RANCANGAN SISTEM KEAMANAN ALGORITMA GRONSFELD CIPHER DENGAN PEMBANGKIT BILANGAN ACAK LCG (*LINEAR CONGRUENTIAL GENERATOR*)

Disusun dan Diajukan Untuk Memenuhi Persyaratan Ujian Akhir Dalam Memperoleh Gelar Sarjana Komputer Pada Fakultas Sains dan Teknologi Universitas Pembangunan Panca Budi Medan

SKRIPSI

OLEH

NAMA	**: SYAHRUL MAULIDIN**
NPM	**: 1514370036**
PROGRAM STUDI	**: SISTEM KOMPUTER**

PROGRAM STUDI SISTEM KOMPUTER

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS PEMBANGUNAN PANCA BUDI

MEDAN

2019

ABSTRAK

SYAHRUL MAULIDIN

Rancangan Sistem Keamanan Algoritma Gronsfeld Cipher dengan Pembangkit Bilangan Acak LCG (*Linear Congruential Generator*)

2019

Untuk menjaga keamanan data supaya data tidak disalahgunakan oleh orang yang tidak bertanggungjawab, salah satu cara yang bisa digunakan yaitu dengan menggunakan metode kriptografi untuk mengenkripsi *file*, sehingga tidak dapat disalahgunakan oleh orang yang tidak berhak. Salah satu algoritma kriptografi adalah Gronsfeld yang merupakan algoritma *plaintext* kriptografi *modern* kunci simetris berbentuk cipher *block*. Enkripsi dilakukan dengan menggunakan penggabungan antara Gronsfeld dengan pembangkit bilangan acak LCG (*Linear Congruential Generator*). Aplikasi yang dibangun ini dapat mengenkripsi *plaintext* sehingga menghasilkan *ciphertext*. *Ciphertext* tersebut dapat dikembalikan seperti semula jika didekripsi menggunakan kunci yang sama sewaktu mengenkripsi *plaintext* tersebut. Kunci yang digunakan tidak membutuhkan batasan maksimum karakter. Metode yang digunakan untuk membangun aplikasi ini adalah metode deskriptif. Perangkat lunak yang digunakan untuk *user system interface*-nya adalah Visual Studio 2012.

Keywords : Dekripsi, Enkripsi, Gronsfeld, LCG (*Linear Congruential Generator*)

DAFTAR ISI

Halaman

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Masalah keamanan adalah masalah yang sangat penting dan yang paling utama dalam sistem komputer yang terhubung dalam suatu jaringan internet. Data maupun informasi menjadi target serangan oleh pihak-pihak yang tidak bertanggung jawab sehingga perlu untuk menjaga data ataupun informasi. Kejahatan yang dilakukan oleh orang yang tidak bertanggungjawab seperti pencurian data, pemalsuan data dan berbagai macam kejahatan dunia maya lainnya jelas sangat merugikan.

Berdasarkan data yang dirilis oleh Facebook, Indonesia menempati urutan ketiga dari perkiraan penyalahgunaan data pribadi oleh Cambridge Analytica setelah Amerika Serikat dan Filipina. Sebanyak 1.096.666 data pribadi pengguna Facebook Indonesia dari total keseluruhan data yang diduga disalahgunakan.

Untuk mengatasi permasalahan di atas, diperlukan sebuah sistem yang dapat mengamankan sebuah sistem agar tidak dicuri ataupun disalahgunakan oleh pihak yang tidak bertanggungjawab. Dalam melakukan pengamanan, dibutuhkan proses enkripsi, Proses enkripsi dari sistem keamanan ini menggunakan Gronsfeld Cipher, dimana cipher tersebut berasal dari hasil perhitungan antara *plaintext,* nilai modulo, nilai A, nilai B dan nilai seed. Agar tingkat keamanannya dapat maksimal, maka membutuhkan pembangkit bilangan acak. Dalam hal ini menggunakan algortima pembangkit bilangan acak LCG (*Linear Congruential*

Generator). LCG (*Linear Congruential Generator*) berfungsi sebagai penghasil kunci acak yang tidak bisa ditebak kunci apa yang akan dihasilkan.

Berdasarkan latar belakang di atas maka penulis merasa tertarik untuk memilih judul **"RANCANGAN SISTEM KEAMANAN ALGORITMA GRONSFELD CIPHER DENGAN PEMBANGKIT BILANGAN ACAK LCG (*LINEAR CONGRUENTIAL GENERATOR*)".**

1.2 Rumusan Masalah

Berdasarkan penjelasan pada bagian latar belakang, maka permasalahan yang menjadi titik utama pembahasan terbagi 3, yaitu :

1. Bagaimana merancang sistem keamanan algoritma Gronsfeld Cipher dengan pembangkit bilangan acak LCG (*Linear Congruential Generator*)?
2. Karakter apa yang bisa diinputkan oleh sistem?
3. Bagaimana *output* yang akan ditampilkan?
4. Bagaimana mengimplementasikan sistem keamanan algoritma Gronsfeld Cipher dengan pembangkit bilangan acak LCG (*Linear Congruential Generator*)?

1.3 Batasan Masalah

Berikut merupakan batasan masalah supaya mudah dipahami :

1. Merancang sebuah sistem keamanan algoritma Gronsfeld Cipher dengan pembangkit bilangan acak LCG (*Linear Congruential Generator*).
2. Karakter yang bisa diinputkan oleh sistem hanya sebuah huruf atau kata, yang disebut dengan plainteks.

3. *Output* yang akan ditampilkan berupa kode enkripsi dan dekripsi algoritma Gronsfeld Cipher dengan pembangkit bilangan acak LCG (*Linear Congruential Generator*).
4. Pengimplementasian algoritma Gronsfeld Cipher dengan pembangkit bilangan acak LCG (*Linear Congruential Generator*) ditampilkan secara rinci proses enkripsi tersebut.

1.4 Tujuan

Adapun tujuan yang ingin dicapai adalah sebagai berikut :

1. Membuat rancangan sebuah sistem keamanan supaya dapat dikembangkan dan diimplementasikan ke dalam sebuah aplikasi utuh.
2. Menampilkan proses enkripsi dan enkripsi secara rinci.
3. Menjelaskan bagaimana menghasilkan kode enkripsi dan dekripsi algoritma Gronsfeld Cipher dengan pembangkit bilangan acak LCG (*Linear Congruential Generator*).

1.5 Manfaat

Dari hasil penelitian ini diharapkan dapat diperoleh beberapa manfaat, yaitu :

1. Dapat dijadikan sumber ketika ada yang ingin mengembangkan pada sebuah sistem.
2. Diharapkan dapat memahami bagaimana cara kerja algoritma Gronsfeld Cipher dengan pembangkit bilangan acak LCG (*Linear Congruential Generator*).

1.6 Metodologi Penelitian

Metode yang digunakan dalam penelitian ini merupakan metode deskriptif. Berikut teknik pengumpulan data yang dilakukan :

1. *Study Literature*

   Data yang dikumpulkan berdasarkan jurnal, buku, *paper*, artikel dan bacaan bacaan yang ada kaitannya dengan judul penelitian.

2. Kajian Pustaka

   Data yang didapat dengan cara menggunakan atau mengumpulkan beberapa sumber yang tertulis yaitu dengan cara membaca mempelajari dan mencatat hal-hal penting yang berhubungan dengan masalah yang sedang dibahas guna mendapat gambaran teoritis.

BAB II

LANDASAN TEORI

2.1 Sistem

Sistem menurut Marliana B. Winanti, S.Si., M.Si. dalam bukunya yang berjudul Sistem Informasi Manajemen (2014), sistem adalah seperangkat komponen yang saling berhubungan dan saling berkerjasama untuk mencapai beberapa tujuan.

Sedangkan definisi sistem menurut Tata Sutabri dalam bukunya yang berjudul Konsep Sistem Informasi (2011), sistem adalah sekelompok unsur yang erat hubungannya satu dengan yang lainnya, yang berfungsi bersama - sama untuk mencapai tujuan tertentu. Suatu sistem mempunyai karakteristik atau sifat tertentu yang mempunyai komponen - komponen (*components*), batas sistem (*boundary*), lingkungan luar sistem (*environments*), penghubung (*interface*), masukan (*input*), keluaran (*output*), pengolah (*process*) dan sasaran (*objectives*) atau tujuan (*goal*).

2.1.1 Klasifikasi Sistem

Menurut Yakub pada buku Pengantar Sistem Informasi (2012), sistem dapat diklasifikasikan dari beberapa sudut pandang, diantaranya adalah sebagai berikut :

1. Sistem diklasifikasikan sebagai sistem abstrak (*abstract system*) dan sistem fisik (*phisycal system*). Sistem abstrak adalah sistem yang berupa pemikiran atau ide-ide yang tidak tampak secara fisik. Misalnya sistem

teologia, yaitu sistem yang berupa pemikiran-pemikiran hubungan antara manusia dengan Tuhan.

2. Sistem fisik merupakan sistem yang ada secara fisik. Misalnya sistem komputer, sistem akuntansi, sistem produksi dan lain sebagainya. Sistem diklasifikasikan sebagai sistem alamiah (*natural system*) dan sistem buatan manusia (*human made system*): 1) Sistem alamiah adalah sistem yang terjadi melalui proses alam, tidak dibuat manusia. Misalnya sistem perputaran bumi. 2) Sistem buatan manusia adalah sistem yang dirancang oleh manusia. Sistem buatan manusia yang melibatkan interaksi antara manusia dengan mesin disebut dengan *human machine system* atau ada yang menyebut dengan *man machine system*. Sistem informasi merupakan contoh *man machine system*, karena menyangkut penggunaan komputer yang berinteraksi dengan manusia.

Sistem diklasifikasikan sebagai sistem tertentu (*deterministic system*) dan sistem tak tentu (*probabilistic system*):

1. Sistem tertentu beroperasi dengan tingkah laku yang sudah dapat diprediksi. Interaksi diantara bagian-bagiannya dapat dideteksi dengan pasti, sehingga keluaran dari sistem dapat diramalkan. Sistem komputer adalah contoh dari sistem tertentu yang tingkah lakunya dapat dipastikan berdasarkan program - program yang dijalankan.
2. Sistem tak tentu adalah sistem yang kondisi masa depannya tidak dapat diprediksi karena mengandung unsur probabilitas.

Sistem diklasifikasikan sebagai sistem tertutup (*closed system*) dan sistem terbuka (*open system*) :

1. Sistem tertutup merupakan sistem yang tidak berhubungan dan tidak terpengaruh dengan lingkungan luarnya. Sistem ini bekerja secara otomatis tanpa adanya turut campur tangan dari pihak diluarnya. Secara teoritis sistem tertutup ini ada, tetapi kenyataannya tidak ada sistem yang benar - benar tertutup, yang ada hanyalah *relatively closed system* (secara relatip tertutup, tidak benar - benar tertutup).
2. Sistem terbuka adalah sistem yang berhubungan dan terpengaruh dengan lingkungan luarnya. Sistem ini menerima masukan dan menghasilkan keluaran untuk lingkungan luar atau subsistem yang lainnya. Karena sistem sifatnya terbuka dan terpengaruh oleh lingkungan luarnya, maka suatu sistem harus mempunyai suatu sistem pengendalian yang baik. Sistem yang baik harus dirancang sedemikian rupa, sehingga secara relatif tertutup karena sistem tertutup akan bekerja secara otomatis dan terbuka hanya untuk pengaruh yang baik saja.

2.2 Keamanan Data

Menurut Sitohang (2013), data dapat diartikan sebagai kenyataan yang digambarkan oleh nilai, bilangan-bilangan, untaian, karakter atau simbol-simbol yang membawa arti tertentu. Informasi sendiri dapat didefinisikan sebagai hasil dari pengolahan data dalam bentuk yang lebih berguna bagi penerimanya, yang digunakan sebagai alat bantu dalam pengambilan.

Keamanan adalah keadaan bebas dari bahaya. Istilah ini dapat digunakan dengan hubungan kepada kejahatan, dan segala bentuk kecelakaan. Keamanan merupakan topik yang luas termasuk keamanan nasional terhadap seorang teroris, keamanan komputer terhadap *hacker*, keamanan rumah terhadap pencuri dan penyusup lainnya, keamanan *financial* terhadap kehancuran ekonomi dan banyak situasi berhubungan lainnya. Kompuer yang terhubung ke jaringan mempunyai ancaman keamanan lebih besar daripada komputer yang tidak terhubung kemana-mana. Dengan mengendalikan *network security* resiko tersebut dapat dikurangi.

2.3 Operasi Modulo

Menurut Khairina, et al. (2017). Operasi modulo merupakan sebuah operasi hitung matematika yang menghasilkan sisa pembagian suatu bilangan. Operasi modulo dalam teori bilangan tergolong kepada aritmetika modulo.

Contoh operasi modulo:

10 mod 6 = 4 → (10 dibagi 6 = 1 sisa 4)

21 mod 11 = 10 → (21 dibagi 11 = 1 sisa 10)

25 mod 8 = 1 → (25 dibagi 8 = 3 sisa 1)

2.4 Algoritma

Algoritma berasal dari kata Al-Khawarizm, yaitu nama seorang tokoh ilmuan Arab yang terkenal. Sehingga kata tersebut diubah orang barat dengan kata "*Algorism*". Yang dulunya kata algoritma banyak dikenal sebagai proses perhitungan dengan angka Arab. Al-Khawarizm menulis buku yang berjudul Kitab *Al Jabar Wal-Muqabala* menjelaskan tentang konsep algoritma. Beliau

menjelaskan bahwa algoritna sangat penting dalam menyelesaikan permasalahan. Algoritma merupakan langkah–langkah menyelesaikan suatu masalah yang efektif dan efisien yang merupakan data *input* dan data *output* sehingga mendapatkan suatu informasi yang asli.

