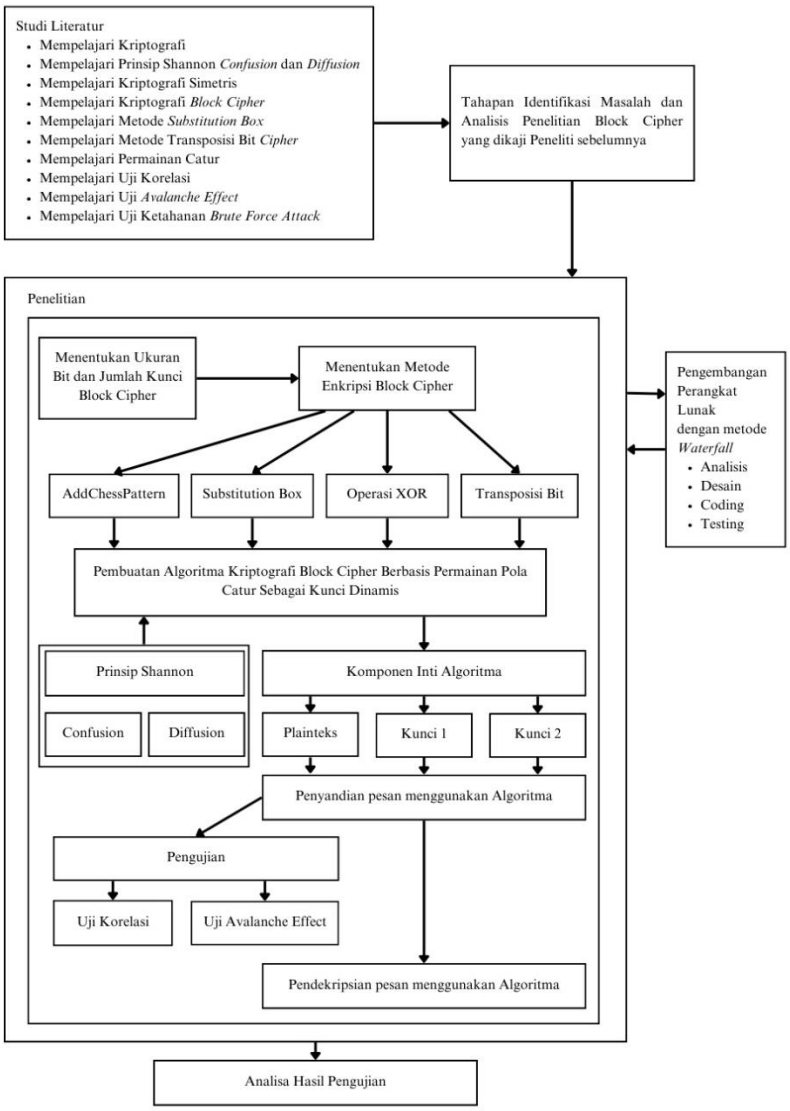
BAB III
METODE PENELITIAN

3.1 Desain Penelitian

Desain penelitian adalah tahapan kegiatan peneliti atau gambaran yang akan dilakukan dalam penelitian. Desain penelitian yang digunakan dalam penelitian ini adalah metode *Design Research Methodology* (DRM). Menurut Blessing & Chalkrabarti (2009) dalam Huda (2023), DRM terdiri dari empat tahap, yaitu Klarifikasi Penelitian (*Research Clarification*), Studi Deskriptif I (*Descriptive Study* I), Studi Preskriptif (*Perscriptive Study*), dan Studi Deskriptif II (*Descriptive Study* II). Gambaran lengkapnya dapat dilihat pada Gambar 3.1 berikut:



Gambar 3.1 Desain Tahapan Penelitian

Penjelasan dari tahapan desain penelitian pada Gambar 3.1 akan dijelaskan pada **subbab** berikut.

3.1.1 Klarifikasi Penelitian

Klarifikasi penelitian atau juga tahap studi literatur merupakan tahap di mana peneliti mempelajari semua aspek yang berhubungan dengan pembuatan algoritma kriptografi *block cipher*. Beberapa aspek yang penting untuk peneliti pelajari yaitu mempelajari dasar-dasar kriptografi, mempelajari prinsip *Shannon* yaitu *confusion* dan *diffusion*, mempelajari sistem kriptografi simetris dan kriptografi *block cipher*. Selain itu peneliti juga mempelajari beberapa metode-metode dalam pembuatan algoritma kriptografi, yaitu metode *substitution box*, dan metode transposisi bit *cipher*. Tidak lupa juga peneliti mempelajari bagaimana cara menguji hasil dari algoritma kriptografi yang sudah dirancang dengan menggunakan uji korelasi dan uji *Avalanche Effect* dan uji Ketahanan *Brute Force Attack*.

