PENERAPAN ALGORITMA KRIPTOGRAFI KUNCI SIMETRIS DENGAN MODIFIKASI *VIGENERE CIPHER* DALAM APLIKASI KRIPTOGRAFI TEKS

Irham Mu'alimin Arrijal [1], Rusdi Efendi[2], Boko Susilo[3]

[1,2,3]Program Studi Teknik Infomatika, Fakultas Teknik, Universitas Bengkulu.
Jl. WR. Supratman Kandang Limun Bengkulu 38371A INDONESIA
(telp: 0736-341022; fax: 0736-341022)

[1] irham_ganna_shine@yahoo.com
[2] r_efendi@yahoo.com
[3] bksusilo@gmail.com

Abstrak: Vigenere Cipher adalah salah satu algoritma kriptografi simetris yang tangguh. Sejak ditemukannya metode kasiski, algoritma ini menjadi sangat di-*decipher*, terutama apabila teks hasil enkripsi cukup panjang. Pada dasarnya algoritma ini bekerja dengan membentuk sejumlah *subsequence* dari teks, di mana banyak *subsequence* sama dengan panjang kunci. Kemudian, pada modifikasi algoritma ini, setiap *subsequence* dienkripsi dengan menggunakan algoritma kriptografi kunci simetris, yang dapat saling berbeda, dan dienkripsi dengan menggunakan kunci yang sama. Karena jumlah algoritma dapat sangat banyak maka algoritma yang digunakan perlu dijadikan suatu *file* yang berisikan konfigurasi dari algoritma-algoritma tersebut, sedemikian sehingga *file* ini kemudian menjadi salah satu kunci yang digunakan untuk dapat membuka *cipher* teks tersebut. Selanjutnya, karena telah terdapat kunci simetris, untuk meningkatkan kesulitan, *file* tersebut harus diamankan dengan suatu mekanisme kriptografi kunci publik. Pada penelitian ini, modifikasi algoritma *Vigenere Cipher* tersebut kemudian diterapkan ke bentuk aplikasi yang memungkinkan pengguna dapat melakukan proses enkripsi dan dekripsi teks dengan menggunakan konsep *Vigenere* Abstrak yang diterapkan oleh aplikasi. Algoritma yang dapat digunakan, hanyalah algoritma yang telah ditambahkan ke aplikasi dan telah tervalidasi oleh *testcase-testcase* yang telah terdaftar pada *database*.
Kata Kunci: Kriptografi Kunci Simetris, *Vigenere Cipher*, Kriptografi Kunci Publik, *Vigenere* Abstrak.

Abstract: Vigenere Cipher was one of the tough symmetric-key cryptography algorithm. Since Kasiski been discovered, this algorithm has been so vulnerable to be deciphered, especially if the cipher text is quite long. Basically the algorithm works on by yielding some subsequences of text, where the number of the subsequences is equal to the length of the key. And thus, on this modification each subsequence is encrypted by symmetric-key cryptography algorithms, that may differ from each other, and using the same key. Because the algorithms could be so much, therefore the algorithms used need to be manifested into a file which contains the configuration of them, such that this file will later be one of the key used to open decrypt the cipher text. Since the symmetric key already occupied, and in order to enhance the complication, the file must be secured using the mechanism of public-key cryptography. On this research, the modification of the Vigenere Cipher is implemented in the form of an application that enable user to do encrpt and decrypt a text with the concept of the Abstract Vigenere. The algorithm may be available only if the algorithm has been installed and validated by the testcases registered in the database.
Keywords: Algorithm, Symmetric-Key Cryptography, Vigenere Cipher, Public-Key Cryptography, Abstract Vigenere

I. PENDAHULUAN

Salah satu algoritma kriptografi klasik yang paling tangguh adalah *Vigenere Cipher* yang

ditemukan pada tahun 1553 oleh Giovan Battista Bellaso[1]. Teks yang dienkripsi dengan algoritma ini sangat sulit untuk didapatkan hasil dekripsinya tanpa menggunakan kunci, hingga ditemukannya metode Kasiski pada tahun 1863. Namun demikian sejak ditemukannya metode ini, algoritma *Vigenere Cipher* menjadi sangat rentan untuk dipecahkan untuk teks yang panjang.

Vigenere Cipher bekerja dengan cara menentukan suatu nilai *n integer* yang merupakan panjang kunci, kemudian dibentuk *n* buah *subsequence* dari teks tersebut berdasarkan indeks karakter dimodulokan *n*, sedemikian sehingga *subsequence* ke – *i* adalah kumpulan karakter dengan indeks *j*, di mana $j \equiv i \bmod n$. dari teks. Kemudian *subsequence* ke- *i* dioperasikan dengan karakter ke- *i* dari kunci[1].

Subsequence dari suatu string *S*, adalah kumpulan karakter di-*S* tanpa mengubah urutan karakter tersebut pada *S*. Contoh : misalkan *S* = "ABCDEFGH". Maka contoh subsequence dari *S* adalah "ABCE", "BDFH", dll. Yang bukan subsequence dari *S* adalah "JKL", "ACHG", "DEGF", dll.

Untuk meningkatkan kesulitan dari algoritma *Vigenere Cipher* dapat dilakukan dengan cara menentukan suatu kunci dan suatu bilangan bulat *n* yang merupakan jumlah *subsequence* yang akan dibentuk (*n* bukan merupakan panjang kunci). Kemudian setiap *subsequence* dienkripsi menggunakan algoritma kriptografi kunci simetris yang berbeda dengan kunci yang sama. Pemilihan algoritma kunci simteris ini didasarkan pada *Vigenere Cipher* merupakan algoritma kriptografi simetris. Setiap algoritma kunci simetris memiliki pola yang unik, yaitu masing-masing algoritma membutuhkan satu kunci untuk mengenkripsi dan mendekripsi teks.

Karena jumlah algoritma dapat sangat banyak maka algoritma yang digunakan perlu dijadikan suatu *file* yang berisikan konfigurasi dari algoritma-algoritma tersebut, sedemikian sehingga *file* ini kemudian menjadi salah satu kunci yang digunakan untuk dapat membuka *cipher* teks tersebut. Akan tetapi, umumnya *file* mudah untuk dibuka. Namun demikian, kerahasiaan isi dari *file* tersebut tetap perlu untuk dijaga.

Salah satu teknik pengamanan data berbentuk *file* yang dapat diterapkan adalah dengan memanfaatkan *object serialization java. Object Serialization Java* memiliki properti yaitu suatu objek yang diserialisasi memiliki suatu ID tertentu, yang berupa suatu bilangan bulat non-negatif, dan tergolong pada suatu *Class* tertentu. Jadi jika suatu objek akan dideserialisasi, maka diperlukan suatu ID dan *Class* yang sama yang merealisasikan objek tersebut.

Selanjutnya, karena telah terdapat kunci untuk enkripsi teks, dan ID merupakan salah satu kebutuhan pada *object serialization*, maka ID perlu untuk ditetapkan sebagai suatu kunci publik. Sehingga pada saat proses dekripsi, penerima dapat menggunakan ID (kunci privat) nya sendiri.