Kemudian menurut Romzi (2012), algoritma adalah prosedur komputasi yang mengambil beberapa nilai atau kumpulan nilai sebagai input kemudian di proses sebagai output sehingga algoritma merupakan urutan langkah komputasi yang mengubah input menjadi output. Definisi algoritma adalah teknik penyusunan langkah-langkah penyelesaian masalah dalam bentuk kalimat dengan jumlah kata terbatas, tetapi tersusun secara sistematis. Dalam ilmu matematika dan komputer, pengertian algoritma merupakan prosedur dari beberapa langkah demi langkah untuk perhitungan.

2.4.1 Kriteria Algoritma

Menurut Donald E. Knuth yang merupakan seorang ilmuwan komputer, matematikawan, dan profesor emeritus di Universitas Stanford Amerika. Beliau juga seorang penulis beberapa buku algoritma abad XX mengemukakan, algoritma yang baik memiliki kriteria sebagai berikut :

1. *Input* dari sisi *input*, minimal program harus memiliki nol atau lebih pengguna. Program pasti memiliki *input*, Yang dimaksud memiliki nol *input* berarti program tidak mendapatkan masukan data dari pengguna secara langsung, namun semua data yang akan digunakan oleh program sudah di deklarasikan di dalam kode program yang akan dieksekusi.

2. *Output* dari sisi *output*, minimal program harus memiliki satu *output*. Program pasti menghasilkan *output* karena program dibuat untuk tujuan tertentu.
3. *Finite* (terbatas) program harus pasti dan berhenti, bukan tak terhingga, suatu program yang dieksekusi haruslah berhenti dan selesai, bukan berjalan terus menerus hingga *hang up* atau *not responding*, dan akhirnya harus.
4. Efisien artinya, program harus efisien, tidak memakan banyak *memory*, tidak melakukan hal – hal yang tidak perlu.

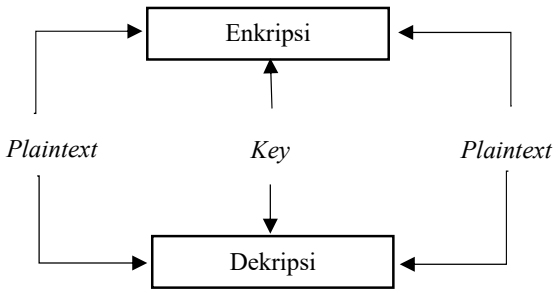
2.5 Kriptografi

Rinaldi Munir (2006), menjelaskan bahwa kriptografi (*cryptography*) berasal dari Bahasa Yunani yaitu *cryptós* artinya *secret* (rahasia), sedangkan "*gráphein*" artinya "writing" (tulisan). Jadi, kriptografi berarti "*secret writing*" (tulisan rahasia).

Sedangkan Rakhmadi, A., dan Nugroho (2016), menjelaskan bahwa kriptografi merupakan sebuah algoritma penyandian untuk menjaga kerahasiaan suatu data. Kriptografi dapat dilakukan dengan cara subtitusi (pergantian huruf) dan transposisi (perpindahan posisi) yang menghasilkan kode-kode rahasia dengan makna yang tidak mudah dipahami orang lain. Istilah-istilah yang sering muncul dalam kriptografi yang penting untuk diketahui yaitu:

1. Plainteks (*plaintext*), pesan asli yang dapat dibaca dan mudah dipahami. Plainteks dapat berupa gambar, pesan tulisan, maupun video bergambar.

2. *Ciphertext*, pesan yang sudah disandikan dari pesan asli menjadi kode-kode yang tidak dapat dipahami karena membentuk kata-kata yang tidak lazim.
3. Enkripsi, proses mengubah *plaintext* menjadi *ciphertext*.
4. Dekripsi, merupakan kebalikan dari enkripsi yaitu proses mengubah *ciphertext* menjadi *plaintext*.



Gambar 2.1. Proses Kriptografi Secara Umum

2.5.1 Tujuan Kriptografi

Menurut Muhammad Khoiruddin Harahap dan Nurul Khairina (2017), tujuan kriptografi terbagi menjadi empat, yaitu:

1. Kerahasiaan (*confidentiality*), adalah layanan yang ditujukan untuk menjaga agar pesan tidak dapat dibaca oleh pihak-pihak yang tidak berhak.
2. Integritas data (data *integrity*), adalah layanan yang menjamin bahwa pesan masih asli dan utuh atau belum pernah dimanipulasi selama pengiriman.
3. Otentikasi (*authentication*), adalah layanan yang berhubungan dengan identifikasi, baik mengidentifikasi kebenaran pihak-pihak yang

berkomunikasi (*user authentication* atau *entity authentication*) maupun mengidentifikasi kebenaran sumber pesan (*origin authentication*).

Nir penyangkalan (*non repudiation*), adalah layanan untuk mencegah entitas yang berkomunikasi melakukan penyangkalan, yaitu pengirim pesan menyangkal melakukan pengiriman atau penerima pesan menyangkal telah menerima pesan.

2.6 Alat Bantu Perancangan Sistem

Alat bantu perancangan sistem merupakan penjelasan tentang peralatan untuk merancang sebuah sistem yang diteliti. Fokus di sini adalah tentang UML *(Unified Modelling Language).*

2.6.1 UML *(Unified Modelling Language)*

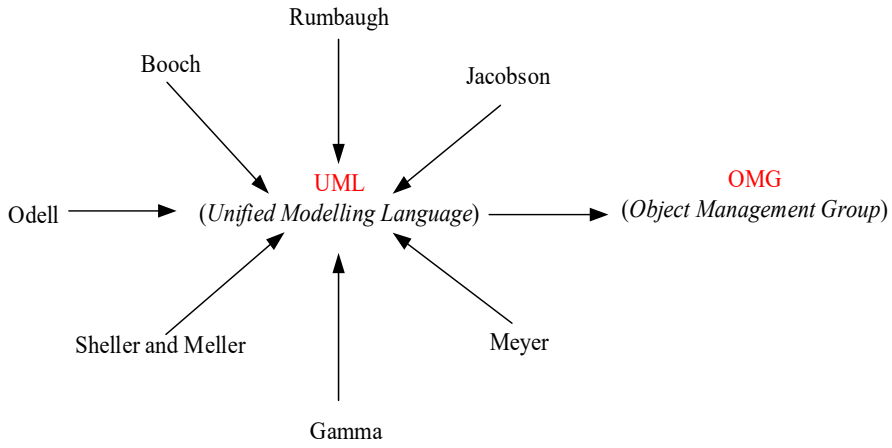
Menurut Windu Gata, Grace (2013) UML *(Unified Modelling Language)* adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem perangkat lunak. UML *(Unified Modelling Language)* menawarkan sebuah standar untuk merancang model sebuah sistem.

Dengan menggunakan UML *(Unified Modelling Language)* kita dapat membuat model untuk semua jenis aplikasi perangkat lunak, dimana aplikasi tersebut dapat berjalan pada perangkat keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML *(Unified Modelling Language)* juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan perangkat lunak dalam bahasa-bahasa berorientasi objek seperti C++, *Java*, C# atau VB.NET. Walaupun

demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C.

Seperti bahasa-bahasa lainnya, UML *(Unified Modelling Language)* mendefinisikan notasi dan *syntax*/semantik. Notasi UML *(Unified Modelling Language)* merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram perangkat lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*).

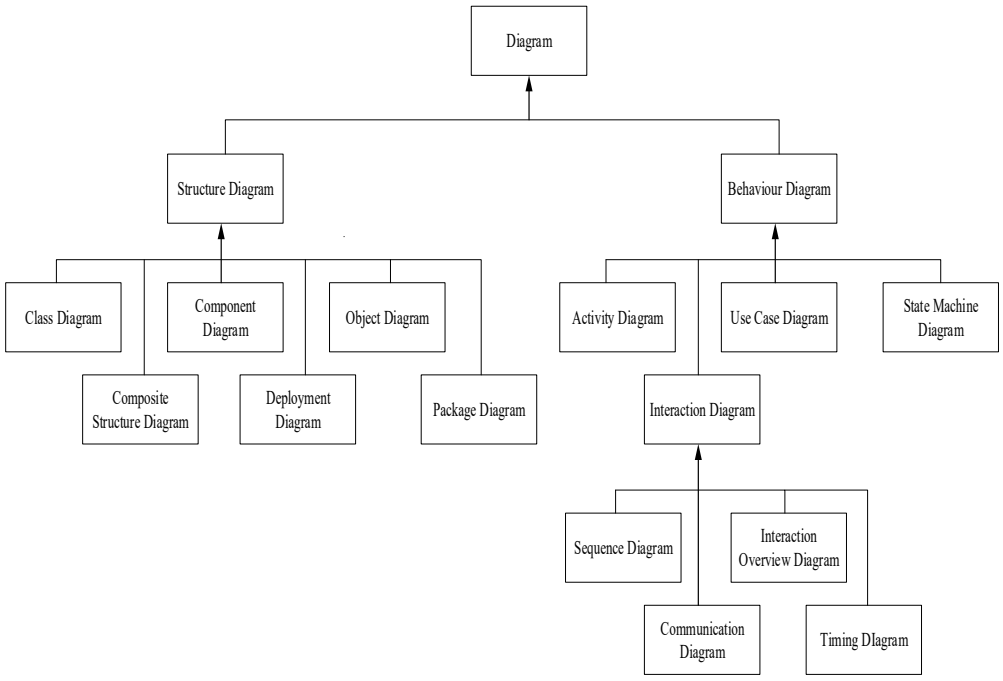
Sejarah UML *(Unified Modelling Language)* sendiri cukup panjang. Sampai era tahun 1990 seperti kita ketahui puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia, diantaranya adalah: metodologi booch, metodologi coad, metodologi OOSE, metodologi OMT, metodologi shlaer-mellor, metodologi wirfs-brock, dsb. Masa itu terkenal dengan masa perang metodologi (*method war*) dalam pendesainan berorientasi objek. Masing-masing metodologi membawa notasi sendiri-sendiri, yang mengakibatkan timbul masalah baru apabila kita bekerjasama dengan *group*/perusahaan lain yang menggunakan metodologi yang berlainan.



Gambar 2.2 Standar Bahasa Pemodelan

Dimulai pada bulan Oktober 1994 Booch, Rumbaugh dan Jacobson, yang merupakan tiga tokoh yang boleh dikata metodologinya banyak digunakan mempelopori usaha untuk penyatuan metodologi pendesainan berorientasi objek. Pada tahun 1995 di-*release draft* pertama dari UML (versi 0.8). Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh *Object Management Group* (OMG – http://www.omg.org). Tahun 1997 UML versi 1.1 muncul, dan saat ini versi terbaru adalah versi 1.5 yang dirilis bulan Maret 2003. Booch, Rumbaugh dan Jacobson menyusun tiga buku serial tentang UML pada tahun 1999. Sejak saat itulah UML telah menjelma menjadi standar bahasa pemodelan untuk aplikasi berorientasi objek.

Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut:

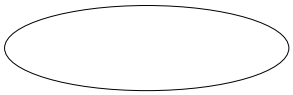


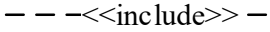
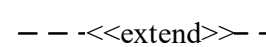
Gambar 2.3 Bagian-bagian dari UML (*Unified Modelling Language*)

1. *Use Case Diagram* (Diagram Kasus Pengguna)

   Use case diagram merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case diagram* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbol-simbol yang digunakan dalam *use case diagram* dapat dilihat dalam tabel 2.1 di bawah ini:

Tabel 2.1 *Use Case Diagram*

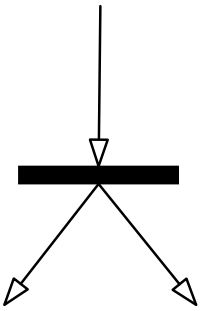
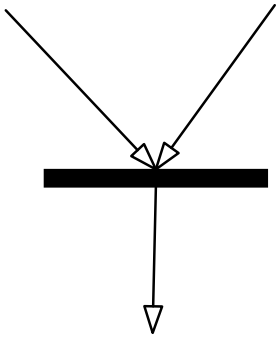
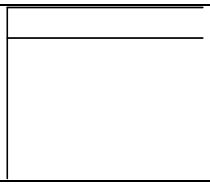
Gambar	Keterangan
	*Use case* menggunakan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan akhir, yang dinyatakan dengan menggunkan kata kerja

Gambar	Keterangan
	*Actor* (Aktor), merupakan *Abstraction* dari orang yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan akhir, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Untuk mengidentifikasikan aktor, ditentukan pembagian kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan *use case,* tetapi tidak memiliki kontrol terhadap *use case.*
	Asosiasi antara aktor dan *use case,* digambarkan dengan garis tanpa panah yang mengidentifikasikan siapa atau siapa yang meminta interaksi secara langsung dan bukannya mengidentifikasikan data.
	Asosiasi antara aktor dan use case, yang menggunankan panah terbuka untuk mengidentifikasikan bila aktor berinteraksi secara pasif dengan sistem.
	*Include,* merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemangilan sebuah fungsi program.
	*Extend,* merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

2. *Activity Diagram* (Diagram Aktivitas)

   Activity Diagram menggambarkan workflow (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *Activity Diagram* terdapat dalam tabel 2.2 di bawah ini:

Tabel 2.2 *Activity Diagram*

Gambar	Keterangan
	*Start Point,* diletakkan pada pojok kiri atas dan merupakan awal aktivitas.
	*End Point,* akhir aktivitas.
	*Activities,* menggambarkan suatu proses/kegiatan bisnis
	*Fork/*percabangan, digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabungkan dua kegiatan paralel menjadi satu.
	*Join,* (penggabungan) atau *rake,* digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Points,* menggambarkan pilihan untuk pengambilan keputusan, benar atau salah.
	*Swimlane,* pembagian *activity diagram* untuk menunjukkan siapa melakukan apa.

