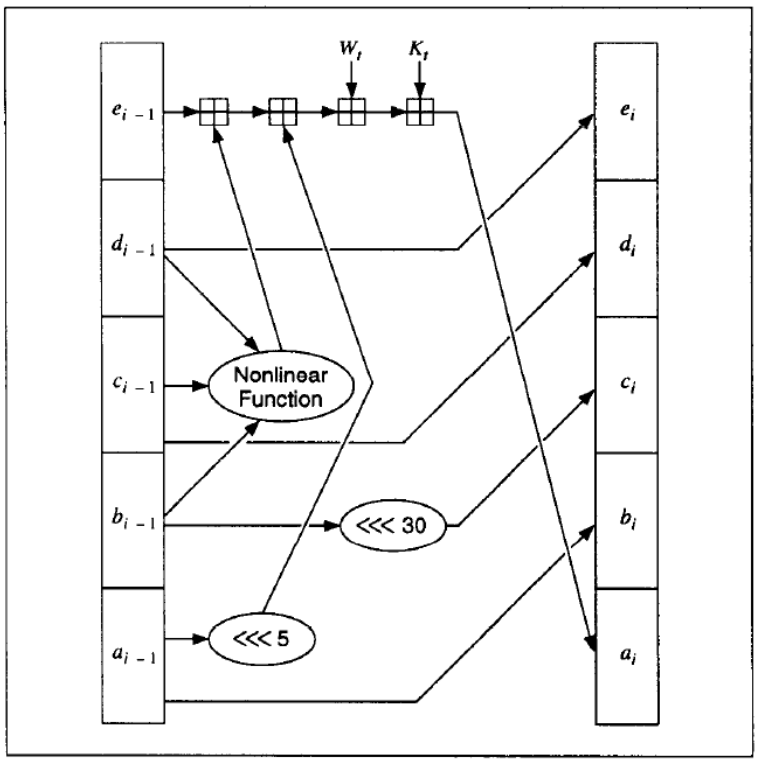
ini, akan digunakan fungsi hash SHA-1 yang memiliki panjang output sebanyak 160 bit.

3.4.1 Proses

Metode Enkripsi *Cipher Block* dan *Stream Cipher* Berdasarkan pada *Collision Resistant Hash Functions* menggunakan fungsi *hash* SHA-1 dan dalam proses kerjanya. Proses kerja dari metode Enkripsi *Cipher Block* dan *Stream Cipher* dapat dijabarkan sebagai berikut:

1. Proses Fungsi Hash SHA-1

Mula-mula, pesan diberi *bit* tambahan untuk membuat panjangnya menjadi kelipatan 512 bit (Padmsg terbagi 4, dan masing-masing dari padmsg tersebut berjumlah total 512 dan dibagi 4 sehingga menjadi 128). Jumlah bit data asal adalah K bit. Tambahkan bit 0 kemudian tambahkan bit 1 sampai ke bit 4 secukupnya sampai 64 bit kurangnya dari kelipatan. Akhirnya tambahkan 64 bit yang menyatakan panjang pesan sebelum diberi tambahan. Pesan dibagi-bagi menjadi blok-blok berukuran sebanyak 512 bit dan setiap blok diolah. Dan keluaran dari setiap blok digabungkan dengan keluaran blok berikutnya. Sehingga akhirnya diperoleh *message digest*.



Gambar 3.2 Proses SHA

Fungsi nonlinier yang digunakan SHA adalah:

$f_t(X, Y, Z) = (X \wedge Y) \vee ((\sim X) \wedge Z)$, untuk t = 0 sampai 19.

$f_t(X, Y, Z) = X \oplus Y \oplus Z$, untuk t = 20 sampai 39.

$f_t(X, Y, Z) = (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$, untuk t = 40 sampai 59.

$f_t(X, Y, Z) = X \oplus Y \oplus Z$, untuk t = 60 sampai 79.

Simbol '<<< 5' menunjukkan bahwa A digeser 5 bit ke arah kiri.

Empat konstanta K_t yang digunakan pada SHA:

K_t =0x5a827999, untuk t = 0 hingga 19 (20 tahap pertama)

K_t =0x6rd9eba1, untuk t = 20 hingga 39 (20 tahap kedua)

K_t =0x8f1bbcdc, untuk t = 40 hingga 59 (20 tahap ketiga)

K_t =0xca62c1d6, untuk t = 60 hingga 79 (20 tahap keempat)

Blok pesan ditransformasikan dari 16 *word* 32-bit (M_0 sampai M_{15}) ke 80 *word* 32-bit menggunakan algoritma berikut:

$W_t = M_t$, untuk t = 0 hingga 15

$W_t = (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16})$ <<< 1, untuk t = 16 hingga 79.

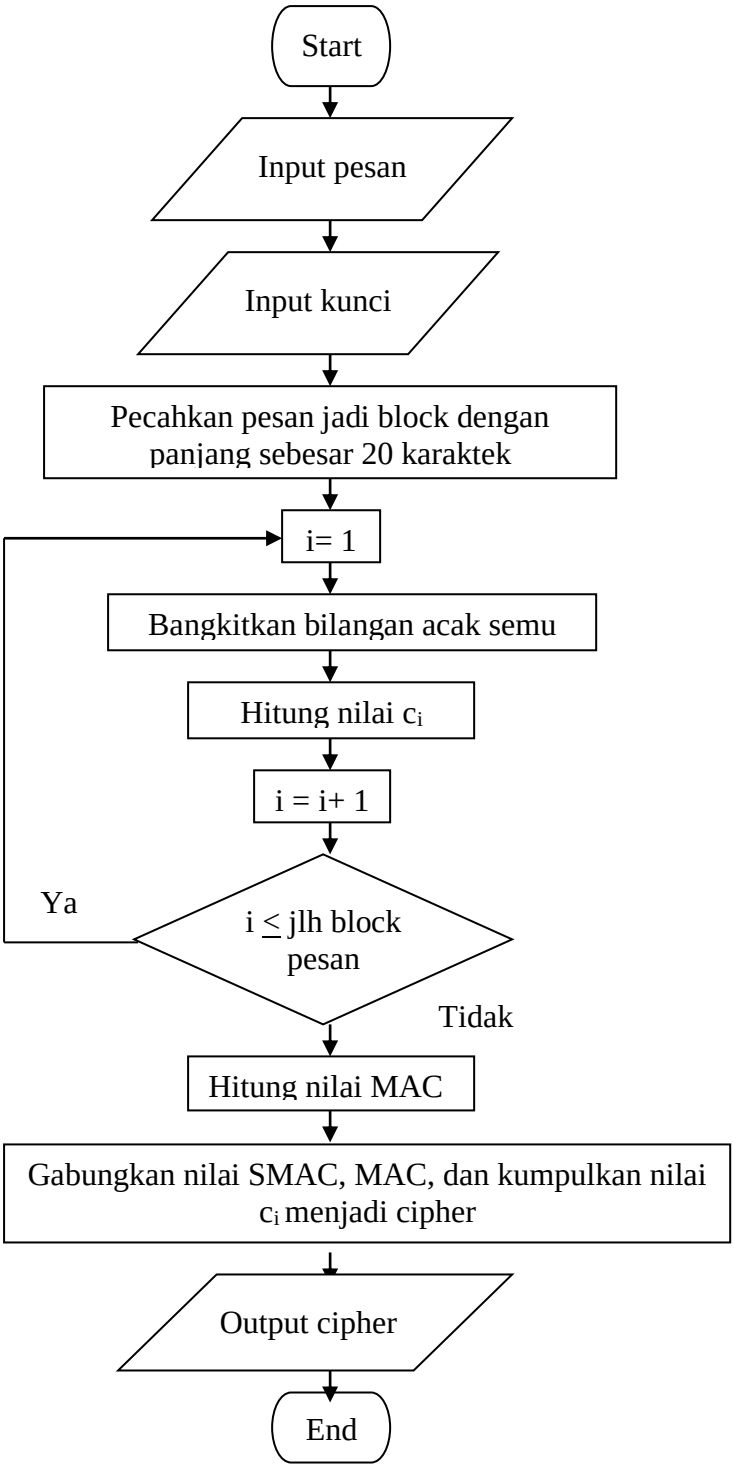
Bila *t* adalah bilangan operasi dari 1 sampai 80, W_t menyatakan sub blok ke-*t* dari pesan yang diekspansikan, dan <<< *s* menyatakan pergeseran ke kiri memutar sejauh *s* bit, maka proses utama setiap tahap akan menjadi bagian berikut:

```
For t = 0 to 79
      TEMP = (a <<< 5) + ft(b,c,d) + e + Wt + Kt
      e = d
      d = c
      c = b <<< 30 (lambang S30 berarti b digeser 30 bit ke
kiri dan hasilnya dimasukkan ke c)
      b = a
      a = Temp
Next i
```

