**PENGEMBANGAN DAN IMPLEMENTASI METODE *CAESAR CIPHER* PADA APLIKASI *CHATTING* SEDERHANA**

**DEVELOPMENT AND IMPLEMENTATION OF CAESAR CIPHER METHOD INTO A SIMPLE CHAT APPLICATION**

**Mochamad Taufikurrohman1, Anief Fauzan Rozi2**

1,2Program Studi Sistem Informasi, Fakultas Teknologi Informasi, Universitas Mercubuana Yogyakarta

Jl. Wates Km. 10 Yogyakarta, 55753, Indonesia

Email: 15121014@student.mercubuana-yogya.ac.id1, anief@mercubuana-yogya.ac.id2

**ABSTRAK**

Kerahasiaan dan keamanan informasi yang ditransaksikan di era revolusi industri 4.0 harus menjadi perhatian khsusus. Pengembangan tentang kriptografi yang ditujukan untuk merahasiakan pesan sudah banyak dilakukan namun tidak sedikit pula para penyadap sudah mengetahui tehnik untuk meretasnya. Maka dari itu di dalam penelitian ini mencoba untuk mengembangkan algoritma yang mampu merahasiakan pesan teks kemudian mencobanya untuk diimplementasikan ke dalam aplikasi chatting sederhana. Pertama dalam melakukan penelitian ini adalah mengidentifikasi masalah yang kemudian dilanjutkan dengan mempelajari literatur tentang kriptografi. Setelah dirasa cukup maka langkah selanjutnya adalah merancang dan mengembangkan algoritma kriptografi serta aplikasi chatting sederhana. Hasil dari penelitian ini adalah sebuah algoritma yang dikembangkan dari metode caesar cipher dan aplikasi chating sederhana berbasis web dengan mengimplementasikan metode kriptografi yang sudah dikembangkan. Bahasa pemrograman yang digunakan untuk mengimplementasikan algoritma adalah PHP dan Javascript. Dari penelitian yang dilakukan dapat ditarik kesimpulan bahwa hasil dari implementasi pengembangan algoritma kriptografi mampu untuk mengamankan pesan teks saat pendistribusiannya baik dari sisi server ke client maupun sebalikanya.

**Kata kunci**: keamanan informasi, kriptografi, caesar cipher, aplikasi chating

**ABSTRACT**

The confidentiality and security of information transacted in the era of the industrial revolution 4.0 must become special attention. Cryptography developments aimed at keeping messages secret have been doing a lot but many tappers already know the techniques to hack it. Therefore in this study tried to develop an algorithm that could keep text message secret then tried it to be implemented in a simple chat application. First in conducting this research was to identify problems then continued with study the literature on cryptography. Once deemed sufficient, the next step was to design and develop a cryptographic algorithm and a simple chat application. The results of this study are an algorithm developed from the caesar cipher method and a simple web-based chat application which has implemented the cryptographic method that has been developed. The programming languages which are used to build a simple chat application and implement the algorithm are PHP and Javascript. From the research can be concluded that the algorithm of the cryptographic method which has been implemented can secure text messages when distributed from the server to the client or on the contrary.

**Keywords:** information security, cryptography, caesar cipher, chat Application

# PENDAHULUAN

Kebutuhan akan keamanan informasi dalam revolusi industri 4.0 semakin meningkat. Mengingat umat manusia sekarang sudah memasuki tahapan baru yaitu masyarakat informasi dimana kegiatan distribusi, penggunaan dan manipulasi informasi dalam aktifitas ekonomi, politik sampai budaya meningkat secara signifikan termasuk informasi yanng berbentuk pesan text.

Adapun kejahatan yang biasa mengancam adalah penyadapan suatu informasi yang dilakukan oleh oknum yang tidak bertanggung jawab melalui suatu jaringan internet. Apabila informasi tersebut tidak diamankan terlebih dahulu dan jatuh ke tangan pelaku maka dapat dipastikan bahwa informasi yang dikirimkan dapat diketahui secara mudah.

Penelitian tentang kriptografi yang menjadi solusi dalam mengmankan informasi dengan tehnik menyandikannya sudah banyak dilakukan. Namun tidak sedikit pula para pelaku berhasil mencari cara untuk meretasnya. Oleh karena itu penelitian tentang kriptografi harus terus dilakukan untuk terus memberikan keamanan dan kenyamanan masyarakat dalam bertransaksi informasi.

Adapun penelitian ini mencoba untuk mengembangkan suatu metode kriptografi klasik yaitu *caesar cipher* dan mencoba untuk mengimplementasikannya ke dalam sebuah aplikasi *chatting* sederhana.

# TINJAUAN PUSTAKA

Dalam penelitian M Miftakul Amin (2016) menyimpulkan bahwa *Caesar Cipher* dapat digunakan untuk melakukan enkripsi dan dekripsi pesan yang dikirimkan dalam aplikasi *chatting* dan dari sisi metode enkripsi dapat tambahkan beberapa pilihan tehnik enkripsi sehingga akan menjadikan aplikasi semakin tangguh untuk menjaga kerahasiaan data.