Berdasarkan uraian di atas, penulis mengangkat judul "Penerapan Algoritma Kriptografi Kunci Simetris Dengan Modifikasi *Vigenere Cipher* Dalam Rancang Bangun Aplikasi Kriptografi Teks".

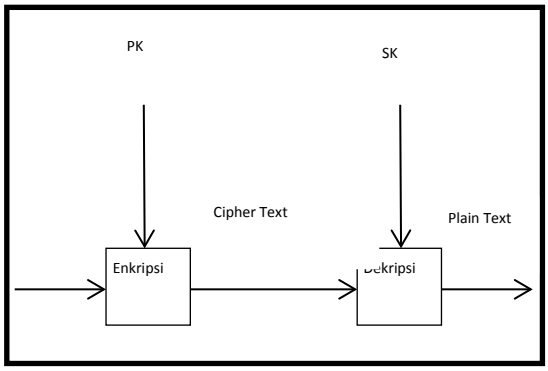
II. LANDASAN TEORI

A. *Algoritma Kriptografi Kunci Simetris*

Algoritma simetris (algoritma kriptografi konvensional) adalah algoritma yang menggunakan kunci yang sama untuk proses enkripsi dan proses deskripsi. Algoritma kriptografi simetris dibagi menjadi dua kategori yaitu algoritma aliran (*Stream Ciphers*) dan algoritma blok (*Block Ciphers*). Dimana pada

algoritma aliran, proses penyandiannya akan beriorientasi pada satu *bit/byte data*. Sedangkan pada algoritma blok, proses penyandiannya berorientasi pada sekumpulan bit/byte data (per blok).

Adapun contoh algoritma kunci simetris adalah DES (*Data Encryption Standard*), *Blowfish*, *Twofish*, MARS, IDEA, 3DES (DES diaplikasikan 3 kali), AES(A*dvanced Encryption Standard*) yang bernama asli *Rijndael*.



Gambar 1. Sistem Kriptografi Kunci Simetris

Suatu *plain* teks dienkripsi menggunakan suatu kunci menghasilkan suatu cipher teks. *Cipher* teks didekripsi menggunakan kunci yang sama untuk menghasilkan *plain* teks[1].

B. *Vigenere Cipher*

Vigenere Cipher termasuk dalam *cipher* abjad majemuk (*Polyalphabetic Substitution Cipher)* yang dipublikasikan oleh diplomat (sekaligus seorang kriptologis) Perancis, Blaise de Vigenere pada tahun 1586. *Vigenere Cipher* adalah metode menyandikan teks alfabet dengan menggunakan deretan sandi *Caesar* berdasarkan huruf-huruf pada kata kunci. Teknik dari substitusi *vigenere cipher* biasa dilakukan dengan dua cara[2] :

1. Angka

Vigenere Cipher dengan angka adalah metode menyandikan teks alfabet dengan menggunakan deretan sandi *Caesar* berdasarkan huruf-huruf pada kata kunci.

2. Huruf

Vigenere Cipher dengan huruf berisi alfabet yang dituliskan dalam 26 baris, masingmasing baris digeser ke kiri dari baris sebelumnya membentuk ke-26 kemungkinan sandi *Caesar* setiap huruf disediakan dengan menggunakan baris yang berbeda-beda sesuai kunci yang diulang.

Rumus dari enkripsi dan dekripsi data *vigenere cipher* adalah :

Enkripsi:

$C_i = (P_i + K_i) \bmod 26$

Dekripsi:

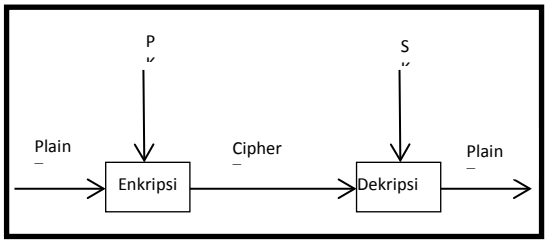
$P_i = (Ci - Ki) \bmod 26$; untuk $C_i \geq K_i$

$P_i = (C_i + 26 - K_i) \bmod 26$; untuk $C_i \leq K_i$

C. *Kriptografi Kunci Publik*

Sampai akhir tahun 1970, hanya ada sistem kriptografi simetri. Karena sistem kriptografi simetri menggunakan kunci yang sama untuk enkripsi dan dekripsi, maka hal ini mengimplikasikan dua pihak yang berkomunikasi saling mempercayai. Konsep sistem kriptografi kunci-publik ditemukan oleh Diffie dan Hellman yang mempresentasikan konsep ini pada Tahun 1976[2].

Ide dasar dari sistem kriptografi kunci-publik adalah bahwa kunci kriptografi dibuat sepasang, satu kunci untuk enkripsi dan satu kunci untuk dekripsi. Kunci untuk enkripsi bersifat publik (tidak rahasia) – sehingga dinamakan **kunci publik** (*public-key*) – sedangkan kunci dekripsi bersifat rahasia – sehingga dinamakan **kunci rahasia** (*private key* atau *secret key*). Kunci-kunci ini dipilih sedemikian sehingga – secara praktek – tidak mungkin menurunkan kunci rahasia dari kunci publik.



Gambar 2 Sistem kriptografi kunci-publik.

III. METODOLOGI PENELITIAN

A. *Jenis Penelitian*

Jenis penelitian yang akan dilakukan adalah penelitian terapan (*applied research*). Penelitian ini menerapkan algoritma kriptografi kunci simetris dengan modifikasi *Vigenere Cipher* pada rancang bangun aplikasi teks.

Jenis data yang dibutuhkan dan digunakan dalam penelitian ini adalah data kualitatif, yaitu data yang tidak dapat diukur secara langsung. Data-data yang digunakan dalam penelitian ini dibagi ke dalam 2 jenis data, yaitu:

1. Data Primer

Data primer pada penelitian ini adalah algoritma-algoritma kriptografi kunci simetris yang telah diterjemahkan ke dalam *source code* bahasa pemrograman java, dan kasus-kasus uji pada untuk proses validasi algoritma. Pada proses validasi, setiap kasus uji terdiri atas suatu string kunci dan suatu teks.

2. Data Sekunder

Data sekunder pada penelitian ini adalah dan kasus uji untuk proses enkripsi dan dekripsi. Pada proses enkripsi, data yang akan digunakan adalah sejumlah kasus uji, yang masing-masing kasus uji terdiri atas 1 buah string kunci, 1 buah teks (*plain text*), dan sebuah bilangan bulat yang berperan sebagai kunci publik.

Pada proses dekripsi, data yang digunakan adalah seluruh data yang merupakan keluaran dari proses enkripsi, berupa 1 buah teks (*cipher text*), 1 buah *file* konfigurasi, 1 buah bilangan yang berperan sebagai *private key*, dan 1 buah kunci simetris yang digunakan untuk mengenkripsi teks tersebut.

