IMPLEMENTASI KRIPTOGRAFI PADA TEKS MENGGUNAKAN ALGORITMA TANAM PADI DAN BAJAK SAWAH

SKRIPSI

OLEH

MIF'ATUL MAHMUDAH

NIM.H72214011

PROGRAM STUDI MATEMATIKA

JURUSAN SAINS

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI SUNAN AMPEL SURABAYA

SURABAYA

2018

PERNYATAAN KEASLIAN

Yang bertandatangan dibawah ini:

Nama :Mif'atul Mahmudah

NIM : H72214011

Program Studi : Matematika

Angkatan : 2014

Menyatakan bahwa saya tidak melakukan plagiat dalam penulisan skripsi saya yang berjudul: **Implementasi Kriptografi Pada Teks Menggunakan Algoritma Tanam Padi Dan Bajak Sawah.** Apabila suatu saat nanti terbukti saya melakukan tindakan plagiat, maka saya akan menerima sanksi yang telah ditetapkan. Demikian pernyataan keaslian ini saya buat dengan sebenar-benarnya.

Surabaya, 3 Agustus 2018

Mif'atul Mahmudah
NIM.H72214011

LEMBAR PENGESAHAN

DETEKSI JARAK PANDANG AMAN SEBAGAI ACUAN UNTUK KESELAMATAN PENERBANGAN DENGAN MENGGUNAKAN METODE *BACKPROPAGATION*

Disusun oleh
Moch. Rizki Kurniawan Hakim
NIM.H72214017

Telah dipertahankan di depan Dewan Penguji
Pada tanggal 20 Juli 2018
Dan dinyatakan telah memenuhi syarat
untuk memperoleh gelar
Sarjana Matematika (S.Mat)

Dewan Penguji

Penguji I

Aris Fanani, M.Kom
NIP.198701272014031002

Penguji II

Yuniar Farida, MT
NIP.197905272014032002

Penguji III

Dian C. Rini N., M.Kom
NIP.198511242014032001

Penguji IV

Wika Dianita Utami, M.Sc
NIP.199206102018012003

Mengesahkan
Dekan Fakultas Sains dan Teknologi
UIN Sunan Ampel Surabaya

Dr. Eni Purwati, M.Ag
NIP.196512211990022001

KEMENTERIAN AGAMA
UNIVERSITAS ISLAM NEGERI SUNAN AMPEL SURABAYA
PERPUSTAKAAN
Jl. Jend. A. Yani 117 Surabaya 60237 Telp. 031-8431972 Fax.031-8413300
E-Mail: perpus@uinsby.ac.id

LEMBAR PERNYATAAN PERSETUJUAN PUBLIKASI
KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademika UIN Sunan Ampel Surabaya, yang bertanda tangan di bawah ini, saya:

Nama : MIF'ATUL MAHMUDAH
NIM : H72214011
Fakultas/Jurusan : SAINS DAN TEKNOLOGI
E-mail address : atulmahmudah88@gmail.com

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Perpustakaan UIN Sunan Ampel Surabaya, Hak Bebas Royalti Non-Eksklusif atas karya ilmiah :
☑ Sekripsi ☐ Tesis ☐ Desertasi ☐ Lain-lain (.....)
yang berjudul :

IMPLEMENTASI KRIPTOGRAFI PADA TEKS MENGGUNAKAN TEKNIK TANAM PADI DAN BAJAK SAWAH

beserta perangkat yang diperlukan (bila ada). Dengan Hak Bebas Royalti Non-Ekslusif ini Perpustakaan UIN Sunan Ampel Surabaya berhak menyimpan, mengalih-media/format-kan, mengelolanya dalam bentuk pangkalan data (database), mendistribusikannya, dan menampilkan/mempublikasikannya di Internet atau media lain secara ***fulltext*** untuk kepentingan akademis tanpa perlu meminta ijin dari saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan atau penerbit yang bersangkutan.

Saya bersedia untuk menanggung secara pribadi, tanpa melibatkan pihak Perpustakaan UIN Sunan Ampel Surabaya, segala bentuk tuntutan hukum yang timbul atas pelanggaran Hak Cipta dalam karya ilmiah saya ini.

Demikian pernyataan ini yang saya buat dengan sebenarnya.

Surabaya, 02-08-2018

Penulis

(Mif'atul Mahmudah)

IMPLEMENTASI KRIPTOGRAFI PADA TEKS MENGGUNAKAN ALGORITMA TANAM PADI DAN BAJAK SAWAH

ABSTRAK

Berkembangnya teknologi dan komunikasi memudahkan manusia dalam bertukar informasi dengan dunia luar, sehingga keamanan data yang dikirim rawan terhadap penyadapan. Untuk menghindari penyadapan data ataupun informasi yang akan dikirim perlu dilakukan pengamanan, salah satu ilmu yang dapat digunakan dalam mengamankan data adalah kriptografi. Tujuan dari penelitian ini adalah penyandian teks dan penguraian proses enkripsi dan dekripsi menggunakan algoritma tanam padi dan bajak sawah. Metode tersebut dipilih karena memiliki kecepatan enkripsi dan dekripsi lebih cepat dibandingkan dengan metode AES (*Advanced Encryption Standard*). Proses pembangkitan kunci dimulai dengan merubah kunci yang telah ditentukan yaitu "muslimah" kedalam bentuk biner, selanjutnya pemasukan dan pengambilan bit dengan pola bajak sawah diproses hingga delapan kali, diperoleh hasil akhir yaitu: 00111111 01100101 01101000 01010000 11001101 11110000 01111100 01011110. Proses enkripsi dimulai dengan merubah *plainteks* "mahmudah" kedalam bentuk biner, selanjutnya pemasukan dan pengambilan bit dilakukan dengan pola tanam padi, hasil dari pengambilan *plainteks* dan pembangkitan kunci kemudian di XoRkan, diproses sampai delapan putaran, sehingga menghasilkan enkripsi *"ú*EOT*è*Z*üü*b*ʊ*". Proses dekripsi yaitu chipperteks di XoR kan dengan kunci yang telah dibangkitkan. Hasil XoR dimasukan kedalam bit menggunakan pola pengambilan *plainteks* dan diambil dengan pola pemasukan *plainteks*, setelah itu hasilnya di XoR kan dengan kunci yang telah dibangkitkan. Proses tersebut diulang sampai delapan kali putaran, menhhasilkan dekripsi yaitu "mahmudah".

Kata kunci : *Plainteks*, Kunci, Kriptografi, Tanam padi, Bajak sawah, *Block chipper.*

IMPLEMENTATION OF CHYPTOGRAPHYON THE TEXS USING PLANTING RICE AND PLOW FIELDS ALGORITHM

ABSTRACT

The development of technology and communication makes it easier for people to exchange information with outside world, so the security of data sent is vulnerable to wiretapping. To avoid tapping data or information to be sent need to be done security, one of science that can be used in securing data is cryptography. The purpose of this research is text encoding and decryption of encryption and decryption process using the method of planting rice and plow fields. The method is chosen because it has faster encryption and decryption speed than the AES method. Key generation process start from change key that has been determined "muslimah" into binary digit, after that input and output bit with plow field technique, this process repeat eight time. There result of key generation is 00111111 01100101 01101000 01010000 11001101 11110000 01111100 01011110. Encryption process start from change plaintext that has been determined "mahmudah" into binary digit, after that input and output bit with planting rice technique, result plaintext and key generation using logica xor, this process repeat eight time, there result process eight time. Decryption process start from chippertext uusing logica xor with key generation, after that input bit with pattern removal plaintekxt, and take bit with entering plaintext, there result logical xor, repeat process eight time. There result decryption "mahmudah".

Keywords: Plaintext, Key, Cryptography, Planting rice, Plow fields, Block Chipper.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1

PENDAHULUAN

A. Latar Belakang

Kemajuan teknologi dan komunikasi yang berkembang begitu pesat, sangat memudahkan manusia dalam berkomunikasi dengan dunia luar dan dengan mudah bertukar data dengan orang luar. Dengan mudahnya pertukaran data maupun informasi, ada hal yang kurang disadari yaitu keamanan dari data yang dikirim kepada orang lain, karena data ataupun informasi yang dikirim kepada pihak lain rawan terhadap penyadapan maupun pencurian data oleh pihak yang tidak bertanggung jawab. Untuk menghindari pencurian data maka data ataupun informasi yang akan dikirim kepada pihak luar perlu diamankan terlebih dahulu. Salah satu yang dapat dipergunakan adalah dengan kriptografi.

Ada beberapa penelitian terdahulu mengenai pengamanaan teks, seperti penelitian yang telah dilakukan oleh Ana Kurniawati dan Muhammad Dwiky Darmawan pada tahun 2016 yang berjudul "Implementasi Algoritma *Advanced Encryption Standard (AES)* untuk Enkripsi dan Dekripsi pada dokumen teks". Penelitian ini membahas tentang cara mengamankan file dokumen dengan menggunakan algoritma kriptografi *Advanced Encryption Standard (AES*) yang dibuat di perangkat lunak. Pengamanan dibuat di perangkat lunak dengan tujuan untuk mengamankan dokumen sebelum mengirimnya melalui jaringan internet. Hanya orang yang berkepentingan

dan yang mempunyai kata sandi yang bisa membuka file tersebut. Penelitian ini melakukan analisa kebutuhan dari *Advanced Encryption Standard (AES*). Kebutuhan yang pertama yakni data berupa dokumen *word* (*.doc/x), *Plain Text* (*.txt) dan PDF (*.pdf), yang kedua adalah kebutuhan fungsional yakni berupa kemampuan untuk meng-enkripsi dan dekripsi dokumen, yang ketiga adalah kebutuhan non fungsional berupa perangkat keras dan perangkat lunak berupa Windows 7 Ultimate 32-bit dan Netbeans 8.1. Perancangan perangkat lunak ini disesuaikan agar aplikasi *user-friendly*. Selanjutnya adalah pengujian 15 dokumen yang berformat pdf, txt, doc. Ukuran yang diuji adalah 16-498 KB (Kurniawati & Darmawan, 2016).

Penelitian kedua tentang pengamanan file dokumen adalah penelitian yang dilakukan oleh Fresly Nandar Pabokory, Indah Fitri Astuti, dan Awang Harsa pada tahun 2015 yang berjudul "Implementasi Kriptografi Pengamanan Data pada Pesan Teks, Isi File Dokumen, dan File Dokumen Menggunakan Algoritma *Advance Encryption Standard*". Penelitian ini membahas tentang sistem keamanan data dengan mengimplementasikan kriptografi dengan algoritma *Advance Encryption Standard (AES)* pada pesan teks, isi file dokumen, dan file dokumen. Hal yang mendasar pada metode ini adalah blok *chipertext* simetrik yang dapat mengenkripsi dan dekripsi informasi. Implementasi pada pengamanan pesan teks, isi file dokumen, dan file dokumen pada penelitian ini menggunakan aplikasi Fres-CAESAS. Dan selanjutnya dilakukan pengujian terhadap aplikasi Fres-

CAESAS untuk mengetahui keakuratan perangkat lunak (Pabokory, Astuti, & Kridalaksana, 2015).

Mayoritas pengamanan file dokumen pada penelitian terdahulu menggunakan metode AES karena metode AES lebih cepat. Namun ada pengembangan penelitian yang membandingkan metode AES dengan metode yang dikembangkan dari *chipper block* yaitu metode tanam padi dan bajak sawah. Peneliti yang mengembangkan metode dari *chipper block* yaitu Ahmad Widodo, Alz Danny Wowor, Evangs Mailoa, dan Magdalena A.Ineke Pakereng pada tahun 2015 dengan penelitian berjudul "Perancangan Kriptografi *Block Chipper* Berbasis Pada Algoritma Tanam Padi dan Bajak Sawah". Dalam penelitiannya ini dibahas tentang pengembangan dari metode *chipper block* yang terinspirasi dari kearifan lokal sehingga diberi nama tanam padi dan bajak sawah, dimana tanam padi sebagai *plainteks* dan bajak sawah sebagai kunci. Setelah algoritma tersebut sudah selesai, untuk mengetahui kecepatan dari algoritma tanam padi dan bajak sawah tersebut dibandingkan dengan metode AES. Setelah dibandingkan, menunjukan bahwa metode tanam padi dan bajak sawah lebih cepat dalam mengamankan teks dibandingkan dengan metode AES-128 (Widodo, Wowor, Mailoa, & Pakereng, 2015).

Kriptografi juga digunakan untuk pengamanan sebuah data ataupun informasi yang bersifat rahasia. Dalam kriptografi terdapat algoritma enkripsi dan dekripsi. Enkripsi adalah suatu proses penyandian dari

plainteks menjadi *chiperteks*, sedangkan dekripsi adalah suatu proses pengembalian dari *chiperteks* kedalam bentuk awal atau *plainteks*.

Chipper block adalah rangkaian dari bit yang dibagi menjadi blok-blok yang panjangnya sudah ditentukan sebelumnya. *Plainteks* akan diproses dengan panjang blok yang tetap. Setiap *Chipper block* akan memproses teks biasanya 64 bit. Agar kriptanalis (orang yang dapat memecahkan sebuah sandi) tidak mudah untuk memecahkan suatu kunci. Dalam proses enkripsi *Chipper block* menggabungkan beberapa algoritma kriptografi klasik. Metode *chipper block* sendiri terdapat beberapa algoritma seperti *Chipper Block Chaining* (CBC), *electronik code book* (ECB), *chipper feedback* (CFB), *output feedback* (OFB).

Pengembangan dari metode *Chipper block* sangat banyak sekali, salah satu yang digunakan dalam penelitian ini yaitu algoritma tanam padi dan bajak sawah. Algoritma Tanam padi digunakan untuk mengubah *plainteks* dalam bentuk *chipperteks*, sedangkan bajak sawah digunakan untuk membangkitkan kunci.

B. Rumusan Masalah

Dari latar belakang yang telah dipaparkan dapat diambil rumusan masalah dalam penelitian ini yaitu :

1. Bagaimana proses dan hasil pembangkitan kunci menggunakan algoritma bajak sawah?

2. Bagaimana proses dan hasil enkripsi menggunakan algoritma tanam padi dan bajak sawah?
3. Bagaimana proses dan hasil dekripsi menggunakan algoritma tanam padi dan bajak sawah?

C. Tujuan

Dari rumusan yang telah dipaparkan, tujuan dari penelitian ini yaitu :

1. Mengetahui proses dan hasil dari pembangkitan kunci menggunakan algoritma bajak sawah.
2. Mengetahui proses dan hasil enkripsi menggunakan algoritma tanam padi dan bajak sawah.
3. Mengetahui proses dan hasil enkripsi menggunakan algoritma tanam padi dan bajak sawah.

D. Batasan Masalah

Dalam penelitian diperlukan batasan masalah karena cakupan pembahasan yang sangat luas, adapun batasan masalah dari penelitian ini adalah

1. Algoritma tanam padi untuk enkripsi *plainteks* dan algoritma bajak sawah untuk pembangkitan kunci.
2. Panjang teks yang disandikan 8 karakter, kunci untuk penyandian mengikuti panjangnya plainteks.

E. Manfaat penelitian

Hasil penelitian diharapkan dapat memberikan manfaat kepada yang bersangkutan, sebagai berikut

1. Bagi Peneliti

   Menambah pengetahuan mengenai penerapan kriptografi untuk menyandikan teks.

2. Bagi Pihak Lain

   Dengan adanya penyandian, maka seseorang akan lebih tenang dalam mengirim pesan kepada pihak lain.