Adam Rotal Yuliandaru (2016), di dalam makalahnya mengatakan bahwa dalam dunia kriptografi ternyata huruf yang sama pada pesan mempunyai image huruf yang sama juga. Hal ini mempunyai tingkat resiko yang tinggi karena mudah ditebak. Unuk menyelesaikan hal ini maka pesan haruslah disandikan (*encoding*). Tujuan membuat *encoding* adalah agar aman dari para pembongkar sandi sehingga hanya penerima saja yang mengetahui isinya.

Penelitian yang dilakukan oleh Anjar Pradipta (2016) menyimpulkan bahwa dengan tidak adanya formula yang pasti dalam metode kriptografi *Caesar Chiper*, maka dapat dikatakan bahwa *Caesar Chiper* sulit untuk dipecahkan. Penyandian sangatlah penting dalam mengirim pesan, apalagi pesan tersebut bersifat sangat rahasia.

Sentot Kromodimoeljo (2009), dalam bukunya mengungkapkan bahwa enkripsi yang menggunakan *shift transformation* seperti *Caesar cipher* sangat rentan terhadap *known plaintext attack*. Jika pasangan naskah asli - naskah acak diketahui, parameter *b* dapat ditemukan dengan mudah.

Dari penelitian Mhd Arief Hasan, Supriadi dan Zamzi (2017) menyimpukan bahwa penerapan algoritma *Fisher-Yates* yang digunakan pada aplikasi CBT (*Computer Based Testing*) dapat mengacak soal yang terlihat pada perbedaan tampilan soal pada setiap peserta ujian sehingga dalam pelaksanaan ujian setiap mahasiswa dalam menjawab soal memiliki nomor yang sama tetapi bentuk soal yang berbeda.

## Algoritma Fisher – Yates Shuffle

Fisher-Yates Shuffle (dinamai berdasarkan penemunya, Ronald Fisher danFrank Yates) digunakan untuk mengubah urutan masukan yang diberikan secara acak. Permutasi yang dihasilkan oleh algoritma ini muncul dengan probabilitas yang sama. Metode dasar yang diberikan untuk menghasilkan permutasi acak dari angka 1 - N berjalan sebagai berikut (Haditama, Slamet, & Rahman, 2016):

1. Tuliskan angka dari 1 sampai n
2. Isi nilai k dengan bilangan acak antara 0 hingga i+1 bulatkan kebawah
3. Hitung dari low end, gantikan nilai k dan tuliskan di tempat lain
4. Ulangi dari langkah 2 sampai semua nomor digantikan
5. Urutan angka yang tertulis di langkah 3 sekarang permutasi acak dari nomor asli
6. Pada versi yang baru (*modern*) angka yang terpilih tidak dicoret, tetapi ditukar posisinya dengan angka terakhir dari angka yang belum terpilih.Tabel 2.1 adalah contoh pengerjaan dari versi *modern*. *Range* adalah jumlah angka yang belum terpilih, *roll* adalah angka acak yang terpilih, *scratch* adalah daftar angka yang belum terpilih, *result* adalah hasil permutasi yang akan didapatkan. Versi *modern* dikenalkan karena lebih optimal dibandingkan dengan versi dahulu. Prosesnya ialah angka terakhir akan dipindahkan ke angka yang ditarik keluar dan mengubah angka yang ditarik keluar menjadi angka akhir yang tidak ditarik lagi untuk setiap kali penarikan dan berlanjut untuk iterasi berikutnya.

## *Diffie – Hellman Key Exchange*

Algoritma ini pertama kali diperkenalkan oleh Whitfield Diffie dan Martin Hellman pada tahun 1975. Mereka berdua adalah peneliti pada universitas Stanford. Mereka memperkenalkan algoritma ini untuk memberi solusi atas pertukaran informasi secara rahasia. Algoritma ini tidak berdasarkan pada proses enkripsi dan dekripsi, melainkan lebih kepada proses matematika yang dilakukan untuk menghasilkan kunci rahasia yang dapat disebarkan secara bebas tanpa harus khawatir karena kunci rahasia tersebut hanya dapat didekripsi hanya oleh pengirim dan penerima pesan. Dasar dari algoritma ini adalah matematika dasar dari aljabar eksponen dan aritmatika modulus.

Langkah-langkah dalam pertukaran kunci dengan menggunakan algoritma Diffie-Hellman adalah sebagai berikut (Rahmatunnisya, 2017):

1. Pilih bilangan prima yang besar, *p* dan bilangan integer yang tidak melebihi dari nilai *p*, *g*, biasa disebut bilangan basis atau *generator*. Kedua bilangan tersebut dapat diketahui secara publik.
2. Pilih sebuah bilangan acak oleh pengirim, *x*, bilangan ini tidak boleh diketahui oleh orang lain.
3. Pilih sebuah bilangan acak oleh penerima, *y*, bilangan ini tidak boleh diketahui oleh orang lain.
4. Pengirim menghitung *A = gx mod p*. Bilangan *A* ini dapat diketahui secara publik.
5. Penerima menghitung *B = gy mod p*. Bilangan *B* ini dapat diketahui secara publik.
6. Lakukan pertukaran bilangan *A* dan *B* terhadap pengirim dan penerima.
7. Lalu Pengirim menghitung *ka = Bx mod p*
8. Penerima menghitung *kb = Ay mod p*

Berdasarkan hukum aljabar nilai ka sama dengan kb atau bisa disebut *ka = kb = k.* Sehingga pengirim dan penerima tersebut mengetahui kunci rahasia tersebut *“k”*.