B. Teknik Pengumpulan Data

Teknik pengumpulan data yang digunakan dalam penelitian ini adalah:

1. Studi Pustaka

Studi pustaka merupakan metode pengumpulan data yang diperoleh dari buku-buku dan/atau jurnal dalam pencarian referensi terkait pengumpulan data maupun perancangan aplikasi yang akan dibangun, yaitu referensi mengenai algoritma kriptografi kunci simetris, *vigenere cipher*, kriptografi kunci publik, dan bahasa pemrograman Java.

2. Observasi

Observasi dilakukan terhadap sejumlah algoritma kriptografi kunci simetris dari sejumlah literature pada saat studi pustraka. Yang diobservasi adalah prilaku algoritma dalam mengekstrak kunci, dan output yang dihasilkan apabila algoritma diterapkan pada suatu string dengan kunci yang diberikan. Sejumlah algoritma kemudian dipisahkan ke dalam dua kelompok, yaitu algoritma yang valid (algoritma yang memenuhi kriteria yang dicari yaitu panjang *cipher text* = panjang *plain text*, dan *decrypted cipher text* = *plain text*), dan algoritma yang tidak valid (algoritma yang tidak memenuhi kriteria).

Seluruh algoritma yang ditinjau kemudian akan digunakan sebagai kasus uji atas kasus-kasus uji untuk validitas algoritma yang dapat diterima pada aplikasi.

3. Teknik Sampling

Teknik *sampling* yang digunakan adalah metode *random sampling*. *Sampling* digunakan untuk menentukan data-data sekunder untuk proses

pengujian. Sampel yang diambil adalah sembarang teks yang *well-written* dan tersusun rapi secara gramatikal dan setiap kata benar sesuai ejaaan yang disempurnakan (EYD). Teks tertulis dalam huruf latin atau teks bahasa asing (non-latin) yang telah diromanisasi (dibaca sesuai ejaan latin).

C. Metode Pengujian

Pengujian aplikasi dilakukan menggunakan metode *Black-Box Testing.* Pengujian dilakukan dengan menerapkan sejumlah kasus uji pada aplikasi, dan mengamati apakah aplikasi telah berjalan sesuai dengan yang seharusnya.

Kasus uji pada pengujian diklasifikasikan sebagai berikut :

1. *Validasi algoritma kriptografi kunci simetris*

Terdapat sejumlah kasus uji pada jenis kasus uji ini yang berguna untuk menguji keabsahan algoritma kunci simetris yang diinputkan ke aplikasi. Masing-masing kasus uji terdiri atas 2 buah string (*kunci* dan *plain text*), dan dinomori secara berurutan dari 1.

Kasus-kasus uji diujikan ke algoritma yang diinputkan satu per satu secara berurutan oleh aplikasi secara otomatis. Kasus uji ke-*i* hanya akan diujikan jika algoritma telah lulus kasus uji ke-(*i-1*). Jika algoritma gagal pada kasus uji ke-*i* maka validasi dihentikan dan algoritma tidak dapat diterima untuk disimpan pada aplikasi. Algoritma dikatakan valid apabila panjang *cipher text* = panjang *plain text*, dan *decrypted cipher text* = *plain text.*

Setiap kasus uji yang diujikan ke algoritma dilakukan pada 5 buah *system runtime* yang dijalankan secara paralel dan berbatas waktu. Batas waktu yang digunakan untuk setiap *testcase* pada satu *system runtime* adalah 5 detik *clock time*. Selanjutnya keluaran untuk setiap kasus uji (*verdict*) diklasifikasikan ke dalam 5 kategori berikut, yaitu :

a. *COMPILE ERROR*

Algoritma dinyatakan *compile error*, apabila terdapat kesalahan *syntax source code* secara bahasa pemrograman *Java* atau terdapat *keyword package* pada bagian *import part*.

b. *RUNTIME ERROR*

Algoritma dinyatakan *runtime error* pada salah satu kasus uji, apabila algoritma tersebut menghasilkan *exception* / *error* yang perlu di-*handle* namun tidak di-*catch* oleh algoritma tersebut.

c. *TIME LIMIT EXCEEDED*

Algoritma dinyatakan *time limit exceeded* pada salah satu kasus uji, apabila algoritma tersebut melampaui batas waktu eksekusi untuk satu *runtime*-nya.

d. *INVALID OUTPUT*

Algoritma dinyatakan *invalid output* pada salah satu kasus uji, apabila algoritma tidak *runtime error* dan tidak *time limit exceeded*, dan *cipher text* ataupun *decrypted cipher text* menyalahi aturan *output*, yaitu panjang *cipher text* tidak sama dengan panjang *plain text* ataupun *decrypted cipher text* tidak sama persis dengan *plain text*.

e. *ACCEPTED (OK)*

Algoritma dinyatakan *accepted (ok)* pada salah satu *testcase*, apabila algoritma tidak melampaui batas waktu dan tidak mengalami *exception* dan menghasilkan *output* seperti yang diharapkan.

Terdapat 50 *testcase* yang akan di-*input*-kan pada *database*, yang rinciannya dapat dilihat pada Tabel 1 berikut :

Tabel 1. Rincian *Testcase*

***Testcase* No**	**Tujuan**
1-3	Uji teks dan kunci dengan huruf capital
4-6	Uji Kombinasi huruf kapital dan tanda baca (ASCII 32 s.d 64)
7-9	Uji tanpa kunci atau tanpa plain teks

10	Uji kunci dan teks pendek (< 10 karakter) dengan kombinasi huruf kapital dan huruf kecil
11-15	Uji kunci dan teks *bigram, trigram*, dan *n-gram*
16-20	Uji kunci berupa tanda baca (ASCII 32 s.d 64) untuk sebarang teks huruf
21-25	Uji sebarang kunci huruf untuk plain teks berupa tanda baca (ASCII 32 s.d 64)
26-35	Uji kunci dan teks pada sebarang karakter (setiap karakter merupakan karakter antara ASCII 1- ASCII 256)
36-45	Uji kunci dan teks pada sebarang karakter (setiap karakter merupakan karakter antara ASCII 1- *Extended* ASCII 65535)
45-50	Uji kunci dan teks panjang (90-100 karakter untuk kunci, dan 800-1000 karakter)

2. Algoritma

Kasus uji pada kategori ini berupa sejumlah algoritma, yang akan dicoba untuk diinputkan ke aplikasi secara manual satu persatu untuk menguji ketepatan pemilihan kasus uji pada validasi algoritma, dan apakah aplikasi dapat menangani setiap kemungkinan *error* yang terjadi yang disebabkan oleh algoritma yang diinputkan, seperti *infinite loop, runtime error*, dll.

Setiap algoritma yang diinputkan dibagi ke dalam 5 bagian kode, yaitu *import part* (bagian deklarasi *library* yang digunakan), *extract key part* (bagian yang berisi kode program untuk proses ekstraksi kunci simetris), *encrypt part* (bagian yang berisi kode program untuk proses enkripsi *plain text*), *decrypt part* (bagian yang berisi kode program untuk proses dekripsi *cipher text*), dan *other methods and variables part* (bagian yang berisi kode program untuk mendklarasikan *method-method* pendukung dan *variable* tambahan yang diperlukan).