Sebagai catatan di sini bahwa tanda + menunjukkan penjumlahan modulo 2^{32}

2. Proses Enkripsi

Proses enkripsi dari metode Enkripsi *Cipher Block* dan *Stream Cipher* memerlukan *input* berupa *plaintext* sebesar 160 bit (= 20 karakter) dan juga kunci dengan panjang maksimal sebesar 10 karakter. Langkah kerja dari proses enkripsi dapat dijabarkan seperti terlihat pada gambar *flowchart* berikut:

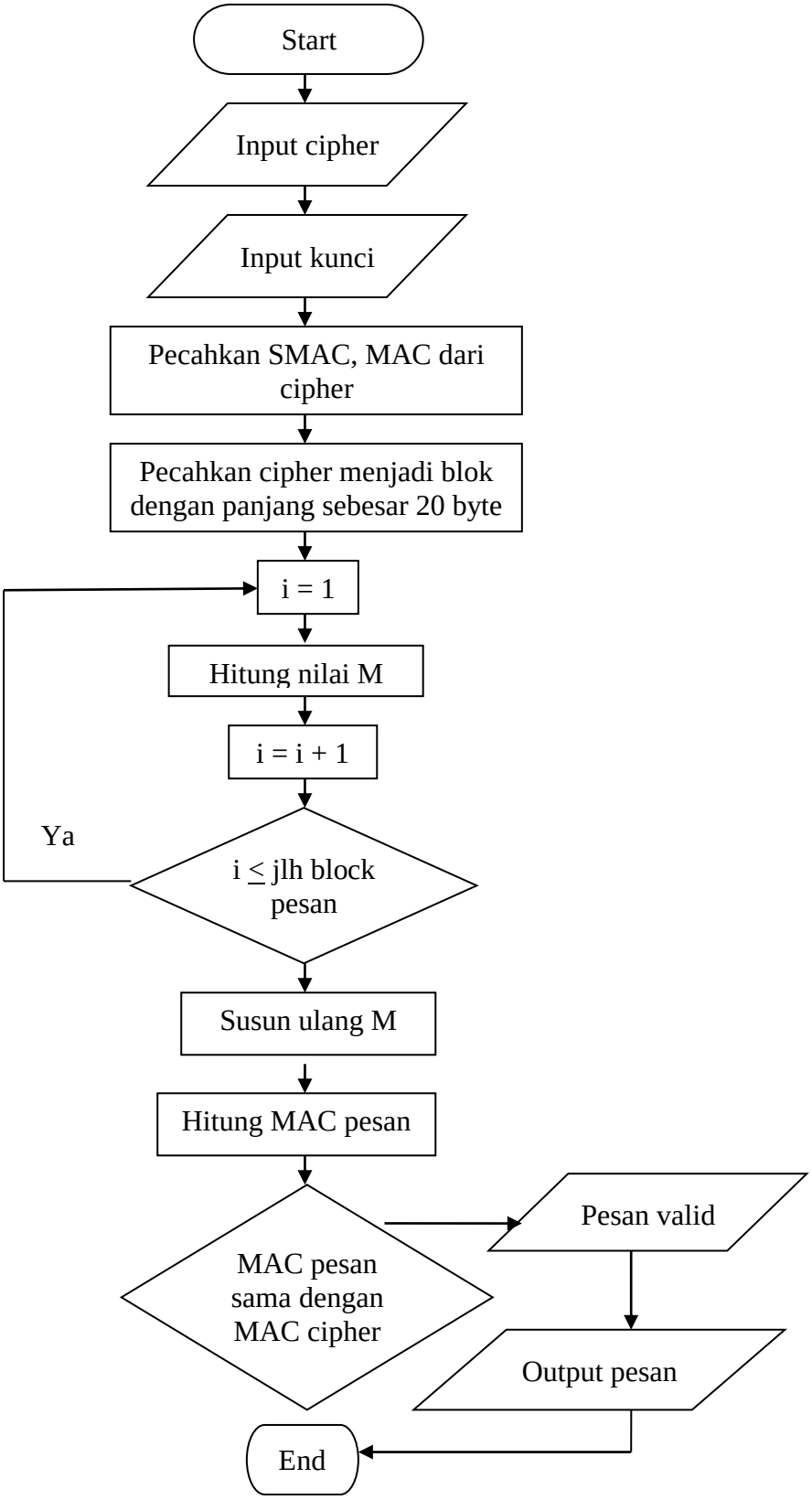


Gambar 3.3 *Flowchart* Proses Enkripsi

3. Proses Dekripsi

Proses dekripsi dari metode Enkripsi *Cipher Block* dan *Stream Cipher* memerlukan *input* berupa *ciphertext* sebesar 160 bit (= 20 karakter) dan juga kunci dengan panjang maksimal sebesar 10 karakter. Langkah kerja dari proses dekripsi dapat dijabarkan seperti terlihat pada gambar *flowchart* berikut:

\\



Gambar 3.4 *Flowchart* Proses Dekripsi

Agar dapat lebih memahami mengenai proses kerja dari metode Enkripsi *Cipher Block* dan *Stream Cipher* menggunakan *Collision Resistant Hash Functions*, maka diberikan sebuah contoh sederhana berikut ini:

1) Proses Enkripsi

1. Misalkan kunci simetris yang digunakan adalah 1234567890, dan pesan yang akan diamankan adalah CRYPTOGRAPHY METHODS.

   Diketahui: Kunci = 1234567890

   Ditanya:

   ASCII Code Kunci = ?

   1 : 49 (ASCII Code dari karakter '1' dalam bentuk bilangan desimal) = 00110001 (nilai biner dari 49)

   2 : 50 = 00110010

   3 : 51 = 00110011

   4 : 52 = 00110100

   5 : 53 = 00110101

   6 : 54 = 00110110

   7 : 55 = 00110111

   8 : 56 = 00111000

   9 : 57 = 00111001

   0 : 48 = 00110000

2. Pecahkan pesan menjadi blok dengan panjang sebesar 20 karakter.

   Blok 1 = CRYPTOGRAPHY METHODS

C : 67 (ASCII Code dari karakter ‘C’ dalam bentuk bilangan desimal)

= 01000011 (nilai biner dari 67) = 43 (nilai heksa dari 67)

R : 82 = 01010010 = 52

Y : 89 = 01011001 = 59

P : 80 = 01010000 = 50

T : 84 = 01010100 = 54

O : 79 = 01001111 = 4F

G : 71 = 01000111 = 47

R : 82 = 01010010 = 52

A : 65 = 01000001 = 41

P : 80 = 01010000 = 50

H : 72 = 01001000 = 48

Y : 89 = 01011001 = 59

: 32 = 00100000 = 20

M : 77 = 01001101 = 4D

E : 69 = 01000101 = 45

T : 84 = 01010100 = 54

H : 72 = 01001000 = 48

O : 79 = 01001111 = 4F

D : 68 = 01000100 = 44

S : 83 = 01010011 = 53

3. Bangkitkan bilangan acak semu S_i (32 bit):

S_1 = (01001100 01000001 00100101 10101100)$_2$ (dalam bentuk biner) = (4C4125AC)$_{16}$ (dalam bentuk heksadesimal)

Nilai S_1 merupakan kumpulan digit biner yang dibangkitkan secara acak sepanjang 32 bit.

4. Hitung nilai c_i:

   $c_i = S_i \,||\, m_i \oplus H(K, S_i, i)$

   Nilai $H(K, S_i, i)$ akan dihitung dengan menggunakan fungsi hash SHA1 dengan input berupa nilai K, S_i dan i. Output dari fungsi hash SHA1 adalah nilai hash sepanjang 160 bit bilangan biner atau sama dengan 40 digit bilangan heksadesimal.