## Kriptografi *Caesar Cipher*

Subttitusi kode yang pertama dalam dunia penyandian tercatat pada waktu pemerintahan Yulius Caesar yang dikenal dengan *Caesar Cipher* (sandi kaisar) yang mengganti posisi huruf awal dari alfabet atau disebut juga dengan algoritma ROT3 (Ariyus & K.R, 2008).

**Tabel ‎2.1 Caesar Cipher (ROT3)**

|  |  |
| --- | --- |
| **Plain Text** | **Encoded Text** |
| ABC | DEF |
| Hello | Khoor |
| Attack | Dwwdfn |

Jika pergeseran dilakukan sebanyak tiga kali maka kunci untuk dekripsinya adalah 3. Pergeseran kunci yang dilakukan tergantung keinginin dari pengirim pesan, bisa saja kunci yang dipakai a = 7, b = 9 dan seterusnya.

Contoh dari algoritma sandi kaisar: Untuk teks asli diberikan simbol “P” dan teks sandinya “C” dan kunci “K”. jadi rumusnya dapat dibuat sebagai berikut:

*C=E(P) = (P+K) mod (26)*

Dari contoh diatas kita bisa memasukan kunci dengan nilai tiga sehingga menjadi:

*C=E(P) = (P+3) mod (26)*

Rumus untuk dekripsinya adalah sebagai berikut:

*P=D(C) = (C-K) mod (26)*

Dari contoh di atas, dengan memasukan kunci tiga maka:

*P=D(C) = (C-3) mod (26)*

# METODOLOGI

Dalam penelitian yang dilakukan akan melalui beberapa tahapan. Hasil akhir dari penelitian adalah sebuah metode enkripsi teks yang dikembangkan dari algoritma caesar cipher dengan menggabungan beberapa metode lainnya dan mengimplementasikannya ke dalam sebuah aplikasi chatting sederhana.. Tahapan penelitan dapat dilihat pada gambar berikut.



Gambar 3.1 Jalan Penelitian

## Algoritma *Fisher – Yates Shuffle*

Nama algortima *Fisher-Yates Shuffle* diadopsi berdasarkan nama penemunya yaitu Ronal Fisher dan Frank Yates. Algoritma ini digunakan untuk mengubah urutan masukan yang diberikan secara acak dengan probabilitas permutasi yang dihasilkannya adalah sama.

Dalam pengembangan metode enkripsi ini, algoritma *Fisher-Yates Shuffle* digunakan untuk mengacak urutan abjad yang nantinya akan digunakan dalam proses enkripsi dan dekripsi. Urutan abjad disebut sebagai *dictionary* yang dalam implementasinya terbagi menjadi 2, antara lain:

1. *dictionary1*, sebuah larik atau vector yang berisi urutan abjad yang digunakan untuk proses *Caesar cipher* dan besifat rahasia.
2. *dictionary2,* sebuah matriks dua dimensi yang memetakan urutan abjad yang digunakan untuk proses *Matrix Cipher* dan bersifat rahasia*.*

## *Diffie-Hellman Key exchange*

Metode pertukaran kunci yang digunakan dalam penelitian adalah pertukaran kunci *Diffie-Hellman*. Pertukaran kunci ini dilakukan untuk menentukan banyaknya pergeseran huruf enkripsi ­*Caesar Cipher* yang akan diterapkan pada urutan abjad atau *dictionay.* Metode ini dipilih karenakeamanan algoritmanya didasakan pada sulitnya menghitung logaritma diskrit.

## Metode *Caesar Cipher*

Algoritma *Caesar cipher* pada penelitian ini digunakan untuk mengenkripsi pesan pada tahap pertama, dimana pada sesi mengenkripsi pesan akan menggunakan urutan abjad yang sebelumnya sudah dilakukan pengacakan menggunakan metode *Fisher-Yates Shuffle*. kemudian setelah enkripsi *Caesar Cipher* dilakukan, pesan akan di enkrispi lagi menggunakan metode enkripsi matriks.

Selain untuk mengenkripsi pesan, metode *Caesar Cipher* juga digunakan untuk mengenkripsi urutan abjad atau yang dalam penelitian ini disebut dengan *dictionary*, baik itu *dictionary1* maupun *dictionary2*. Dalam melakukan enkripsi *dictionary*, proses enkripsi menggunakan urutan abjad publik. Sesuai dengan Namanya, urutan abjad publik bersifat umum dan boleh ditransmisikan melalui jaringan umum.