3. Bagi Universitas

   Agar dapat dijadikan sebagai acuan untuk penelitian selajutnya, dan dijadikan sebagai referensi bagi pihak perpustakaan Universitas Islam Negeri Sunan Ampel Surabaya.

F. Sistematika Penyusunan

Sistematika penyusunan proposal dalam penelitian ini dibagi menjadi tiga bab yaitu:

1. Bab kesatu adalah Pendahuluan, yang berisi mengenai latar belakang penelitian, rumusan masalah penelitian, tujuan penelitian, batasan masalah penelitian, dan sistematika penyusunan penelitian.
2. Bab kedua adalah Kajian Teori, berisi tentang mengenai teori-teori yang digunakan dalam penelitian ini yaitu, mengenai file dokumen, kriptografi, chiper blok, dan algoritma tanam padi dan bajak sawah.

3. Bab ketiga adalah Metode Penelitian, berisi mengenai alur atau jalannya penelitian ini yaitu jenis penelitian, data yang digunakan, tahapan-tahapan yang dilakukan dalam penelitian, dan tahapan mengenai metode yang dilakukan dalam penelitian.
4. Bab keempat adalah Hasil dan Pembahasan, berisi mengenai proses enkripsi, pembangkitan kunci, dan proses Dekripsi.
5. Bab kelima adalah Penutup, berisi kesimpulan dari penelitian ini dan saran untuk kepenulisan selanjutnya.

BAB II

TINJAUAN PUSTAKA

A. Teks

Menurut Alex Sobur teks adalah seperangkat tanda yang ditransmisikan dari seorang pengirim kepada seorang penerima melalui medium tertentu atau kode-kode tertentu.

Sedangkan menurut Kamus Besar Bahasa Indonesia teks didefinisikan sebagai Naskah yang berisi kata-kata asli dari pengarang, kutipan dari kitab suci untuk pangkal ajaran atau alasan, bahan tertulis untuk dasar memberikan pelajaran, berpidato, dan sebagainya, dan wacana tertulis (Sunenda, 2018).

Jenis-jenis teks dibedakan menjadi beberapa diantaranya :

a. Teks Anekdot yaitu teks yang berisi peristiwa-peristiwa lucu.
b. Teks Dekripsi yaitu menggambarkan keadaan secara lebih detail mengenai sesuatu.
c. Teks Diskusi yaitu berisi tinjauan terhadap sebuah isu dari dua sudut pandang.

B. Kriptografi

Kata Kriptografi berasal dari bahasa Yunani yaitu *crypto* yang berarti rahasia dan *graphia* yang berarti tulisan. Berarti kata kriptografi adalah tulisan rahasia (Nurhardian & Pudoli, 2016).

Kriptografi merupakan ilmu dan seni untuk menjaga kerahasiaan sebuah pesan dengan cara menyandikannya ke dalam bentuk yang tidak dapat dipahami lagi maknanya. Menurut Menezes, kriptografi adalah sebuah ilmu yang mempelajari algoritma-algoritma matematika yang berhubungan dengan aspek keamanan informasi, misalnya kerahasiaan, integritas data, serta otentikasinya (Munir, 2006). Kriptografi adalah ilmu sekaligus seni untuk menjaga keamanan pesan (*message*). Kriptografi juga digunakan untuk pengamanan sebuah data ataupun informasi yang bersifat rahasia.

Berdasarkan sejarah, kriptografi sudah digunakan oleh bangsa Mesir pada ribuan tahun yang lalu hingga abad ke 20. Ada empat kelompok yang membantu perkembangan kriptografi yaitu kalangan militer, kalangan diplomatik, penulis buku harian, dan pecinta. Namun dalam hal ini kalangan militer adalah kalangan yang sangat berkontribusi karena proses enkripsi dan deskripsi sangat dibutuhkan di medan perang. Secara umum kriptografi klasik dibagi menjadi dua yaitu algoritma transposisi dan algoritma subsitusi. Algoritma transposisi (*transposition chipper*) adalah merubah susunan huruf-huruf dalam bentuk pesan, sedangkan algoritma subsitusi (*subsitution chipper*) adalah mengganti setiap huruf atau kelompok huruf dengan sebuah huruf (Munir, 2006).

Sejarah kriptografi mencatat penggunaan kriptografi oleh tentara Sparta di Yunani pada tahun 400 SM, dimana telah ada alat untuk mengirim pesan rahasia dengan nama *Scytale*. *Scytale* adalah alat yang memiliki pita panjang dari daun *papyrus* dan sebatang silinder. Pesan ditulis di atas pita yang

dililitkan dari batang silinder lalu dikirim, untuk membaca pesan maka pita tersebut dililitkan kembali pada sebatang silinder yang diameternya sama sehingga yang menjadi kunci pada *scytale* adalah diameter silindernya (Nurhardian & Pudoli, 2016).

Kriptografi juga digunakan untuk menangani masalah keamanan yang terdiri dari empat hal berikut.

1. Kerahasiaan (*confidentiality*)

   Hal tersebut berkaitan dengan keaslian pengirim atau masalah tersebut dapat dinyatakan sebagai pertanyaan "Apakah pesan yang diterima benar-benar dari pengirim sesungguhnya?". Untuk memeriksanya menggunakan kriptografi dengan tujuan tersebut.

2. Integritas data (*data integrity*)

   Hal tersebut berkaitan dengan keutuhan atau perubahan pesan. Dengan arti lain masalah tersebut dapat dinyatakan dalam pertanyaan "Apakah pesan yang diterima tidak mengalami perubahan (modifikasi)?"

3. Nirpenyangkalan (*non-repudiation*)

   Hal tersebut akan menunjukkan bahwa pengirim tidak dapat menyangkal (berbohong) bahwa dia adalah yang mengirim pesan.

4. Otentikasi (*authentication*)

   Layanan yang berhubungan dengan identifikasi, baik mengidentifikasi kebenaran pihak-pihak yang berkomunikasi maupun mengidentifikasi kebenaran sumber pesan.

Enkripsi merupakan proses penyandian *plaintext* menjadi *chipertext* (Rosnawan, 2011). Sedangkan dekripsi merupakan proses mengembalikan *chipertext* menjadi *plaintext*-nya. Adapun pengertian lain, enkripsi adalah proses yang dilakukan untuk mengamankan pesan (biasanya disebut *plaintext*) menjadi pesan yang tersembunyi (disebut *chipertext*). *Chipertext* merupakan suatu pesan yang sudah tidak dapat dibaca dengan mudah. Sedangkan proses sebaliknya yaitu poses dekripsi adalah untuk mengubah *chipertext* menjadi *plaintext*. Pada bidang kriptografi, enkripsi mempunyai arti yaitu proses mengamankan suatu informasi dengan membuat informasi tersebut tidak dapat dibaca tanpa bantuan pengetahuan/ilmu khusus.

Algoritma kriptografi juga mempunyai konsep matematis yang merupakan relasi antara dua buah himpunan yaitu himpunan yang berisi elemen-elemen *plaintext* dan himpunan yang berisi *chipertext*. Enkripsi dan dekripsi adalah fungsi yang memetakan elemen-elemen antara kedua himpunan tersebut. Misalkan P menyatakan *plaintext* dan C menyatakan *chipertext*, maka fungsi enkripsi E memetakan $P\ ke\ C$, $E(P) = C$ Dan fungsi deskripsi D memetakan $C\ ke\ P$, $D(C) = P$, Kemudian proses dekripsi mengembalikan pesan ke pesan asal maka persamaannya menjadi berikut ini, $D(E(P)) = P$.

C. Chipper Block

Berdasarkan cara memproses teks (*plainteks*), *chipper* di kategorikan menjadi dua jenis yaitu *chipper block* dan *stream chipper*. *Chipper block* bekerja dengan memeproses data secara blok, dimana beberapa

karakter/data digabungkan menjadi satu blok. Setiap proses satu blok menghasilkan keluaran satu blok juga. Sementara itu *stream cipher* bekerja memproses masukan (karakter atau data) secara terus menerus dan menghasilkan data pada saat yang bersamaan.

Chipper block merupakan suatu algoritma yang membagi rangkaian bit menjadi blok-blok bit yang panjangnya sudah ditentukan sebelumnya. *Plainteks* akan diproses dengan panjang blok yang tetap. Pada data yang panjang maka dilakukan pemecahan dalam bentuk blok yang lebih kecil. Jika dalam pemecahan dihasilkan blok data yang kurang dari jumlah data dalam blok maka akan dilakukan proses *pading*. Pada umumnya, setiap *chipper block* memproses teks dengan blok yang relatif panjangnya lebih dari 64-bit, untuk mempersulit algoritma kriptanalisis dalam membongkar kunci. Dalam *chipper block* menggunakan kunci simetris untuk proses enkripsi dan dekripsinya karena kunci merupakan parameter yang digunakan untuk transformasi *plainteks* ke *chipperteks* (Munir, 2006).

Misalkan blok *plainteks* (P) yang berukuran *n* bit dinyatakan sebagai vektor

$$P = (P_1, P_2, \ldots, P_n) \quad (2.1)$$

Dalam hal ini p_i adalah 0 atau 1 untuk $i = 1,2,\ldots,n$ dan blok *chipperteks* (C) adalah :

$$C = (c_1, c_2, \ldots, c_n) \quad (2.2)$$

Dalam hal ini c_i adalah 0 atau 1 untuk $i = 1,2,\ldots,n$ bila *plainteks* dibagi menjadi m buah blok, barisan blok-blok plainteks dinyatakan sebagai

$(P_1, P_2, \ldots, P_m)$ untuk setiap blok plainteks p_i, bit-bit penyusunannya dapat dinyatakan sebagai vektor

$$p_i = (P_1, P_2, \ldots, p_m) \quad (2.3)$$

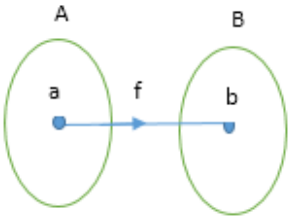
Enkripsi dan dekripsi dengan kunci K dinyatakan berturut-turut $E_k(p) = C$ adalah enkripsi dan $D_k(C) = P$. Fungsi E harus berkorespondensi satu satu. $E^1 = D$ (Munir, 2006), seperti yang ditunjukan oleh Gambar 2.1.

D. Fungsi

Misalkan A dan B himpunan, relasi biner f dari $A\ ke\ B$ merupakan suatu fungsi jika setiap elemen di dalam A dihubungkan dengan tepat satu elemen di dalam B. Jika f adalah fungsi dari $A\ ke\ B$ dituliskan

$f: A \rightarrow B$ yang artinya f memetakan $A\ ke\ B$.

Nama lain untuk fungsi adalah pemetaan atau transformasi. Dapat dituliskan dengan $f(a)b$ jika elemen a di dalam A dihubungkan dengan elemen b di dalam B. Himpunan A disebut daerah asal (*domain*) dari f dan himpunan B disebut daerah hasil (*codomain*) dari f . Gambar dibawah ini merepresentasikan fungsi dari $A\ ke\ B$.

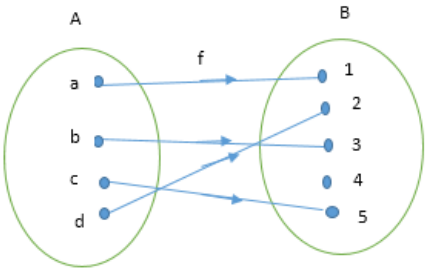


Gambar 2.1 Fungsi

Fungsi adalah relasi yang khusus. Kekhususan ini tercakup pada dua hal penting yaitu:

1) Tiap elemen di dalam himpunan A, yang merupakan daerah asal f, harus digunakan oleh produser atau kaidah yang mendefinisikan f.
2) Frasa ”dihubungkan dengan tepat satu elemen didalam B” berarti bahwa jika $(a, b) \in f$ dan $(a, c) \in f$, maka $b = c$.

Fungsi f dari himpunan A ke himpunan B dikatakan satu-ke-satu *(one-to-one)* atau injektif *(injektive)* jika tidak ada elemen himpunan A yang memiliki bayangan sama, dengan kata lain, jika $a \; dan \; b$ adalah anggota himpunan A, maka $f(a) \neq f(b)$ bilamana $a \neq b$. Jika $f(a) = f(b)$ maka implikasinya adalah $a = b$, Gambar 2.2 mengilustrasikan fungsi satu-ke-satu.



Gambar 2.2 Fungsi satu-satu

Relasi $f = \{(1, w), (2, u)\ (3, v)\ \}$ dari $A\ =\ \{1,2,3\}$ ke $B = \{u, v, w, x\}$ adalah fungsi satu-ke-satu. Relasi $f = \{(1, \mathrm{w}), (2, \mathrm{u}), (3, \mathrm{v})\}$ dari $A = \{1,2,3\}$ ke $B = \{u, v, w\}$ juga fungsi satu-ke-satu, tetapi relasi $f = \{(1, u), (2, u), (3, v)\}$ dari $A = \{1,2,3\}$ ke $B = \{u, v, w\}$ bukan fungsi satu-ke-satu, karena ada satu himpunan dari kodomain yang mempunyai 2 pasangan di domain, yaitu $f(1) = f(2) = u$.

E. Logika Proposisi

Logika proposisi adalah bidang logika yang membentuk proposisi pada pernyataan yang mengandung peubah. Proposisi sendiri adalah kalimat yang mengandung nilai benar atau salah. Benar dalam hal ini adalah kesesuaian antara apa yang dinyatakan dengan fakta yang sebenarnya. Konsep logika terdapat beberapa operator logika untuk menggabungkan proposisi proposisi yang ada (Irawati, 2017). Macam-macam operator logika ada banyak seperti konjungsi, disjungsi, implikasi, dan biimplikasi. Namun dalam penelitian ini hanya menggunakan operator logika disjungsi.

Disjungsi adalah gabungan dua pernyataan yang menggunakan kata penghubung logika "atau" sehingga membentuk dua pernyataan majemuk. Disjungsi dilambangkan dengan operator " $\vee$ " . Pernyataan *p dan q* dalam disjungsi dapat ditulis "$p \vee q$". Disjungsi ada dua macam yaitu disjungsi *inklusif* dan disjungsi *eksklusif*. Disjungsi inklusif adalah dua pernyataan yang bernilai benar apabila paling sedikit satu dari keduannya bernilai benar, sedangkan disjungsi eksklusif (XOR) adalah dua pernyataan bernilai benar apabila hanya satu dari dua pernyataan bernilai benar. Tabel 2.2

menjelaskan disjungsi *inklusif* (OR), sedangkan Tabel 2.3 menjelaskan disjungsi *eksklusif* (XOR).

Tabel 2.1: Tabel kebenaran OR

P	q	$p \vee q$
B	B	B
B	S	B
S	B	B
S	S	S

Tabel 2.2: Tabel kebenaran XOR

P	q	$p \ominus q$
B	B	S
B	S	B
S	B	B
S	S	S

F. Representasi Bilangan dan Operasi Aritmatika

Bilangan mempunyai dua tipe yaitu bilangan bertanda dan bilangan tak bertanda. Bilangan bertanda (*signed*) adalah bilangan yang memuat nilai positif dan negatif, sedangkan bilangan tak bertanda (*unsigned*) adalah bilangan yang hanya memuat nilai positif.