   $H(K, S_i, i)$ = SHA1(1234567890, 4C4125AC, 1)

   = CDFEA2D166C639BA60D3CF943EB7957C7105AE35 (dalam heksadesimal)

   $m_1 \oplus H(K, S_i, i)$ =

   43525950 544F4752 41504859 204D4554 484F4453 ⊕

   CDFEA2D1 66C639BA 60D3CF94 3EB7957C 7105AE35

   = 8EACFB81 32897EE8 218387CD 1EFAD028 394AEA66 (dalam heksadesimal)

   $c_1 = \mathbf{S_1} \,||\, m_i \oplus H(K, S_i, i)$

   c_1 =

   4C4125AC 8EACFB81 32897EE8 218387CD 1EFAD028 394AEA66

5. Hitung nilai MAC

   $S_{MAC} = S_1$ = 4C4125AC

$H(K, M, S_{MAC})$ =

SHA1(1234567890CRYPTOGRAPHY METHODS4C4125AC) =

7CB92E921DB022994CB9DEC196A2A28E94C27B17

6. C = $\mathbf{S_{MAC}} \| \mathit{MAC} \| c_1 \| c_2 \| ... \| c_n$

C =

4C4125AC *7CB92E92 1DB02299 4CB9DEC1 96A2A28E 94C27B17*

4C4125AC 8EACFB81 32897EE8 218387CD 1EFAD028 394AEA66

2) Proses Dekripsi

1 : Input Cipher:

C =

4C4125AC 7CB92E92 1DB02299 4CB9DEC1 96A2A28E 94C27B17

4C4125AC 8EACFB81 32897EE8 218387CD 1EFAD028 394AEA66

2 : Input Kunci:

Kunci :

1	: 49	=	00110001
2	: 50	=	00110010
3	: 51	=	00110011
4	: 52	=	00110100
5	: 53	=	00110101
6	: 54	=	00110110
7	: 55	=	00110111
8	: 56	=	00111000
9	: 57	=	00111001

0 : 48 = 00110000

3 : Pecahkan S_{MAC} dan MAC dari cipher C:

C =

4C4125AC 7CB92E92 1DB02299 4CB9DEC1 96A2A28E 94C27B17

4C4125AC 8EACFB81 32897EE8 218387CD 1EFAD028 394AEA66

S_{MAC} = 4C4125AC

MAC = 7CB92E92 1DB02299 4CB9DEC1 96A2A28E 94C27B17

4 : Pecahkan *cipher* C:

c_1 =

4C4125AC 8EACFB81 32897EE8 218387CD 1EFAD028 394AEA66

5 : Hitung nilai m_1:

m_1 = 8EACFB81 32897EE8 218387CD 1EFAD028 394AEA66 ⊕

CDFEA2D1 66C639BA 60D3CF94 3EB7957C 7105AE35

= 43525950 544F4752 41504859 204D4554 484F4453

6 : Susun ulang M:

43 (heksa) = C

52 = R

59 = Y

50 = P

54 = T

4F = O

47 = G

52 = R

41 = A

50 = P

48 = H

59 = Y

20 = {spasi}

4D = M

45 = E

54 = T

48 = H

4F = O

44 = D

53 = S

7 : Hitung nilai MAC:

$S_{MAC} = S_1$ = 4C4125AC

$H(K, M, S_{MAC})$ =

SHA1(1234567890CRYPTOGRAPHY METHODS4C4125AC) =

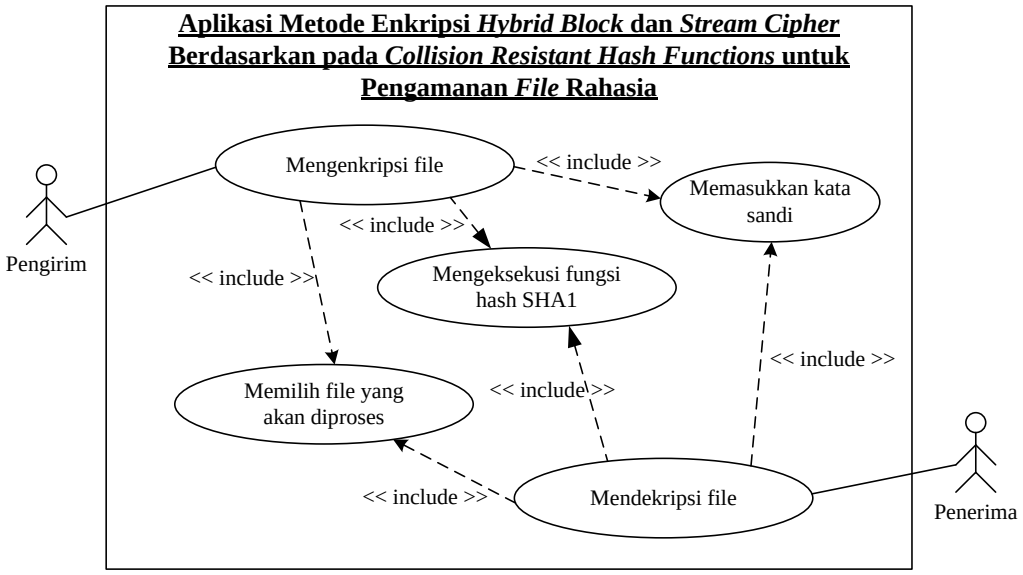
7CB92E921DB022994CB9DEC196A2A28E94C27B17

8 : Nilai MAC pesan sama dengan nilai MAC cipher → Pesan Valid.

9 : Pesan asli = CRYPTOGRAPHY METHODS

3.4.2 Pemodelan Sistem

Setelah menganalisis prosedur kerja dari metode Enkripsi *Cipher Block* dan *Stream Cipher*, maka tahapan selanjutnya adalah memodelkan sistem yang akan dirancang. Alat bantu yang digunakan untuk menganalisis dan memodelkan sistem adalah *use case*. Gambar 3.4 berikut menunjukkan *use case* dari sistem:



Gambar 3.5 *Use Case Diagram*

Narasi dari *use case* pada gambar 3.4 dapat dirincikan sebagai berikut:

Tabel 3.1 Narasi Mengeksekusi Fungsi Hash SHA-1

Nama *use case*	Mengeksekusi fungsi hash SHA-1	
Actor	Pemakai	
Deskripsi	*Use case* ini mendeskripsikan proses perhitungan fungsi *hash* SHA-1	
	Aksi Aktor	**Respons Sistem**
		1. Baca data *input* dari fungsi *hash* SHA-1. 2. Eksekusi fungsi *hash* SHA-1. 3. Tampilkan *output* SHA-1.

Tabel 3.2 Narasi Mengenkripsi *File*

Nama *use case*	Mengenkripsi *file*	
Actor	Pengirim	
Deskripsi	*Use case* ini mendeskripsikan kejadian proses enkripsi *file*	
	Aksi Aktor	**Respons Sistem**
	1. Klik link ’Enkripsi *File*’. 3. Klik tombol Browse. 5. Pilih *file* yang ingin dienkripsi. 6. Klik tombol Open	2. Sistem menampilkan form ’Enkripsi’. 4. Sistem menampilkan kotak dialog Open *File*. 7. Sistem membaca isi *file*. 8. Sistem mengenkripsi *file* dengan menggunakan metode metode Enkripsi *Hybrid Block* dan *Stream Cipher*.

Tabel 3.3 Narasi Mendekripsi *File*

Nama *use case*	Mendekripsi *file*	
Actor	Penerima	
Deskripsi	*Use case* ini mendeskripsikan kejadian proses dekripsi *file*	
	Aksi Aktor	**Respons Sistem**
	1. Klik link ’Dekripsi *File*’. 3. Klik tombol Browse. 5. Pilih *file* yang ingin didekripsi. 6. Klik tombol Open	2. Sistem menampilkan form ’Dekripsi’. 4. Sistem menampilkan kotak dialog Open *File*. 7. Sistem membaca isi *file*. 8. Sistem mendekripsi *file* dengan menggunakan metode metode Enkripsi *Hybrid Block* dan *Stream Cipher*.