Jumlah pergeseran huruf yang terdapat dalam metode enkripsi *Caesar Cipher* pada penelitian ini bersifat kondisional. Ketika metode ini digunakan untuk mengenkripsi pesan, maka jumlah pergeseran huruf sesuai dengan panjang pesan itu sendiri dan kemudian ketika metode ini digunakan untuk mengenkrispi urutan abjad atau *dictionary,* maka jumlah pergeseran huruf ditentukan oleh hasil dari metode pertukaran kunci *Diffie-Hellman*. Dalam kata lain pertukaran kunci yang dihasilkan dari metode *Diffie-Hellman* digunakan untuk mengenkripsi *private key* yang berupa urutan acak abjad.

## Metode *Matrix Cipher*

*Matrix Cipher* adalah sebuah metode enkripsi yang merubah setiap huruf di dalam *plaintext* menjadi angka, yang mana angka tersebut adalah merupakan sebuah indeks dari *array* dua dimensi. Jadi prinsip dasar dari metode ini adalah mencocokan setiap huruf yang ada di dalam sebuah *plaintext* dengan urutan abjad yang sudah dipetakan menjadi array dua dimensi yang dalam penelitian ini disebut dengan *dictionary* kemudian merubah huruf yang ada di dalam *plaintext* tersebut dengan angka yang menjadi indeks dari huruf yang mencocoki di dalam *array*. Berikut adalah gambaran bagaimana metode ini merubah huruf menjadi angka:

* Asumsikan terdapat sebuah urutan abjad yang sudah dipetakan menjadi array dua dimensi yang kita sebut sebagai *dictionary* sebagai berikut:

Gambar 3.2 *Dictionary* urutan abjad

* Terdapat sebuah *plaintext* yang berupa OPICK
* Setelah melalui proses enkripsi maka akan menjadi 1415080210

Pada penelitan ini, metode *matrix cipher* merupakan proses enkripsi yang dilakukan pada tahap kedua setelah proses enkripsi pertama yang mengunakan metode *caesar cipher* dijalannkan. Pada metode ini enkripsi menggunakan *dictionary2* yang mana ini adalah sebuah urutan abjad yang sudah dipetakan menjadi array dua dimensi.

Di dalam pelaksanaanya, *dictionary2* dibuat dari urutan abjad satu atau *dictionary1* yang sudah diacak menggunakan algoritma *fisher-yates shuffle* dan kemudian dipetakan menjadi array dua dimensi. Setelah proses pemetaan selesai dilakukan, *dictionary2* diacak ulang menggunakan algoritma *fisher-yates shuffle.* Namun pengacakan dilakukan di setiap indeks baris dari array tersebut.

Tugas utama yang dilakukan oleh enkripsi matriks adalah selain untuk mengenkripsi pesan, metode ini juga digunakan untuk mengenkripsi *dictionary*, baik *dictionary1* maupun *dictionary2*. Dalam proses enkripsi *dictionary1* dan *dictionary2* metode *matrix cipher* menggunakan *dictionary public dua* yang mana urutan abjad ini dibentuk dari *dictionary public satu* yang dipetakan menjadi array dua dimensi.Sama dengan metode enkripsi *caesar cipher*, *matrix* *cipher* juga dijadikan sebuah fungsi yang ada di dalam *class*. Adapun parameter yang dibutuhkan ada dua yaitu: *plaintext* dan *dictionary* yang akan digunakan.

## Proses Enkripsi

Sebelum melakukan proses enkripsi ada dua tahapan yang harus dijalankan terlebih dahulu, yaitu: melakukan pertukaran kunci yang dilakukan dengan menggunakan metode *diffie-hellman key exchange* dan melakukan proses inisialisasi urutan abjad atau *dictionary* dengan menggunakan algoritma *fisher-yates shuffle.* Kemudian untuk proses enkripisi dijadikan sebuah fungsi yang di dalamnya ada beberapa fungsi yang harus dijalankan, antara lain: fungsi untuk melakukan proses enkripsi dengan metode c*aesar cipher*, fungsi untuk melakukan proses enkripsi dengan metode *matrix cipher* kemudian yang terakhir adalah melakukan proses mengatur atau memformat ulang text yang sudah terenkripsi. Perlu diketahui bahwa ketika akan mengenkripsi *dictionary2* yang merupakan array dua dimensi maka sebelumnya harus dijadikan string terlebih dahulu.

Adapun isitilah yang digunakan untuk proses memformat ulang *ciphertext* adalah *packaging*, yaitu sebuah proses akhir setelah pengenkripsian pesan dan *dictionary* dilakukan yang di dalamnya terdapat proses memotong, mengacak dan menggabungkan kembali semua elemen yaitu pesan dan *dictionary*. Secara garis besar konsep kerja *packaging* adalah pesan dan *dictionary* yang sudah terenkripsi digabungkan menjadi satu string. Ini bertujuan selain untuk menambah kerumitan juga untuk mengirimkan *dictionary* yang digunakan dalam proses pengenkripsian. Berikut adalah gambaran konsep *packaging*:

* Asumsikan terdapat sebuah pesan dan *dictionary* terenkripsi.