1. Bilangan Integer dan Desimal

Bilangan desimal atau yang disebut dengan bilangan radix-10 atau base-10, karena digitnya mempunyai nilai 10 yang mungkin dan tiap digit berbobot pangkat 10. Dalam sistem desimal atau *unsigned integer* memuat digit yang mempunyai nilai 0-9. Bilangan desimal n-digit dapat dinyatakan sebagai $D = d_{n-1}d_{n-2} \dots d_1 d_0$. Bilangan D tersebut mewakili nilai *integer* yang dapat direpresentasikan :

$$V(D) = d_{n-1} \times 10^{n-1} + d_{n-2} \times 10^{n-2} + \cdots + d_1 \times 10^1 + d_0 \times 10^0 \qquad (2.4)$$

Dimana d adalah bilangan desimal dan n adalah banyaknya bilangan desimal.

Misalkan diberikan contoh bilangan berikut 8547, bilangan tersebut dapat disajikan dalam bentuk

$$8 \times 10^3 + 5 \times 10^2 + 4 \times 10^1 + 7 \times 10^0$$

Representasi bilangan diatas disebut representasi posisional.

2. Bilangan Biner

Bilangan biner adalah bilangan yang baik untuk penyimpanan data maupun untuk pemrosesan suatu operasi karena hanya mempunyai 2 nilai yaitu 0 atau 1, bilangan ini juga bisa dikatakan bilangan dengan basis 2. Representasi posisional bilangan biner n-digit adalah $B = b_{n-1}b_{n-2} \dots b_1 b_0$

$$V(B) = b_{n-1} \times 2^{n-1} + b_{n-2} \times 2^{n-2} + \cdots + b_1 \times 2^1 + b_0 \times 2^0 = \sum_{i=0}^{n-1} b_i \times 2^i \quad (2.5)$$

Sebagai contoh dari representasi bilangan biner

$$(1101)_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = (13)_{10}$$

Tabel 2.3 Tabel Biner

	2^3	2^2	2^1	2^0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10/A	1	0	1	0
11/B	1	0	1	1
12/C	1	1	0	0
13/D	1	1	0	1
14/E	1	1	1	0
15/F	1	1	1	1

3. Bilangan Oktal Hexadesimal

Bilangan oktal adalah bilangan dengan radix 8 yaitu digit bernilai dari $0, 1, 2, \ldots, 7$, sedangkan hexadesimal adalah bilangan dengan radix 16 yaitu digit bernilai dari $0, 1, 2, \ldots, 9$ dan $A, B, \ldots, F$. Representasi posisional dapat digunakan untuk sebarang radix (r)

$$K = K_{n-1}K_{n-2\ldots}K_1K_0 \quad (2.6)$$

mempunyai nilai integer

$$\sum_{i=0}^{n-1} K_i r^i \quad (2.7)$$

Dimana K adalah bilangan oktal dan n adalah banyaknya bilangan.

4. Representasi bilangan dan nilai ekivalennya

Dibawah ini adalah tabel dari representasi bilangan, yaitu dari bilangan desimal, biner, oktal, dan hexadesimal.

Tabel 2.4 Representasi Bilangan

Desimal	Biner	Oktal	Hexa
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

5. Konversi bilangan biner ke desimal dan sebaliknya

Dari persamaan (2.5) dilakukan konversi bilangan dari biner ke desimal, Misal diberikan sebuah bilangan dengan radix 2 sebagai berikut $(11101011)_2$, bilangan tersebut dapat diubah kedalam bentuk desimal, yaitu

$$= 1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

$$= 2^7 + 2^6 + 2^5 + 2^3 + 2^1 + 2^0$$

$$= 128 + 64 + 32 + 8 + 2 + 1$$

$$= (235)_{10} = 235$$

Untuk konversi dari desimal ke biner dapat dilakukan dengan langkah-langkah sebagai berikut :

1) Bagi bilangan desimal D dengan 2, memberikan hasil bagi (*quotient*) dan sisa. Sisa nilainya 0 atau 1. Sisanya akan menjadi LSB.

2) Bagi *quotient* dengan 2, memberikan hasil bagi dan sisa. Ulangi pembagian *quotient* sampai *quotient* = 0.
3) Untuk setiap pembagian, sisa akan mempresentasikan satu bit bilangan binernya.

Misalkan diberikan bilangan desimal 235, ubahlah kedalam bentuk biner !

$2 \times 10^2 + 3 \times 10^1 + 5 \times 10^0$

pembagi	Hasil bagi	sisa
2	235	
2	117	1
2	58	1
2	29	0
2	14	1
2	7	0
2	2	1
2	1	1
2	0	1

Sehingga didapatkan bilangan biner 11101011.

Bilangan desimal juga dapat direpresentasikan kedalam bentuk huruf atau dalam kriptografi disebut plainteks, yaitu pada Tabel 2.5 sebagai berikut:

Tabel 2.5 Tabel Convert Decimal to character

Dec	Character	Description
0	NUL	null
1	SOH	start of header
2	STX	start of text
3	ETX	end of text
4	EOT	end of transmission
5	ENQ	enquiry
6	ACK	acknowledge
7	BEL	bell
8	BS	backspace
9	HT	horizontal tab
10	LF	line feed
11	VT	vertical tab
12	FF	form feed
13	CR	enter / carriage return
14	SO	shift out
15	SI	shift in
16	DLE	data link escape
17	DC1	device control 1
18	DC2	device control 2
19	DC3	device control 3
20	DC4	device control 4
21	NAK	negative acknowledge
22	SYN	synchronize
23	ETB	end of trans. block
24	CAN	cancel
25	EM	end of medium
26	SUB	substitute
27	ESC	escape
28	FS	file separator

Dec	Character	Description
29	GS	group separator
30	RS	record separator
31	US	unit separator
127	DEL	delete

6. Konversi bilangan biner-oktal-hexadesimal

Konversi dari bilangan biner ke oktal, dimana 1 digit oktal merupakan grup 3 digit biner.

Tabel 2.6 Konversi bilangan biner-oktal dan sebaliknya

Konversi biner ke oktal	Biner	001	000	110	100
	Oktal	1	0	6	4
Konversi oktal ke biner	Oktal	2	3	6	7
	Biner	010	011	110	111

Konversi biner ke hexadesimal, dimana 1 digit hexa merupakan grup 4 digit biner.

Tabel 2.7 Konversi bilangan biner-hexa dan sebaliknya

Konversi biner ke hex	biner	1111	0000	0110	0100
	hex	F	0	6	4
Konversi hex ke biner	hex	2	A	C	7
	biner	0010	1010	1100	0111

7. Kode ASCII (*American Standard Code for Information Interchange*)

Kode ASCII merupakan kode yang sering digunakan untuk merepresentasikan informasi komputasi yaitu merepresentasikan nilai *alphanumeric* (huruf, bilangan, dan simbol menjadi nilai-nilai biner). Nilai-nilai akan dibaca dan diproses oleh peralatan digital misal :

komputer, *microprocessor* dalam bentuk biner. Kode ASCII terdiri dari 7 bit biner yang terdiri dari 3 bit MSB dan 4 bit LSB , dengan kombinasi kode 2^7 yaitu 128 kombinasi kode yang terdiri dari digit bilangan 0-9, karakter a-z dan A-Z, tanda baca dan tanda kontrol. Didekripsikan pada Tabel 2.8.

Tabel 2.8 Tabel ASCII

MSB / LSB	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	‘	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	OR	GS	-	=	M	]	m	}
1110	SOH	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

Contoh merubah huruf kedalam bentuk biner dengan menggunakan tabel ASCII

UINSA = 55 49 4E 53 41

G. Teori Bilangan

Teori bilangan *(Number Theory)* adalah teori yang mendasar dalam memahami kriptografi, khususnya sistem kriptografi kunci-publik. Bilangan yang dimaksudkan di sini hanya bilangan bulat *(integer).* Bilangan bulat adalah bilangan yang tidak mempunyai pecahan desimal,

misalnya $8{,}21{,}8765, -34, 0$. Berlawanan dengan bilangan bulat adalah bilangan riil yang mempunyai titik desimal, seperti $8.0,\ 34.25,\ 0.02$.

1. Sifat pembagian pada bilangan bulat

Misalkan $a\ dan\ b$ adalah dua buah bilangan bulat dengan syarat $a \neq 0$. Dinyatakan bahwa a habis membagi b $(a\ divedes\ b)$ jika terdapat bilangan bulat c sedemikian sehinga $b = ac$.

Notasi: $a|b\ jika\ b = ac, c\ \in\ Z\ dan\ a\ \neq\ 0$. ($Z$ = Himpunan bilangan bulat).

Contoh

$4|12$ karena $12 \div 4 = 3$ (bilangan bulat) atau $12 = 4 \times 3$. Begitu juga $134|0$ karna $0 + 134 = 0$ (bukan bilangan bulat). Tetapi, 4 tidak habis membagi 14 karna $14 \div 4 = 3.5$ (bukan bilangan bulat).

2. Aritmatika modulo *(Modular Arithmetic)*

Teorema *Euclidean* : Misalkan *m* dan *n* adalah dua bilangan bulat dengan syarat $n\ >\ 0$. Jika *m* dibagi dengan *n* maka terdapat dua buah bilangan bulat unik *q (quotient)* dan *r* (*remainder)* (Brualdi, 2009).

$M = nq + r$ dengan $0\ \leq\ r\ <\ n$ (8)

Keterangan

M adalah bilangan bulat.

n adalah hasil bagi.

q adalah bilangan yang membagi.

r adalah sisa (*remainder).*

Misalkan a adalah bilangan bulat dan m adalah bilangan bulat yang lebih dari 0. Operasi *a mod m* (dibaca "$a\ modulo\ m$") memberikan sisa jika a dibagi dengan m. Bilangan m disebut modulus atau modulo, dan hasil aritmatika modulo m terletak di dalam himpunan $\{0,1,2,\ldots,m-1\}$.

Notasi: $a\ mod\ m = r$ sedemikian sehingga $a = mq + r$, dengan

$0 \leq r < m$.

Keterangan :

a adalah bilangan bulat

m adalah hasil bagi

r adalah sisa

Misal diberikan $m = 80$, $n = 12$ dan dipenuhi syarat $m \geq n$

$80 = 6.12 + 8$ dimana 6 adalah hasil bagi dari $80 : 12$, dan 12 adalah pembagi dari 80, sedangkan 8 adalah sisa (r).

$12 = 1.8 + 4$ dimana 1 adalah hasil bagi dari $12 : 8$, dan 8 adalah pembagi dari 12, sedangkan 4 adalah sisa (r).

sehingga sisa pembagian terakhir sebelum 0 adalah 4, maka PBB (80,12) adalah 4.

Aritmatika modulo dapat digunakan untuk kriptografi karena dua alasan:

1. Nilai-nilai aritmatika modulo berada dalam himpunan berhingga (0 sampai modulo $(m-1)$), maka kita tidak perlu khawatir hasil perhitungan berada diluar himpunan.
2. Bekerja dengan bilangan bulat, maka kita tidak khawatir kehilangan informasi akibat pembulatan *(round off)* pada operasi bilangan riil.

H. Algoritma Tanam Padi dan Bajak Sawah

Pengembangan dari metode *Chipper block* sangat banyak sekali. Salah satu yang digunakan dalam penelitian ini yaitu algoritma tanam padi dan bajak sawah yang terinspirasi dari kearifan lokal sehingga diberi nama tanam padi dan bajak sawah. Algoritma tanam padi digunakan untuk mengubah plainteks kedalam bentuk *chipperteks*, sedangkan bajak sawah digunakan membangkitkan kunci.

1. Tanam Padi

Algoritma tanam padi digunakan untuk sebuah *plainteks*. Proses tanam padi biasanya dilakukan dengan menyesuaikan dengan panjang petakan sawah. Penanaman dilakukan secara horisontal yang berkesinambungan. Rancangan ini menggunakan cara yang sama dengan menempatkan bit seperti proses penanaman padi, dengan menggunakan kotak berukuran 8×8 yang secara keseluruhan terdapat 64

kotak. Setiap kotak ditempatkan satu bit, sehingga untuk melakukan satu kali proses dibutuhkan 8 karakter. Misal *plainteks*

$$Y = \{Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, Y_7, Y_8\}$$

$$Y_1 = \{C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8\}$$

$$Y_2 = \{C_9, C_{10}, C_{11}, C_{12}, C_{13}, C_{14}, C_{15}, C_{16}\}$$

$$\vdots$$

$$Y_n = \{C_{8-7}, C_{8-6}, C_{8-5}, \dots, C_8\} \quad (9)$$

Diubah dalam bentuk bit yang akan menghasilkan 64 bit, dimana bit-bit tersebut diubah kedalam biner berdasarkan karakternya yaitu :

$$Kar\ 1 \rightarrow \{C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8\}$$

$$Kar\ 2 \rightarrow \{C_9, C_{10}, C_{11}, C_{12}, C_{13}, C_{14}, C_{15}, C_{16}\}$$

$$Kar\ 3 \rightarrow \{C_{17}, C_{18}, C_{19}, C_{20}, C_{21}, C_{22}, C_{23}, C_{24}\}$$

$$Kar\ 4 \rightarrow \{C_{25}, C_{26}, C_{27}, C_{28}, C_{29}, C_{30}, C_{31}, C_{32}\} \quad (10)$$

$$Kar\ 5 \rightarrow \{C_{33}, C_{34}, C_{35}, C_{36}, C_{37}, C_{38}, C_{39}, C_{40}\}$$

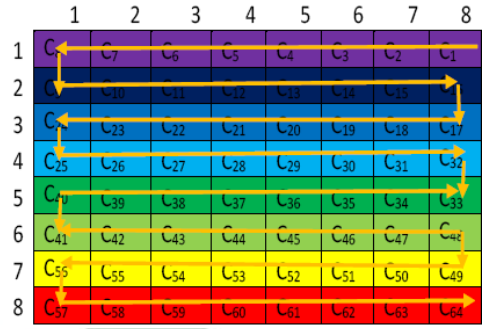
$$Kar\ 6 \rightarrow \{C_{41}, C_{42}, C_{43}, C_{44}, C_{45}, C_{46}, C_{47}, C_{48}\}$$

$$Kar\ 7 \rightarrow \{C_{49}, C_{50}, C_{51}, C_{52}, C_{53}, C_{54}, C_{55}, C_{56}\}$$

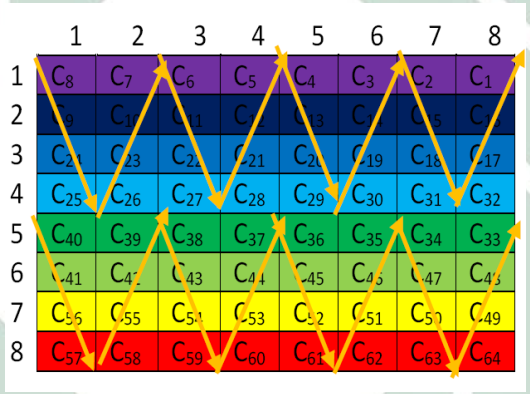
$$Kar\ 8 \rightarrow \{C_{57}, C_{58}, C_{59}, C_{60}, C_{61}, C_{62}, C_{63}, C_{64}\}$$