Pada dasarnya, seluruh kasus uji validasi algoritma yang terdapat pada *database* diasumsikan akan mampu menyeleksi algoritma kriptografi kunci simetris, sedemikian sehingga algoritma tersebut akan *valid* pada seluruh kemungkinan *input* pada proses enkripsi dan dekripsi. Akan tetapi, tidak menutup kemungkinan akan terdapat kasus di mana algoritma menghasilkan *output* tidak *valid* ataupun mengalami *runtime error* ataupun melampaui batas waktu yang ditetapkan. Algoritma tersebut kemudian dinamai dengan *TROLL*.

Berdasarkan kemungkinan keluaran (*verdict*) dari algoritma yang diinputkan, ditambah dengan kemungkinan *TROLL*, selanjutnya algoritma yang akan diujikan diklasifikasikan ke dalam 5 kategori, yaitu algoritma dengan *verdict compile error*, algoritma dengan *verdict runtime error*, algoritma dengan *verdict time limit exceeded*, algoritma dengan *verdict invalid output*, algoritma dengan *verdict accepted*, dan algoritma *troll*.

Terdapat sebanyak 23 algoritma kriptografi kunci simetris yang telah diterjemahkan ke dalam *source code* bahasa pemrograman *Java*. Daftar algoritma yang akan diujikan dapat dilihat pada Tabel 2 berikut.

Tabel 2. Daftar Algoritma yang Diujikan

No	**Nama Algoritma**
1	*CE Cipher*
2	*CE Cipher 2*
3	*RTE Identity Cipher*
4	*RTE Reverse Cipher*
5	*TLE Infinite Loop*
6	*TLE Identity Cipher*
7	*Invalid Triple Transposition Vigenere Cipher*
8	*Play Fair Cipher*
9	*Vigenere Cipher*
10	*Caesar Cipher*
11	*Identity Cipher*
12	*Reverse Cipher*
13	*Extended Vigenere Cipher*
14	*16-Bits Vigenere Cipher*

15	*XOR-Bit Cipher*
16	*XOR 48 Cipher*
17	*Left-Right-Bit Cipher*
18	*Transposition Cipher*
19	*Play Fair 64 Cipher*
20	*Complement-Bit Cipher*
21	*Reverse-Bit Cipher*
22	*Triple Transposition Vigenere Cipher*
23	*Troll Identity Cipher*

3. Enkripsi dan Dekripsi

Kasus uji pada kategori ini digunakan untuk menguji apakah aplikasi mampu melakukan enkripsi dan dekripsi terhadap input yang dimasukkan. Terdiri atas sejumlah kasus uji diinputkan satu persatu ke aplikasi secara manual.

Kasus uji untuk proses enkripsi masing-masing terdiri atas 1 buah string kunci, 1 buah teks (*plain text*), dan sebuah bilangan bulat yang berperan sebagai kunci publik. Yang akan diamati pada proses pengujian ini adalah apakah aplikasi telah berhasil menghasilkan *cipher text* sesuai dengan perhitungan secara manual, dan apakah aplikasi telah berhasil menghasilkan suatu *file* konfigurasi yang berisikan konfigurasi algoritma yang diterapkan pada proses enkripsi.

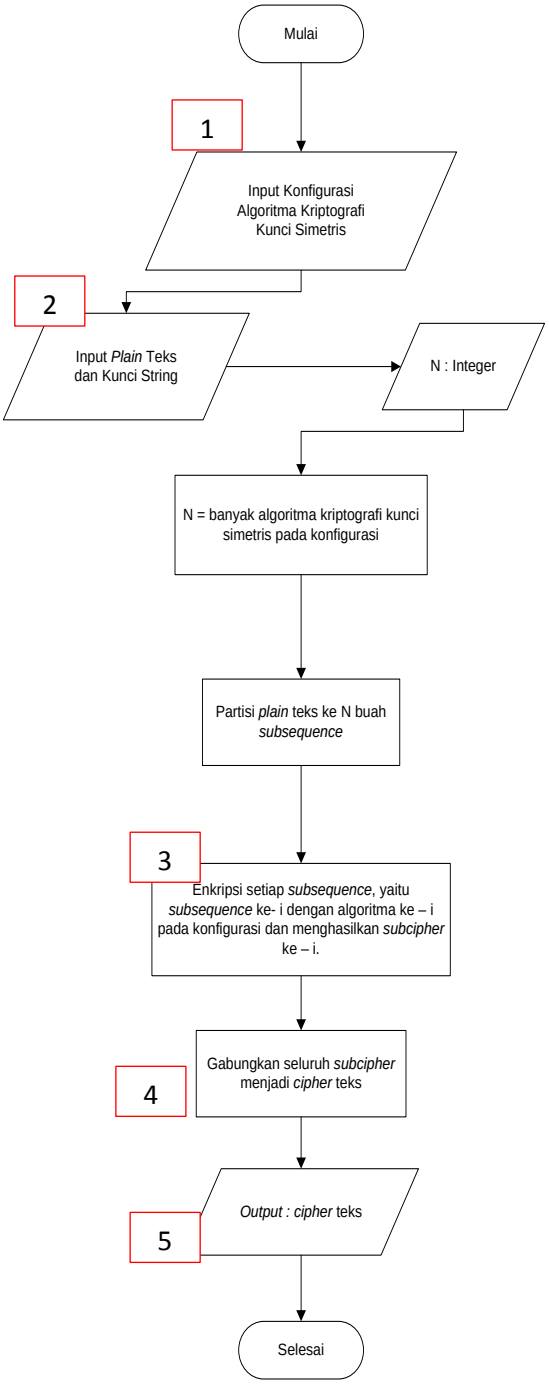
Kasus uji untuk dekripsi menggunakan data yang merupakan keluaran dari proses enkripsi, berupa 1 buah teks (*cipher text*), 1 buah *file* konfigurasi, 1 buah bilangan yang berperan sebagai *private key*, dan 1 buah kunci simetris yang digunakan untuk mengenkripsi teks tersebut.

Hal yang diamati pada pengujian ini adalah, apakah aplikasi telah berhasil menerapkan konsep kriptografi kunci publik yaitu kunci privat yang diinputkan dapat membuka file konfigurasi. File konfigurasi mampu dideserialisasi untuk diambil kontennya yang berupa konfigurasi algoritma kriptografi kunci simetris. Dan apakah aplikasi mampu menghasilkan *decrypted cipher text* yang sama dengan *plain text*.

IV. Analisa dan Perancangan Sistem

A. *Enkripsi*

Proses enkripsi untuk algoritma kriptografi kunci simetris dengan modifikasi *vigenere cipher* ini dapat dilihat pada Gambar 3 berikut.