3.5 Perancangan

Aplikasi Enkripsi *Cipher Block* dan *Stream Cipher* ini dirancang dengan menggunakan bahasa pemrograman Microsoft Visual C# 2013 dengan menggunakan beberapa objek dasar seperti :

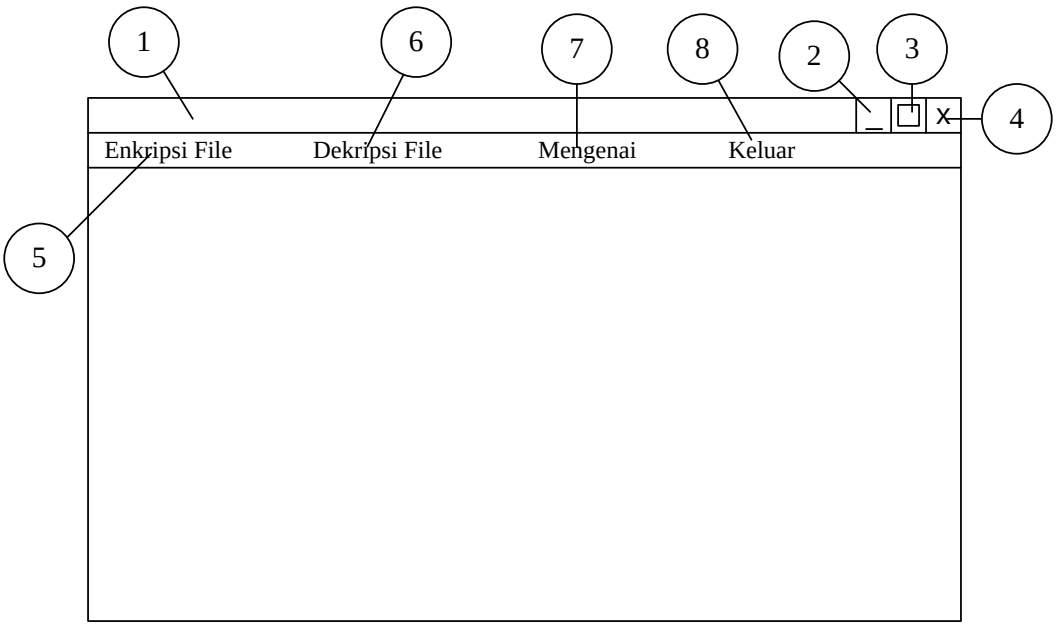
1. *Label*, yang digunakan untuk menampilkan keterangan.
2. *Link Label*, yang digunakan sebagai *link* untuk menghubungkan *form-form* yang terdapat pada perangkat lunak.
3. *Button*, yang digunakan sebagai tombol eksekusi.
4. *Textbox*, yang digunakan sebagai tempat pengisian data dan untuk menampilkan nilai hasil perhitungan.
5. *RichTextbox*, yang digunakan untuk menampilkan hasil proses perhitungan.
6. *Panel*, yang digunakan untuk menggabungkan objek.

Sedangkan, rancangan tampilan dari aplikasi Enkripsi *Cipher Block* dan *Stream Cipher* ini dapat dirincikan sebagai berikut:

1. *Form* Main.
2. *Form* Proses Enkripsi.
3. *Form* Proses Dekripsi.
4. *Form* Mengenai.

3.5.1 *Form Main*

Form ini merupakan *form* inti dari perangkat lunak yang berfungsi sebagai penghubung (*link*) ke *form-form* lainnya. Rancangan tampilan dari *form Main* ini dapat dilihat pada gambar berikut ini :



Gambar 3.6 Rancangan *Form Main*

Keterangan :

1 : *Title bar*.

2 : Tombol _, berfungsi untuk me-*minimize form*.

3 : Tombol 🗗, berfungsi untuk mengatur ukuran *form*.

4 : Tombol X, berfungsi untuk keluar dari perangkat lunak.

5 : *Link* Enkripsi *File*, berfungsi untuk menampilkan *form* Enkripsi *File*.

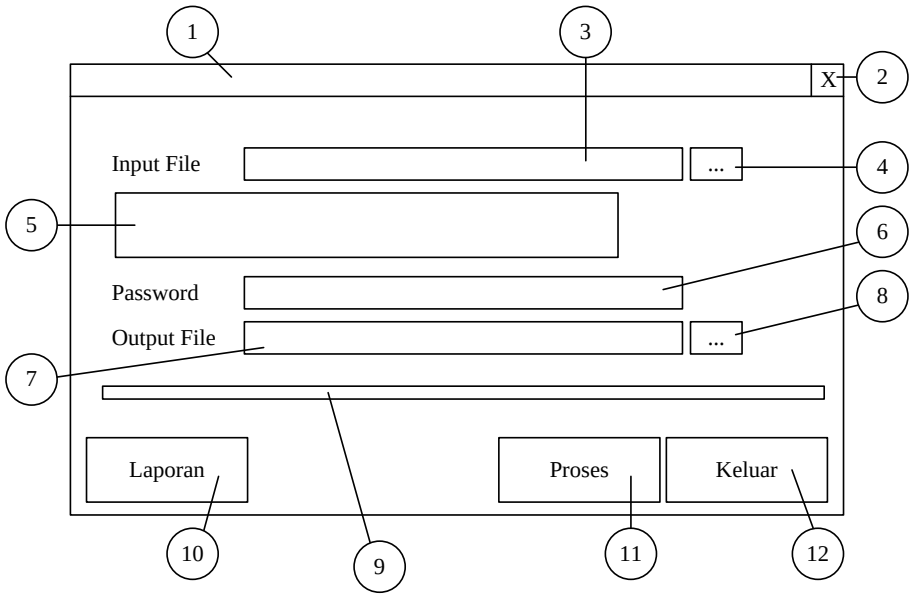
6 : *Link* Dekripsi *File*, berfungsi untuk menampilkan *form* Dekripsi *File*.

7 : *Link* Mengenai, berfungsi untuk menampilkan *form About*.

8 : *Link* Keluar, berfungsi untuk keluar dari perangkat lunak.

3.5.2 *Form* Proses Enkripsi

Form ini digunakan untuk melakukan proses enkripsi terhadap *file* yang dimasukkan. Rancangan tampilan dari *form* Proses Enkripsi ini dapat dilihat pada gambar berikut ini:



Gambar 3.7 Rancangan *Form* Proses Enkripsi

Keterangan :

1 : *Title bar*.

2 : Tombol x, berfungsi untuk menutup *form*.

3 : *Textbox File* Input, berfungsi untuk menampilkan lokasi dan nama *file* yang akan diproses.

4 : Tombol Browse, berfungsi untuk membuka kotak dialog *open*.

5 : Informasi *file* yang akan diproses.

6 : *Textbox* Password, berfungsi sebagai tempat pengisian data *password*.

7 : *Textbox File* Output, berfungsi untuk menampilkan lokasi dan nama *file* yang akan dihasilkan.

8 : Tombol Browse, berfungsi untuk membuka kotak dialog *save*.

9 : *Progress bar*.

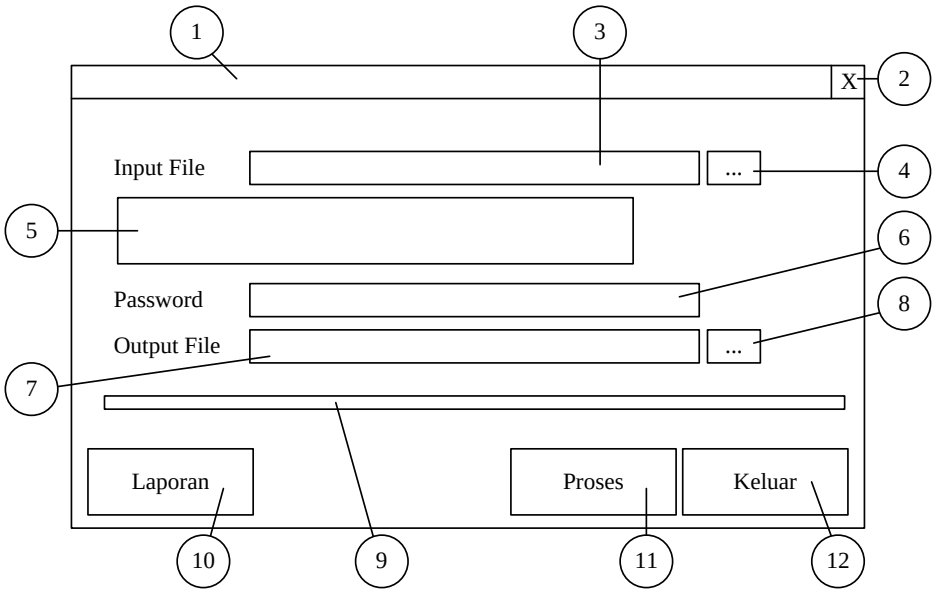
10 : Tombol Laporan, berfungsi untuk menampilkan *form* Laporan.

11 : Tombol Proses, berfungsi untuk memulai proses enkripsi *file*.

12 : Tombol Keluar, berfungsi untuk menutup *form*.

3.5.3 *Form* Proses Dekripsi

Form ini digunakan untuk melakukan proses dekripsi terhadap *file* yang dimasukkan. Rancangan tampilan dari *form* Proses Dekripsi ini dapat dilihat pada gambar berikut ini:



Gambar 3.8 Rancangan *Form* Proses Dekripsi

Keterangan :

1 : *Title bar*.

2 : Tombol x, berfungsi untuk menutup *form*.

3 : *Textbox File* Input, berfungsi untuk menampilkan lokasi dan nama *file* yang akan diproses.