Selanjutnya bit-bit dari plainteksnya yaitu pada Persamaan 10 akan dimasukkan kedalam matriks dengan menggunakan pola massukan tanam padi. Seperti pada Gambar 2.3. dan selanjutnya akan di ambil bit baru dengan menggunakan pola pengambilan tanam padi, seperti pada Gambar 2.4. Didapatkan bit baru yaitu :

$$P_8 = C_{40,}C_{41}, C_{56}, \dots, C_{32}, C_{17}, C_{16}, C_1$$



Gambar 2.3. Pemasukan bit plainteks menggunakan pola tanam padi



Gambar 2. 4. Pengambilan bit plainteks menggunakan pola tanam padi

2. Bajak Sawah

Algoritma bajak sawah digunakan untuk membangkitkan sebuah kunci. Pola dalam algoritma bajak sawah adalah spiral. Misal sebuah kunci

$$X = \{X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8\}$$

$$X_1 = \{C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8\}$$

$$X_2 = \{C_9, C_{10}, C_{11}, C_{12}, C_{13}, C_{14}, C_{15}, C_{16}\}$$

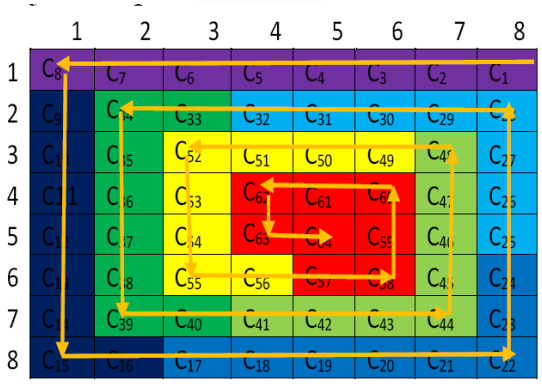
$$\vdots$$

$$X_n = \{C_{8-7}, C_{8-6}, C_{8-5}, \dots, C_8, \} \quad (11)$$

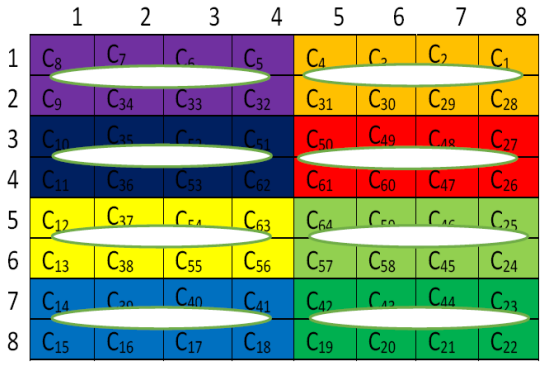
Diubah kedalam bentuk bit yang akan menghasilkan 64 bit, dimana bit-bit tersebut berdasarkan karakternya yaitu :

$$kun\ 1 \rightarrow \{C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8\}$$
$$kun\ 2 \rightarrow \{C_9, C_{10}, C_{11}, C_{12}, C_{13}, C_{14}, C_{15}, C_{16}\}$$
$$kun\ 3 \rightarrow \{C_{17}, C_{18}, C_{19}, C_{20}, C_{21}, C_{22}, C_{23}, C_{24}\}$$
$$kun\ 4 \rightarrow \{C_{25}, C_{26}, C_{27}, C_{28}, C_{29}, C_{30}, C_{31}, C_{32}\} \quad (12)$$
$$kun\ 5 \rightarrow \{C_{33}, C_{34}, C_{35}, C_{36}, C_{37}, C_{38}, C_{39}, C_{40}\}$$
$$kun\ 6 \rightarrow \{C_{41}, C_{42}, C_{43}, C_{44}, C_{45}, C_{46}, C_{47}, C_{48}\}$$
$$kun\ 7 \rightarrow \{C_{49}, C_{50}, C_{51}, C_{52}, C_{53}, C_{54}, C_{55}, C_{56}\}$$
$$kun\ 8 \rightarrow \{C_{57}, C_{58}, C_{59}, C_{60}, C_{61}, C_{62}, C_{63}, C_{64}\}$$

Proses masukan bit-bit kunci dilakukan dengan mengikuti pola bajak sawah yang ditunjukan pada Gambar 2.5. Selanjutnya akan diambil bit bit baru dengan menggunakan pola bajak sawah yang ditunjukan pada Gambar 2.6.



Gambar 2.5. Pemasukan bit kunci menggunakan pola bajak sawah



Sumber : SETISI, 2015

Gambar 2.6. Pengambilan bit kunci menggunakan pola bajak sawah

I. Kriptografi dalam Al-Qur'an

Kriptografi merupakan ilmu yang mempelajari mengenai pengamanan suatu pesan. Tujuan kriptografi adalah menjaga kerahasiaan pesan. Selain itu, dapat digunakan untuk mencegah adanya suatu penghianatan. Penghianatan tersebut dapat dilakukan oleh setiap orang dengan berbagai macam sebab. Padahal hal tersebut sangat dibenci oleh Allah. Sesuai dengan firman Allah dalam Qur'an Surah Al-Anfal ayat 58.

وَاِمَّا تَخَا فَنَّ مِنْ قَوْ مٍ خِيَا نَةً فَا نْبِذْ اِ لَيْهِمْ عَلٰى سَوَاۤءٍ ۗاِنَّ اللّٰهَ لَا يُحِبُّ ا لْخَا ۤئِنِيْنَ30

Artinya:

"*Dan jika kamu hawatir akan (terjadinya) penghianatan dari suatu golongan, maka kembalikanlah perjanjian itu kepada mereka dengan cara yang jujur. Sesunguhnya allah tidak menyukai orang-orang yang berkhianat (Al-anfal:58)*".

Pada ayat tersebut telah dijelaskan bahwa jika telah terlihat tanda-tanda sebuah pengkhianatan dari seseorang, maka haruslah dapat dikembalikan sebuha perjanjian itu dengan jujur. Sehingga tidak ada lagi

perjanjian yang akan mengikat satu sama lain dan mengurangi terjadinya penkhianatan. Allah tidak menyukai orang-orang yang berkhianat, dan juga tidak membolehkan pengkhianatan secara mutlak.selain itu, Allah juga memberi peringatan pula kepada orang-orang yang berkhianat dengan azab yang akan menimpa sebagai akibat dari pengkhianatan.

BAB III

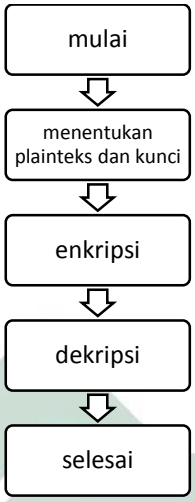
METODE PENELITIAN

A. Jenis Penelitian

Penelitian implementasi kriptografi pada teks menggunakan algoritma tanam padi dan bajak sawah termasuk jenis penelitian deskriptif kuantitatif. Jika dilihat dari aspek fungsinya merupakan penelitian terapan, dengan membuat kunci pada file dokumen yang selanjutnya akan dilakukan proses enkripsi dan deskripsi dengan algoritma tanam padi dan bajak sawah.

B. Proses Pengolahan Data

Hal pertama yang dilakukan dalam penelitian ini adalah menentukan teks yang akan disandikan selanjutnya membentuk kunci awal yang bersesuaian dengan panjang teksnya, selanjutnya akan dienkripsi dengan menggunakan metode tanam padi dan kunci akan dibangkitkan dengan metode bajak sawah sehingga didapatkan sebuah *chiperteks*. Setelah didapatkan *chiperteks* akan dilakukan proses deskripsi dengan kunci awal yang telah dibangkitkan untuk mengembalikan *chiperteks* dalam bentuk semula. Untuk lebih jelasnya alur pengolahan data akan disajikan dalam bentuk diagram alir pada Gambar 3.1.



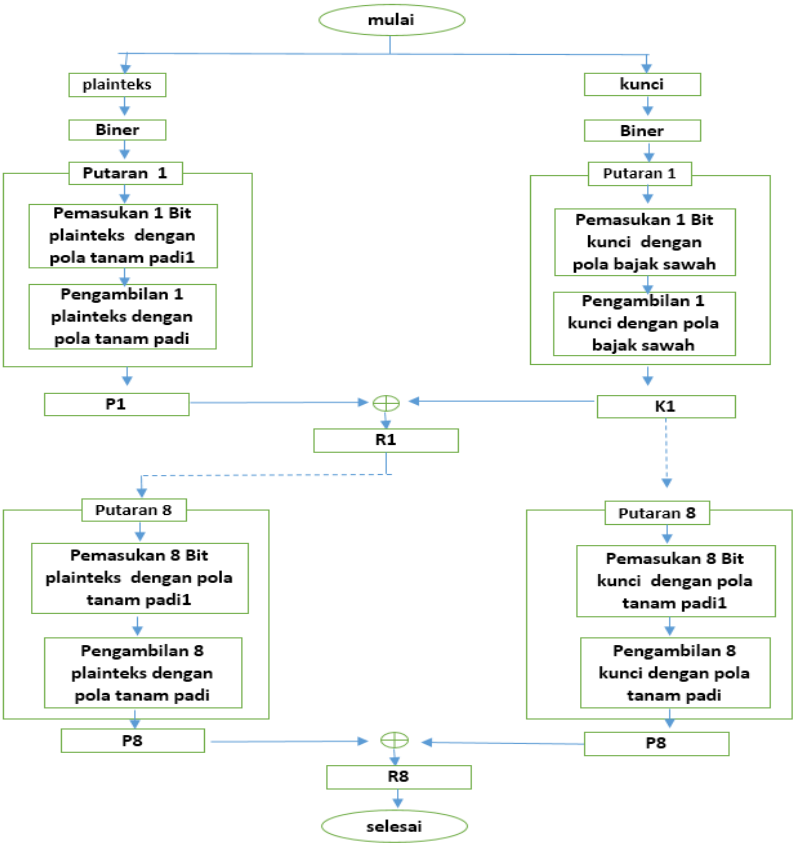
Gambar 3.1. Alur pengolahan data

Proses enkripsi dalam penelitian ini ada beberapa langkah, yaitu sebagai berikut:

- Langkah 1 : Merubah plainteks kedalam bentuk hexa .
- Langkah 2 : Merubah bentuk plainteks dari hexa kedalam bentuk biner.
- Langkah 3 : Memasukan bit-bit plainteks dengan menggunakan pola tanam padi.
- Langkah 4 : Pengambilan bi-bit plainteks dengan menggunakkan pola tanam padi, sehingga akan didapatkan P1.
- Langkah 5 : Merubah kunci kedalam bentuk hexa, seperti langkah 1 dan 2.
- Langkah 6 : Masukan bit-bit kunci dengan menggunakan pola bajak sawah.
- Langkah 7 : Pengambilan nit-bit kunci dengan menggunakan pola bajak sawah, sehingga diperoleh K1.

- Langkah 8 : XORkan antara P1 dan K1. Hasil XOR dari P1 dan K1 disebut R1.
- Langkah 9 : Dari R1 proses menggunakan langkah 2 dan 3. Sehingga akan didapatkan P2.
- Langkah 10 : Dari K1 proses menggunakan langkah 6 dan 7, sehingga akan didapatkan K2.

Dari proses ke 9 dan 10 menunjukan bahwa yang menjadi input untuk proses selanjutnya adalah hasil sebelumnya. Sampai putaran ke 8 yaitu R8 yang akan menjadi chiperteks. Untuk lebih lengkapnya proses enkripsi disajikan dalam diagram alir pada Gambar 3.2.

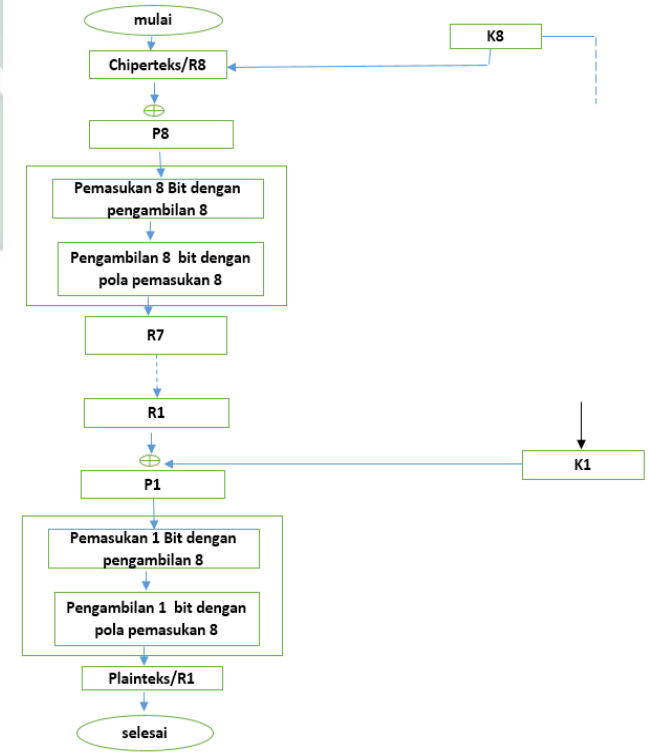


Gambar 3.2. Alur Enkripsi

Setelah proses enkripsi, didapatkan *chiperteks*, dari *chiperteks* tersebut akan di kembalikan lagi kedalam bentuk semula yaitu *plainteks*. Adapun langkah-langkah dari dekripsi adalah sebagai berikut :

- Putaran yang ke delapan yaitu R8 di-XOR dengan K8 menjadi P8.
- Hasil dari P8 dimasukkan mengikuti pola pengambilan kedelapan.
- Setelah itu pengambilan *bit* sesuai pola pemasukan ke delapan untuk mendapatkan R7.
- Proses ini diulang sampai pada putaran -1.
- Hasil dari putaran-1 akan dijadikan sebagai plainteks.

Untuk lebih jelasnya, proses dekripsi disajikan dalam bentuk diagram alir pada gambar 3.3



Gambar 3.3 pola dekripsi

BAB IV

PEMBAHASAN

A. Enkripsi

Pada pembahasan ini dilakukan penyandian dengan plainteks "mahmudah" dengan menggunakan kata kunci "muslimah". Untuk keperluan tersebut teks yang ada dirubah terlebih dahulu kedalam bentuk desimal dengan ketentuan sebagaimana dalam Tabel 2.7

Tabel 4.1 Hasil Konversi huruf plainteks

Plainteks	Desimal
m	109
a	97
h	104
m	109
u	117
d	100
a	97
h	104

Tabel 4.2 Hasil Konversi kunci dalam angka

Kunci	Desimal
m	109
u	117
s	115
l	108
i	105
m	109
a	97
h	104

Setelah itu, langkah selanjutnya adalah merubah bilangan desimal tersebut kedalam bentuk biner dengan menggunakan Persamaan 2.5.

$$V(B) = b_{n-1} \times 2^{n-1} + b_{n-2} \times 2^{n-2} + \cdots + b_1 \times 2^1 + b_0 \times 2^0 = \sum_{i=0}^{n-1} b_i \times 2^i$$

Sehingga didapatkan hasil sebagai berikut,

$m = 109 = 01101101$

Tabel 4.3 Biner m

pembagi	Hasil bagi	sisa
2	109	
2	54	1
2	27	0
2	13	1
2	6	1
2	3	0
2	1	1
2	0	1

$a = 97 = 01100001$

Tabel 4.4 Biner a

pembagi	Hasil bagi	sisa
2	97	
2	48	1
2	24	0
2	12	0
2	6	0
2	3	0
2	1	1
2	0	1

$h = 104 = 01101000$

Tabel 4.5 Biner h

pembagi	Hasil bagi	sisa
2	104	
2	52	0
2	26	0
2	13	0
2	6	1
2	3	0
2	1	1
2	0	1

$u = 117 = 01110101$

Tabel 4.6 Biner u

pembagi	Hasil bagi	sisa
2	117	
2	58	1
2	29	0
2	14	1
2	7	0
2	3	1
2	1	1
2	0	1

$d = 100 = 01100100$

Tabel 4.7 Biner d

pembagi	Hasil bagi	sisa
2	100	
2	50	0
2	25	0
2	12	1
2	6	0
2	3	0
2	1	1
2	0	1

$s = 115 = 01110011$

Tabel 4.8 Biner s

pembagi	Hasil bagi	sisa
2	115	
2	57	1
2	28	1
2	14	0
2	7	0
2	3	1
2	1	1
2	0	1

$l = 108 = 01101100$ $i = 105 = 01101001$

Tabel 4.9 Biner l

pembagi	Hasil bagi	sisa
2	108	
2	54	0
2	27	0
2	13	1
2	6	1
2	3	0
2	1	1
2	0	1

Tabel 4.10 Biner i

pembagi	**Hasil bagi**	**sisa**
2	**105**	
2	**52**	**1**
2	**26**	**0**
2	**13**	**0**
2	**6**	**1**
2	**3**	**0**
2	**1**	**1**
2	**0**	**1**