Gambar 3. *Flowchart* Enkripsi

Algoritma untuk *flowchart* tersebut secara tertulis adalah sebagai berikut :

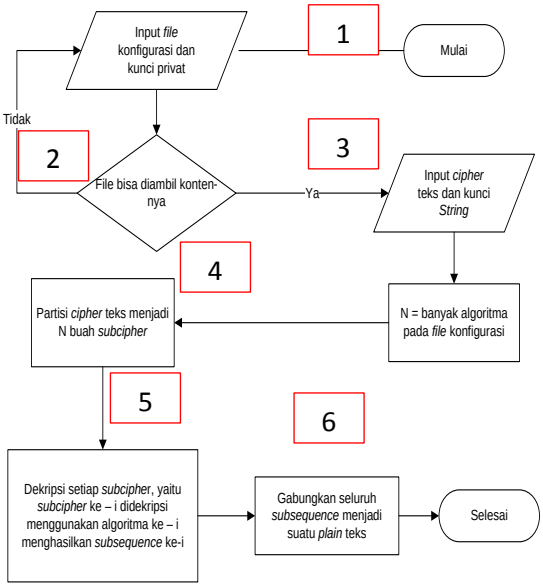
1. Pertama-tama *user* meng-*input*-kan dan mengkonfigurasikan algoritma yang akan digunakan. Mengkonfigurasi algoritma adalah menentukan urutan algoritma-algoritma kriptografi kunci simetris (misal *Caesar Cipher* pada urutan pertama, *Vigenere Cipher* urutan kedua, dst).
2. Kemudian *user* menginputkan *plain text*, kunci *string*, dan suatu bilangan bulat yang merupakan kunci publik.
3. *Plain text* lalu dipecah menjadi *N* buah *subsequence* (di mana *N* adalah banyak algoritma kriptografi kunci simetris yang digunakan), dengan *subsequence*[*i*] berisikan seluruh karakter dengan indeks (*i mod N*) pada *plain* teks, (diasumsikan bahwa karakter pertama dari *plain* teks memiliki indeks = 0).
4. Setelah itu, *subsequence*[*i*] dienkripsi dengan menggunakan algoritma ke-*i*, yang akan menghasilkan *subcipher*[*i*].
5. Kemudian, seluruh *subcipher* diintegrasikan menjadi *cipher text*, sesuai dengan urutan karakter pada *subsequence* pada *plain text*.

Contoh :

Misalkan *plain text* = “ABCDEFGHIJ”, *N* = 3. Maka *subsequence*[1] = “ADGJ”, *subsequence*[2] = “BEH”, *subsequence*[3] = “CFI”. Misalkan *subsequence plain text* pada contoh di atas menghasilkan *subcipher-subcipher* sebagai berikut, *subcipher*[1] = “KNQT”, *subcipher*[2] = “LOR”, *subcipher*[3] = “MPS”, maka *cipher text* yang dihasilkan adalah “KLMNOPQRST”.

B. *Dekripsi*

Proses dekripsi untuk algoritma kriptografi kunci simetris dengan modifikasi *vigenere cipher* ini dapat dilihat pada Gambar 4 berikut.



Gambar 4. *Flowchart* Dekripsi

Algoritma untuk *flowchart* tersebut secara tertulis adalah sebagai berikut :

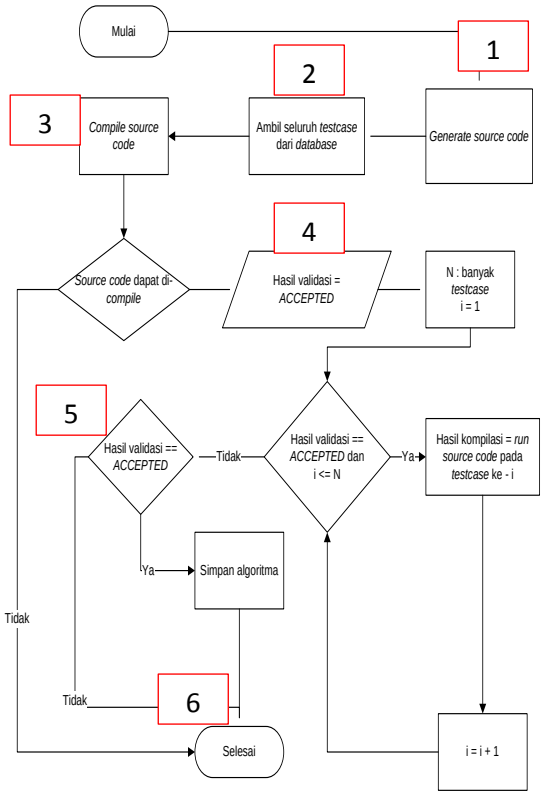
1. Pertama-tama *user* harus menginputkan *file* konfigurasi dan suatu bilangan bulat yang berperan sebagai kunci privat (pasangan kunci publik pada proses enkripsi).
2. Ulangi langkah 1 sampai *file* tersebut dapat diambil kontennya. Konten dari file konfigurasi adalah daftar algoritma kriptografi kunci simetris yang digunakan pada saat proses enkripsi berikut dengan nomor urut algoritma-algoritma tersebut.
3. Kemudian, *user* menginputkan *cipher text* dan satu buah kunci *string*.
4. Selanjutnya, *cipher text* dipartisi ke dalam *N* buah *subcipher* seperti pada proses enkripsi, di mana *N* adalah jumlah algoritma kriptografi kunci simetris yang digunakan yang terdaftar pada *file* konfigurasi
5. Berikutnya *subcipher*[*i*] didekripsi dengan menggunakan algoritma ke-i, menghasilkan *subsequence*[*i*].
6. Terakhir, seluruh *subsequence* diintegrasikan berdasarkan indeks *subcipher* pada *cipher text* untuk menghasilkan *plain text*.

Contoh :

Misalkan *cipher text* = “KLMNOPQRST”, N = 3. Maka *subcipher*[1] = “KNQT”, *subcipher*[2] = “LOR”, *subcipher*[3] = “MPS”. Misalkan *subcipher* dari *cipher text* pada contoh di atas menghasilkan *subsequnce-subsequence* sebagai berikut, *subsequence*[1] = “ADGJ”, *subsequence*[2] = “BEH”, *subsequence*[3] = “CFI”, maka *plain text* yang dihasilkan adalah “ABCDEFGHIJ”.

C. Flowchart Validasi Algoritma

Flowchart untuk proses validasi suatu algoritma kriptografi kunci simetris pada aplikasi yang akan dibangun dapat dilihat pada Gambar 5 berikut.