Gambar 3.3 Asumsi dictionary dan pesan yang sudah terenkripsi

* Potong *dictionary1* dan pesan menjadi dua bagian.

Gambar 3.4 potongan dictionary dan pesan

* Susun dan gabungkan semua elemen yang ada dengan pengacakan sebagai berikut

Gambar 3.5 mengacak dan menggabungkan elemen menjadi satu baris *string*

## Proses Dekripsi

Adapun di dalam proses dekripsi terdapat sebuah fungsi *unpackaging* yaitu kebalikan dari fungsi *packaging* yang bertugas untuk mengembalikan rangkaian atau susunan *ciphertext* yang sudah dibentuk dalam proses enkripsi menjadi bentuk seperti semula yang terdiri dari tiga bagian, yaitu: *dictionary1*, *dictionary2* dan pesan yang sudah di enkripsi itu sendiri. Kemudian dilanjutkan dengan proses dekripsi dari metode *caesar cipher* dan *matrix cipher*. Namun sebelum proses dekripsi dijalankan ada beberapa syarat yang harus dipenuhi terlebih dahulu, yaitu: mengetahui pergeseran kunci yang didapatkan dari proses *Diffie-Hellman*.

## Perancangan Aplikasi *Chatting Sederhana*

Aplikasi *chatting* yang akan dibangun bertujuan untuk mengimplementasikan metode enkripsi yang sudah dikembangkan. Maka dari itu pengembangan yang dilakukan masih bersifat sederhana, dalam artian belum layak untuk diproduksi secara industri dan belum layak untuk dipublikasikan.

## Relasi Tabel

Pada penelitian ini relasi yang digunakan adalah relasi satu ke banyak karena satu pengirim dapat memiliki banyak pesan akan tetapi satu pesan hanya dimiliki oleh satu pengirim. Adapun relasi tabel tersebut dapat di lihat pada gambar 3.7 berikut.

## Rancangan Halaman *Login*

Gambar 3.6 Relasi Tabel

Halaman *login* digunakan untuk memeriksa apakah user memang benar benar berhak untuk mengakses halaman yang dituju. Berikut adalah reancangan halaman login yang memberi hak user untuk memberikan informasi hak akasesnya.



Gambar ‎3.7 Rancangan Halaman Login

## Rancangan Halaman Selamat Datang

Halaman awal yang akan muncul apabila user telah melakukan *login* dan berhasil dalam proses pengecekan. Tampilan halaman dapat dilihat pada gambar berikut.

Gambar ‎3.8 Rancangan Halaman Selamat Datang

## Halaman *Chatting*

Halaman *chatting* dapat dikases setelah *user* melakukan login. Halaman ini digunakan oleh *user* untuk mengirimkan pesan yang akan di kirim kepada member yang terdapat didalam aplikasi.

Gambar 3.9 Rancangan Halaman *Chatting*

## PEMBAHASAN

## Pengembangan Metode Kriptografi

Penjelasan tentang metode yang sudah dikembangkan akan diilustrasikan dengan asumsi ada seorang yang bernama Alice ingin mengirim pesan kepada Bob. Adapun pesan yang akan dikirim adalah sebagai berikut.

Tabel 4.1 Contoh Pesan Teks

|  |
| --- |
| Opick Tamvan Sekali, ya…! |

Kemudian asumsikan bahwa Alice dan Bob sudah menyepakati karakter yang akan digunakan adalah sebagai berikut.



Gambar 4.1 Huruf Untuk Berkomunikasi (*Dictionary Public*)

### Pertukaran Kunci

Jumlah kunci akan diciptakan sebanyak dua buah karena pergeseran huruf dalam metode *caesar cipher* diberlakukan untuk dua urutan abjad yang digunakan dalam proses pengenkripsian. Adapun proses pertukaran kuncinya adalah sebagai berikut:

1. Alice dan Bob menyepakati bilangan prima *g1 = 19, n1 = 37* untuk menciptakan kunci satu dan *g2 = 31, n2 = 43* untuk menciptakan kunci dua. Nilai *g* lebih kecil dari *n* dan bersifat umum serta dapat ditransmisikan melalui jaringan tidak aman.
2. Alice membangkitkan bilangan bulat acak *x1 = 71* dan *x2 = 67*. Bilangan ini bersifat rahasia.
3. Bob membangkitkan bilangan bulat acak *y1 = 41* dan *y2 = 69*. Bilangan ini berifat rahasia.
4. Alice menghitung

*X1 = g1x1 mod n1 =* *1971 mod 37 = 2*

*X2 = g2x2 mod n2 = 3167 mod 43 = 10*

kemudian mengirimkannya kepada Bob. Bilangan ini bersifat umum.

1. Bob menghitung

*Y1 = g1y1 mod n1 = 1941 mod 37 = 22*

*Y2 = g2y2 mod n2 = 3169 mod 43 = 21*

kemudian mengirmkannya kepada Alice. Bilangan ini bersifat umum.