3. *Class Diagram* (Diagram Kelas)

   Class Diagram merupakan diagram antar kelas dan penjelasan rinci tiap model dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggungjawab entitas yang menentukan perilaku sistem.

   Class Diagram juga menunjukkan atribut-atribut dan operasi dari sebuah kelas dan *constaint* yang berhubungan dengan objek yang dihubungkan.

   Class Diagram secara khas meliputi *class, assosiations, generalitions* dan *aggregation, attributes, operation* dan *visibility* tingkat akses objek eksternal kepada suatu operasi atau atribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau *cardinality.*

   Untuk jenis-jenis *multiplicity* dan penjelasannya, terdapat pada tabel 2.3 di bawah ini:

Tabel 2.3 *Multiplicity Class Diagram*

Multiplicity	**Penjelasan**
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimal 4

4. *Sequence Diagram* (Diagram Urutan)

   Sequence Diagram menggambarkan kelakuan objek pada *use case,* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterina antar objek. Simbol-simbol yang digunakan dalam *Sequence Diagram* terdapat dalam tabel 2.4 di bawah ini:

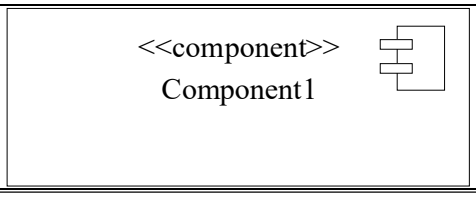
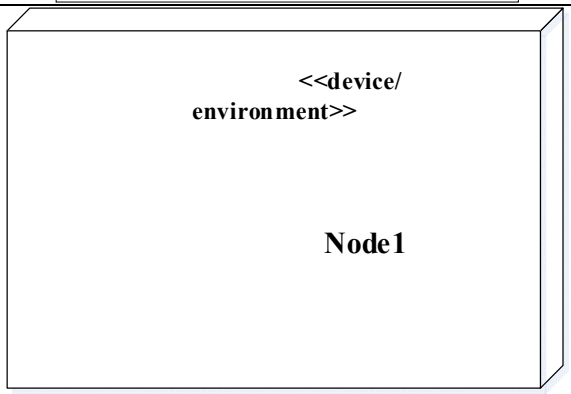
Tabel 2.4 *Sequence Diagram*

Gambar	Keterangan
	*Entity Class,* merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.
	*Boundary Class,* berisi kumpulan kelas yang menjadi *interfaces* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan *form entry* dan *form* cetak
	*Control Class,* suatu objek yang berisi logika aplikasi yang tidak memiliki tanggungjawab kepada entitas. Contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek.
	*Message,* simbol mengirim pesan antar *class.*
	*Recursive,* menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.
	*Activation,* mewakilik sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivasi sebuah operasi.
	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*

5. *Deployment Diagram*

   Deployment diagram digunakan untuk menggambarkan secara rinci bagaimana komponen disusun di insfrastrukstur sistem. Simbol-simbol yang digunakan dalam *deployment diagram* terdapat dalam tabel 2.5 di bawah ini:

Tabel 2.5 *Deployment Diagram*

Gambar	Keterangan
	Pada deployment diagram, komponen-komponen yang ada diletakkan di dalam node untuk memastikan keberadaan posis mereka.
	Node menggambarkan bagian-bagian hardware dalam sebuah sistem. Notasi untuk node digambarkan sebagai sebuah kubus 3 dimensi.
	Sebuah association digambarkan sebagai sebuah garis yang menghubungkan dua node yang mengindikasikan jalur komunikasi antara elemen-elemen *hardware.*

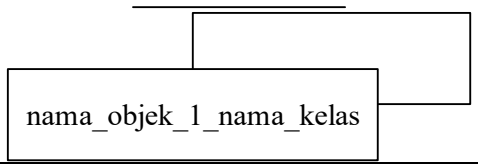
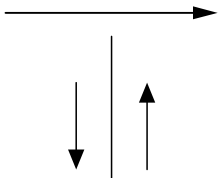
Sumber: Ade Hendini (2016)

6. *Collaboration Diagram*

   Collaboration diagram juga menggambarkan interaksi antar objek seperti *sequence diagram*, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian *message*.

   Setiap *message* memiliki *sequence number*, di mana *message* dari level tertinggi memiliki nomor 1. Messages dari level yang sama memiliki prefiks yang sama. Simbol-simbol yang digunakan dalam *collaboration diagram* terdapat dalam tabel 2.6 di bawah ini:

Tabel 2.6 *Collaboration Diagram*

Gambar	Keterangan
nama_objek_1_nama_kelas	Objek, objek melakukan interaksi pesan.
	*Link,* merupakan relasi antar objek yang menghubungkan objek satu dengan yang lainnya atau dengan dirinya sendiri.
	Stimulus, arah pesan yang terjadi, jika pada suatu *link* ada dua arah pesan yang berbeda maka arah juga digambarkan dua arah pada dua sisi *link*

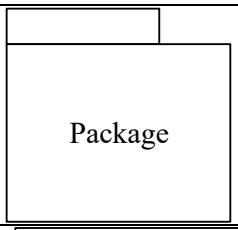
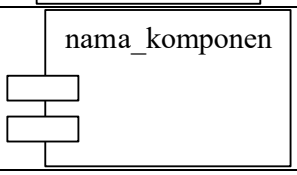
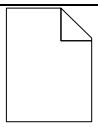
7. *Component Diagram*

Component Diagram menggambarkan struktur dan hubungan antar komponen perangkat lunak, termasuk ketergantungan (*dependency*) diantaranya.

Komponen perangkat lunak adalah modul berisi kode, baik berisi sumber kode maupun *binary code,* baik *library* maupun *executable,* baik yang muncul pada *compile time, link time,* maupun *run time.* Umumnya komponen terbentuk dari beberapa *class* dan/atau *package,* tapi dapat juga dari komponen komponen yang lebih kecil.

Komponen dapat juga berupa *interface,* yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain. Simbol-simbol yang digunakan dalam *component diagram* terdapat dalam tabel 2.7 di bawah ini:

Tabel 2.7 *Component Diagram*

Gambar	Keterangan
	Package, merupakan merupakan sebuah bungkusan dari dari satu atau lebih komponen.
	Komponen sistem.
	Depedency, kebergantungan antar komponen, arah panah mengarah pada komponen yang dipakai.
	*Interface,* sama dengan interface pada pemrograman berorientasi objek, yaitu sebagai antarmuka komponen agar tidak mengakses langsung komponen.
	Link, relasi antar komponen.
	Dokumen, dapat berupa *file* atau *library*

2.6.2 Langkah-Langkah Penggunaan UML

Berikut ini adalah langkah-langkah penggunaan UML:

1. Buatlah daftar *business process* dari level tertinggi untuk mendefinisikan aktivitas dan proses yang mungkin muncul.
2. Petakan *use case* untuk tiap *business process* untuk mendefinisikan dengan tepat fungsionalitas yang harus disediakan oleh sistem. Kemudian perhalus *use case diagram* dan lengkapi dengan *requirement, constraints* dan catatan-catatan lain.

3. Buatlah *deployment diagram* secara kasar untuk mendefinisikan arsitektur fisik sistem.
4. Definisikan *requirement* lain (non-fungsional, *security* dan sebagainya) yang juga harus disediakan oleh sistem.
5. Berdasarkan *use case diagram*, mulailah membuat *activity diagram*.
6. Definisikan objek-objek *level* atas (*package* atau *domain*) dan buatlah *sequence* dan/atau *collaboration diagram* untuk tiap alir pekerjaan. Jika sebuah *use case* memiliki kemungkinan alir normal dan *error*, buatlah satu diagram untuk masing-masing alir.
7. Buatlah rancangan *user interface* model yang menyediakan antarmuka bagi pengguna untuk menjalankan skenario *use case*.
8. Berdasarkan model-model yang sudah ada, buatlah *class diagram*. Setiap *package* atau *domain* dipecah menjadi hirarki *class* lengkap dengan atribut dan metodanya. Akan lebih baik jika untuk setiap *class* dibuat *unit test* untuk menguji fungsionalitas *class* dan interaksi dengan *class* lain.
9. Setelah *class diagram* dibuat, kita dapat melihat kemungkinan pengelompokan *class* menjadi komponen-komponen. Karena itu buatlah component diagram pada tahap ini. Juga, definisikan tes integrasi untuk setiap komponen meyakinkan ia berinteraksi dengan baik.
10. Perhalus *deployment diagram* yang sudah dibuat. Detilkan kemampuan dan *requirement* perangkat lunak, sistem operasi, jaringan, dan sebagainya. Petakan komponen ke dalam node.
11. Mulailah membangun sistem. Ada dua pendekatan yang dapat digunakan :

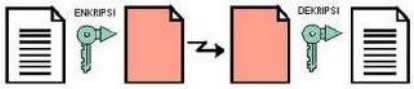
1) Pendekatan *use case*, dengan meng-*assign* setiap *use case* kepada tim pengembang tertentu untuk mengembangkan *unit code* yang lengkap dengan tes.
2) Pendekatan komponen, yaitu meng-*assign* setiap komponen kepada tim pengembang tertentu.

12. Lakukan uji modul dan uji integrasi serta perbaiki model berserta *code*nya. Model harus selalu sesuai dengan *code* yang aktual.
13. Perangkat lunak siap dirilis.

2.7 Kriptografi Simetris

Menurut Basri (2016), kriptografi simteris atau disebut juga algoritma kriptografi konvensional adalah algoritma yang menggunakan kunci untuk proses enkripsi sama dengan kunci untuk proses dekripsi. Algoritma kriptografi simetris dibagi menjadi 2 kategori yaitu algoritma aliran (s*tream ciphers*) dan algoritma blok (*block ciphers*). Pada algoritma aliran, proses penyandiannya berorientasi pada satu bit atau satu *byte* data. Sedang pada algoritma blok, proses penyandiannya berorientasi pada sekumpulan bit atau *byte* data (per blok). Ini adalah jenis kriptografi yang paling umum dipergunakan. Kunci untuk membuat pesan yang disandikan sama dengan kunci untuk membuka pesan yang disandikan itu. Jadi pembuat pesan dan penerimanya harus memiliki kunci yang sama persis. Siapapun yang memiliki kunci tersebut, termasuk pihak-pihak yang tidak diinginkan dapat membuat dan membongkar rahasia *ciphertext*. Problem yang paling jelas disini terkadang bukanlah masalah pengiriman *ciphertext*-nya, melainkan masalah bagaimana menyampaikan kunci simetris tersebut kepada

pihak yang diinginkan. Contoh algoritma kunci simetris yang terkenal adalah DES (Data *Encryption Standard*) dan RC-4, sebagaimana ditunjukkan pada gambar berikut :



Gambar 2.4 Kunci Simetris
(Sumber : Basri, 2016)

Ada beberapa kelebihan menggunakan kunci simetris yang sudah diketahui menurut beberapa ahli. Yang pertama menurut Quasim dan Md.Tabrez (2013), kelebihan menggunakan kunci simetris yaitu kecepatan operasi lebih tinggi apabila dibandingkan dengan algoritma asimetris, walupun hal ini berbanding lurus dengan penambahan ukuran *file*.

Sedangkan menurut Sitinjak et.al (2010), kecepatan proses enkripsi atau dekripsi bergantung pada besarnya ukuran *file*, semakin besar ukuran *file* semakin banyak waktu yang dibutuhkan untuk enkripsi atau dekripsi. Selain itu karena kecepatannya yang cukup tinggi, maka dapat digunakan pada sistem *real-time*. Namun terdapat pula kelemahannya, yaitu untuk tiap pengiriman pesan dengan pengguna yang berbeda dibutuhkan kunci yang berbeda juga, sehingga akan terjadi kesulitan dalam manajemen kunci tersebut, dan permasalahan dalam pengiriman kunci itu sendiri yang disebut "*key distribution problem*".

2.8 Gronsfeld

Menurut Zuhri Ramadhan, et.al (2018), penyandian gronsfeld adalah algoritma kriptografi yang bekerja dengan cara yang sama seperti algoritma penyandian Vigenere. Perbedaan mendasar adalah bahwa Vigenere menggunakan alfabet untuk menggantikan plaintext sementara Gronsfeld menggunakan angka untuk menggeser karakter *plaintext* ke tombol. Algoritma ini menggunakan karakter ASCII 256 untuk proses perhitungan di dalam tabel substitusi. Persamaan berikut adalah rumus yang digunakan untuk menerapkan algoritma Gronsfeld.