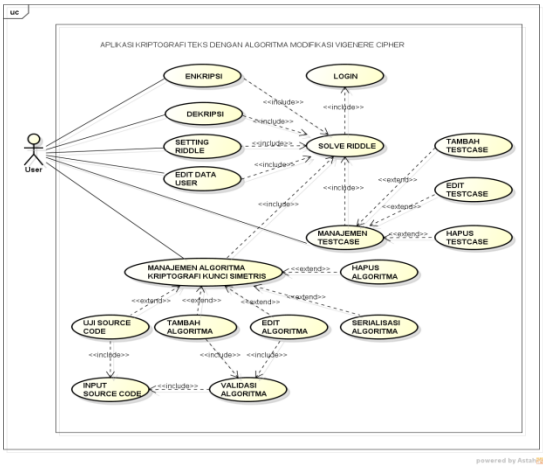
Gambar 5. *Flowchart* Proses Validasi

Algoritma untuk *flowchart* tersebut secara tertulis adalah sebagai berikut :

1. Pertama *source code* algoritma yang telah di-*chunked* (dibagi ke dalam 6 bagian, *nama class*, *import part*, *extract key part*, *encrypt part*, *decrypt part*, *other methods and variables part*) di-*generate* (disatukan kembali menjadi satu *class* tunggal dengan struktur sesuai format yang dibutuhkan aplikasi) oleh aplikasi.
2. Aplikasi kemudian mengambil seluruh *testcase* yang ada pada *database*.
3. *Generated source code* tersebut kemudian di-*compile*.
4. Jika *generated source code* tersebut dapat di-*compile* (*compilable*), maka aplikasi akan memvalidasi algoritma dengan cara menjalankan algoritma tersebut dari *testcase* 1 sampai dengan *testcase* terakhir dengan syarat selama keluaran validasi pada *testcase* yang dijalankan bersifat *ACCEPTED* (validasi dihentikan jika terdapat *testcase* yang menghasilkan keluaran tidak *ACCEPTED*).
5. Hasil validasi adalah nilai keluaran dari *testcase* terakhir yang menyebabkan validasi dihentikan (artinya jika algoritma divalidasi sampai *testcase* terakhir, maka hasil validasi adalah hasil validasi *testcase* terakhir).
6. Jika hasil validasi akhir adalah *ACCEPTED*, maka algoritma tersebut dapat disimpan pada database.

D. *Use Case Diagram*

Use case diagram aplikasi kriptografi teks ini dapat dilihat pada Gambar 6.

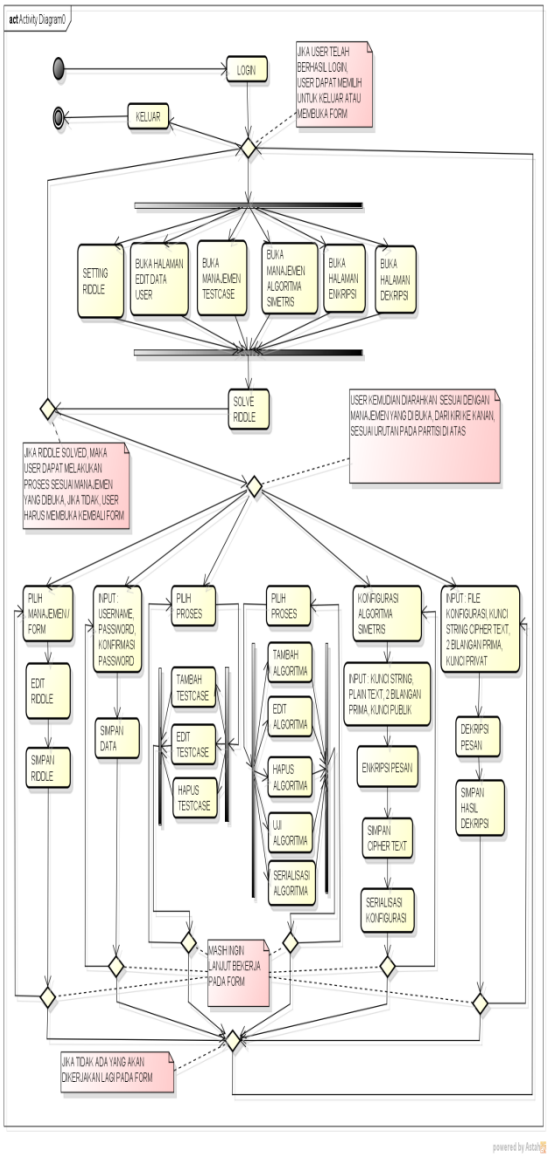


Gambar 6. *Use case Diagram*

Pada aplikasi ini hanya terdapat seorang aktor yang dinamakan *user*. Hanya ada 1 *user* yang bisa mengoperasikan aplikasi. Terdapat sejumlah manajemen data yang dapat dilakukan oleh *user*, dengan terlebih dahulu *user* harus *login* ke aplikasi. Masing-masing manajemen berada pada *form* yang masing-masing berbeda.

E. *Activity Diagram*

Berdasarkan *use case* yang telah didefinisikan sebelumnya, dihasilkanlah *Activity Diagram* aplikasi kriptografi teks dengan modifikasi *vigenere cipher* yang dapat dilihat pada Gambar 7.

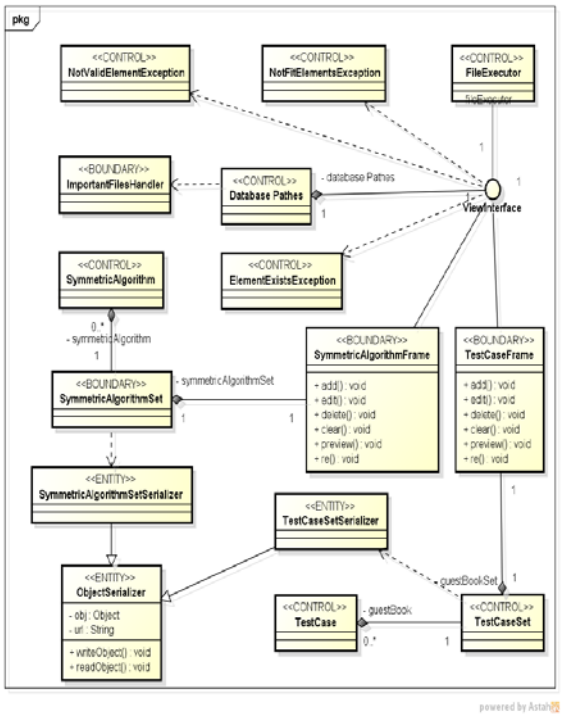


Gambar 7. *Activity Diagram*

Penggunaan aplikasi kriptografi teks ini dimulai dengan *user login* ke aplikasi dengan menginputkan *username* dan *password* pada *field* yang tersedia pada *form login*. Jika telah berhasil *login*, *user* akan diarahkan pada halaman utama aplikasi. Pada halaman utama, *user* dapat memilih untuk melakukan suatu pekerjaan pada aplikasi atau keluar dari aplikasi.

F. *Class Diagram*

Class Diagram aplikasi kriptografi teks ini dapat dilihat pada Gambar 8

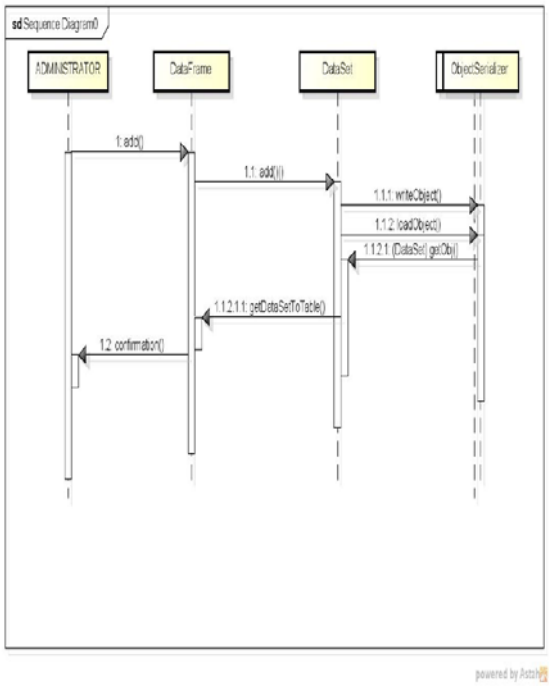


Gambar 8. *Class Diagram*

Pada sistem informasi ini, pembangunan aplikasi menggunakan bahasa pemrograman *java* dengan gaya pemrograman *object-oriented-programming* (OOP) dan metode pengklasifikasian *source code Controller-Model-View* (CMV).