1. Alice menghitung kunci pergeseran

*K1 = Y1x1**mod n1 = 2271 mod 37 = 32*

*K2 =Y2x2 mod n2 = 2167 mod 43 = 35*

1. Bob menghitung kunci pergeseran

*K1 = X1y1* *mod n1 = 241 mod 37 = 32*

*K2 = X2y2 mod n2 = 1069 mod 43 = 35*

Dengan metode pertukaran kunci *diffie – hellman* maka orang pertama yang dalam contoh adalah Alice dan Bob sebagai orang kedua akan dapat menentukan jumlah pergeseran huruf yang sama tanpa pihak lain mengetahuinya karena nilai *x* dan *y* hanya diketahui oleh yang membangkitkannya saja dan yang dapat diketahui oleh *public* hanyalah nilai *n, g,* *X* dan *Y*.

### Proses Inisialisasi

Proses inisialisasi dilakukan untuk membentuk *dictionary1* dan *dictionary2*. Untuk *dictionary1* ini dilakukan dengan mengacak *dicitonary public* dengan algoritma *fisher yates shuffle* dan untuk *dictionary2* dibuat dari pemetaan *dictionary1* menjadi *array* dua dimensi dan diacak di setiap indeks baris. Berikut adalah hasilnya.



Gambar 4.2 Huruf Yang Sudah Diacak (*Dictionary1*)



Gambar 4.3 Hasil Pengacakan *Array* Dua Dimensi (*Dictionary2*)

### Proses Enkripsi

Proses enkripsi akan diberlakukan untuk pesan, *dictionary1* dan *dictionary2* kemudian ditutup dengan proses *packaging*. Adapun proses enkripsi dilakukan dengan dua metode yaitu: *Caesar cipher* dan *matrix cipher*.

Dalam metode *caesar cipher* kunci pergeserannya tergantung apa yang akan dienkripsi. Jika pesan maka kuncinya adalah panjang dari pesan itu sendiri, jika *dictionary1* maka kuncinya adalah kunci satudan jika *dictionary2* maka kuncinya adalah kunci dua

Adapun penggunaan *dictionary* untuk setiap metode enkripsi akan berbeda tergantung *plaintext* apa yang akan dienkripsi. Berikut adalah detailnya.

Tabel 4.1 Detail Penggunaan D*ictionary*

|  |  |  |
| --- | --- | --- |
| Plainext | Caesar Cipher | Matrix Cipher |
| Pesan | Dictionary1 | Dictionary2 |
| Dictionary1 | Dictionary public | Dictionary2 Public |
| Dictionary2 | Dictionary1 | Dictionary2 Public |

 Setelah semua proses enkripsi dijalankan maka akan menghasilkan keluaran berikut

Tabel 4.2 Hasil Enkripsi Pesan Tahap Satu Dengan *Caesar Cipher*

|  |
| --- |
| DozXs vH6YHN itsH[z8 SHgggh |

Tabel 4.3 Hasil Enkripsi Pesan Tahap Dua Dengan *Matrix Cipher*

|  |
| --- |
| 7602570831ZO341812521832PL20613118735737QS701817171724 |

Dapat dilihat hasil dari enkripsi *matrix cipher* di atas mempunyai huruf yang ada di antara deretan angka. Hal ini dikarenakan dalam pesan yang dienkripsi terdapat karakter spasi dan huruf – huruf tersebut menggantikan karakter spasi itu sendiri. Adapun huruf tersebut dihasilkan secara acak dengan rentang antara A – Z dan setiap karakter spasi akan digantikan dengan dua huruf.

Tabel 4.4 Hasil Enkripsi *Dictionary1* Tahap Satu Dengan *Ceasar* *Cipher*

|  |
| --- |
| YWA[z4]dR!PleLk;nqgVarxwGX6:StB3b.N/8Tiv,'(O1?u2EfoF\\CI9\"50h|Ks)MQ7DUZcpmjJyH |

Tabel 4.5 Hasil Enkripsi *Dicitionary1* Tahap Dua Dengan *Matrix Cipher*

|  |
| --- |
| 242200715155722917681537301136643942322126434948066623576518450154276213765919344763746914526746530431400575020860735661337710447012165803202528413835095007 |

Tabel 4.6 Hasil Enkripsi *Dictionary2* Tahap Satu Dengan *Caesar cipher*

|  |
| --- |
| 8qfb9C:lPsjOVwzry|IUBtmcMYp3d5D'7x.aG[J-n4o)XS2T!u,6g\\/FHk0EQAi1(]KR?L\";WZNevh |

Tabel 4.7 Hasil Enkripsi *Dictionary2* Tahap Dua Dengan *Matrix Cipher*

|  |
| --- |
| 594231276002653715443514214851435077082001453828122441542956037458496226067109663955407023185319684663573275760507366104160034526972101767117364222513304733 |

Tabel 4.8 Ciphertext Hasil *Packaging*

|  |
| --- |
| 2422007151557229176815373011366439423221264349480666235765184501542762137659197602570831ZO341812521832PL25942312760026537154435142148514350770820014538281224415429560374584962260671096639554070231853196846635732757605073661041600345269721017671173642225133047330613118735737QS701817171724344763746914526746530431400575020860735661337710447012165803202528413835095007 |

Dapat dilihat hasil dari proses terakhir yaitu *packaging* dimana warna kuning menunjukan *dictionary1*, warna hijau menunjukan *dicionary2* dan warna merah adalah pesan itu sendiri. Setelah tahap ini kemudian Alice akan mengirimkan pesan kepada Bob.