Enripsi :

$$E(x) = (P(x) + K(x)) \text{ Mod } 256$$

Dekripsi :

$$E(x) = (P(x) + K(x)) \text{ Mod } 256$$

Proses penambahan terjadi pada enkripsi dan pengurangan terjadi di dekripsi. Jika karakter yang diproses melebihi 256 atau lebih dari 0, maka karakter akan mengalami proses modulo untuk mendapatkan *ciphertext* atau karakter *plaintext*.

Dalam melakukan proses enkripsi teks, tentukan *plaintext* untuk dienkripsi dan kemudian mengubahnya dalam bentuk modal/huruf besar jika diperlukan. Tentukan kunci dalam bentuk angka. Jika panjang kunci tidak sama dengan panjang *plaintext*, maka kuncinya diulang secara berkala sehingga jumlah karakter kunci sama dengan jumlah plaintexts. *Plaintext* akan diubah menjadi desimal. Kode ASCII akan ditambahkan dengan nilai desimal ke kunci. Hasil dari jumlah

jika melebihi 256 akan mengalami proses modulo. Hasil akhir diubah kembali ke bentuk karakter.

2.9 Pembangkit Bilangan Acak

Thomas (2004), menjelaskan bahwa pembangkit bilangan acak atau *random number generator* adalah suatu algoritma yang digunakan untuk menghasilkan urutan-urutan atau *sequence* dari angka-angka sebagai hasil dari perhitungan dengan komputer yang diketahui distribusinya sehingga angka-angka tersebut muncul secara random dan digunakan terus-menerus.

Menurut Zeenat Mahmood, et.al (2012), metode linier kongruen adalah algoritma paling terkenal dan paling banyak digunakan untuk menghasilkan angka acak. Keuntungan praktis mereka adalah kecepatan, kemudahan implementasi, dan ketersediaan kode portabel, parameter serta hasil tes.

2.9.1 LCG (*Linear Congruential Generator*)

Menurut Jhessica Clawdia, et.al (2017), LCG (*Linear Congruential Generator*) merupakan algoritma pembangkit bilangan acak kongruen. Algoritma ini mudah dipahami dan dapat diimplementasikan secara cepat. LCG dapat diterapkan untuk menghasilkan nilai acak atau digunakan untuk mengacak posisi dari sekumpulan nilai, dimana bilangan acak tersebut muncul berdasarkan rumus aritmatika yang sudah ditetapkan.

LCG didefinisikan dalam relasi rekurens (Schneier 1996) dengan rumus :

$Zi = (a * Zi - 1 + b) \bmod m$

Keterangan :

Zi = bilangan acak ke i dari deretnya

Zi - 1 = bilangan acak sebelumnya

a = faktor pengali

b = penambah (*increment*)

m = modulus (a, b, dan m semuanya konstans)

LCG mempunyai periode tidak lebih besar dari m, dan kebanyakan kasus periodenya kurang dari m, maksudnya adalah deret bilangan acak yang dihasilkan tidak lebih banyak dari modulonya. Menurut Rinaldi Munir (2006), LCG mempunyai periode penuh (m – 1) jika memenuhi syarat berikut :

1. b relative prima terhadap m
2. a – 1 dapat dibagi dengan semua faktor prima dari m
3. a – 1 adalah kelipatan 4, jika m adalah kelipatan 4
4. m > maks (a, b, X0)
5. a > 0, b > 0

X0 adalah kunci pembangkit atau disebut juga umpan (seed). Secara teori LCG mampu menghasilkan bilangan acak, namun sensitif terhadap pemilihan nilai-nilai a, b, dan m. Pemilihan nilai-nilai yang tidak sesuai dapat mempengaruhi

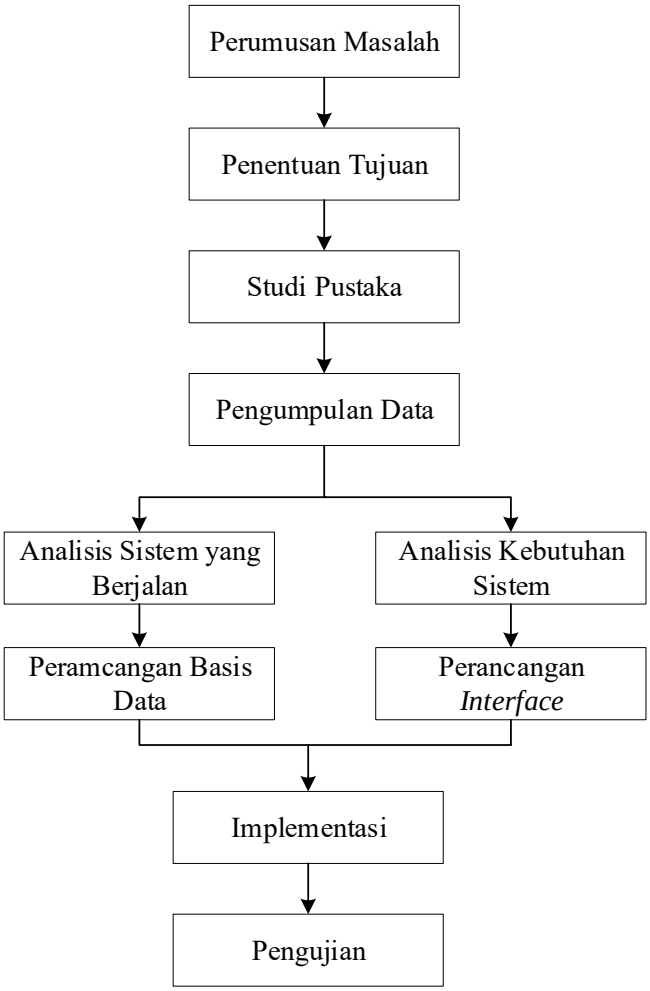
implementasi pada LCG. LCG memiliki kelebihan pada kecepatannya karena sedikit membutuhkan operasi bit. namun kemunculan bilangan acaknya mudah diprediksi sehingga tidak aman secara kriptografi, namun demikian LCG tetap berguna untuk latihan awal penerapan enkripsi dengan metode stream cipher menggunakan kunci yang dibangkitkan oleh algoritma LCG.

BAB III

METODE PENELITIAN

3.1 Tahapan Penelitian

Tahapan penelitian merupakan cara-cara memperoleh data yang digunakan untuk penelitian. Untuk lebih memahami peneliti dan pembaca memahami penelitian, maka dibawah ini merupakan tahapan penelitian berbentuk *flowchart*.



Gambar 3.1 *Flowchart* Tahapan Penelitian

3.2 Metode Pengumpulan Data

Berikut merupakan metode pengumpulan data yang dilakukan :

1. *Study Literature*

   Data yang dikumpulkan berdasarkan jurnal, buku, *paper*, artikel dan bacaan bacaan yang ada kaitannya dengan judul penelitian.

2. Kajian Pustaka

   Data yang didapat dengan cara menggunakan atau mengumpulkan beberapa sumber yang tertulis yaitu dengan cara membaca mempelajari dan mencatat hal-hal penting yang berhubungan dengan masalah yang sedang dibahas guna mendapat gambaran teoritis.

3.3 Analisa

Berdasarkan Kamus Besar Bahasa Indonesia daring (2016), analisa berasal dari kata analisis, yang artinya penyelidikan terhadap suatu peristiwa untuk mengatahui keadaan sebenarnya. Jadi, analisa merupakan penguraian suatu pokok atas berbagai bagiannya dan penelaahan bagian itu sendiri, serta hubungan antar bagian itu sendiri serta hubungan antar bagian untuk memperoleh pengertian yang tepat dan pemahaman arti keseluruhan.

Berdasarkan definisi di atas, dapat disimpulkan bahwa analisa merupakan sebuah teknik pemacahan masalah yang menguraikan sebuah sistem menjadi bagian-bagian komponen dengan tujuan mempelajari seberapa baik bagian-bagian komponen tersebut bekerja dan berinteraksi.

3.3.1 Analisa Kebutuhan Sistem

1. Analisa Kebutuhan Fungsional

Kebutuhan fungsional merupakan daftar kebutuhan apa saja yang nantinya dilakukan oleh sistem. Kebutuhan fungsional juga berisi inlayerasi apa saja yang harus ada dan dihasilkan oleh sistem. Berikut kebutuhan fungsional yang terdapat pada sistem yang dibangun:

a. Mengimplementasikan penggunaan *Visual Basic.Net* 2012 dalam membuat sistem keamanan algoritma Gronsfeld Cipher dengan pembangkit bilangan acak LCG *(Linear Congruential Generator).*
b. Kode enkripsi dan dekripsi yang didapat berdasarkan hasil perhitungan Gronsfeld Cipher dan pembangkit bilangan acak LCG *(Linear Congruential Generator).*

2. Analisa Kebutuhan Non Fungsional

Kebutuhan ini adalah tipe kebutuhan yang berisi properti perilaku yang dimiliki oleh sistem. Berikut adalah kebutuhan nonfungsional yang dimiliki sistem:

a. Operasional
    a) Dapat digunakan pada sistem operasi *Microsoft Windows 7/8/10* secara *stand alone*.
    b) Sistem dibangun dengan menggunakan komponen *IDE Visual Studio 2010.*

c) Spesifikasi komputer standard untuk menjalankan *Visual Studio 2012* yaitu *Processor Pentium IV* 2,6 GHz, Memori 512 MB, Kartu Grafik 128 MB.

b. Kinerja

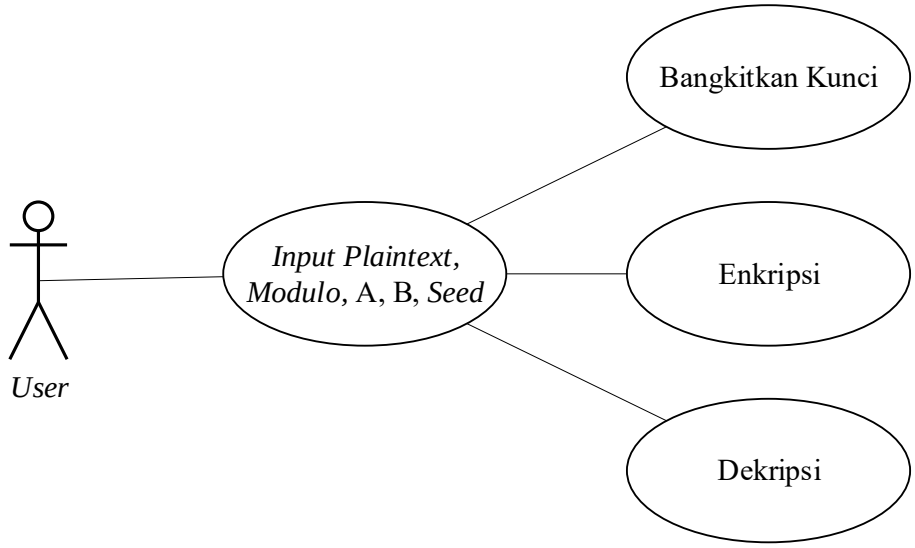
Waktu yang diperlukan dalam mengeksekusi sistem rancangan keamanan algoritma Grondsfeld Cipher dengan pembangkit bilangan acak LCG *(Linear Congruential Generator)* dibangun sangat sederhana dan cukup ringan, sehingga eksekusi tampilannya sangat cepat dan mudah digunakan.

3.4 Perancangan Sistem

Tahap selanjutnya yang dilakukan dalam penyusunan skirpsi ini adalah perancangan sistem. Perancangan sistem menjelaskan bagaimana sebuah sistem akan dibangun yang bertujuan untuk mempermudah membangun sebuah sistem dan juga digunakan untuk dokumentasi.

3.4.1 *Use Case Diagram* (Diagram Kasus Pengguna)

Berikut ini merupakan diagram *use case* dari "Sistem Rancangan Keamanan Algoritma Grondsfeld Cipher dengan Pembangkit Bilangan Acak LCG *(Linear Congruential Generator)*" Terlihat pada gambar 3.2 di bawah ini:



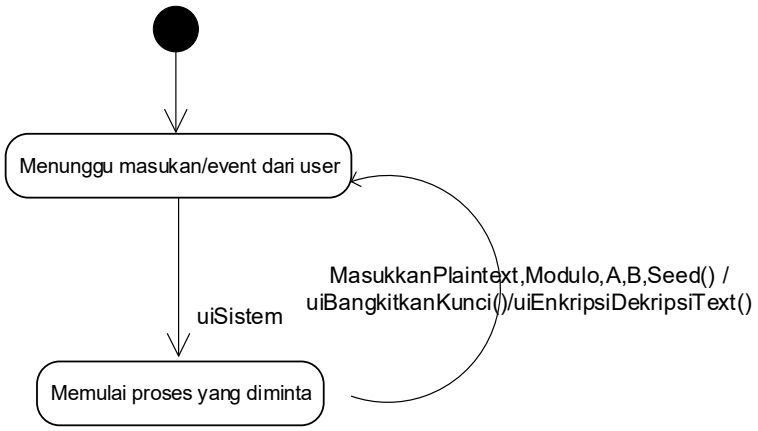
Gambar 3.2 Diagram *Use Case* Sistem

Pada diagram *use case* di atas, aktor yang didefenisikan pada sistem hanya satu, yaitu pengguna. Pengguna adalah orang yang menjalankan sistem. Ketika sistem dijalankan akan menampilkan *User Interface* kemudian pengguna memasukkan *Plaintext,* Modulo, A, B, Seed, kemudian pengguna meng-klik tombol "Bangkitkan Kunci" untuk mendapatkan kunci yang acak, setelah itu *user* meng-klik tombol "Enkrip", maka kunci enkripsi didapatkan. Jika ingin mengembalikan kunci enkripsi, pengguna bisa meng-klik tombol dekripsi.