Setelah diubah kedalam bentuk biner selanjutkan *plainteks* dimasukan kedalam bit 8×8 dengan pola masukan tanam padi seperti yang terdapat dalam Gambar 2.3, kemudian pengambilan bit *plainteks* dilakukan dengan menggunakan pola pengambilan tanam padi seperti pada Gambar 2.4. Sedangkan pembangkitan kunci yaitu kunci yang telah diubah kedalam bentuk biner di masukan kedalam bit dengan pola masukan kunci seperti pada Gambar 2.5, kemudian pengambilan kunci dilakukan dengan pola pengambilan kunci seperti pada Gambar 2.6. selanjutnya hasil dari pengambilan *plainteks* dan kunci di XoRkan. Hasil dari XoR diproses diulang sampai dengan 8 putaran, yaitu sebagai berikut:

Putaran 1

Plainteks

Tabel 4.11 Pemasukan bit plainteks putaran 1

1	0	1	1	0	1	1	0
0	1	1	0	0	0	0	1
0	0	0	1	0	1	1	0
0	1	1	0	1	1	0	1
1	0	1	0	1	1	1	0
0	1	1	0	0	1	0	0
1	0	0	0	0	1	1	0
0	1	1	0	1	0	0	0

Tabel 4.12 Pengambilan plainteks putaran 1

1	0	1	1	0	1	1	0
0	1	1	0	0	0	0	1
0	0	0	1	0	1	1	0
0	1	1	0	1	1	0	1
1	0	1	0	1	1	1	0
0	1	1	0	0	1	0	0
1	0	0	0	0	1	1	0
0	1	1	0	1	0	0	0

Pada Tabel 4.11 adalah proses pemasukan bit plainteks sesuai dengan pola pada Gambar 2.3 yaitu dimulai dari bit paling atas dan kanan sampai dengan yang paling bawah. Sedangkan pada Tabel 4.12 adalah pola pengambilan bit-bit plainteks sesuai dengan Gambar 2.4, yaitu dimulai dari kolom yang keempat dari bawah dan dimulai dari kiri sesuai dengan pola sampai pada baris ke 8, selanjutnya adalah 4 baris yang atas dimulai dari kiri sampai dengan baris ke 8, sehingga didapatkan hasil yaitu :

p_1 = 10101010 11010000 10010111 10100000 10001010 11010101 00011101 10101010

Ketika dikerjakan menggunakan MATLAB 2013 dengan *source code* sebagai berikut :

```
      clear all;
clc;

%data='muslimah';
%nama=double(data);
d=dec2bin([109;97;104;109;117;100;97;104],8); %merubah biner
kedalam bentuk ASCII
% c=zeros(8,1);
% d=[c,a];
disp (d) %menampilkan hasil dari nilai ASCII
```

```
g1 = [d(1,8) d(1,7) d(1,6) d(1,5) d(1,4) d(1,3) d(1,2) d(1,1)];
g3 = [d(3,8) d(3,7) d(3,6) d(3,5) d(3,4) d(3,3) d(3,2) d(3,1)];
g5 = [d(5,8) d(5,7) d(5,6) d(5,5) d(5,4) d(5,3) d(5,2) d(5,1)];
g7 = [d(7,8) d(7,7) d(7,6) d(7,5) d(7,4) d(7,3) d(7,2) d(7,1)];
plainText = [g1; d(2,:); g3; d(4,:); g5; d(6,:); g7; d(8,:)] %pola
pemasukan bit-bit plainteks
pola1 = plainText(5:8,:) %pola pengambilan bit yang bawah
p2 = [pola1(4,2), pola1(3,2), pola1(2,2), pola1(1,2)];
p4 = [pola1(4,4), pola1(3,4), pola1(2,4), pola1(1,4)];
p6 = [pola1(4,6), pola1(3,6), pola1(2,6), pola1(1,6)];
p8 = [pola1(4,8), pola1(3,8), pola1(2,8), pola1(1,8)];
Pengambilan1 = [pola1(:,1)' p2 pola1(:,3)' p4 pola1(:,5)' p6
pola1(:,7)' p8]
pola2 = plainText(1:4,:) %pola pengambilan bit yang atas
p22 = [pola2(4,2), pola2(3,2), pola2(2,2), pola2(1,2)];
p24 = [pola2(4,4), pola2(3,4), pola2(2,4), pola2(1,4)];
p26 = [pola2(4,6), pola2(3,6), pola2(2,6), pola2(1,6)];
p28 = [pola2(4,8), pola2(3,8), pola2(2,8), pola2(1,8)];
Pengambilan2 = [pola2(:,1)' p22 pola2(:,3)' p24 pola2(:,5)' p26
pola2(:,7)' p28]
PP = [Pengambilan1 Pengambilan2] %penggabungan pengambilan bit-bit
plainteks
```

Gambar 4.1 Pseudo-code enkripsi plainteks

pseudo-code pada gambar 4.1 adalah inputan awal untuk enkripsi plaintekas yang selanjutnya dimasukan kedalam bit-bit sesuai dengan pola pada Gambar 2.3, selanjutnya pengambilan bit-bit plaintekas sesuai dengan pola pada Gambar 2.4. dihasilkan output sebagai berikut :

```
PP =

1010101011010000100101111010000010001010110101010001110110101010
```

Gambar 4.2 Hasil output enkripsi plainteks

Hasil output dari program menunjukan hasil yang sama dengan hasil manual yang telah dihitung.

Kunci

Tabel 4.13 pemasukan bitkunci putaran 1

1	0	1	1	0	1	1	0
0	1	0	0	0	1	1	0
1	1	0	1	1	0	1	1
1	0	0	0	1	0	0	1
1	1	0	0	0	1	1	0
0	0	0	1	0	1	1	1
1	0	1	0	1	1	0	1
0	1	0	1	1	1	0	0

Tabel 4.14 pengambilan kunci putaran 1

1	0	1	1	0	1	1	0
0	1	0	0	0	1	1	0
1	1	0	1	1	0	1	1
1	0	0	0	1	0	0	1
1	1	0	0	0	1	1	0
0	0	0	1	0	1	1	1
1	0	1	0	1	1	0	1
0	1	0	1	1	1	0	0

K_1 = 10110010 11010001 11001000 10101010 11010011 01101110 10111001 01100110

Pada Tabel 4.13 adalah pemasukan bit kunci sesuai dengan Gambar 2.5 yaitu dimulai dari baris pertama dimulai dari kanan dan diputar sesuai dengan pola, selanjutnya adalah pengambilan bit-bit kunci sesuai dengan gambar 2.6 yaitu pada kolom pertama dan baris petama sampai dengan baris keempat lalu kebawah yaitu ke kolom kedua baris keempat sampai dengan kolom kedua baris pertama. Sampai seterusnya pada kolom kedua baris pertama.

Untuk pembangkitan kunci dalam proses enkripsi ketika dilakukan dengan MATLAB 2013 dengan psedou code sebagai berikut :

```
k=dec2bin([109;117;115;108;105;109;97;104],8); % perubahan kunci
ASCII
knc =[k(1,8) k(1,7) k(1,6) k(1,5) k(1,4) k(1,3) k(1,2) k(1,1); %
pola pemasukan bit-bit kunci
    k(2,1) k(5,2) k(5,1) k(4,8) k(4,7) k(4,6) k(4,5) k(4,4)
    k(2,2) k(5,3) k(7,4) k(7,3) k(7,2) k(7,1) k(6,8) k(4,3)
    k(2,3) k(5,4) k(7,5) k(8,6) k(8,5) k(8,4) k(6,7) k(4,2)
    k(2,4) k(5,5) k(7,6) k(8,7) k(8,8) k(8,3) k(6,6) k(4,1)
    k(2,5) k(5,6) k(7,7) k(7,8) k(8,1) k(8,2) k(6,5) k(3,8)
    k(2,6) k(5,7) k(5,8) k(6,1) k(6,2) k(6,3) k(6,4) k(3,7)
    k(2,7) k(2,8) k(3,1) k(3,2) k(3,3) k(3,4) k(3,5) k(3,6)]
kc1 = knc(:,1:4); %pengambilan bit-bit kunci 1
k2 = [kc1(2,4) kc1(2,3) kc1(2,2) kc1(2,1)];
k4 = [kc1(4,4) kc1(4,3) kc1(4,2) kc1(4,1)];
k6 = [kc1(6,4) kc1(6,3) kc1(6,2) kc1(6,1)];
k8 = [kc1(8,4) kc1(8,3) kc1(8,2) kc1(8,1)];
kunci1 = [kc1(1,:) k2 kc1(3,:) k4 kc1(5,:) k6 kc1(7,:) k8]
kc2 = knc(:,5:8); %pengambilan bit-bit kunci 2
k22 = [kc2(2,4) kc2(2,3) kc2(2,2) kc2(2,1)];
k24 = [kc2(4,4) kc2(4,3) kc2(4,2) kc2(4,1)];
```

```
k26 = [kc2(6,4) kc2(6,3) kc2(6,2) kc2(6,1)];
k28 = [kc2(8,4) kc2(8,3) kc2(8,2) kc2(8,1)];
kunci2 = [kc2(7,:) k28 kc2(5,:) k26 kc2(3,:) k24 kc2(1,:) k22]
kunci = [kunci1 kunci2]
```

Gambar 4.3 Pseudo code pembangkitan kunci

Pada Gambar 4.3 adalah Pseudo code untuk pembangkitan kunci pada putaran pertama mulai dari inputan kunci sampai dengan pengambilan bit-bit kunci, sehingga dihasilkan output sebagai berikut :

```
kunci =

1011001011010001110010001010101011010011011011101011100101100110
```

Gambar 4.4 Output pembangkitan kunci

Gambar 4.4 Menunjukan bahwa hasil dari program sama dengan hasil manual yang telah dihitung pada putaran pertama.

Setelah dilakukan proses enkripsi dan pembangkitan kunci pada putaran pertama didapatkan *plainteks* sementara yaitu p_1 dan kunci yang telah dibangkitkan yaitu K_1, langkah selanjutnya adalah p_1 dan K_1 di XoRkan, yaitu

p_1 = 10101010 11010000 10010111 10100000 10001010 11010101 00011101 10101010

K_1 = 10110010 11010001 11001000 10101010 11010011 01101110 10111001 01100110

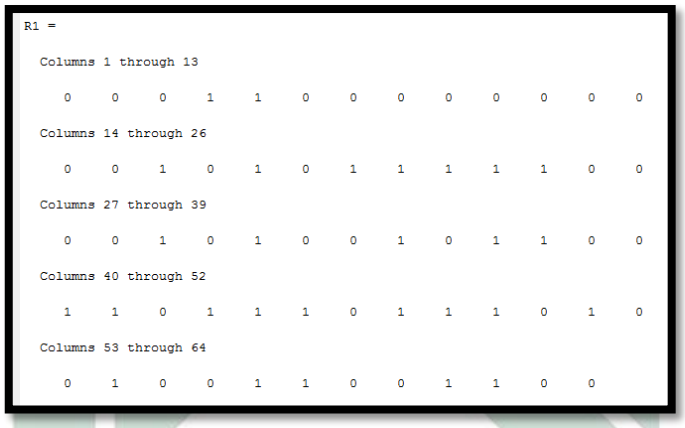
----------XoR----------

Sehingga menghasilkan *chipperteks* sementara yaitu R_1 yang akan diproses lagi dalam putaran 2. Hasil biner dari R_1 adalah

R_1 = 00011000 00000001 01011111 00001010 01011001 10111011 10100100 11001100

```
PT = double(PP); K=double(kunci);
        R1 = bitxor(PT,K
```

Gambar 4.5 Pseudo code xor plainteks dan kunci



Gambar 4.6 Hasil xor dari plainteks dan kunci

Gambar 4.6 adalah hasil dari xor antara plainteks dengan kunci yang telah di bangkitkan dan menunjukan hasil yang sama dengan perhitungan manualnya. Selanjutnya hasil dari plainteks dan kunci yang telah dibangkitkan disimpan kedalam bentuk save .mat untuk nantinya dijadikan inputan pada putaran kedua.

```
save R1.mat R1 %buat ngeload data selanjutnya
  save K.mat K
```

Gambar 4.7 Pseudo code save.mat

- **Putaran 2**

Pada putaran 2 yang akan dimasukkan kedalam bit tanam padi adalah *chipperteks* sementara yang diperoleh dari proses enkripsi pada putaran 1 yaitu R_1. Pembangkitan kunci pada putaran 2 ini adalah hasil dari pembangkitan kunci pada putaran pertama yaitu K_1, yang selanjutnya akan di massukkan kedalam bit-bit pemasukan kunci dan juga bit-bit pengambilan kunci pada putaran 2.

Plainteks

Tabel 4.15 Pengambilan plainteks putaran 2

0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	1
1	1	1	1	1	0	1	0
0	0	0	0	1	0	1	0
1	0	0	1	1	0	1	0
1	0	1	1	1	0	1	1
0	0	1	0	0	1	0	1
1	1	0	0	1	1	0	0

Tabel 4.16 Pengambilan plainteks putaran 2

0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	1
1	1	1	1	1	0	1	0
0	0	0	0	1	0	1	0
1	0	0	1	1	0	1	0
1	0	1	1	1	0	1	1
0	0	1	0	0	1	0	1
1	1	0	0	1	1	0	0

Pada Tabel 4.15 adalah proses pemasukan bit plainteks pada putaran 2 sesuai dengan pola pada Gambar 2.3 yaitu dimulai dari bit paling atas dan kanan sampai dengan yang paling bawah. Sedangkan pada Tabel 4.16 adalah pola pengambilan bit-bit plainteks sesuai dengan Gambar 2.4, yaitu dimulai dari kolom yang keempat dari bawah dan dimulai dari kiri sesuai dengan pola sampai pada baris ke 8, selanjutnya adalah 4 baris yang atas dimulai dari kiri sampai dengan baris ke 8.