### Proses Dekripsi

Proses dekripsi dilakukan berdasarkan kebalikan dari proses enkripsi. Adapapun urutan prosesnya adalah *unpackaging*, dekripsi *matrix cipher* dan dekripsi *caesar cipher*.

Langkah pertama adalah mengambil *dictionary1* dari keseluruhan *ciphertext*. Hal ini dilakukan dengan cara sebagai berikut:

1. *x* *=* ambil karakter dari *ciphertext* mulai dari *0* sampai *panjang\_dictionary\_public*
2. *y =* ambil karakter dari *ciphertext* mulai dari *panjang\_ciphertext* –*panjang\_dictionary\_public* sampai *panjang ciphertext*.
3. *dicitionary1 = x + y*

Selanjutnya adalah mengambil *dictionary2* dari *ciphertext* yang dikirimkan Alice dimana prosesnya adalah sebagai berikut:

1. *panjang\_pesan = panjang\_ciphertext – (panjang\_dictionary\_public \* 4)*

*x =* ambil karakter dari *ciphertext* mulai dari *panjang\_dictionary\_public* *+ panjang\_pesan / 2* sampai *panjang\_dictionary\_public \* 2*.

Proses *unpackaging* yang terakhir adalah bertujuan untuk mengambil pesan itu sendiri dimana prosesnya adalah sebagai berikut:

1. *panjang\_pesan = panjang\_ciphertext – (panjang\_dictionary\_public \* 4)*
2. *x =* ambil karakter dari *ciphertext* mulai dari *panjang\_dictionary\_public* sampai *panjang\_pesan / 2*.
3. *y =* ambil karakter dari *ciphertext* mulai dari *panjang\_ciphertext – (panjang\_dictionary\_public + panjang\_pesan / 2)* sampai *panjang\_ciphertext / 2*.
4. *pesan = x + y*

dari proses *unpackaging* di atas maka menghasilkan keluaran berikut.

Tabel 4.9 *Dictionary1* Hasil Proses *Unpackaging*

|  |
| --- |
| 242200715155722917681537301136643942322126434948066623576518450154276213765919344763746914526746530431400575020860735661337710447012165803202528413835095007 |

Tabel 4.10 *Dictionary2* Hasil Proses *Unpackaging*

|  |
| --- |
| 594231276002653715443514214851435077082001453828122441542956037458496226067109663955407023185319684663573275760507366104160034526972101767117364222513304733 |

Tabel 4.11 Pesan Hasil Proses *Unpackaging*

|  |
| --- |
| 7602570831ZO341812521832PL20613118735737QS701817171724 |

Proses selanjutnya adalah mendekripsi setiap *dictionary* dan yang terakhir pendekripsian pesan itu sendiri. Adapun dalam penggunaan *dictionary* dalam setiap proses dekripsi sama halnya dengan pada saat proses enkripsi. Berikut adalah hasilnya.

Tabel 4.12 Hasil Dekripsi *Dictionary1* Tahap Satu Dengan *Matrix Cipher*

|  |
| --- |
| YWA[z4]dR!PleLk;nqgVarxwG-X6:StB3b.N/8Tiv,'(O1?u2EfoF\\CI9\"50h|Ks)MQ7DUZcpmjJyH |

Tabel 4.13 Hasil Dekripsi *Dictionary1* Tahap Dua Dengan *Caesar Cipher*

|  |
| --- |
| )!unTXo\,k0F/6EgHKA?]LRQ1i(Zh;NvW"e8sb:CPfql9UjOVy|Izrw3cpYdBt5Mm7.ax-['JGD4S2 |

Tabel 4.14 Hasil Dekripsi *Dictionary2* Tahap Satu Dengan *Matrix Cipher*

|  |
| --- |
| 8qfb9C:lPsjOVwzry|IUBtmcMYp3d5D'7x.aG[J-n4o)XS2T!u,6g\\/FHk0EQAi1(]KR?L\";WZNevh |

Tabel 4.15 Hasil Dekripsi *Dictionary2* Tahap Dua Dengan *Caesar Cipher*

|  |
| --- |
| )\oukTn,X!F/6?KAEgH0i(;LhQR]1ZCsN"vW:8bejPylVfqO9UIcY|3wdzrp.txa-MB75mSJ4[G2D' |

Tabel 4.16 Hasil Dekripsi Pesan Tahap Satu Dengan *Matrix Cipher*

|  |
| --- |
| DozXs vH6YHN itsH[z8 SHgggh |

Tabel 4.16 Hasil Dekripsi Pesan Tahap Dua Dengan *Caesar Cipher*

|  |
| --- |
| Opick Tamvan Sekali, ya…! |

## Aplikasi *Chatting* sederhana

 Pengembagan aplikasi *chatting* sederhana yang dilakukan menggunakan *framework codeigniter*, *javascript*, *jquery* dan *bootstrap* mampu dalam mengimplementasikan algoritma enkripsi dan dekripsi yang sudah direncanakan. Ada beberapa teknik yang digunakan dalam aplikasi ini. Berikut akan dijelaskan beberapa teknik yang digunakan.