3.4.2 Diagram Status

Berikut ini merupakan diagram status dari sistem "Sistem Rancangan Keamanan Algoritma Grondsfeld Cipher dengan Pembangkit Bilangan Acak LCG *(Linear Congruential Generator)*".

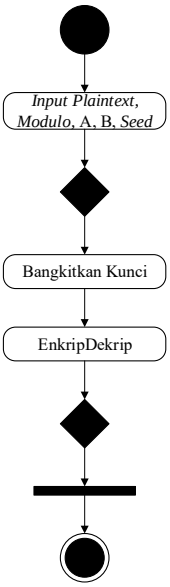
Objek : main dari kelas Main



Gambar 3.3 Diagram Status Objek : main dari kelas Main

3.4.3 *Activity Diagram* (Diagram Aktivitas)

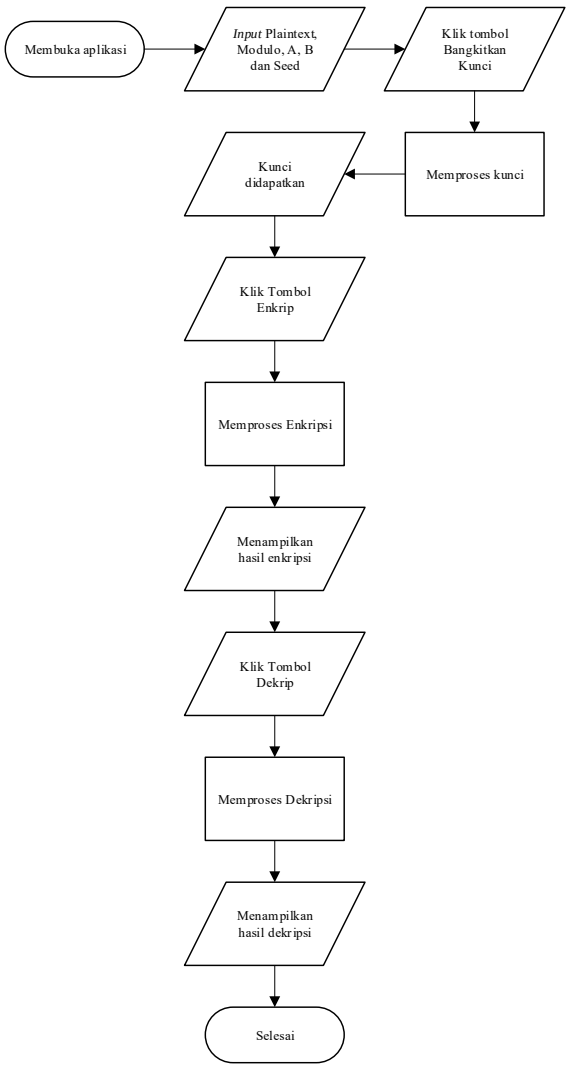
Berikut ini merupakan diagram aktivitas dari aplikasi "Sistem Rancangan Keamanan Algoritma Grondsfeld Cipher Dengan Pembangkit Bilangan Acak LCG *(Linear Congruential Generator)*". Terlihat pada gambar 3.4 di bawah ini:



Gambar 3.4 *Activity Diagram*

3.4.4 Perancangan *Flowchart*

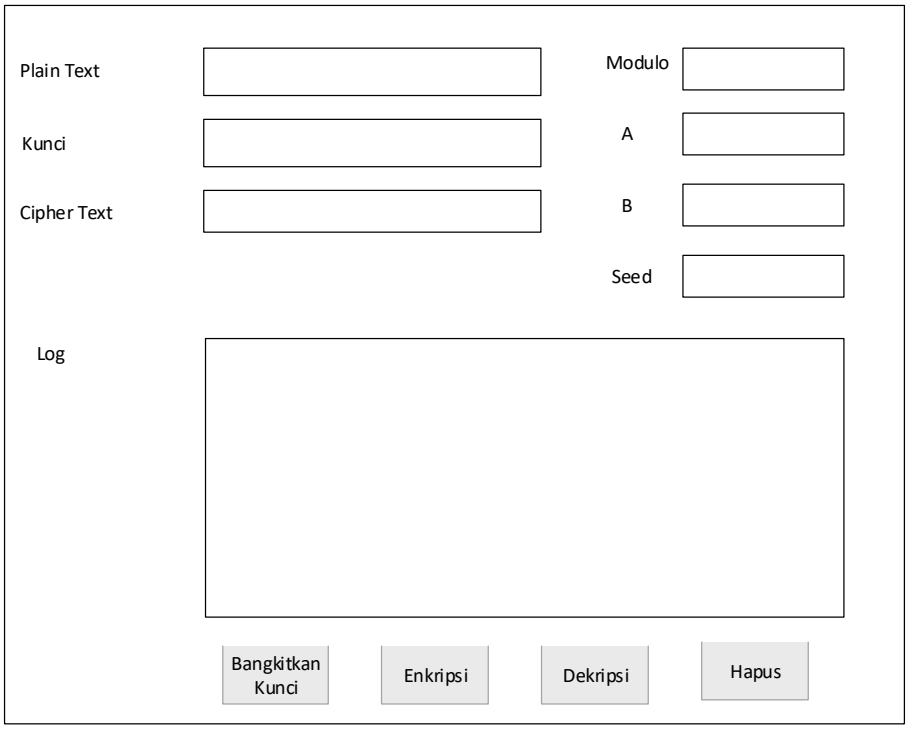
Rancangan ini digunakan untuk merancang dan merepresentasikan program. Sebelum pembuatan program, fungsinya adalah mempermudah *programmer* dalam menentukan alur logika program yang akan dibuat. Sesudah pembuatan program fungsinya adalah untuk menjelaskan alur program kepada orang lain atau user. *Flowchart* untuk perancangan enkripsi dan deskripsi dapat dilihat pada gambar 3.5 berikut ini:



Gambar 3.5 *Flowchart* Mendapatkan Kode Enkripsi dan Dekripsi

3.5 Perancangan Tampilan

Di bawah ini merupakan rancangan tampilan aplikasi yang akan dibuat.



Gambar 3.6 Perancangan Tampilan

Keterangan Gambar 3.3 :

1. *Plaintext*

   Fungsi dari kolom tersebut sebagai *input* kata-kata yang nantinya akan diproses menjadi kunci oleh sistem.

2. Kunci

   Fungsi dari kolom tersebut menampilkan hasil kunci setelah tombol "Bangkitkan Kunci" di-klik.

3. *Cipher Text*

   Fungsi dari kolom tersebut menampilkan *ciphertext* hasil enkripsi dari *Plaintext.*

4. LOG

   Fungsi LOG yaitu menampilkan hasil bagaimana perhitungan pembangkitan kunci, dan proses enkripsi-dekripsi yang sudah diproses oleh sistem.

5. A

   A merupakan variabel pertama dari bilangan prima kedua.

6. B

   B merupakan variabel kedua dari bilangan prima kedua.

7. Seed

   Seed merupakan *input* angka dari kunci pengulangan.

8. Modulo

   Fungsi modulo sebagai *input* angka ketentuan agar menghasilkan hasil sisa pembagian dari *input* A, B dan Seed.

9. Enkrip

   Tombol enkrip berfungsi untuk menampilkan kode enkripsi dari plain teks.

10. Dekrip

    Tombol dekrip berfungsi untuk mendekripsikan plain teks yang sebelumnya sudah di enkripsi.

BAB IV

IMPLEMENTASI DAN PENGUJIAN APLIKASI

4.1 Kebutuhan Spesifikasi Minimum *Hardware* dan *Software*

Dalam implementasi aplikasi yang sudah dirancang, maka dibutuhkan beberapa perangkat yang sangat penting agar aplikasi yang sudah dirancang dapat berjalan pada saat diimplementasikan.

4.1.1 *Hardware* (Perangkat Keras)

Perangkat keras yang dapat disediakan adalah berupa aplikasi komputer yang lengkap. pada saat aplikasi ini diuji, digunakan spesifikasi PC *(Personal computer)* sebagai berikut:

Processor : Intel® Celeron ® CPU 1007U @ 1.50GHz 1.50GHz

RAM : 2.00 GB (1.80 GB usable)

HD : 500 GB

Monitor : 19”

4.1.2 *Software* (Perangkat Lunak)

Untuk mengimplementasikan aplikasi ini, dibutuhkan beberapa perangkat lunak yang sudah ada di komputer yang digunakan. Berikut perangkat lunak yang digunakan:

1. Sistem Operasi

   Sistem operasi yang digunakan dalam pengujian ini adalah *Windows* 10, dan aplikasi ini masih bisa berjalan dengan baik pada aplikasi operasi *Windows XP*, *Windows* 7, *Windows* 8, dan *Windows* 8,1.

Adapun perangkat minimum agar aplikasi ini tetap berjalan secara optimal adalah sebagai berikut:

Processor : Pentium IV 2,6 GHz

RAM : 512 MB

HDD : 64 GB

Layar : 14 *inch*

4.2 Pembahasan dan Pengujian Aplikasi

Pada sub bab ini, akan dijelaskan secara rinci bagaimana aplikasi dapat menghasilkan kunci LCG, kunci enkripsi dan dekripsi.

4.2.1 Pembahasan Bangkitkan Kunci LCG

Dalam proses pembangkitan kunci LCG, maka akan dijelaskan di bawah ini:

Teks yang akan dihitung:

UNIVERSITAS PEMBANGUNAN PANCA BUDI

MOD = 256, A = 7, B = 9, Seed = 40

Untuk menghitung pembangkitan kunci LCG, maka terdapat persamaan seperti di bawah ini:

$$\text{Kunci LCG [i]} = (\text{seed} * A + B) \bmod 256$$

Berdasarkan persamaan di atas, maka perhitungan untuk menghasilkan kunci LCG yaitu:

Kunci LCG [1] = (40 * 7 + 9) Mod 256

Kunci LCG [1] = 33

Kunci LCG [2] = (33 * 7 + 9) Mod 256

Kunci LCG [2] = 240

Kunci LCG [3] = (240 * 7 + 9) Mod 256

Kunci LCG [3] = 153

Kunci LCG [4] = (153 * 7 + 9) Mod 256

Kunci LCG [4] = 56

Kunci LCG [5] = (56 * 7 + 9) Mod 256

Kunci LCG [5] = 145

Kunci LCG [6] = (145 * 7 + 9) Mod 256

Kunci LCG [6] = 0

Kunci LCG [7] = (0 * 7 + 9) Mod 256

Kunci LCG [7] = 9

Kunci LCG [8] = (9 * 7 + 9) Mod 256

Kunci LCG [8] = 72

Kunci LCG [9] = (72 * 7 + 9) Mod 256

Kunci LCG [9] = 1

Kunci LCG [10] = (1 * 7 + 9) Mod 256

Kunci LCG [10] = 16

Kunci LCG [11] = (16 * 7 + 9) Mod 256

Kunci LCG [11] = 121

Kunci LCG [12] = (121 * 7 + 9) Mod 256

Kunci LCG [12] = 88

Kunci LCG [13] = (88 * 7 + 9) Mod 256

Kunci LCG [13] = 113

Kunci LCG [14] = (113 * 7 + 9) Mod 256

Kunci LCG [14] = 32

Kunci LCG [15] = (32 * 7 + 9) Mod 256

Kunci LCG [15] = 233

Kunci LCG [16] = (233 * 7 + 9) Mod 256

Kunci LCG [16] = 104

Kunci LCG [17] = (104 * 7 + 9) Mod 256

Kunci LCG [17] = 225

Kunci LCG [18] = (225 * 7 + 9) Mod 256

Kunci LCG [18] = 48

Kunci LCG [19] = (48 * 7 + 9) Mod 256

Kunci LCG [19] = 89

Kunci LCG [20] = (89 * 7 + 9) Mod 256

Kunci LCG [20] = 120

Kunci LCG [21] = (120 * 7 + 9) Mod 256

Kunci LCG [21] = 81

Kunci LCG [22] = (81 * 7 + 9) Mod 256

Kunci LCG [22] = 64

Kunci LCG [23] = (64 * 7 + 9) Mod 256

Kunci LCG [23] = 201

Kunci LCG [24] = (201 * 7 + 9) Mod 256

Kunci LCG [24] = 136

Kunci LCG [25] = (136 * 7 + 9) Mod 256

Kunci LCG [25] = 193

Kunci LCG [26] = (193 * 7 + 9) Mod 256

Kunci LCG [26] = 80

Kunci LCG [27] = (80 * 7 + 9) Mod 256

Kunci LCG [27] = 57

Kunci LCG [28] = (57 * 7 + 9) Mod 256

Kunci LCG [28] = 152

Kunci LCG [29] = (152 * 7 + 9) Mod 256

Kunci LCG [29] = 49

Kunci LCG [30] = (49 * 7 + 9) Mod 256

Kunci LCG [30] = 96

Kunci LCG [31] = (96 * 7 + 9) Mod 256

Kunci LCG [31] = 169

Kunci LCG [32] = (169 * 7 + 9) Mod 256

Kunci LCG [32] = 168

Kunci LCG [33] = (168 * 7 + 9) Mod 256

Kunci LCG [33] = 161

Berdasarkan perhitungan di atas, maka hasil pembangkitan kunci LCG dari teks "UNIVERSITAS PEMBANGUNAN PANCA BUDI" adalah:

33 240 153 56 145 0 9 72 1 16 121 88 113 32 233 104 225 48 89 120 81

64 201 136 193 80 57 152 49 96 169 168 161

Agar dapat lebih memahami perhitungan pembangkitan kunci LCG, maka perhitungan dapat di dilihat dalam tabel 4.1 di bawah ini:

Tabel 4.1 Tabel perhitungan pembangkitan kunci

PLAINTEXT	UNIVERSITAS PEMBANGUNAN PANCA BUDI
PERHITUNGAN	
Kunci LCG [1] = (40 * 7 + 9) Mod 256 Kunci LCG [1] = 33	
Kunci LCG [2] = (33 * 7 + 9) Mod 256 Kunci LCG [2] = 240	

PLAINTEXT	UNIVERSITAS PEMBANGUNAN PANCA BUDI
Kunci LCG [3] = (240 * 7 + 9) Mod 256 Kunci LCG [3] = 153	
Kunci LCG [4] = (153 * 7 + 9) Mod 256 Kunci LCG [4] = 56	
Kunci LCG [5] = (56 * 7 + 9) Mod 256 Kunci LCG [5] = 145	
Kunci LCG [6] = (145 * 7 + 9) Mod 256 Kunci LCG [6] = 0	
Kunci LCG [7] = (0 * 7 + 9) Mod 256 Kunci LCG [7] = 9	
Kunci LCG [8] = (9 * 7 + 9) Mod 256 Kunci LCG [8] = 72	
Kunci LCG [9] = (72 * 7 + 9) Mod 256 Kunci LCG [9] = 1	
Kunci LCG [10] = (1 * 7 + 9) Mod 256 Kunci LCG [10] = 16	
Kunci LCG [11] = (16 * 7 + 9) Mod 256 Kunci LCG [11] = 121	
Kunci LCG [12] = (121 * 7 + 9) Mod 256 Kunci LCG [12] = 88	
Kunci LCG [13] = (88 * 7 + 9) Mod 256 Kunci LCG [13] = 113	
Kunci LCG [14] = (113 * 7 + 9) Mod 256 Kunci LCG [14] = 32	
Kunci LCG [15] = (32 * 7 + 9) Mod 256 Kunci LCG [15] = 233	
Kunci LCG [16] = (233 * 7 + 9) Mod 256 Kunci LCG [16] = 104	
Kunci LCG [17] = (104 * 7 + 9) Mod 256 Kunci LCG [17] = 225	
Kunci LCG [18] = (225 * 7 + 9) Mod 256 Kunci LCG [18] = 48	
Kunci LCG [19] = (48 * 7 + 9) Mod 256 Kunci LCG [19] = 89	
Kunci LCG [20] = (89 * 7 + 9) Mod 256 Kunci LCG [20] = 120	
Kunci LCG [21] = (120 * 7 + 9) Mod 256 Kunci LCG [21] = 81	
Kunci LCG [22] = (81 * 7 + 9) Mod 256 Kunci LCG [22] = 64	
Kunci LCG [23] = (64 * 7 + 9) Mod 256 Kunci LCG [23] = 201	
Kunci LCG [24] = (201 * 7 + 9) Mod 256 Kunci LCG [24] = 136	
Kunci LCG [25] = (136 * 7 + 9) Mod 256 Kunci LCG [25] = 193	
Kunci LCG [26] = (193 * 7 + 9) Mod 256 Kunci LCG [26] = 80	
Kunci LCG [27] = (80 * 7 + 9) Mod 256 Kunci LCG [27] = 57	
Kunci LCG [28] = (57 * 7 + 9) Mod 256 Kunci LCG [28] = 152	
Kunci LCG [29] = (152 * 7 + 9) Mod 256 Kunci LCG [29] = 49	

PLAINTEXT	UNIVERSITAS PEMBANGUNAN PANCA BUDI
Kunci LCG [30] = (49 * 7 + 9) Mod 256 Kunci LCG [30] = 96	
Kunci LCG [31] = (96 * 7 + 9) Mod 256 Kunci LCG [31] = 169	
Kunci LCG [32] = (169 * 7 + 9) Mod 256 Kunci LCG [32] = 168	
HASIL	**33 240 153 56 145 0 9 72 1 16 121 88 113 32 233 104 225 48 89 120 81 64 201 136 193 80 57 152 49 96 169 168 161**

4.2.2 Pembahasan Enkripsi

Untuk menghitung kunci enkripsi dari kunci LCG, dapat dituliskan persamaan di bawah ini:

Enkripsi = Bilangan ASCII + Seed (Kunci LCG) Mod 256

Dalam proses pembangkitan kunci LCG, didapat hasil:

33 240 153 56 145 0 9 72 1 16 121 88 113 32 233 104 225 48 89 120 81

64 201 136 193 80 57 152 49 96 169 168 161

Maka, perhitungan kunci enkripsi dari kunci LCG yaitu:

U = 85 + 40 Mod 256 = 125 = }

N = 78 + 33 Mod 256 = 111 = o

I = 73 + 240 Mod 256 = 57 = 9

V = 86 + 153 Mod 256 = 239 = ï

E = 69 + 56 Mod 256 = 125 = }

R = 82 + 145 Mod 256 = 227 = ã

S = 83 + 0 Mod 256 = 83 = S

I = 73 + 9 Mod 256 = 82 = R

T = 84 + 72 Mod 256 = 156 = œ

A = 65 + 1 Mod 256 = 66 = B

S = 83 + 16 Mod 256 = 99 = c

= 32 + 121 Mod 256 = 153 = ™

P = 80 + 88 Mod 256 = 168 = ¨

E = 69 + 113 Mod 256 = 182 = ¶

M = 77 + 32 Mod 256 = 109 = m

B = 66 + 233 Mod 256 = 43 = +

A = 65 + 104 Mod 256 = 169 = ©

N = 78 + 225 Mod 256 = 47 = /

G = 71 + 48 Mod 256 = 119 = w

U = 85 + 89 Mod 256 = 174 = ®

N = 78 + 120 Mod 256 = 198 = Æ

A = 65 + 81 Mod 256 = 146 = ’

N = 78 + 64 Mod 256 = 142 =

= 32 + 201 Mod 256 = 233 = é

P = 80 + 136 Mod 256 = 216 = Ø

A = 65 + 193 Mod 256 = 2 = _

N = 78 + 80 Mod 256 = 158 =

C = 67 + 57 Mod 256 = 124 = |

A = 65 + 152 Mod 256 = 217 = Ù

= 32 + 49 Mod 256 = 81 = Q

B = 66 + 96 Mod 256 = 162 = ¢

U = 85 + 169 Mod 256 = 254 = þ

D = 68 + 168 Mod 256 = 236 = ì

I = 73 + 161 Mod 256 = 234 = ê

Berdasarkan perhitungan di atas, maka hasil dekripsi dari kunci enkripsi "UNIVERSITAS PEMBANGUNAN PANCA BUDI" adalah:

}o9ï}ãSRœBc™¨¶m+©/w®Æ' éØ |ÙQ¢þìê

Agar dapat lebih memahami perhitungan kunci enkripsi LCG dan gronsfeld maka dapat di dilihat dalam tabel 4.2 di bawah ini:

Tabel 4.2 Tabel perhitungan kunci enkripsi

PLAINTEXT	ANGKA ASCII	KUNCI AWAL	MOD	HASIL PERHITUNGAN	KARAKTER ASCII
U	85	40	256	125	}
N	78	33	256	111	o
I	73	240	256	57	9
V	86	153	256	239	ï
E	69	56	256	125	}
R	82	145	256	227	ã
S	83	0	256	83	S
I	73	9	256	82	R
T	84	72	256	15	œ
A	65	1	256	66	B
S	83	16	256	99	c
	32	121	256	153	™
P	80	88	256	168	¨
E	69	113	256	182	¶
M	77	32	256	109	m
B	66	233	256	43	+
A	65	104	256	169	©
N	78	255	256	47	/
G	71	48	256	119	w
U	85	89	256	174	®
N	78	120	256	198	Æ
A	65	81	256	146	'
N	78	64	256	142	

PLAINTEXT	ANGKA ASCII	KUNCI AWAL	MOD	HASIL PERHITUNGAN	KARAKTER ASCII
	32	201	256	233	é
P	80	136	256	216	Ø
A	65	193	256	2	
N	78	80	256	158	
C	67	57	256	124	|
A	65	152	256	217	Ù
	32	49	256	81	Q
B	66	96	256	162	¢
U	85	169	256	254	þ
D	68	168	256	236	ì
I	73	161	256	234	ê
HASIL ENKRIPSI		**}o9ï}ãSRœBc™¨¶m+©/w®Æ' éØ |ÙQ¢þìê**			

4.2.3 Pembahasan Dekripsi

Untuk menghitung kunci dekripsi dari kunci enkripsi, dapat dituliskan persamaan di bawah ini:

Dekripsi = Hasil enkripsi - Seed (Kunci LCG) Mod 256

Kemudian, dalam proses pembangkitan kunci LCG, didapat hasil:

33 240 153 56 145 0 9 72 1 16 121 88 113 32 233 104 225 48 89 120 81 64 201 136 193 80 57 152 49 96 169 168 161

Dalam proses perhitungan kunci enkripsi, didapat hasil:

}o9ï}ãSRœBc™¨¶m+©/w®Æ' éØ |ÙQ¢þìê

Berdasarkan hasil dari proses pembangkitan kunci ke enkripsi di atas, maka proses perhitungan dekripsi akan dijelaskan di bawah ini:

} = 125 - 40 Mod 256 = 85 = U

o = 111 - 33 Mod 256 = 78 = N

9 = 57 - 240 Mod 256 = 73 = I

ï = 239 - 153 Mod 256 = 86 = V

} = 125 - 56 Mod 256 = 69 = E

ã = 227 - 145 Mod 256 = 82 = R

S = 83 - 0 Mod 256 = 83 = S

R = 82 - 9 Mod 256 = 73 = I

œ = 156 - 72 Mod 256 = 84 = T

B = 66 - 1 Mod 256 = 65 = A

c = 99 - 16 Mod 256 = 83 = S

™ = 153 - 121 Mod 256 = 32 =

¨ = 168 - 88 Mod 256 = 80 = P

¶ = 182 - 113 Mod 256 = 69 = E

m = 109 - 32 Mod 256 = 77 = M

+ = 43 - 233 Mod 256 = 66 = B

© = 169 - 104 Mod 256 = 65 = A

/ = 47 - 225 Mod 256 = 78 = N

w = 119 - 48 Mod 256 = 71 = G

® = 174 - 89 Mod 256 = 85 = U

Æ = 198 - 120 Mod 256 = 78 = N

’ = 146 - 81 Mod 256 = 65 = A

 = 142 - 64 Mod 256 = 78 = N

é = 233 - 201 Mod 256 = 32 =

Ø = 216 - 136 Mod 256 = 80 = P

_ = 2 - 193 Mod 256 = 65 = A

= 158 - 80 Mod 256 = 78 = N

| = 124 - 57 Mod 256 = 67 = C

Ù = 217 - 152 Mod 256 = 65 = A

Q = 81 - 49 Mod 256 = 32 =

¢ = 162 - 96 Mod 256 = 66 = B

þ = 254 - 169 Mod 256 = 85 = U

ì = 236 - 168 Mod 256 = 68 = D

ê = 234 - 161 Mod 256 = 73 = I

Berdasarkan perhitungan di atas, maka hasil dekripsi dari kunci enkripsi " }o9ï}ãSRœBc™¨¶m+©/w®Æ' éØ |ÙQ¢þìê " adalah:

UNIVERSITAS PEMBANGUNAN PANCA BUDI

Agar dapat lebih memahami perhitungan dekripsi LCG dan Gronsfeld, maka dapat di dilihat dalam tabel 4.3 di bawah ini:

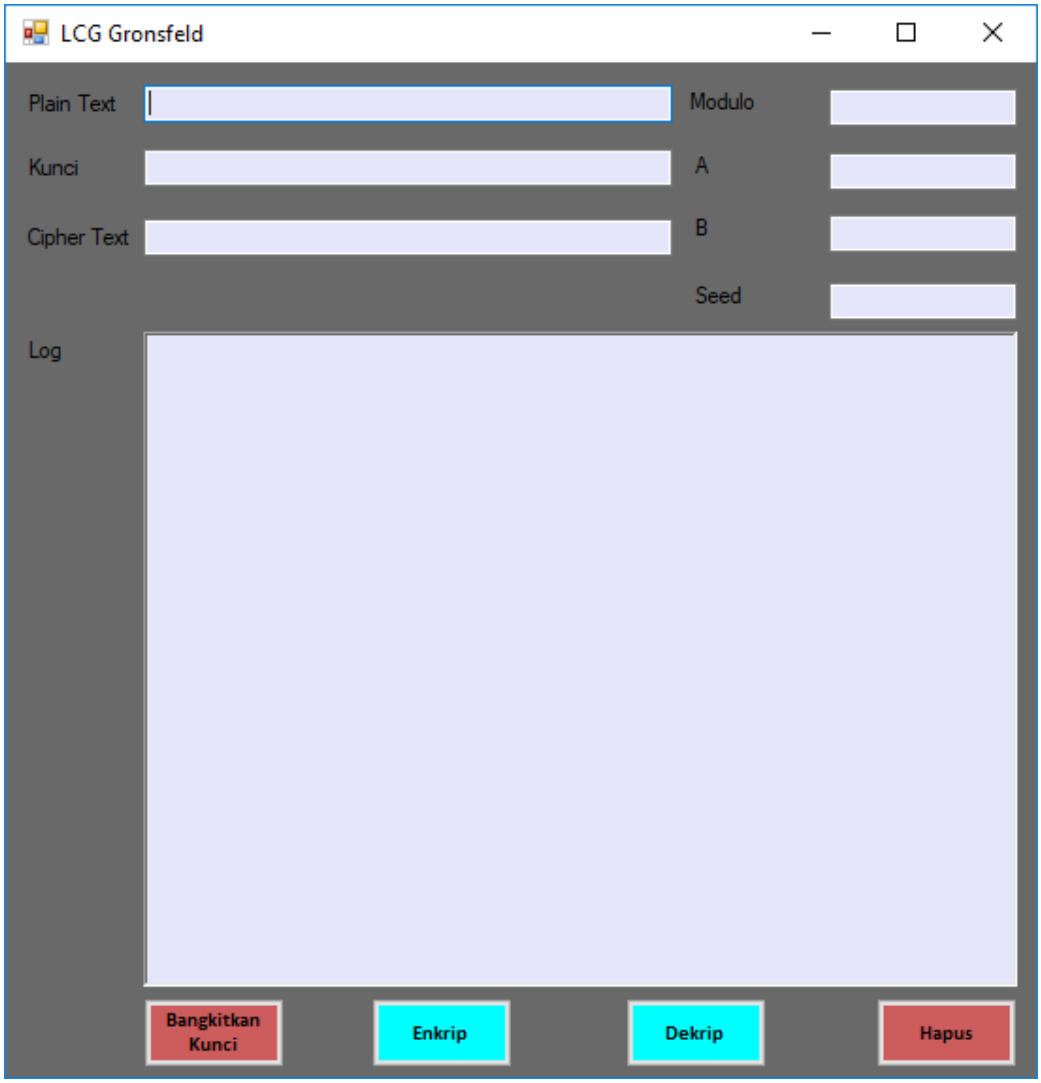
Tabel 4.3 Tabel perhitungan kunci dekripsi

KARAKTER ASCII	HASIL PERHITUNGAN	KUNCI AWAL	MOD	ANGKA ASCII	PLAINTEXT
}	125	40	256	85	U
o	111	33	256	78	N
9	57	240	256	73	I
ï	239	153	256	86	V
}	125	56	256	69	E
ã	227	145	256	82	R
S	83	0	256	83	S
R	82	9	256	73	I
œ	15	72	256	84	T
B	66	1	256	65	A
c	99	16	256	83	S
™	153	121	256	32	

KARAKTER ASCII	HASIL PERHITUNGAN	KUNCI AWAL	MOD	ANGKA ASCII	PLAINTEXT
¨	168	88	256	80	P
¶	182	113	256	69	E
m	109	32	256	77	M
+	43	233	256	66	B
©	169	104	256	65	A
/	47	255	256	78	N
w	119	48	256	71	G
®	174	89	256	85	U
Æ	198	120	256	78	N
’	146	81	256	65	A
	142	64	256	78	N
é	233	201	256	32	
Ø	216	136	256	80	P
	2	193	256	65	A
	158	80	256	78	N
|	124	57	256	67	C
Ù	217	152	256	65	A
Q	81	49	256	32	
¢	162	96	256	66	B
þ	254	169	256	85	U
ì	236	168	256	68	D
Ê	234	161	256	73	I
HASIL DEKRIPSI		**UNIVERSITAS PEMBANGUNAN PANCA BUDI**			

4.2.4 Pengujian Aplikasi

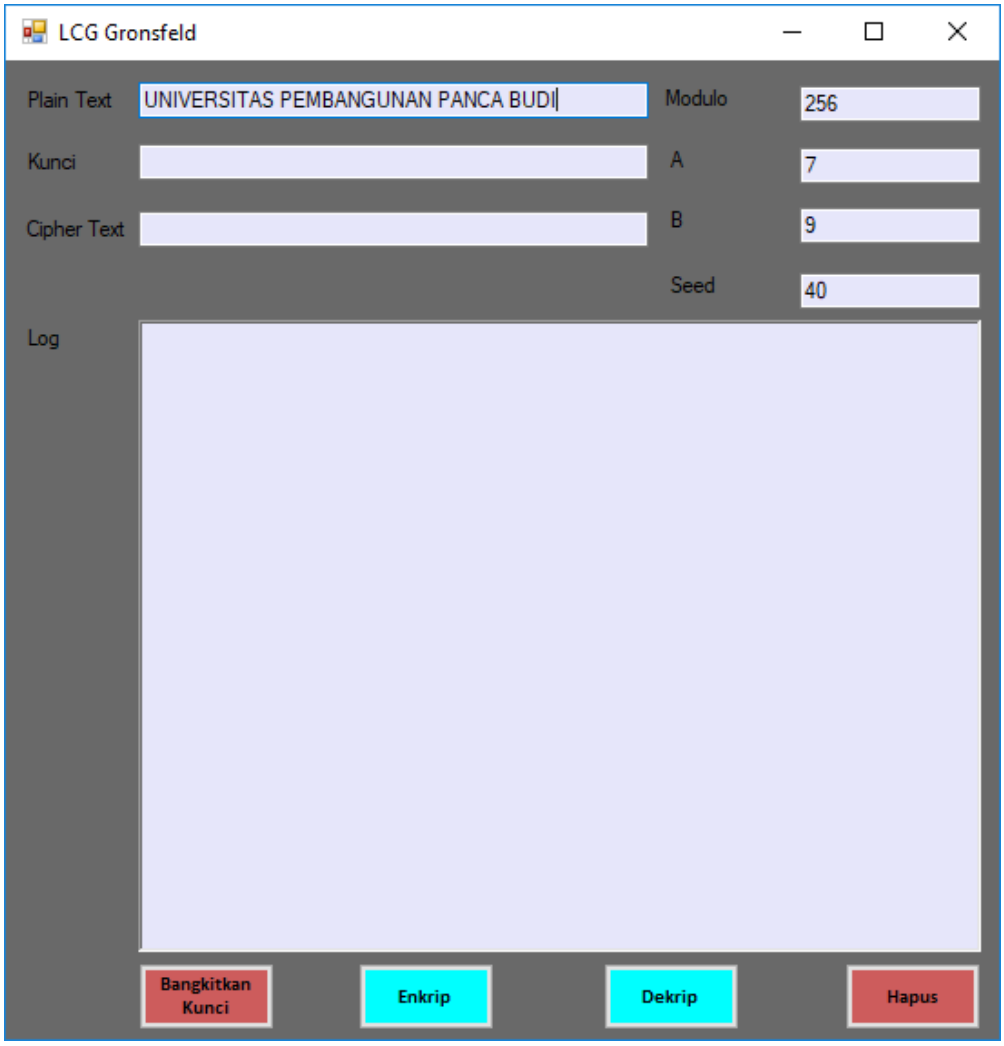
Pengujian aplikasi yang dilakukan oleh penulis, akan dijelaskan gambar 4.1 di bawah ini:



Gambar 4.1 Tampilan awal aplikasi

Kolom *Plaintext*, modulo, A, B, dan seed masih kosong. Maka pengguna harus mengisinya. Isi kolom *plaintext* dengan kata-kata apapun, kemudian nilai

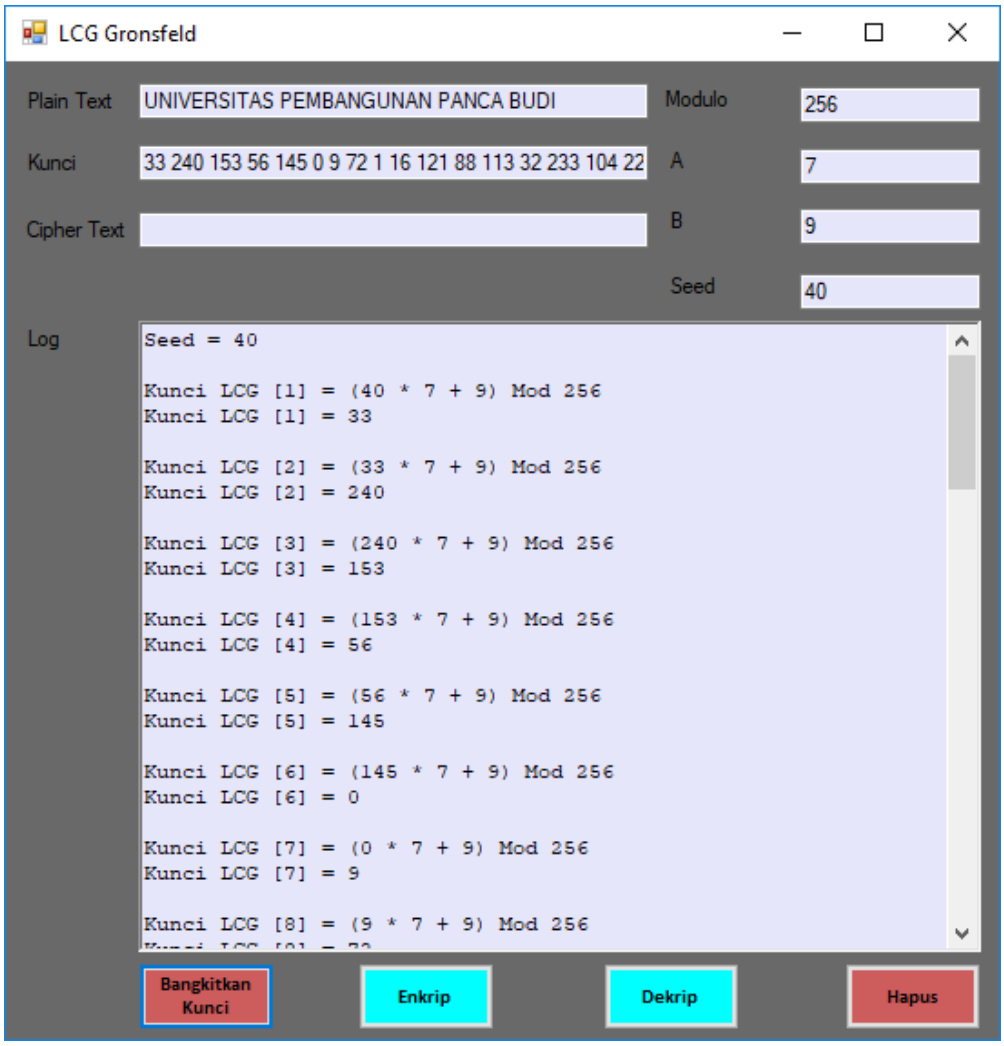
modulo harus diisi dengan bilangan 256, karena bilangan 256 merupakan jumlah bilangan ASCII dari 0-255, nilai A dan B harus diisi dengan bilangan prima, dan seed bisa diisi dengan bilangan desimal berapapun, karena seed merupakan bilangan pengulangan awal dari sebuah aplikasi Gronsfeld dan LCG.



Gambar 4.2 Tampilan aplikasi setelah diisi plaintext, nilai modulo, A, B dan seed

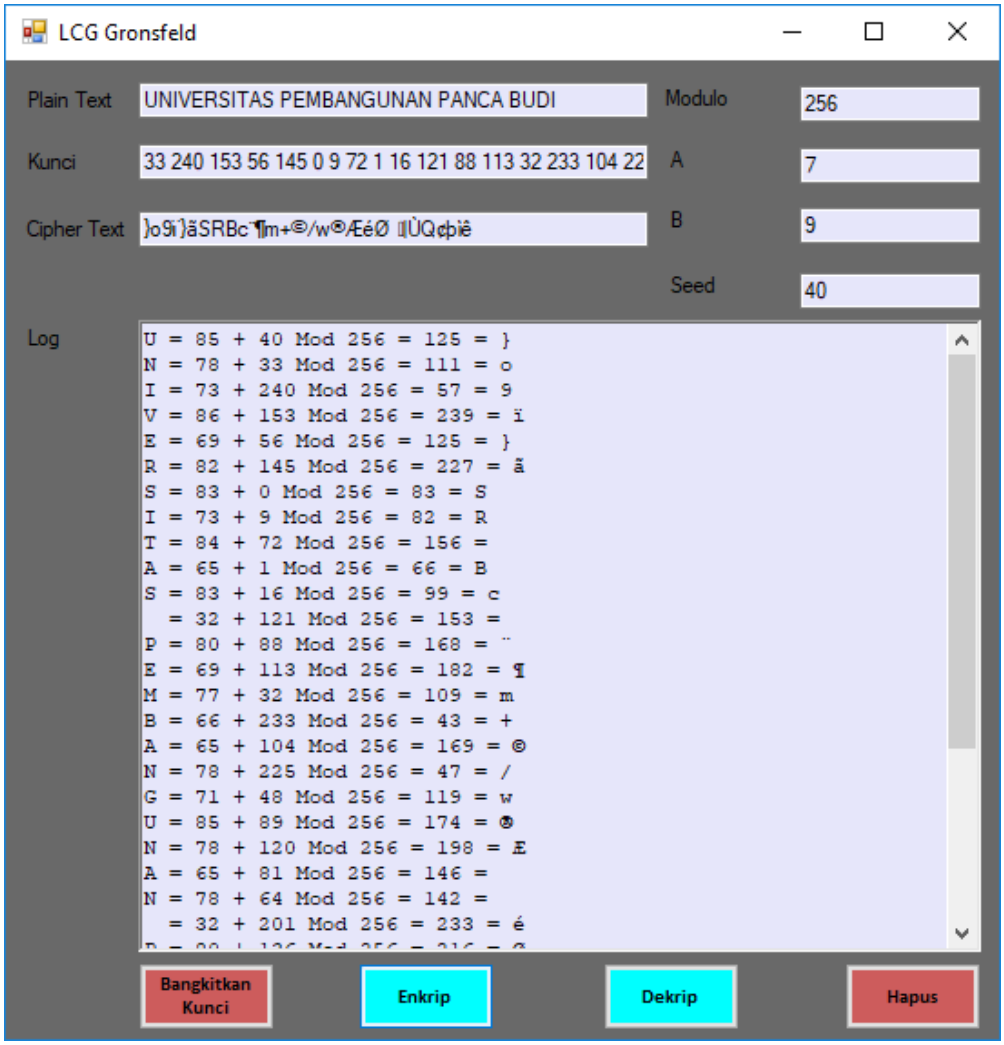
Kolom *plaintext* sudah diisi dengan kalimat "UNIVERSITAS PEMBANGUNAN PANCA BUDI", kolom modulo diisi dengan nilai 256,

kolom A dan B masing masing diisi dengan bilangan prima, 7 dan 9. Terakhir kolom seed diisi dengan nilai 40.