4 : Tombol Browse, berfungsi untuk membuka kotak dialog *open*.

5 : Informasi *file* yang akan diproses.

6 : *Textbox* Password, berfungsi sebagai tempat pengisian data *password*.

7 : *Textbox File* Output, berfungsi untuk menampilkan lokasi dan nama *file* yang akan dihasilkan.

8 : Tombol Browse, berfungsi untuk membuka kotak dialog *save*.

9 : *Progress bar*.

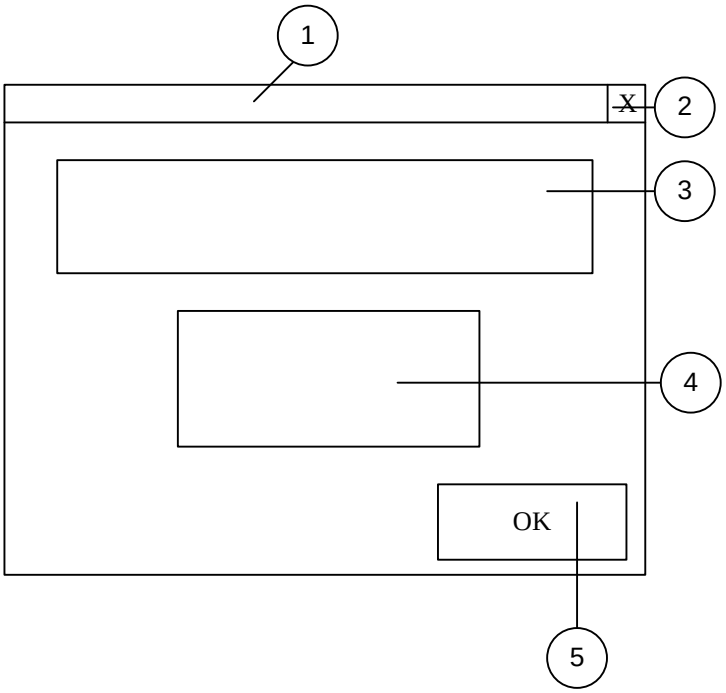
10 : Tombol Laporan, berfungsi untuk menampilkan *form* Laporan.

11 : Tombol Proses, berfungsi untuk memulai proses dekripsi *file*.

12 : Tombol Keluar, berfungsi untuk menutup *form*.

3.5.4 *Form About*

Form ini berfungsi untuk menampilkan data-data pribadi dari perancang perangkat lunak. Rancangan tampilan dari *form About* dapat dilihat pada gambar berikut :



Gambar 3.9 Rancangan *Form About*

Keterangan :

1 : *Title bar* yang berisi tulisan *About.*

2 : Tombol X, berfungsi untuk menutup *form* dan kembali ke *form Main*.

3 : Daerah tampilan nama perangkat lunak.

4 : Daerah tampilan data penyusun.

5 : Tombol OK, berfungsi untuk menutup *form* dan kembali ke *form Main*.

BAB IV

HASIL DAN PEMBAHASAN

4.1 Spesifikasi *Hardware* dan *Software*

Untuk menjalankan sistem yang dirancang, diperlukan beberapa faktor pendukung sebagai berikut :

1) Kebutuhan Perangkat Keras (*Hardware*)

   Dalam merancang dan menguji sistem, *hardware* yang digunakan adalah satu set lengkap perangkat komputer yang memiliki spesifikasi sebagai berikut:

   1. laptop
   2. Windows 7 ultimate 32-bit
   3. System manufacture acer
   4. Model 4739Z
   5. RAM 1 GB
   6. Harddisk 80 GB
   7. Monitor SVGA dengan resolusi layar minimal 1024 x 768
   8. Keyboard dan Mouse

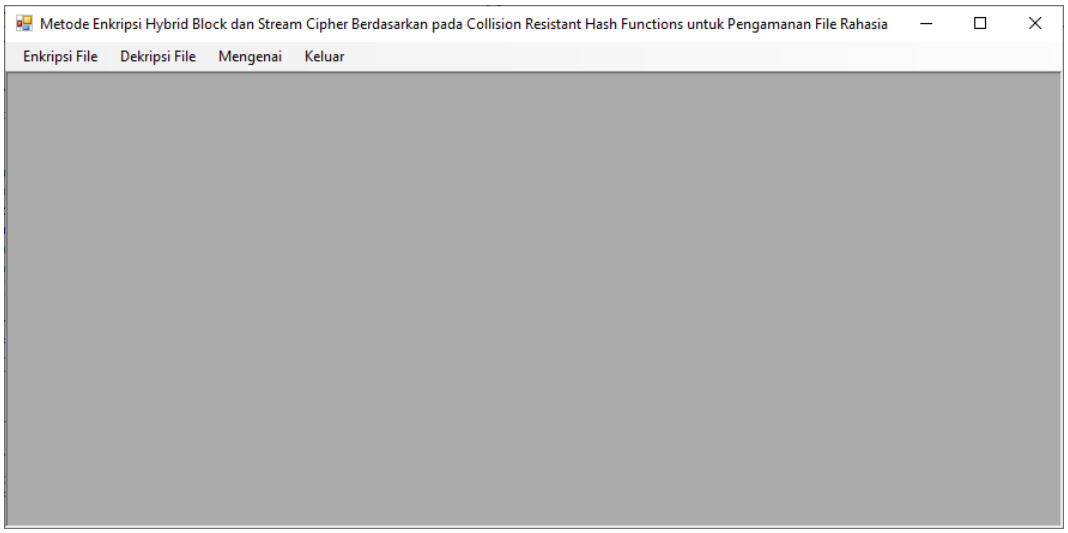
2) Kebutuhan Perangkat Lunak (*Software*)

   Adapun perangkat lunak untuk merancang program ini adalah:

   1. Sistem operasi Windows 7 ke atas.
   2. Microsoft Visual Studio 2013.

4.2 Hasil

Tampilan pertama yang akan muncul pada saat menjalankan perangkat lunak adalah *form Main* seperti terlihat pada gambar 4.1:

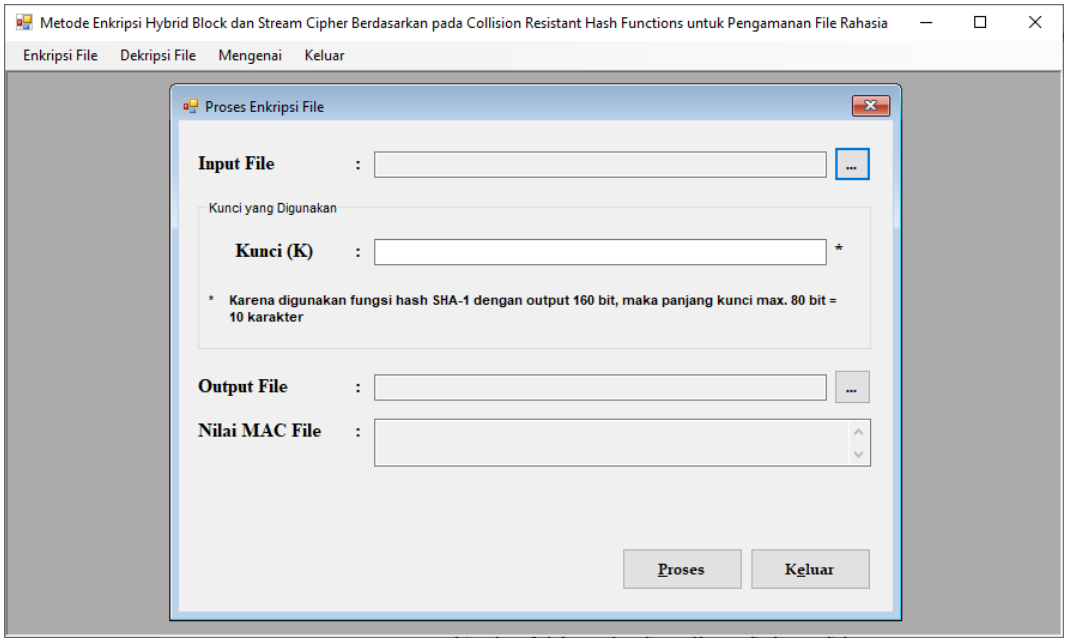


Gambar 4.1 Tampilan Form Main

Pada *form* Main ini terdapat beberapa menu yang berfungsi untuk mengakses *form-form* yang terdapat dalam sistem. Berikut perincian dari menu yang terdapat dalam sistem:

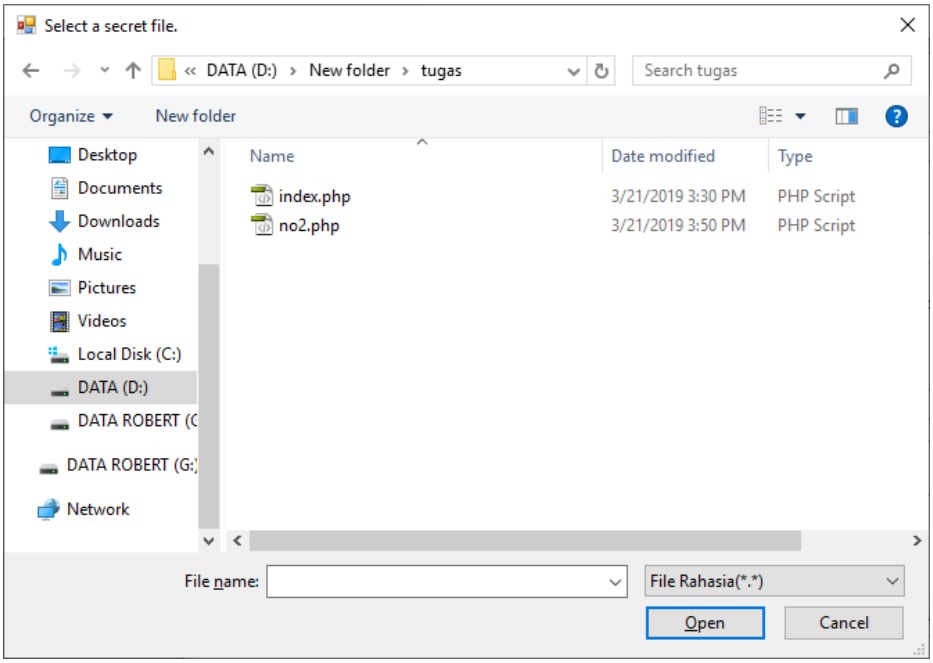
1. Menu Proses Enkripsi File

Pada saat menjalankan menu Proses Enkripsi File ini, maka sistem akan menampilkan *form* Enkripsi File seperti terlihat pada gambar 4.2.



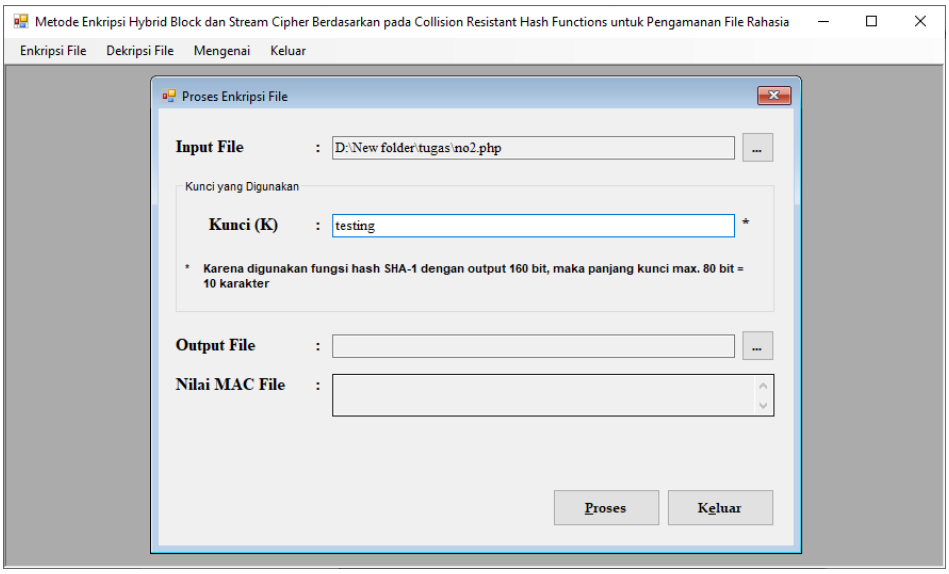
Gambar 4.2 Tampilan *Form* Proses Enkripsi File

Langkah pertama dari proses enkripsi *file* adalah memilih *file* yang akan diproses. Caranya adalah dengan mengklik tombol '…', sehingga sistem akan menampilkan kotak dialog Open seperti terlihat pada gambar 4.3:



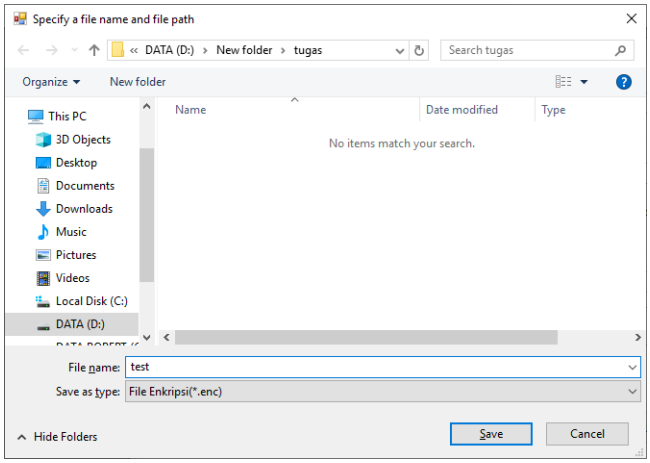
Gambar 4.3 Tampilan Kotak Dialog *Open*

Pilihlah *file* yang diinginkan. Setelah itu, klik tombol ‘Open’ untuk membuka *file* tersebut. Setelah itu, proses akan dilanjutkan dengan pengisian data kunci yang digunakan seperti gambar 4.4:



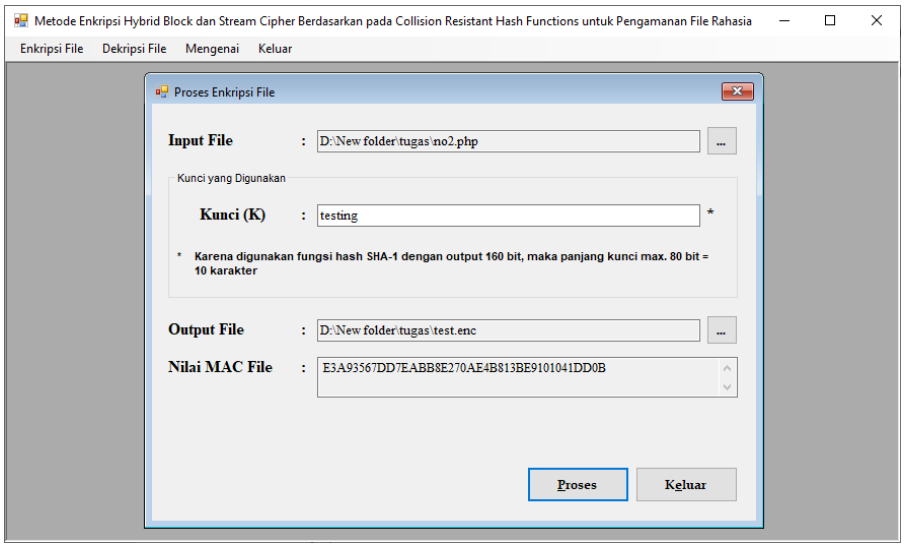
Gambar 4.4 Tampilan *Form* Enkripsi File Setelah Pemilihan File dan Pengisian Password

Setelah itu, klik tombol *Browse Output File* untuk memilih lokasi penyimpanan *file* dan nama *file* yang dihasilkan, sehingga sistem akan menampilkan kotak dialog seperti terlihat pada gambar 4.5:



Gambar 4.5 Tampilan Kotak Dialog Simpan

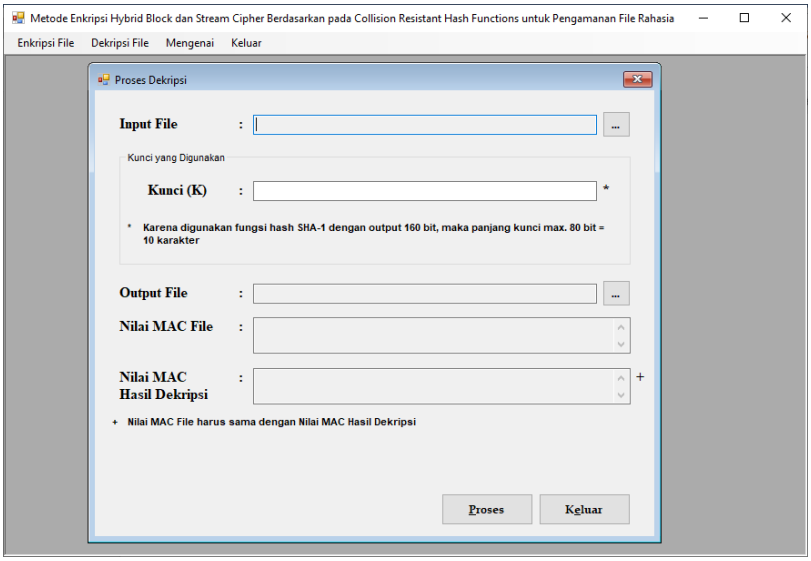
Setelah itu, kliklah tombol ‘Proses’ untuk memulai proses enkripsi. Hasil enkripsi akan disimpan ke dalam sebuah *file*, seperti terlihat pada gambar 4.6:



Gambar 4.6 Tampilan *Form* Proses Enkripsi

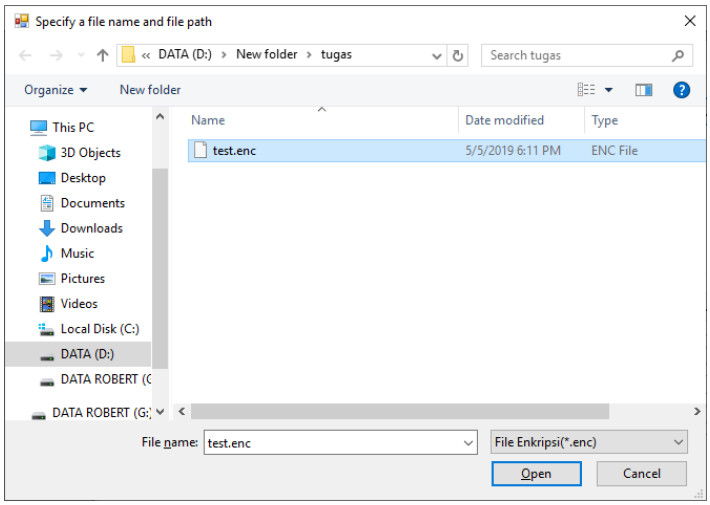
2. Menu Proses Dekripsi *File*

Untuk melakukan proses dekripsi *file* rahasia kembali dapat dilakukan dengan mengakses menu Proses Dekripsi *File* sehingga sistem akan menampilkan *form* Dekripsi *File* seperti terlihat pada gambar 4.7.



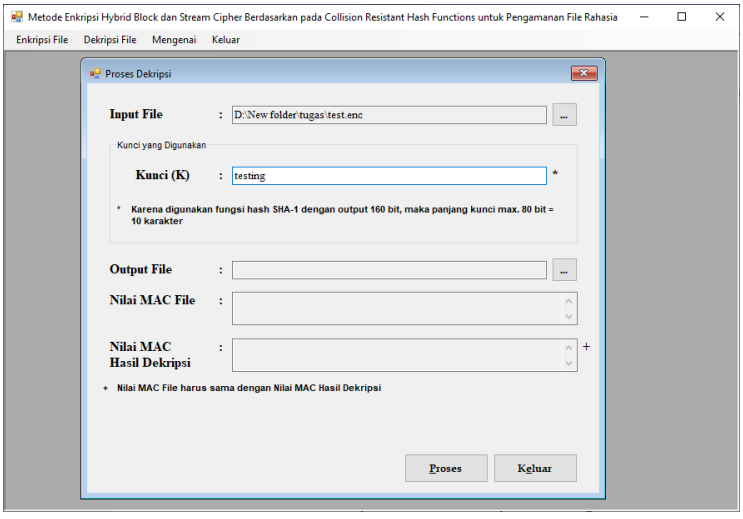
Gambar 4.7 Tampilan *Form* Dekripsi *File*

Langkah pertama dari proses dekripsi *file* adalah memilih *file* yang akan diproses. Caranya adalah dengan mengklik tombol '…', sehingga sistem akan menampilkan kotak dialog *Open* seperti terlihat pada gambar 4.8:



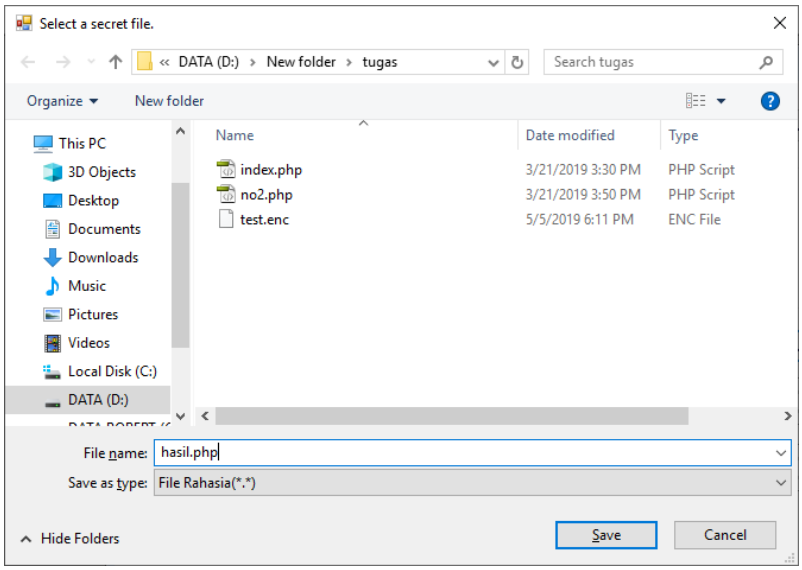
Gambar 4.8 Tampilan Kotak Dialog *Open*

Pilihlah *file* yang diinginkan. Setelah itu, klik tombol 'Open' untuk membuka *file* tersebut. Setelah itu, proses akan dilanjutkan dengan pengisian *password* yang digunakan. Tampilan *form* Proses Dekripsi akan terlihat seperti gambar 4.9:



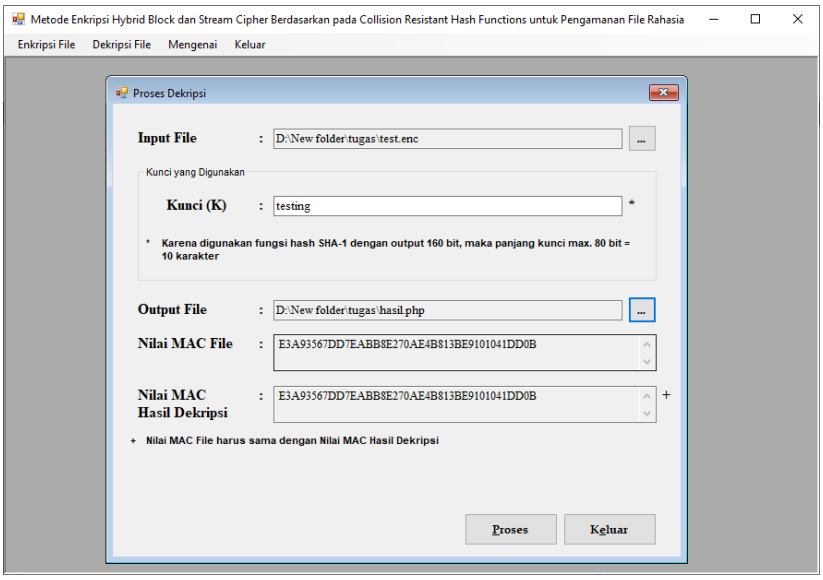
Gambar 4.9 Tampilan *Form* Proses Dekripsi Setelah Pengisian Data

Setelah itu, klik tombol *Browse Output File* untuk memilih lokasi penyimpanan *file* dan nama *file* yang dihasilkan, sehingga sistem akan menampilkan kotak dialog seperti terlihat pada gambar 4.10:



Gambar 4.10 Tampilan Kotak Dialog Simpan

Setelah itu, kliklah tombol 'Proses' untuk memulai proses dekripsi. Hasil dekripsi akan disimpan ke dalam sebuah *file*, seperti terlihat pada gambar 4.11:



Gambar 4.11 Tampilan *Form* Proses Dekripsi

4.3 Pembahasan

Adapun keunggulan dari perangkat lunak pengamanan *file* dengan metode Enkripsi *Cipher Block* dan *Stream Cipher* menggunakan *Collision Resistant Hash Functions* yaitu:

1. Aplikasi yang dibuat dapat digunakan untuk melakukan pengamanan semua jenis *file* dengan menggunakan kunci yang telah disepakati bersama sebelumnya.
2. Lama proses enkripsi dan dekripsi sangat cepat sehingga metode Enkripsi *Cipher Block* dan *Stream Cipher* menggunakan *Collision Resistant Hash Functions* ini dapat diterapkan secara praktikal.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Setelah menyelesaikan pembuatan perangkat lunak ini, penulis dapat menarik beberapa kesimpulan sebagai berikut:

1. Aplikasi yang dibuat dapat digunakan untuk melakukan pengamanan semua jenis *file* dengan menggunakan kunci yang telah disepakati bersama sebelumnya, dimana panjang kunci maksimal sebesar 10 karakter.
2. Lama proses enkripsi dan dekripsi sangat cepat sehingga metode Enkripsi *Cipher Block* dan *Stream Cipher* menggunakan *Collision Resistant Hash Functions* ini dapat diterapkan secara praktikal.