$P_2 = 11011000\ 01100011\ 11011100\ 11000110\ 00100100\ 00100101\ 10110000\ 00110010$

```
load R1; %merubah biner kedalam bentuk ASCII
d=[R1(1,1:8);R1(1,9:16);R1(1,17:24);R1(1,25:32);R1(1,33:40);R1(1,4
1:48);R1(1,49:56);R1(1,57:64)];
% c=zeros(8,1);
% d=[c,a];
%menampilkan hasil dari nilai ASCII
g1 = [d(1,8) d(1,7) d(1,6) d(1,5) d(1,4) d(1,3) d(1,2) d(1,1)];
g3 = [d(3,8) d(3,7) d(3,6) d(3,5) d(3,4) d(3,3) d(3,2) d(3,1)];
g5 = [d(5,8) d(5,7) d(5,6) d(5,5) d(5,4) d(5,3) d(5,2) d(5,1)];
g7 = [d(7,8) d(7,7) d(7,6) d(7,5) d(7,4) d(7,3) d(7,2) d(7,1)];
plainText = [g1; d(2,:); g3; d(4,:); g5; d(6,:); g7; d(8,:)];
%pola pemasukan bit-bit plainteks
pola1 = plainText(5:8,:); %pola pengambilan bit yang bawah
p2 = [pola1(4,2), pola1(3,2), pola1(2,2), pola1(1,2)];
p4 = [pola1(4,4), pola1(3,4), pola1(2,4), pola1(1,4)];
p6 = [pola1(4,6), pola1(3,6), pola1(2,6), pola1(1,6)];
p8 = [pola1(4,8), pola1(3,8), pola1(2,8), pola1(1,8)];
Pengambilan1 = [pola1(:,1)' p2 pola1(:,3)' p4 pola1(:,5)' p6
pola1(:,7)' p8];
pola2 = plainText(1:4,:); %pola pengambilan bit yang atas
p22 = [pola2(4,2), pola2(3,2), pola2(2,2), pola2(1,2)];
p24 = [pola2(4,4), pola2(3,4), pola2(2,4), pola2(1,4)];
p26 = [pola2(4,6), pola2(3,6), pola2(2,6), pola2(1,6)];
p28 = [pola2(4,8), pola2(3,8), pola2(2,8), pola2(1,8)];
```

```
Pengambilan2 = [pola2(:,1)' p22 pola2(:,3)' p24 pola2(:,5)' p26
pola2(:,7)' p28];
PP = [Pengambilan1 Pengambilan2]
```

Gambar 4.8 Pseudo code enkripsi pada putaran 2

Pada Gambar 4.8 menunjukan bahwa yang menjadi inputan pada putaran 2 adalah hasil enkripsi plainteks pada putaran satu yang selanjutnya di simpan kedalam bentuk save .mat, sehingga dapat di load pada putaran 2. Sehingga didapatkan hasil sebagai berikut:

1101100001100011110111001100011000100100001001011011000000110010

Hasil yang didapatkan dari program menunjukan sama dengan hasil perhitungan manual.

Kunci

Tabel 4.17 pemasukan kunci putaran 2

0	1	0	0	1	1	0	1
1	1	1	0	1	0	1	0
1	0	1	1	0	1	0	1
0	1	1	1	0	0	1	0
1	0	0	1	0	1	1	1
0	0	0	1	0	1	1	0
0	1	1	0	1	1	0	0
0	1	1	1	0	0	1	0

Tabel 4.18 pengambilan kunci putaran 2

0	1	0	0	1	1	0	1
1	1	1	0	1	0	1	0
1	0	1	1	0	1	0	1
0	1	1	1	0	0	1	0
1	0	0	1	0	1	1	1
0	0	0	1	0	1	1	0
0	1	1	0	1	1	0	0
0	1	1	1	0	0	1	0

Pada Tabel 4.17 adalah pemasukan bit kunci sesuai dengan Gambar 2.5 yaitu dimulai dari baris pertama dimulai dari kanan dan diputar sesuai dengan pola, selanjutnya adalah pengambilan bit-bit kunci sesuai dengan Gambar 2.6 yaitu pada kolom pertama dan baris petama sampai dengan baris keempat lalu kebawah yaitu ke kolom kedua baris keempat sampai dengan kolom kedua baris pertama. Sampai seterusnya pada kolom kedua baris pertama, sehingga didapatkan hasil sebagai berikut:

$K_2 = 01000111\ 10111110\ 10011000\ 01101110\ 11000100\ 01110110\ 01010100\ 11010101$

```
load K % perubahan kunci ASCII
k
=[K(1,1:8);K(1,9:16);K(1,17:24);K(1,25:32);K(1,33:40);K(1,41:48);K
(1,49:56);K(1,57:64)];
knc =[k(1,8) k(1,7) k(1,6) k(1,5) k(1,4) k(1,3) k(1,2) k(1,1); %
pola pemasukan bit-bit kunci
    k(2,1) k(5,2) k(5,1) k(4,8) k(4,7) k(4,6) k(4,5) k(4,4)
    k(2,2) k(5,3) k(7,4) k(7,3) k(7,2) k(7,1) k(6,8) k(4,3)
    k(2,3) k(5,4) k(7,5) k(8,6) k(8,5) k(8,4) k(6,7) k(4,2)
    k(2,4) k(5,5) k(7,6) k(8,7) k(8,8) k(8,3) k(6,6) k(4,1)
    k(2,5) k(5,6) k(7,7) k(7,8) k(8,1) k(8,2) k(6,5) k(3,8)
    k(2,6) k(5,7) k(5,8) k(6,1) k(6,2) k(6,3) k(6,4) k(3,7)
    k(2,7) k(2,8) k(3,1) k(3,2) k(3,3) k(3,4) k(3,5) k(3,6)];
kc1 = knc(:,1:4); %pengambilan bit-bit kunci 1
k2 = [kc1(2,4) kc1(2,3) kc1(2,2) kc1(2,1)];
k4 = [kc1(4,4) kc1(4,3) kc1(4,2) kc1(4,1)];
k6 = [kc1(6,4) kc1(6,3) kc1(6,2) kc1(6,1)];
k8 = [kc1(8,4) kc1(8,3) kc1(8,2) kc1(8,1)];
kunci1 = [kc1(1,:) k2 kc1(3,:) k4 kc1(5,:) k6 kc1(7,:) k8];
kc2 = knc(:,5:8); %pengambilan bit-bit kunci 2
k22 = [kc2(2,4) kc2(2,3) kc2(2,2) kc2(2,1)];
k24 = [kc2(4,4) kc2(4,3) kc2(4,2) kc2(4,1)];
k26 = [kc2(6,4) kc2(6,3) kc2(6,2) kc2(6,1)];
k28 = [kc2(8,4) kc2(8,3) kc2(8,2) kc2(8,1)];
kunci2 = [kc2(7,:) k28 kc2(5,:) k26 kc2(3,:) k24 kc2(1,:) k22];
kunci = [kunci1 kunci2]; %penggabungan bit-bit kunci
```

Gambar 4.9 Pseudo code pembangkitan kunci putaran 2

Pada Gambar 4.9 menunjukan codingan untuk pembangkitan knci pada putaran 2, dimana inputan untuk kunci adalah load kunci dari putaran 1.

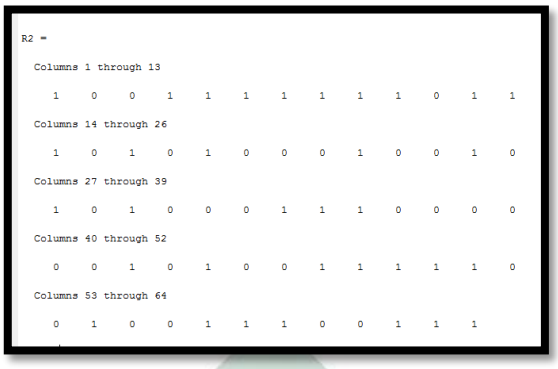
Pada putaran kedua didapatkan P_2 dan K_2 yang selanjutnya akan di XoRkan , yaitu

$P_2 = 11011000\ 01100011\ 11011100\ 11000110\ 00100100\ 00100101\ 10110000\ 00110010$

$K_2 = 01000111\ 10111110\ 10011000\ 01101110\ 11000100\ 01110110\ 01010100\ 11010101$

----------XoR----------

Sehingga didapatkan *chipperteks* sementara pada putaran 2 yaitu R2 yang selanjutnya kan di gunakan untuk putaran ketiga dalam proses enkripsi.



Gambar 4.10 hasil xor plainteks dan kunci putaran 2

$R_2 = 10011111\ 11011101\ 01000100\ 10101000\ 11100000\ 01010011\ 11100100\ 11100111$

Putaran 3

Plainteks

Tabel 4.19 Pemasukan Plainteks putaran 3

1	1	1	1	1	0	0	1
1	1	0	1	1	1	0	1
0	0	1	0	0	0	1	0
1	0	1	0	1	0	0	0
0	0	0	0	0	1	1	1
0	1	0	1	0	0	1	1
0	0	1	0	0	1	1	1
1	1	1	0	0	1	1	1

Tabel 4.20 Pemasukan Plainteks putaran 3

1	1	1	1	1	0	0	1
1	1	0	1	1	1	0	1
0	0	1	0	0	0	1	0
1	0	1	0	1	0	0	0
0	0	0	0	0	1	1	1
0	1	0	1	0	0	1	1
0	0	1	0	0	1	1	1
1	1	1	0	0	1	1	1

$P_3 = 00011010\ 00110010\ 00001101\ 11111111\ 11010011\ 10110011\ 11010010\ 00100011$

Kunci

Tabel 4.21 Pemasukan Kunci putaran 3

1	1	1	0	0	0	1	0
1	1	1	0	1	1	1	0
0	0	1	0	1	0	0	1
1	0	0	1	0	1	1	1
1	0	1	0	1	0	1	0
1	1	0	0	1	1	0	0
1	0	0	0	1	1	1	0
1	0	1	0	0	1	1	0

Tabel 4.22 Pengambilan Kunci putaran 3

1	1	1	0	0	0	1	0
1	1	1	0	1	1	1	0
0	0	1	0	1	0	0	1
1	0	0	1	0	1	1	1
1	0	1	0	1	0	1	0
1	1	0	0	1	1	0	0
1	0	0	0	1	1	1	0
1	0	1	0	0	1	1	0

$K_3 = 11100111\ 00101001\ 10100011\ 10000101\ 11100110\ 10100011\ 10011110\ 00100111$

Pada putaran kedua didapatkan P_3 dan K_3yang selanjutnya akan di XoRkan yaitu:

P_3 = 00011010 00110010 00001101 11111111 11010011 10110011 11010010 00100011

K_3 = 11100111 00101001 10100011 10000101 11100110 10100011 10011110 00100111

----------XoR----------

Sehingga didapatkan chipperteks sementara pada putaran 3 yaitu R_3 yang selanjutnya kan di gunakan untuk putaran selanjutnya dalam proses enkripsi.

R_3 = 11111101 00011011 10101110 01111010 00110101 00010000 01001100 00000100

Putaran 4

Plainteks

Tabel 4.23 Pemasukan Plainteks putaran 4

1	0	1	1	1	1	1	1
0	0	0	1	1	0	1	1
0	1	1	1	0	1	0	1
0	1	1	1	1	0	1	0
1	0	1	0	1	1	0	0
0	0	0	1	0	0	0	0
0	0	1	1	0	0	1	0
0	0	0	0	0	1	0	0

Tabel 4.24 Pengambilan Plainteks putaran 4

1	0	1	1	1	1	1	1
0	0	0	1	1	0	1	1
0	1	1	1	0	1	0	1
0	1	1	1	1	0	1	0
1	0	1	0	1	1	0	0
0	0	0	1	0	0	0	0
0	0	1	1	0	0	1	0
0	0	0	0	0	1	0	0

P_4 = 10000000 10100110 10001001 00100000 10001100 10111111 11010101 11010111

Kunci

Tabel 4.25 Pemasukan Kunci putaran 4

1	1	1	0	0	1	1	1
0	1	1	1	0	1	0	0
0	1	1	0	0	1	1	0
1	0	1	1	0	0	1	0
0	0	1	1	1	1	0	1
1	1	1	0	0	0	0	1
0	1	0	1	0	1	0	1
0	1	1	0	1	0	0	0

Tabel 4.26 Pemasukan Kunci putaran 4

1	1	1	0	0	1	1	1
0	1	1	1	0	1	0	0
0	1	1	0	0	1	1	0
1	0	1	1	0	0	1	0
0	0	1	1	1	1	0	1
1	1	1	0	0	0	0	1
0	1	0	1	0	1	0	1
0	1	1	0	1	0	0	0

k_4 = 11101110 01101101 00110111 01010110 01010001 11011000 01100100 01110010

Pada putaran kedua didapatkan P_4 dan k_4 yang selanjutnya akan di XoRkan , yaitu:

P_4 = 10000000 10100110 10001001 00100000 10001100 10111111 11010101 11010111

k_4 = 11101110 01101101 00110111 01010110 01010001 11011000 01100100 01110010

----------XoR----------

Sehingga didapatkan chipperteks sementara pada putaran 4 yaitu R_4 yang selanjutnya akan di gunakan untuk putaran selanjutnya dalam proses enkripsi.

R_4 = 01101110 11001011 10111110 01110110 11011101 01100111 10110001 10100101

Putaran 5

Plainteks

Tabel 4.27 Pemasukan Plainteks putaran 5

0	1	1	1	0	1	1	0
1	1	0	0	1	0	1	1
0	1	1	1	1	1	0	1
0	1	1	1	0	1	1	0
1	0	1	1	1	0	1	1
0	1	1	0	0	1	1	1
1	0	0	0	1	1	0	1
1	0	1	0	0	1	0	1

Tabel 4.28 Peng ambilan Plainteks putaran 5

0	1	1	1	0	1	1	0
1	1	0	0	1	0	1	1
0	1	1	1	1	1	0	1
0	1	1	1	0	1	1	0
1	0	1	1	1	0	1	1
0	1	1	0	0	1	1	1
1	0	0	0	1	1	0	1
1	0	1	0	0	1	0	1

P_5 = 10110010 11010001 10101110 11001111 01001111 10111101 01101101 11010110

Kunci

Tabel 4.29 Pemasukan Kunci putaran 5

0	1	1	1	0	1	1	1
0	1	0	0	1	1	0	1
1	0	0	1	1	0	0	0
1	1	0	0	0	1	0	1
0	0	1	1	0	1	0	0
1	0	0	0	0	1	1	1
1	0	1	1	1	0	1	1
0	1	0	0	1	1	0	1

Tabel 4.30 Pengambilan Kunci putaran 5

0	1	1	1	0	1	1	1
0	1	0	0	1	1	0	1
1	0	0	1	1	0	0	0
1	1	0	0	0	1	0	1
0	0	1	1	0	1	0	0
1	0	0	0	0	1	1	1
1	0	1	1	1	0	1	1
0	1	0	0	1	1	0	1

K_5 = 01110010 10010011 00110001 10110010 10111011 01001110 10001010 01111011

Pada putaran kedua didapatkan P_5 dan K_5 yang selanjutnya akan di XoRkan , yaitu:

P_5 = 10110010 11010001 10101110 11001111 01001111 10111101 01101101 11010110

K_5 = 01110010 10010011 00110001 10110010 10111011 01001110 10001010 01111011

----------XoR----------

Sehingga didapatkan chipperteks sementara pada putaran 5 yaitu R_5 yang selanjutnya akan di gunakan untuk putaran selanjutnya dalam proses enkripsi.