G. *Sequence Diagram*

Sequence Diagram aplikasi kriptografi teks dapat dilihat pada Gambar 9.



Gambar 9. *Sequence Diagram*

V. HASIL DAN PEMBAHASAN

Aplikasi kriptografi teks pada penelitian ini dibangun dengan menggunakan bahasa pemrograman Java dan IDE Netbeans. Berikut tampilan aplikasi ini.

A. *Halaman Utama*

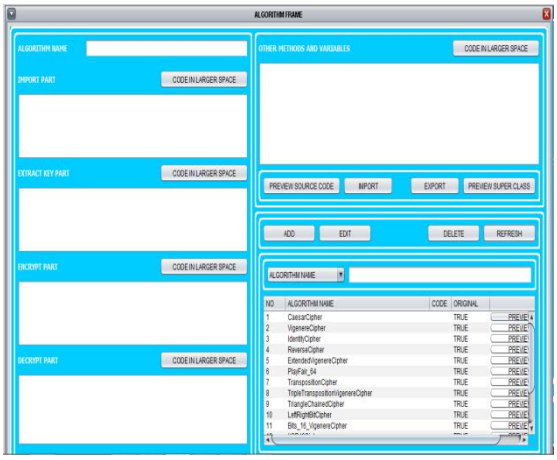
Halaman ini digunakan sebagai *outer frame* dari seluruh fitur yang dapat digunakan diaplikasi ini. Tampilan halaman utama aplikasi kriptografi teks dengan modifikasi *vigenere cipher* dapat dilihat pada Gambar 10 berikut :

Gambar 10. Halaman Utama

Fitur-fitur yang ada pada aplikasi ini antara lain : halaman *user*, manajemen *riddle*, manajemen *testcase*, manajemen algoritma kriptografi kunci simetris, enkripsi dan dekripsi. Fitur-fitur tersebut dapat diaktifkan dengan mengklik *submenu* yang tersedia pada *menu-menu* di *menu bar* pada halaman ini.

B. *Halaman Manajemen Kriptografi Kunci Simetris*

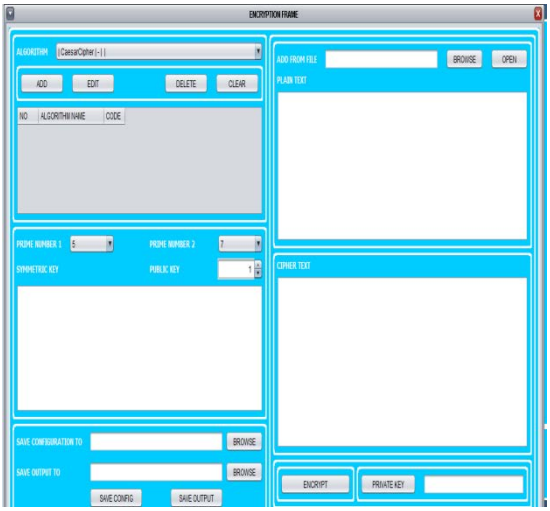
Halaman manajemen algoritma kriptografi kunci simetris digunakan untuk memanajemen algoritma kriptografi kunci simetris yang akan digunakan untuk proses enkripsi dan dekripsi pada aplikasi ini. Tampilan halaman manajemen algoritma kriptografi kunci simetris dapat dilihat pada Gambar 11 berikut :



Gambar 11. Halaman Manajemen Algoritma Kriptografi Simetris

C. Halaman Enkripsi

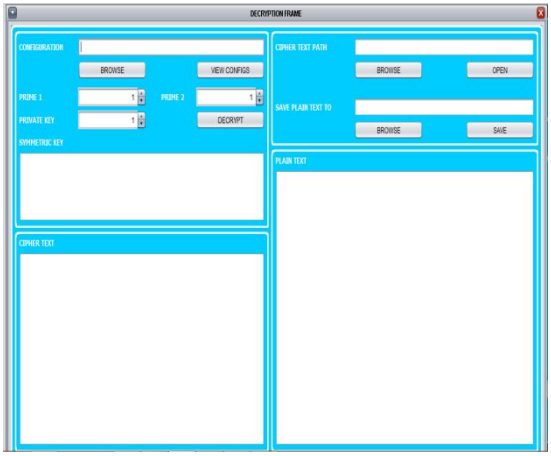
Halaman enkripsi digunakan untuk melakukan proses enkripsi pesan teks dengan metode modifikasi *Vigenere* yang diperkenalkan pada aplikasi ini. Tampilan halaman enkripsi dapat dilihat pada Gambar 12 berikut :



Gambar 12. Halaman Enkripsi

D. Halaman Dekripsi

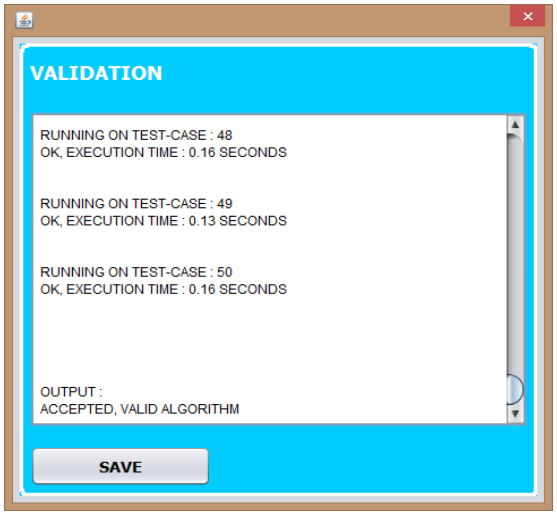
Halaman dekripsi digunakan untuk mendekripsi *cipher text* yang merupakan keluaran dari halaman enkripsi. Tampilan halaman dekripsi aplikasi kriptografi teks ini, dapat dilihat pada Gambar 13 berikut :



Gambar 13. Halaman Dekripsi

E. Validasi Algoritma Kriptografi Kunci Simetris

Hasil validasi algoritma yang telah diterjemahkan ke bentuk *source code* untuk diinputkan ke aplikasi dapat dilihat pada Gambar 14.