5.2 Saran

Penulis ingin memberikan beberapa saran yang mungkin berguna untuk pengembangan lebih lanjut pada perangkat lunak, yaitu :

1. Perangkat lunak dapat dibuat dalam bentuk aplikasi *website* atau *mobile* sehingga dapat diakses oleh pemakai dimanapun.
2. Perangkat lunak dapat dikembangkan lebih lanjut dengan menerapkan skema untuk melakukan verifikasi dan pengamanan sebuah layanan *email*.
3. Perangkat lunak dapat dikembangkan lagi dengan menambahkan fitur lainnya seperti fitur tutorial yang mampu menjelaskan prosedur kerja dari algoritma yang dibahas secara terperinci.

DAFTAR PUTAKA

Asaziduhu, Et. All. 2019. “Implementasi *Triple Transposition Vigenere Chipper* Pada Pengiriman Data Teks Dalam Jaringan *LAN*”. *Jurnal Manajemen Informatika Dan Komputerisasi Akuntansi. Vol. 3. No. 1. Hal. 147.*

Fachri, barany, agus perdana windarto, and ikhsan parinduri. "penerapan backpropagation dan analisis sensitivitas pada prediksi indikator terpenting perusahaan listrik." jepin (jurnal edukasi dan penelitian informatika) 5.2 (2019): 202-208.

Fachri, b., windarto, a. P., & parinduri, i. (2019). Penerapan backpropagation dan analisis sensitivitas pada prediksi indikator terpenting perusahaan listrik. Jepin (jurnal edukasi dan penelitian informatika), 5(2), 202-208.

Fachri, barany; windarto, agus perdana; parinduri, ikhsan. Penerapan backpropagation dan analisis sensitivitas pada prediksi indikator terpenting perusahaan listrik. Jepin (jurnal edukasi dan penelitian informatika), 2019, 5.2: 202-208

Fergiawan. Et. All. 2017. Aplikasi *Commerce* Berbasis *Web Mobile* Pada Industri Konveksi Seragam *Drumband* Di Pekon Klaten Gadingrejo Kabupaten Pringsewu”. *Jurnal TAM (Technologi Acceptance Model). Vol. . No. 2. Hal 147.*

Firlhi Kurniawan, *et, al*. 2017. “ Analisis Dan Implementasi Algoritma SHA-1 dan SHA-3 Pada Sistem *Autentikasi* Garuda *Training Cost*” *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*. Vol. 1. No. 9. Hal. 804.

Hamdi, nurul. "model penyiraman otomatis pada tanaman cabe rawit berbasis programmable logic control." jurnal ilmiah core it: community research information technology 7.2 (2019).

Hendri Pandiangan, Sijabat Salomo. 2016. “Perancangan Media Pengiriman Pesan Teks Dengan Penyandian Pesan Menggunakan Algoritma RC4 Berbasis *Web”. Jurnal Matik Penusa*. Vol. 19. No. 1. Hal 64.

Irham Arrizal, *et, al*. 2016 “Penerapan Algoritma Kriptografi Kunci Simetris Dengan Modifikasi *Vigenere Cipher* Dalam Aplikasi Kriptografi Teks” Jurnal Pseudocoe, Vol. 3. No. 1. Hal. 70.

Juansyah Andi. 2015. “Pembangunan Aplikasi *Child Tracker* Berbasis *Assisted-Global Positioning System (A-GPS)* Dengan Platform Android”. *Jurnal Ilmiah Komputer Dan Informatika (KOMPUTA). Vol. 1. No. 1.Hal. 2.*

Marcio Rosemberg, *et, al*. 2017. “ *A Hybrid Block And Stream Cipher Encription Scheme Based On Collision Resistant Hash Function*” Jurnal *Monografias em Ciência da Computação*. Vol. 1 No. 13. Hal. 6.

Meti Atiroh, *et. al*. 2014. “Rancang Bangun Aplikasi Fiqih Ibadah Shalat Berbasis Android” *Jurnal Algoritma*. Vol. 11. No. 1. Hal. 2

Munir, Rinaldi. 2019. KRIPTOGRAFI. Edisi 2. Bandung. Informatika. . Novi Natashia, Anang Wicaksono. 2011. “ Penerapan Teknik Kriptografi *Stream Cipher* Untuk Pengamanan Basis Data. *Jurnal Basis Data*, Vol. 6. No. 1. Hal. 4

Nuniek Fahriani, *et. al*. 2016. “ Pembangkit *Key Polyalphabetic Cipher* Pada Kriptografi *Simetri* Menggunakan *Java*”. *Jurnal Tekno*. Vol. 26. No. 2 Hal. 138.

Paulus Irawan. 2015. “Implementasi Teknik Kriptografi *Stream Cipher* Salsa20 Untuk Pengamanan Basis Data”. Jurnal STIKI Informatika. Vol. 5. No. 2. Hal. 89.

Permana, aminuddin indra. "kombinasi algoritma kriptografi one time pad dengan generate random keys dan vigenere cipher dengan kunci em2b." (2019).

Putra, randi rian. "sistem informasi web pariwisata hutan mangrove di kelurahan belawan sicanang kecamatan medan belawan sebagai media promosi." jurnal ilmiah core it: community research information technology 7.2 (2019).

Putra, randi rian, et al. "decision support system in selecting additional employees using multi-factor evaluation process method." (2019).

Putra, randi rian. "implementasi metode backpropagation jaringan saraf tiruan dalam memprediksi pola pengunjung terhadap transaksi." jurti (jurnal teknologi informasi) 3.1 (2019): 16-20.

Rifkie Primartha. 2011. “Penarapan *Enkripsi* Dan *Dekripsi File* Menggunakan Algoritma Data *Encription* Standart (*DES*)”. *Jurnal Sistem Informasi (JSI)*. Vol. 3. No, 2. Hal. 372.

Ruri Zain. 2012. “ Perancangan Dan Implementasi *Crypthography* Dengan Metode Algoritma RC4 Pada *Type File Document* Menggunakan Bahasa Programan *Viual Basic* 6.0” *Jurnal Momentum*. Vol. 12. No. 1. Hal. 73.

Saputra, muhammad juanda, and nurul hamdi. "rancang bangun aplikasi sejarah kebudayaan aceh berbasis android studi kasus dinas kebudayaan dan pariwisata aceh." journal of informatics and computer science 5.2 (2019): 147-157

Sastra Ramadanu. 2019. "Perancangan Aplikasi Pencarian *File* Dengan Menggunakan Algoritma *Collusi* dan Algoritma *Simon*" *jurna Riset Komputer (JURIKOM)*. Vol. 6 No. 1. Hal. 25.

Sidik, a. P., efendi, s., & suherman, s. (2019, june). Improving one-time pad algorithm on shamir's three-pass protocol scheme by using rsa and elgamal algorithms. In journal of physics: conference series (vol. 1235, no. 1, p. 012007). Iop publishing.

Sitepu, n. B., zarlis, m., efendi, s., & dhany, h. W. (2019, august). Analysis of decision tree and smooth support vector machine methods on data mining. In journal of physics: conference series (vol. 1255, no. 1, p. 012067). Iop publishing.

Sudi Suryadi. 2014. "Perancangan Aplikasi Pencarian *File* Dengan Menggunakan Metode *Best first Search*" *Jurnal Ilmiah AMIK Labuhan batu*. Vol. 2. No. 2. Hal. 21.

Tasril, v., wijaya, r. F., & widya, r. (2019). Aplikasi pintar belajar bimbingan dan konseling untuk siswa sma berbasis macromedia flash. Jurnal informasi komputer logika, 1(3).

Wiwiek Nurwiyati, Indra Yatini. 2013. "*Enkripsi Dekripsi* Data Menggunakan Metode *Stream* Dan *Vigenere Cipher*. *Jurnal Teknik*. Vol. 3. No. 23. Hal. 152.