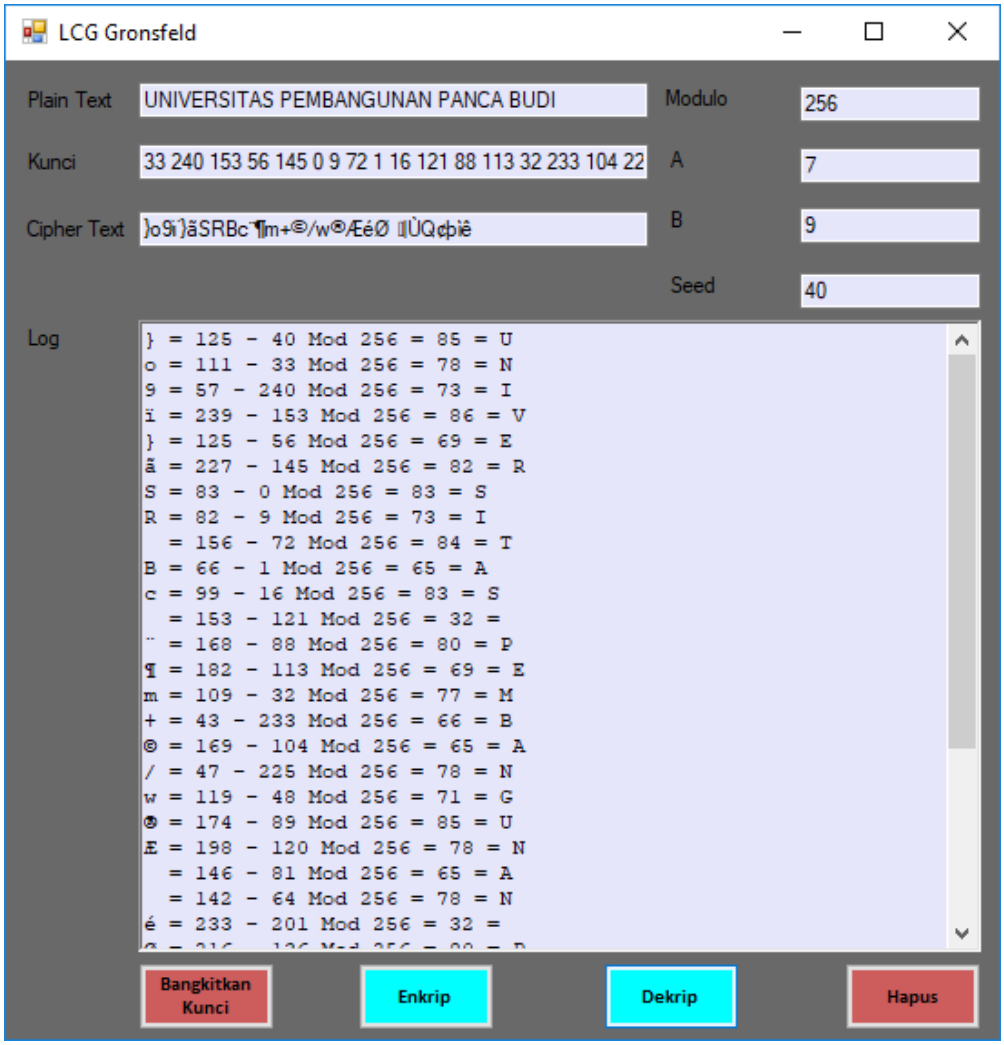
Gambar 4.3 Menampilkan hasil dari bangkitkan

Hasil yang keluar setelah *user* meng-klik tombol "Bangkitkan Kunci", perhitungan pembangkitan kunci LCG ditampilkan di *textbox* log dan hasil kunci LCG ditampilkan kolom kunci.



Gambar 4.4 Menampilkan hasil dari enkripsi

Hasil yang keluar setelah *user* meng-klik tombol “Enkrip”, perhitungan kunci enkripsi ditampilkan di *textbox* log dan kunci enkripsi ditampilkan di kolom *ciphertext*.



Gambar 4.5 Menampilkan hasil dekripsi

Hasil yang keluar setelah *user* meng-klik tombol “Dekrip”, perhitungan kunci dekrip ditampilkan di *textbox* log dan kunci enkripsi tetap ditampilkan di kolom *ciphertext*.

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan analisa dan pengujian sistem pada bab sebelumnya, maka dapat disimpulkan bahwa:

1. Algoritma Gronsfeld Cipher merupakan algoritma yang cukup sederhana, karena menggunakan perhitungan standar dalam proses enkripsi dan dekripsi data.
2. Hasil kunci, enkripsi dan dekripsi dihasilkan berdasarkan panjang karakter, nilai modulo, A, B dan seed.
3. Kunci awal di baris pertama yang dihasilkan sebelum di-enkripsi, menghasilkan kunci yang sama, apapun karakternya.
4. Hasil kunci awal di baris pertama dapat berbeda dengan karakter sebelumnya jika antara nilai A, B dan seed dirubah.
5. Sistem yang dirancang merupakan sistem yang sangat sederhana dan cukup ringan. Sehingga lebih mudah digunakan dan dipahami jika kedepannya ada yang ingin mengembangkan sistem ini.

5.2 Saran

Supaya rancangan sistem ini dapat dikembangkan lagi, maka diperlukan saran yang bisa membuat rancangan sistem ini lebih berkembang. Adapun saran penulis untuk rancangan sistem ini yaitu:

1. Rancangan sistem ini jika dikembangkan, akan menjadi sistem keamanan *login* jika ditambahkan algoritma OTP *(One Time Password)* di dalam sistem ini.
2. Algoritma kriptografi yang dipakai dalam skripsi ini kurang kuat keamanannya, artinya masih ada celah untuk masuk ke dalam sistem keamanan ini. Harapan penulis, jika rancangan sistem ini ingin dikembangkan untuk pengamanan berupa *file, image,* maupun *video,* maka dapat ditambahkan algoritma yang lebih kuat lagi. Contoh algortima yang bisa diterapkan diantaranya yaitu RSA, MD5, HASH, dan lain sebagainya.
3. Tidak ada sistem yang benar-benar aman. Maka penulis menyadari bahwa rancangan sistem ini masih diperlukan pengembangan yang membuat sistem ini bisa lebih baik lagi.

DAFTAR PUSTAKA

Basri. (2016). Kriptografi Simetris dan Asimetris Dalam Perspektif Keamanan Data dan Kompleksitas Komputasi. Jurnal Ilmiah Ilmu Komputer, Vol. 2, No. 2.

Clawdia, J., Khairani, N., & Harahap, M.K. (2017). Implementasi Algoritma Kriptografi One Time Pad (OTP) dengan Dynamic Key Linear Congruential Generator (LCG), 13, ISSN: 2597-4645.

Harahap, M.K., & Khairani, N. (2017). Analisis Algoritma One Time Pad dengan Algoritma Cipher Transposisi Sebagai Pengamanan Pesan Teks, Jurnal & Penelitian Teknik Informatika, Vol. 1 No. 2, 59, ISSN: 2541-2019.

Kamus Besar Bahasa Indonesia (KBBI) Daring. (2019). Analisis. https://kbbi.kemdikbud.go.id/entri/analisis, diakses pada tanggal 28 Agustus 2019

Mahmood, Z. J.L Rana, & Khare, A. (2012). Symmetric Key Cryptography Using Dynamic Key And Linear Congruential Generator (LCG). International Journal of Computer Applications, Vol. 50, No. 19, doi:10.5120/7908-0973.

Munir, R. (2006). Kriptografi. Jakarta: Informatika.

Quasim., Tabrez, M.D. (2013). Security Issues in Distributed Database System Model. COMPUSOFT, Vol. 2-12, 396-399.

Rakhmadi, A., & Nugroho. (2016). Web-Based Application for Single Elimination Tournament Using Linear Congruential Generator. ISETH 2016 (The 2nd International Conference on Science, Technology, and Humanity), 273-285.

Ramadhan, Z., Siahaan, A.P.U. (2018). Protection of Important Data and Information using Gronsfeld Cipher, Vol. 4, ISSN: 2455-0620.

Romzi, M. (2012). Logika dan Algoritma, 141, 1736.

Sitinjak, S., Fauziah, Y., Juwairiah, (2015). Aplikasi Kriptografi File Menggunakan Algoritma Blowfish. Seminar Nasional Informatika UPN "Veteran", 78-86, ISSN: 1979-2328.

Sitohang, (2013) Perangkat Aplikasi Keamanan Data Text Menggunakan Electronic CodeBook dengan Algoritma DES. V, 2-3.

Thomas, J., Kakiay. (2004). Pengantar Sistem Simulasi. Yogyakarta: Andi.

Mariance, U. C. (2018). Analisa dan Perancangan Media Promosi dan Pemasaran Berbasis Web Menggunakan Work System Framework (Studi Kasus di Toko Mandiri Prabot Kota Medan). Jurnal Ilmiah Core IT: Community Research Information Technology, 6(1).

Sarif, M. I. Classification Of Feasibility Of Basic Food Recipients In Kelurahan Tanjung Morawa A, Tanjung Morawa Sub-District Using Naïve Bayes Classifier Algorithm.

Sarif, M. I. (2017). Penemuan Aturan yang Berkaitan dengan Pola dalam Deret Berkala (Time Series).

Putri, N. A. (2018). Sistem Pakar untuk Mengidentifikasi Kepribadian Siswa Menggunakan Metode Certainty Factor dalam Mendukung Pendekatan Guru. INTECOMS: Journal of Information Technology and Computer Science, 1(1), 78-90.

Hendrawan, J. (2018). Rancang Bangun Aplikasi Mobile Learning Tuntunan Shalat. INTECOMS: Journal of Information Technology and Computer Science, 1(1), 44-59.

Dhany, H. W., Izhari, F., Fahmi, H., Tulus, M., & Sutarman, M. (2017, October). Encryption and decryption using password based encryption, MD5, and DES. In International Conference on Public Policy, Social Computing and Development 2017 (ICOPOSDev 2017) (pp. 278-283). Atlantis Press.

Sumartono, I., Siahaan, A. P. U., & Mayasari, N. (2016). An overview of the RC4 algorithm. IOSR J. Comput. Eng, 18(6), 67-73.

Badawi, A. (2018). Evaluasi Pengaruh Modifikasi Three Pass Protocol Terhadap Transmisi Kunci Enkripsi.

Fuad, R. N., & Winata, H. N. (2017). Aplikasi Keamanan File Audio Wav (Waveform) Dengan Terapan Algoritma Rsa. Infotekjar: Jurnal Nasional Informatika Dan Teknologi Jaringan, 1(2), 113-119.

Sitorus, Z., Saputra, K, S., Sulistianingsih, I. (2018) C4.5 Algorithm Modeling For Decision Tree Classification Process Against Status UKM.

Hariyanto, E., Lubis, S. A., & Sitorus, Z. (2017). Perancangan prototipe helm pengukur kualitas udara. KOMIK (Konferensi Nasional Teknologi Informasi dan Komputer), 1(1).

Sitorus, Z. (2018). Kebutuhan Web Service untuk Sinkronisasi Data Antar Sistem Informasi dalam Universitas. Jurnal Teknik dan Informatika, 5(2), 87-90.

Iqbal, M., Siahaan, A. P. U., Purba, N. E., & Purwanto, D. (2017). Prim's Algorithm for Optimizing Fiber Optic Trajectory Planning. Int. J. Sci. Res. Sci. Technol, 3(6), 504-509.

Rahim, R. (2018, October). A Novelty Once Methode Power System Policies Based On SCS (Solar Cell System). In International Conference of ASEAN Prespective and Policy (ICAP) (Vol. 1, No. 1, pp. 195-198).

Fachri, Barany. "Aplikasi Perbaikan Citra Efek Noise Salt & Papper Menggunakan Metode Contraharmonic Mean Filter." Seminar Nasional Royal (Senar). Vol. 1. No. 1. 2018.