Gambar 14. Validasi Algoritma

F. *Pengujian Algoritma Kriptografi Kunci Simetris*

Hasil pengujian lengkap seluruh algoritma dapat dilihat pada Tabel 3 berikut :

Tabel 3 Hasil Pengujian Algoritma

No	Nama Algoritma	Output
1	*CE Cipher*	*Compile Error*
2	*CE Cipher 2*	*Compile Error*
3	*RTE Identity Cipher*	*Runtime Error on TC – 37*
4	*RTE Reverse Cipher*	*Runtime Error on TC – 6*
5	*TLE Infinite Loop*	*Time Limit Exceeded on TC – 1*
6	*TLE Identity Cipher*	*Time Limit Exceeded on TC – 17*
7	*Invalid Triple Transposition Vigenere Cipher*	*Invalid Output on TC – 10*
8	*Play Fair Cipher*	*Invalid Output on TC – 1*
9	*Vigenere Cipher*	*Accepted (OK)*
10	*Caesar Cipher*	*Accepted (OK)*
11	*Identity Cipher*	*Accepted (OK)*
12	*Reverse Cipher*	*Accepted (OK)*
13	*Extended Vigenere Cipher*	*Accepted (OK)*
14	*16-Bits Vigenere Cipher*	*Accepted (OK)*
15	*XOR-Bit Cipher*	*Accepted (OK)*
16	*XOR 48 Cipher*	*Accepted (OK)*
17	*Left-Right-Bit Cipher*	*Accepted (OK)*
18	*Transposition Cipher*	*Accepted (OK)*
19	*Play Fair 64 Cipher*	*Accepted (OK)*
20	*Complement-Bit Cipher*	*Accepted (OK)*
21	*Reverse-Bit Cipher*	*Accepted (OK)*
22	*Triple Transposition Vigenere Cipher*	*Accepted (OK)*
23	*Troll Identity Cipher*	*Accepted (OK)*

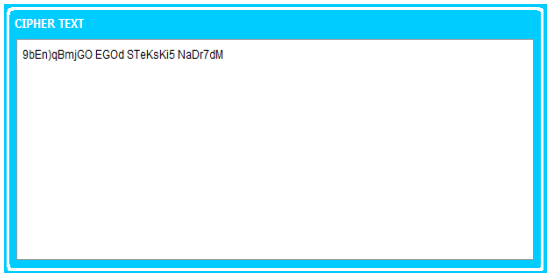
G. Pengujian Enkripsi

Misalkan terdapat teks : "Perkenalkan Saya *Vigenere* "Abstrak" (tanpa kutip), dan kunci *string* yang digunakan adalah : "Kripto" (tanpa kutip). Misalkan konfigurasi algoritma yang digunakan dapat dilihat pada Tabel 4 berikut:

Tabel 4. Konfigurasi Algoritma

No	Algoritma
1	*Left-Right-Bit Cipher*
2	*Play Fair 64 Cipher*

Hasil enkripsi menggunakan aplikasi dapat dilihat pada Gambar 15 berikut :



Gambar 15. *Cipher* Teks Enkripsi

Pembahasan hasil enkripsi tersebut adalah sebagai berikut :

1. Pertama-tama dibentuk 2 buah *subsequence* untuk masing-maisng metode sebagai berikut :

Subsequence untuk metode *Left-Right-Bit Cipher :*

PreaknSy ieeeAsrk

Subsequence untuk metode *Play Fair 64 Cipher :*

Eknla aaVgnr bta

2. Selanjutnya, setiap *subsequence* dienkripsi dengan menggunakan kunci string dengan metode masing-masing, dan diperoleh hasil sebagai berikut :

Subcipher untuk metode *Left-Right-Bit Cipher :*

9E)BjOEO TKK5ND7M

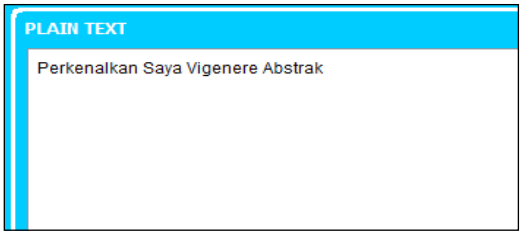
Subcipher untuk metode *Play Fair 64 Cipher :*

bnqmG GdSesi ard

3. Terakhir gabungkan seluruh *subcipher* dan diperoleh hasil seperti Gambar 15.

H. Pengujian Dekripsi

Pada pengujian proses dekripsi ini, *cipher* teks yang digunakan adalah *cipher* teks yang dihasilkan pada Gambar 15, dan konfigurasi serta kunci *string* yang digunakan adalah yang digunakan pada poin G. Hasil proses dekripsi tersebut dapat dilihat pada Gambar 16.



Gambar 16. Hasil Dekripsi *Cipher* Teks

Pembahasan hasil dekripsi tersebut adalah sebagai berikut :

1. Pertama-tama dibentuk 2 buah *subcipher* untuk masing-maisng metode sebagai berikut :

Subcipher untuk metode *Left-Right-Bit Cipher :*

9E)BjOEO TKK5ND7M

Subcipher untuk metode *Play Fair 64 Cipher :*

bnqmG GdSesi ard

2. Selanjutnya, setiap *subsequence* dienkripsi dengan menggunakan kunci string dengan metode masing-masing, dan diperoleh hasil sebagai berikut :

Subsequence untuk metode *Left-Right-Bit Cipher :*

PreaknSy ieeeAsrk

Subsequence untuk metode *Play Fair 64 Cipher :*

Eknla aaVgnr bta

3. Terakhir gabungkan seluruh *subsequence* dan diperolehlah hasil seperti Gambar 16.

VI. PENUTUP

A. *Kesimpulan*

Berdasarkan analisa perancangan sistem, implemantasi, dan pengujian sistem, maka dapat disimpulkan bahwa: penelitian ini telah berhasil menghasilkan suatu *proto-type* aplikasi yang menerapkan algoritma kriptografi kunci simetris

dengan modifikasi *Vigenere Cipher*. Modifikasi yang dilakukan adalah dengan mengenkripsi setiap *subsequence* yang dipartisi dari *plain text* sesuai metode *Vigenere Cipher* dengan algoritma kriptografi kunci simetris yang dapat saling berbeda, di mana jumlah *subsequence* yang dihasilkan adalah sebanyak algoritma kriptografi kunci simetris yang digunakan. Modifikasi ini menghasilkan suatu *cipher text* yang sulit untuk di-d*ecipher* oleh *cryptanalyst* tanpa adanya kunci ataupun konfigurasi algoritma (susunan algoritma) yang digunakan.

B. *Saran*

Berdasarkan hasil dari penelitian ini, penulis menyarankan : aplikasi ini dapat dikembangkan menjadi suatu aplikasi *client-server* yang dilengkapi fitur *chatting*, pengiriman pesan, pengiriman algoritma, atau pengiriman *file*, sehingga *User* dapat menggunakannya sebagai sarana mengirimkan kunci *string*, *file* konfigurasi, *cipher text*, ataupun algoritma kriptografi kunci simetris ke sesama pengguna aplikasi ini.

REFERENSI

[1] Arius, Dony. *Pengantar Ilmu Kriptografi dan Implementasi*. Yogyakarta. 2008.

[2] Hermawan, B. *Menguasai Java 2 & Object Oriented Programming.* Yogyakarta: ANDI. 2004.