$R_5 = 11000000\ 01000010\ 10011111\ 01111101\ 11110100\ 11110011\ 11100111\ 1010110$

Putaran 6

Plainteks

Tabel 4.31 pemasukan Plainteks putaran 6

0	0	0	0	0	0	1	1
0	1	0	0	0	0	1	0
1	1	1	1	1	0	0	1
0	1	1	1	1	1	0	1
0	0	1	0	1	1	1	1
1	1	1	1	0	0	1	1
1	1	1	0	0	1	1	1
1	0	1	0	1	1	0	1

Tabel 4.32 pemasukan Plainteks putaran 6

0	0	0	0	0	0	1	1
0	1	0	0	0	0	1	0
1	1	1	1	1	0	0	1
0	1	1	1	1	1	0	1
0	0	1	0	1	1	1	1
1	1	1	1	0	0	1	1
1	1	1	0	0	1	1	1
1	0	1	0	1	1	0	1

$P_6 = 01110110\ 11110010\ 10011101\ 11101111\ 00101110\ 00111100\ 00111000\ 11001101$

Kunci

Tabel 4.33 Pengambilan kunci putaran 6

0	1	0	0	1	1	1	0
1	0	1	0	1	0	0	1
0	1	0	0	0	1	0	1
0	1	1	0	1	1	1	0
1	1	0	1	1	1	1	1
0	0	1	0	0	1	1	1
0	1	1	0	1	0	0	0
1	1	0	0	1	1	0	0

Tabel 4.34 Pengambilan kunci putaran 6

0	1	0	0	1	1	1	0
1	0	1	0	1	0	0	1
0	1	0	0	0	1	0	1
0	1	1	0	1	1	1	0
1	1	0	1	1	1	1	1
0	0	1	0	0	1	1	1
0	1	1	0	1	0	0	0
1	1	0	0	1	1	0	0

$K_6 = 01000101\ 01000110\ 11010100\ 01100011\ 10000011\ 11111110\ 01010111\ 11101001$

Pada putaran kedua didapatkanP_6 dan K_6 yang selanjutnya akan di XoRkan , yaitu:

$P_6 = 01110110\ 11110010\ 10011101\ 11101111\ 00101110\ 00111100\ 00111000\ 11001101$

$K_6 = 01000101\ 01000110\ 11010100\ 01100011\ 10000011\ 11111110\ 01010111\ 11101001$

----------XoR----------

Sehingga didapatkan chipperteks sementara pada putaran 6 yaitu R_6 yang selanjutnya akan di gunakan untuk putaran selanjutnya dalam proses enkripsi.

$R_6 = 00110011\ 10110100\ 01001001\ 10001100\ 10101101\ 11000010\ 01101111\ 00100100$

Putaran 7

Plainteks

Tabel 4.35 Pemasukan Plainteks putaran 7

1	1	0	0	1	1	0	0
1	0	1	1	0	1	0	0
1	0	0	1	0	0	1	0
1	0	0	0	1	1	0	0
1	0	1	1	0	1	0	1
1	1	0	0	0	0	1	0
1	1	1	1	0	1	1	0
0	0	1	0	0	1	0	0

Tabel 4.36 Pengambilan Plainteks putaran 7

1	1	0	0	1	1	0	0
1	0	1	1	0	1	0	0
1	0	0	1	0	0	1	0
1	0	0	0	1	1	0	0
1	0	1	1	0	1	0	1
1	1	0	0	0	0	1	0
1	1	1	1	0	1	1	0
0	0	1	0	0	1	0	0

$P_7 =$ 11100110 10110101 00001101 01100001 11110001 01000110 10011011 00100000

Kunci

Tabel 4.37 Pemasukan Kunci putaran 7

1	0	1	0	0	0	1	0
0	0	1	1	1	0	0	0
1	0	1	0	1	0	0	1
0	0	0	0	1	0	1	1
0	0	1	0	1	1	1	0
0	0	1	1	1	1	1	0
1	1	1	1	1	1	1	0
1	0	1	1	0	1	0	1

Tabel 4.38 Pemasukan Kunci putaran 7

1	0	1	0	0	0	1	0
0	0	1	1	1	0	0	0
1	0	1	0	1	0	0	1
0	0	0	0	1	0	1	1
0	0	1	0	1	1	1	0
0	0	1	1	1	1	1	0
1	1	1	1	1	1	1	0
1	0	1	1	0	1	0	1

$K_7 = 10101100\ 10100000\ 00101100\ 11111101\ 11101010\ 11100111\ 10011101\ 00100001$

Pada putaran kedua didapatkan P_7 dan K_7 yang selanjutnya akan di XoRkan , yaitu:

$P_7 =$ 11100110 10110101 00001101 01100001 11110001 01000110 10011011 00100000

$K_7 = 10101100\ 10100000\ 00101100\ 11111101\ 11101010\ 11100111\ 10011101\ 00100001$

----------XoR----------

Sehingga didapatkan chipperteks sementara pada putaran 7 yaitu R_7 yang selanjutnya akan di gunakan untuk putaran selanjutnya dalam proses enkripsi.

$R_7 = 01001010\ 00010101\ 00100001\ 10011100\ 00011011\ 10100001\ 00000110\ 00000001$

Putaran 8

Plainteks

Tabel 4.39 Pemasukan Plainteks putaran 8

0	1	0	1	0	0	1	0
0	0	0	1	0	1	0	1
1	0	0	0	0	1	0	0
1	0	0	1	1	1	0	0
1	1	0	1	1	0	0	0
1	0	1	0	0	0	0	1
0	1	1	0	0	0	0	0
0	0	0	0	0	0	0	1

Tabel 4.40 Pengambilan Plainteks putaran 8

0	1	0	1	0	0	1	0
0	0	0	1	0	1	0	1
1	0	0	0	0	1	0	0
1	0	0	1	1	1	0	0
1	1	0	1	1	0	0	0
1	0	1	0	0	0	0	1
0	1	1	0	0	0	0	0
0	0	0	0	0	0	0	1

$P_8 = 11000101\ 01100001\ 10000000\ 00001010\ 00110001\ 00001011\ 00011110\ 10000010$

Kunci

Tabel 4.41 Pemasukan Kunci putaran 8

0	0	1	1	0	1	0	1
1	1	1	1	0	1	1	1
0	1	1	0	0	1	1	1
1	0	1	0	0	0	1	1
0	1	1	0	1	1	1	1
0	0	0	1	0	0	0	0
0	1	0	1	1	1	0	0
0	0	0	0	1	0	1	1

Tabel 4.42 Pemasukan Kunci putaran 8

0	0	1	1	0	1	0	1
1	1	1	1	0	1	1	1
0	1	1	0	0	1	1	1
1	0	1	0	0	0	1	1
0	1	1	0	1	1	1	1
0	0	0	1	0	0	0	0
0	1	0	1	1	1	0	0
0	0	0	0	1	0	1	1

$K_8 = 00111111\ 01100101\ 01101000\ 01010000\ 11001101\ 11110000\ 01111100\ 01011110$

Pada putaran kedua didapatkan P_8 dan K_8 yang selanjutnya akan di XoRkan , yaitu:

$P_8 = 11000101\ 01100001\ 10000000\ 00001010\ 00110001\ 00001011\ 00011110\ 10000010$

$K_8 = 00111111\ 01100101\ 01101000\ 01010000\ 11001101\ 11110000\ 01111100\ 01011110$

----------XoR----------

Sehingga didapatkan chipperteks sementara pada putaran 8 yaitu R_8 yang selanjutnya akan di gunakan untuk putaran selanjutnya dalam proses enkripsi.

$R_8 =$ 11111010 00000100 11101000 01011010 11111100 11111011 01100010 11011100

Dari hasil enkripsi kata "mahmudah" dengan sandi "muslimah" didapatkan *chipperteks* dalam bentuk biner yaitu:

11111010 00000100 11101000 01011010 11111100 1111101101100010 11011100

Langkah terakhir dari proses penyandian adalah merubah cipperteks biner dalam bentuk desimal yang selanjutnya akan diubah kedalam bentuk alfabet. Berikut ini adalah konvert biner kedalam bentuk desimal sesuai dengan Persamaan 2.4 yaitu :

$$V(D) = d_{n-1} \times 10^{n-1} + d_{n-2} \times 10^{n-2} + \cdots + d_1 \times 10^1 + d_0 \times 10^0$$

1. Biner dari 11111010 akan dirubah kedalam bentuk desimal dengan menggunakan (Persamaan 2.5).

$$1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 250$$

2. Decimal dari 00000100 adalah

$$0 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 4$$

3. Desimal dari 11101000 adalah

$$1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 232$$

4. Desimal dari 01011010 adalah

$$0 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 90$$

5. Desimal dari 11111100 adalah

$$1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 252$$

6. Desimal dari 11111011 adalah

$$1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 251$$

7. Desimal dari 01100010 adalah

$$0 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 98$$

8. Desimal dari 11011100 adalah

$$1 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 220$$

Setelah diubah kedalam desimal, selanjutnya diubah kedalam bentuk karakter sesuai dengan Tabel 2.7

Tabel 4.43 Convert chipperrteks to ASCII

Biner	desimal	ASCII
11111010	250	*ú*
00000100	4	EOT
11101000	232	*è*
01011010	90	Z
11111100	252	*ü*
11111011	251	*ü*
01100010	98	b
11011100	220	*ʊ*

Dari proses enkripsi yang telah dilakukan dengan plainteks awal "mahmudah" dengan kunci "muslimah" didapatkan *chipperteks "ú*EOT*è*Z*üü*b*ʊ*" yang selanjutnya akan digunakan dalam proses dekripsi.

B. Dekripsi

Proses dekripsi dalam penelitian ini dimulai dari chipperteks di XoR kan dengan kunci yang telah dibangkitkan. Hasil XoR dari chipperteks di masukan kedalam bit sesuai pada Gambar 2.4, lalu pengambilan bit sesuai dengan pola pada Gambar 2.3. setelah itu hasilnya di XoR kan dengan kunci yang telah di xoR kan. Proses tersebut diulang sampai dengan 8 putaran.

R_8 = 11111010 00000100 11101000 01011010 11111100 11111011 01100010 11011100

K_8 = 00111111 01100101 01101000 01010000 11001101 11110000 01111100 01011110

----------XoR----------

P_8 = 11000101 01100001 10000000 00001010 00110001 00001011 00011110 10000010

Putaran 1

Pada putaran pertama ini yang dimasukkan kedalam bit adalah chipperrteks yang telah di xorkan dengan kunci yang telah dibangkitkan yaitu P_8, yang selanjutnya akan di masukan kedalam bit menggunakan pola pengambilan bit plainteks. Setelah bit-bit P8 dimasukan selanjutnya adalah mengambil bit dengan pola masukan plainteks, yaitu

Tabel 4.44 Pemasukan R7

0	1	0	1	0	0	1	0
0	0	0	1	0	1	0	1
1	0	0	0	0	1	0	0
1	0	0	1	1	1	0	0
1	1	0	1	1	0	0	0
1	0	1	0	0	0	0	1
0	1	1	0	0	0	0	0
0	0	0	0	0	0	0	1

Tabel 4.45 Pemasukan R7

0	1	0	1	0	0	1	0
0	0	0	1	0	1	0	1
1	0	0	0	0	1	0	0
1	0	0	1	1	1	0	0
1	1	0	1	1	0	0	0
1	0	1	0	0	0	0	1
0	1	1	0	0	0	0	0
0	0	0	0	0	0	0	1

Setelah bit-bit P_8 dimasukan selanjutnya adalah mengambil bit dengan pola masukan plainteks. Sehngga didapatkan chipperteks sementara yang selanjutnya akan di xorkan dengan kunci ketujuh yang telah dibangkitkan pada proses enkripsi.

R_7 = 01001010 00010101 00100001 10011100 00011011 10100001 00000110 00000001

K_7 = 10101100 10100000 00101100 11111101 11101010 11100111 10011101 00100001

----------XoR----------

Sehingga didapatkan plainteks sementara dari proses dekripsi, yaitu

P_7 = 11100110 10110101 00001101 01100001 11110001 01000110 10011011 00100000

```
load R8;
load K8;
a=(bitxor(R8,K8)-48);
b=a(1,1:32);
e1=reshape(b,4,8);
f2 = [e1(4,2), e1(3,2), e1(2,2), e1(1,2)];
f4 = [e1(4,4), e1(3,4), e1(2,4), e1(1,4)];
f6 = [e1(4,6), e1(3,6), e1(2,6), e1(1,6)];
f8 = [e1(4,8), e1(3,8), e1(2,8), e1(1,8)];
matriks1 = [e1(:,1)'; f2; e1(:,3)'; f4; e1(:,5)'; f6; e1(:,7)';
f8]';
c=a(1,33:64);
e2=reshape(c,4,8);
f22 = [e2(4,2), e2(3,2), e2(2,2), e2(1,2)];
f24 = [e2(4,4), e2(3,4), e2(2,4), e2(1,4)];
f26 = [e2(4,6), e2(3,6), e2(2,6), e2(1,6)];
f28 = [e2(4,8), e2(3,8), e2(2,8), e2(1,8)];
matriks2 = [e2(:,1)'; f22; e2(:,3)'; f24; e2(:,5)'; f26; e2(:,7)';
f28]';
d=[matriks2; matriks1];
g1 = [d(1,8) d(1,7) d(1,6) d(1,5) d(1,4) d(1,3) d(1,2) d(1,1)];
g3 = [d(3,8) d(3,7) d(3,6) d(3,5) d(3,4) d(3,3) d(3,2) d(3,1)];
g5 = [d(5,8) d(5,7) d(5,6) d(5,5) d(5,4) d(5,3) d(5,2) d(5,1)];
g7 = [d(7,8) d(7,7) d(7,6) d(7,5) d(7,4) d(7,3) d(7,2) d(7,1)];
DR7 = [g1 d(2,:) g3 d(4,:) g5 d(6,:) g7 d(8,:)]
save DR7.mat DR7
```

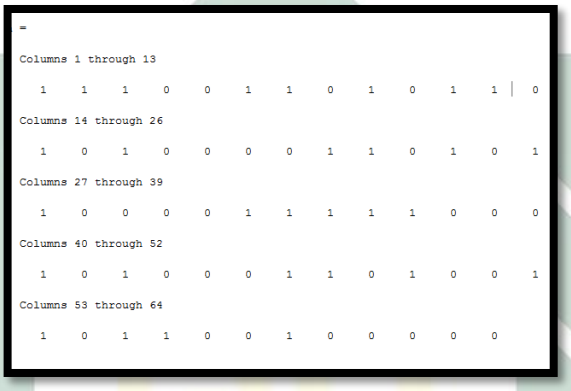
Gambar 4.11 Pseudo code pemasukan dan pengambilan bit pada dekripsi putaran 1

Pada Gambar 4.11 adalah codingan untuk proses dekripsi pada putaran pertama, yang dijadikan inputan pada putaran pertama ini adalah hasil xor antara chipperteks dengan kunci yang telah dibangkitkan. Sedangkan hasil codingan dari Gambar 4.11 adalah sebagai berikut :

```
DR7 =

  Columns 1 through 13

     0     1     0     0     1     0     1     0     0     0     0     1     0

  Columns 14 through 26

     1     0     1     0     0     1     0     0     0     0     1     1     0

  Columns 27 through 39

     0     1     1     1     0     0     0     0     0     1     1     0     1

  Columns 40 through 52

     1     1     0     1     0     0     0     0     1     0     0     0     0

  Columns 53 through 64

     0     1     1     0     0     0     0     0     0     0     0     1
```

Gambar 4.12 Hasil R_7

Gambar 4.12 adalah hasil dari pemasukan dan pengambilan bit-bit chipperteks. Sehingga menghasilkan R_7 yang nantinya akan digunakan untuk xor dengan kunci yang telah dibangkitkan pada putaran ke 7 yaitu $\boldsymbol{K_7}$ yang menghasilkan P_7. P_7 adalah plainteks sementara yang akan digunakan dalam putaran kedua yaitu sebagai berikut :



Gambar 4.13 Hasil plainteks sementara

Penjabaran dari Gambar 4.13 adalah

$P_7 =$ 11100110 10110101 00001101 01100001 11110001 01000110 10011011 00100000

Putaran 2

Pada putaran 2 dilakukan proses yang sama dengan putaran 1, namun yang menjadi masukan pada putaran kedua adalah hasil dekripsi pada putaran pertama, yaitu:

Tabel 4.46 Pengambilan R8

1	1	0	0	1	1	0	0
1	0	1	1	0	1	0	0
1	0	0	1	0	0	1	0
1	0	0	0	1	1	0	0
1	0	1	1	0	1	0	1
1	1	0	0	0	0	1	0
1	1	1	1	0	1	1	0
0	0	1	0	0	1	0	0

Tabel 4.47 Pengambilan R8

1	1	0	0	1	1	0	0
1	0	1	1	0	1	0	0
1	0	0	1	0	0	1	0
1	0	0	0	1	1	0	0
1	0	1	1	0	1	0	1
1	1	0	0	0	0	1	0
1	1	1	1	0	1	1	0
0	0	1	0	0	1	0	0

Sehingga didapatkan chipperteks sementara yang selanjutnya akan diproses sesuai dengan proses pada putaran pertama, namun dengan kunci yang keenam.