Gambar 4.5 Halaman *Welcome*

Gambar 4.4 Halaman *Login*

## Pertukaran Data

Aplikasi *chatting* yang dikembangkan dalam melakukan pertukaran data terutama dalam mengirim dan menerima pesan menggunakan teknik *Asynchronous JavaScript and XML* atau yang biasa di singkat sebagai *AJAX*. Hal ini dilakukan karena dengan teknik ini dapat memungkinkan sisi klien dengan sisi server dapat bertukar data tanpa harus menyegarkan halaman *website*. Ini sangat membantu kelancaran proses pertukaran kunci pergeseran.

### *Realtime Data Update*

Pembaruan data yang dilakukan secara berkala dilakukan ketika pengguna memasuki halaman *chatting*. Hal ini bertujuan untuk memastikan bahwa pesan terbaru yang masuk dapat langsung dibaca se – segera mungkin oleh penerima pesan tersebut.

Adapaun teknik yang digunakan dalam pembaruan data adalah *short polling*, yaitu klien akan mengirimkan *request* ke *server* setiap 15 detik sekali untuk memastikan apakah ada pesan terbaru yang masuk atau tidak dan dalam setiap kali melakukannya klien akan menyiapkan propereti untuk melakukan pertukaran kunci.

### Antar Muka

Aplikasi *chating* sederhana yang dihasilkan dalam penelitian ini tidak jauh berbeda dengan apa yang sudah dirancang sebelumnya. Proses utama yang sudah direncanakan dalam perancangan tidak ada perbedaan. Namun dalam hal desain antarmuka sedikit berbeda dengan hasil yang sudah dikerjakan. Berikut adalah hasilnya.



# KESIMPULAN

Hasil dari pengembangan dan implementasi algotitma *caesar cipher* ini, ada beberapa kesimpulan yang didapat diantaranya:

1. Algoritma yang dikembangkan dapat mengamankan proses pengiriman pesan dari *server* ke klien ataupun sebaliknya.
2. Aplikasi *chatting* sederhana mampu mengimplementasikan algoritma kriptografi yang dikembangkan dalam penelitian ini.

# DAFTAR PUSTAKA

Arie. (2015, November 07). *Terminologi Kriptografi*. Retrieved from Blog Binadarma: http://blog.binadarma.ac.id/ariezaki/?p=505

Ariyus, D., & K.R, R. A. (2008). *Komunikasi Data.* Yogyakarta: Andi.

Haditama, I., Slamet, C., & Rahman, D. F. (2016). IMPLEMENTASI ALGORITMA FISHER-YATES DAN FUZZY TSUKAMOTO DALAM GAME KUIS TEBAK NADA SUNDA BERBASIS ANDROID. *JOIN* .

Harumy, T. H., Windarto, A. P., & Sulistianingsih, I. (2016). *Belajar Dasar Algoritma dan Pemrograman C++.* Medan.

Hasan, M. A., Supriadi, & Zamzami. (2017). Implementasi Algoritma Fisher-Yates Untuk Mengacak Soal Ujian Online Penerimaan Mahasiswa Baru (Studi Kasus : Universitas Lancang Kuning Riau). *Jurnal Tteknologi dan Sistem Iinformasi*.

Kromodimoeljo, S. (2009). *Teori Dan Aplikasi Kriptografi.* SPK IT Consulting.

Musfiroh, A. (2017, Oktober 19). *Pengertian, Tujuan dan Jenis-jenis Kriptografi & Rumus Penyelesaian*. Retrieved from Mastekno: https://www.mastekno.com/id/pengertian-tujuan-dan-jenis-jenis-kriptografi-rumus-penyelesaian/

Pradipta, A. (2016). Implementasi Metode Caesar Chiper Alphabet Majemuk Dalam Kriptografi Untuk Pengamanan Informasi. *Indonesian Journal on Networking and Security*.

Gambar 4.6 Halaman *Chating*

Pratama, A. R. (2019). *Belajar UML*. Retrieved juli 23, 2019, from www.codepolitan.com

Rahmatunnisya. (2017, Oktober 29). *Enkripsi*. Retrieved from UGM: http://rahmatunnisya.web.ugm.ac.id/2017/10/29/enkripsi/

Yuliandru, A. R. (2016). Teknik Kriptografi Hill Cipher Menggunakan Matriks. *Makalah IF2123*.