R_6 = 00110011 10110100 01001001 10001100 10101101 11000010 01101111 00100100

K_6 = 01000101 01000110 11010100 01100011 10000011 11111110 01010111 11101001

----------XoR----------

P_6 = 01110110 11110010 10011101 11101111 00101110 00111100 00111000 11001101

Putaran 3

Tabel 4.48 Pengambilan R5

0	0	0	0	0	0	1	1
0	1	0	0	0	0	1	0
1	1	1	1	1	0	0	1
0	1	1	1	1	1	0	1
0	0	1	0	1	1	1	1
1	1	1	1	0	0	1	1
1	1	1	0	0	1	1	1
1	0	1	0	1	1	0	1

Tabel 4.49 Pengambilan R5

0	0	0	0	0	0	1	1
0	1	0	0	0	0	1	0
1	1	1	1	1	0	0	1
0	1	1	1	1	1	0	1
0	0	1	0	1	1	1	1
1	1	1	1	0	0	1	1
1	1	1	0	0	1	1	1
1	0	1	0	1	1	0	1

R_5 = 11000000 01000010 10011111 01111101 11110100 11110011 11100111 10101101

K_5 = 01110010 10010011 00110001 10110010 10111011 01001110 10001010 01111011

----------XoR----------

P_5 = 10110010 11010001 10101110 11001111 01001111 10111101 01101101 11010110

Putaran 4

Tabel 4.50 Pengambilan R4

0	1	1	1	0	1	1	0
1	1	0	0	1	0	1	1
0	1	1	1	1	1	0	1
0	1	1	1	0	1	1	0
1	0	1	1	1	0	1	1
0	1	1	0	0	1	1	1
1	0	0	0	1	1	0	1
1	0	1	0	0	1	0	1

Tabel 4.51 Pengambilan R4

0	1	1	1	0	1	1	0
1	1	0	0	1	0	1	1
0	1	1	1	1	1	0	1
0	1	1	1	0	1	1	0
1	0	1	1	1	0	1	1
0	1	1	0	0	1	1	1
1	0	0	0	1	1	0	1
1	0	1	0	0	1	0	1

R_4 = 01101110 11001011 10111110 01110110 11011101 01100111 10110001 10100101

k_4 = 11101110 01101101 00110111 01010110 01010001 11011000 01100100 01110010

----------XoR----------

$P_4 =$ 10000000 10100110 10001001 00100000 10001100 10111111 11010101 11010111

Putaran 5

Tabel 4.52 pengambilan R3

1	0	1	1	1	1	1	1
0	0	0	1	1	0	1	1
0	1	1	1	0	1	0	1
0	1	1	1	1	0	1	0
1	0	1	0	1	1	0	0
0	0	0	1	0	0	0	0
0	0	1	1	0	0	1	0
0	0	0	0	0	1	0	0

Tabel 4.53pengambilan R3

1	0	1	1	1	1	1	1
0	0	0	1	1	0	1	1
0	1	1	1	0	1	0	1
0	1	1	1	1	0	1	0
1	0	1	0	1	1	0	0
0	0	0	1	0	0	0	0
0	0	1	1	0	0	1	0
0	0	0	0	0	1	0	0

$R_3 =$ 11111101 00011011 10101110 01111010 00110101 00010000 01001100 00000100

$K_3 =$ 11100111 00101001 10100011 10000101 11100110 10100011 10011110 00100111

----------XoR----------

$P_3 =$ 00011010 00110010 00001101 11111111 11010011 10110011 11010010 00100011

Putaran 6

Tabel 4.54 Pengambilan R2

1	1	1	1	1	0	0	1
1	1	0	1	1	1	0	1
0	0	1	0	0	0	1	0
1	0	1	0	1	0	0	0
0	0	0	0	0	1	1	1
0	1	0	1	0	0	1	1
0	0	1	0	0	1	1	1
1	1	1	0	0	1	1	1

Tabel 4.55 Pengambilan R2

1	1	1	1	1	0	0	1
1	1	0	1	1	1	0	1
0	0	1	0	0	0	1	0
1	0	1	0	1	0	0	0
0	0	0	0	0	1	1	1
0	1	0	1	0	0	1	1
0	0	1	0	0	1	1	1
1	1	1	0	0	1	1	1

$R_2 =$ 10011111 11011101 01000100 10101000 11100000 01010011 11100100 11100111

$K_2 =$ 01000111 10111110 10011000 01101110 11000100 01110110 01010100 11010101

----------XoR----------

$P_2 =$ 11011000 01100011 11011100 11000110 00100100 00100101 10110000 00110010

Putaran 7

Tabel 4.56 Pengambilan R1

0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	1
1	1	1	1	1	0	1	0
0	0	0	0	1	0	1	0
1	0	0	1	1	0	1	0
1	0	1	1	1	0	1	1
0	0	1	0	0	1	0	1
1	1	0	0	1	1	0	0

Tabel 4.57 Pengambilan R1

0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	1
1	1	1	1	1	0	1	0
0	0	0	0	1	0	1	0
1	0	0	1	1	0	1	0
1	0	1	1	1	0	1	1
0	0	1	0	0	1	0	1
1	1	0	0	1	1	0	0

$R_1 = 00011000\ 00000001\ 01011111\ 00001010\ 01011001\ 10111011\ 10100100\ 11001100$

$K_1 = 10110010\ 11010001\ 11001000\ 10101010\ 11010011\ 01101110\ 10111001\ 01100110$

----------XoR----------

$p_1 = 10101010\ 11010000\ 10010111\ 10100000\ 10001010\ 11010101\ 00011101\ 10101010$

Putaran 8

Pada putaran kedelapan ini plainteks sementara pada putaran ketujuh dimasukan kedalam bit dengan pola pengambilan plainteks pada proses enkripsi, selanjutnya bit-bit tersebut diambil sesuai dengan pola pemasukan pada proses enkripsi plainteks

Tabel 4.58 pengambilan plainteks

1	0	1	1	0	1	1	0
0	1	1	0	0	0	0	1
0	0	0	1	0	1	1	0
0	1	1	0	1	1	0	1
1	0	1	0	1	1	1	0
0	1	1	0	0	1	0	0
1	0	0	0	0	1	1	0
0	1	1	0	1	0	0	0

Tabel 4.59 pengambilan plainteks

1	0	1	1	0	1	1	0
0	1	1	0	0	0	0	1
0	0	0	1	0	1	1	0
0	1	1	0	1	1	0	1
1	0	1	0	1	1	1	0
0	1	1	0	0	1	0	0
1	0	0	0	0	1	1	0
0	1	1	0	1	0	0	0

Hasil dari pengambilan bit-bit tersebut yang akan menjadi plainteks awal, yaitu

$P = 01101101\ \ 01100001\ 01101000\ 01101101\ 01110101\ 01100100\ 01100001\ 01101000$

Dari proses dekripsi didapatkan hasil biner yaitu

01101101 01100001 01101000 01101101 01110101 01100100 01100001 01101000

yang selanjutnya dirubah kedalam huruf dengan menggunakan tabel ASCII sehingga diperoleh hasil sebagaimana dalam Tabel 4.60

Tabel 4.60 convert biner to ASCII

Biner	desimal	Huruf ASCII
01101101	109	m
01100001	97	a
01101000	104	h
01101101	109	m
01110101	117	u
01100100	100	d
01100001	97	a
01101000	104	h

Dengan mengetahui Tabel 4.60 tersebut maka disimpulkan chipperteks *"ú*EOT*è*Z*üüü*b*u*" berubah menjadi "mahmudah" setelah dilakukan proses dekripsi dengan kunci "muslimah" menggunakan algoritma tanam padi dan bajak sawah.

Dari hasil enkripsi plainteks "mahmudah" dan kunci "muslimah" dengan menggunakan algoritma tanam padi dan bajak sawah didapatkan hasil chipperteks *"ú*EOT*è*Z*üüü*b*u*"selanjutnya dilakukan proses dekripsi dengan algoritma tanam padi dan bajak sawah dengan kunci yang sama yaitu "muslimah" didapatkan hasil plainteks awal yaitu "mahmudah". Hasil dari proses enkripsi dan dekripsi dideskripsikan sebagai berikut:

Tabel 4.61 Hasil enkripsi dan dekripsi

Teks	proses	kunci	hasil
mahmudah	enkripsi	muslimah	*ú*EOT*è*Z*üüü*b*u*
*ú*EOT*è*Z*üüü*b*u*	dekripsi	muslimah	mahmudah

C. Algoritma Pengujian

Dalam algoritma pengujian ini penulis melakukan dekripsi dengan chipperteks hasil enkripsi yaitu *"úEOTèZüübu"*dengan menggunakan kunci pada putaran ke 6 yaitu :

Tabel 4.62 kunci ke-6 (kunci salah)

Biner	Desimal	karakter
01000101	68	D
01000110	70	F
11010100	212	Ô
01100011	98	b
10000011	130	e
11111110	254	℘
01010111	102	f
11101001	232	è

Hasil dekripsi dengan menggunakan kunci yang salah adalah :plainteks yang berbeda dengan plainteks awal yaitu : hasil ini didekripsikan sebagai berikut :

Tabel 4.63 Tabel hasil dengan kunci yang salah

Biner	Desimal	Karakter
110110111	214	
00001011	10	NL
00110110	54	6
01111011	122	z
10101001	168	K
11001010	204	|
10110010	178	
01001010	74	J

Hasil enkripsi plainteks "mahmudah" dengan kunci "muslimah" *chipperteks "úEOTèZüübu"*yang selanjutnya didekripsi dengan menggunakan kunci yang berbeda yaitu "DFÔbe℘fè" menghasilkan

plainteks yang berbeda dengan plainteks awal yaitu "𝒪NL6zK|ȧJ" yang didekripsikan dalama tabel berikut:

Tabel 4.64 Dekripsi pengujian kedua

Proses	kunci	hasil
Enkripsi	muslimah	*"úEOTèZüüb𝓊"*
Dekripsi	DF𝒪be𝔊𝑜𝑓è	𝒪NL6zK|ȧJ"

Pengujian yang kedua yaitu pada proses enkripsi plainteks "mahmudah" dengan kunci "muslimah" yang menghasilkan chipperteks, namun pada pengujian kedua pada proses dekripsi chipperteks dari hasil enkripsi dirubah menjadi chipperteks sementara pada putaran ketiga, lalu di dekripsi dengan kunci yang sama yaitu "muslimah", menghasilkan plainteks dalam biner yaitu 10100001 10110011 01101000 00011100 01101110 10000101 10000011 11010011. Selanjutnya dirubah kedalam bentuk desimal yang selanjutnya di rubah kedalam bentuk karakter. Perubahan dari biner kedesimal, desimal ke dalam bentuk karakter didekripsikan kedalam bentuk tabel sebagai berikut:

Tabel 4.65 Hasil pengujian ketiga

Biner	Desimal	Karakter
10100001	160	
10110011	178	ȧ
01101000	104	h
00011100	28	FS
01101110	110	n
10000101	132	*ä*
10000011	130	*è*
11010011	210	*𝒪*

BAB V

PENUTUP

A. Simpulan

Dari hasil pembahasan yang telah dipaparkan dapat disimpulkan bahwa:

1. Proses pembangkitan kunci dimulai dari merubah kunci yang telah ditentukan yaitu "muslimah" kedalam bentuk biner, selanjutnya dimasukan dan diambil bit-bitnya menggunakan algoritma bajak sawah. Hasil dari pembangkitan kunci pada putaran pertama dibangkitkan lagi sampai 8 putaran. Didapatkan hasil dari pembangkitan kunci yaitu:

   00111111 01100101 01101000 0101000011001101 11110000 01111100 01011110

2. Proses enkripsi dimulai dari merubah plainteks kedalam bentuk biner, selanjutnya dimasukan kedalam bit dan diambil bit-bitnya menggunakan algoritma tanam padi. selanjutnya hasil dari pengambilan *plainteks* dan pembangkitan kunci di XoRkan. Hasil dari XoR diproses lagi dan diulang sampai dengan 8 putaran, hingga didapatkan hasil akhir atau *chipperteks "ú*EOT*è*Z*üü*b*ɯ*"
3. Proses dekripsi dimulai dari chipperteks di XoR kan dengan kunci yang telah dibangkitkan. Hasil XoR dari *chipperteks* di masukan kedalam bit dengan pola pengambilan pada plainteks, lalu pengambilan bit dengan pola masukan pada plainteks. Proses tersebut diulang sampai dengan 8 putaran hingga didapatkan hasil dari proses dekripsi yaitu "mahmudah".

B. Saran

Dari penelitian yang telah dilakukan, peneliti menyarankan untuk penelitian selanjutnya bisa pada file dokumen yang di enkripsi, jadi tidak hanya delapan karakter. Untuk sintak juga bisa dilakukan hingga n karakter.

DAFTAR PUSTAKA

(2018, juli 12). Retrieved from http://www.sumberpengertian.co/pengertian-teks-menurut-para-ahli.

(2018, juli 12). Retrieved from http://sciencebuddies.org/science-fair-projects/references/table-of-8-bit-ascii-character-codes.

Brualdi, R. A. (2009). *Introductory combinatorics.* China: China Machine Press.

Irawati, A. R. (2017). *Logika dasar*. Lampung: Laboratorium komputasi dasar.

Kurniawati, A., & Darmawan, D. M. (2016). Implementasi algoritma advanced encryption standard (AES) untuk enkripsi dan dekripsi pada dokumen teks.

Lusiana, V. (2011). Implementasi kriptografi pada file dokumen. *Nitro PDF*.

Munir, R. (2006). *Kriptografi*. Bandung: Departemen Algoritma Informatika.

Nurhardian, & Pudoli, A. (2016). Implementasi keamanan file dengan kompresi Huffman dan Kriptografi menggunakan algoritma RC4 serta Steganografi menggunakan End of File berbasis desktop pada SMK Negeri 3 Kota tangerang. *jurnal TICOM*.

Pabokory, F. N., Astuti, I. ,., & Kridalaksana, A. H. (2015). Implementasi kriptografi pengamanan data pada pesan teks, isi file dokumen, dan file dokumen menggunakan algoritma advanced encryption standard. *Jurnal Informatika Mulawarman*.

Rosnawan, D. (2011). *Aplikasi algoritma RSA untuk keamanan data pada sistem informasi berbasis web.* Semarang: Universitas Negeri Semarang.

Sunenda, D. H. (2018, juli 12). *KBBI(5th)*. Retrieved from https://kbbi.kemendikbud.go.id/entri/religius.

Widodo, A., Wowor, d. a., Mailoa, E., & Pakereng, M. A. (2015). Perancangan Kriptografi Block Cipher Berbasis pada algoritma tanam padi dan bajak sawah. *Seminar Nasional Algoritma Informatika dan Sistem Informasi (SETISI)*.