3.1.2 Studi Deskriptif I

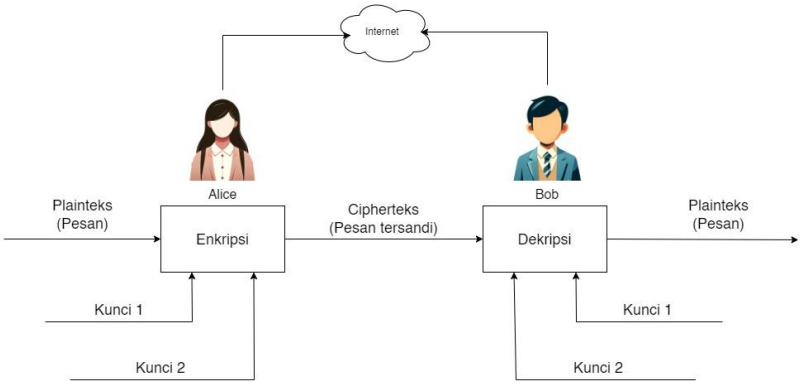
Tahap identifikasi masalah dimulai dengan mengidentifikasi masalah yang akan diteliti. Masalah tersebut ditemukan melalui analisis terhadap algoritma kriptografi block cipher yang telah dikaji oleh peneliti sebelumnya. Ditemukan bahwa banyak algoritma kriptografi block cipher hanya menggunakan satu pola statis dalam proses enkripsinya. Hal ini mendorong peneliti untuk merancang sebuah algoritma kriptografi yang tidak hanya bergantung pada satu pola statis, melainkan menggunakan pola dinamis.

3.1.3 Studi Preskriptif

Melakukan penelitian untuk membuat algoritma kriptografi *block cipher* baru.

1. Sebelum penelitian dimulai, peneliti menetapkan ukuran bit dan jumlah kunci yang akan digunakan dalam pengembangan algoritma *block cipher*. Dengan melihat penelitian sebelumnya yang umumnya menggunakan 64-bit, peneliti memutuskan untuk mengadopsi ukuran yang sama. Ukuran ini cocok untuk dikombinasikan dengan pola

permainan catur, karena papan catur memiliki matriks 8x8 yang setara dengan 64-bit dalam representasi bit dengan bit. Dalam penelitian ini, akan digunakan dua kunci utama, yaitu kunci 1 yang berupa 8 karakter ASCII dan kunci 2 yang berupa seluruh langkah dari salah satu bidak catur dalam satu permainan. Kunci ini akan digunakan sebagai pola aturan untuk pengacakan bit dalam algoritma. Ilustrasi proses enkripsi/dekripsi pesan dapat dilihat pada Gambar 3.2 berikut.



Gambar 3.2 Ilustrasi proses enkripsi/dekripsi pesan algoritma

Gambar 3.2 mengilustrasikan bahwa proses enkripsi dan dekripsi membutuhkan tiga *input*, yaitu satu plainteks dan dua kunci, yaitu kunci 1 dan kunci 2. Dalam proses transmisi, pengguna dapat menggunakan internet sebagai media pertukaran cipherteks maupun kunci. Namun, untuk kunci, pengguna harus terlebih dahulu melakukan kesepakatan (contohnya dengan menentukan permainan catur yang akan digunakan sebagai kunci 2 melalui saluran aman seperti pertemuan langsung). Setelah itu, pemilihan bidak catur dapat ditransmisikan melalui internet.

2. Untuk mengembangkan algoritma kriptografi *block cipher* baru, peneliti memilih menggunakan empat metode enkripsi yang berbeda. Pemilihan metode ini juga didasarkan pada prinsip *confusion* dan *diffusion*. Metode-metode yang digunakan adalah sebagai berikut: 1.) Metode substitusi S-box Rijndael. Metode ini merupakan bagian dari algoritma kriptografi *block cipher Advanced Encryption Standard* (AES). S-box yang digunakan mengikuti substitusi yang telah disusun oleh algoritma Rijndael, yang menggantikan blok-blok data

input dengan blok-blok *output* berdasarkan substitusi tertentu. 2.) Metode *AddChessPattern* atau sering disebut juga sebagai metode *Bit Shuffling*. Metode ini digunakan untuk mengubah urutan bit dalam suatu blok data tanpa mengubah nilai bit tersebut. Dalam penelitian ini, menggunakan dua pola berbeda: pola menyamping (*row*) dan pola menurun (*column*) sebagai aturan pengacakan bit.

p1*	p2*	p3*	p4*	p5*	p6*	p7*	p8*
p9*	p10*	p11*	p12*	p13*	p14*	p15*	p16*
p17*	p18*	p19*	p20*	p21*	p22*	p23*	p24*
p25*	p26*	p27*	p28*	p29*	p30*	p31*	p32*
p33*	p34*	p35*	p36*	p37*	p38*	p39*	p40*
p41*	p42*	p43*	p44*	p45*	p46*	p47*	p48*
p49*	p50*	p51*	p52*	p53*	p54*	p55*	p56*
p57*	p58*	p59*	p60*	p61*	p62*	p63*	p64*

Gambar 3.3 Alur penyusunan bit *input* plainteks secara *default* (dapat berubah-ubah)

Pola *input* bit plainteks statis pada Gambar 3.3 dapat disesuaikan dengan *input* kunci 2, yang ditunjukkan dalam Gambar 3.4 dan *input* kunci 2 ditandai dengan warna kuning.

p3	p2	p4	p1	p5	p6	p7	p8
p9	p10	p11	p12	p13	p14	p15	p16
p17	p18	p19	p20	p21	p22	p23	p24
p25	p26	p27	p28	p29	p30	p31	p32
p33	p34	p35	p36	p37	p38	p39	p40
p41	p42	p43	p44	p45	p46	p47	p48
p49	p50	p51	p52	p53	p54	p55	p56
p57	p58	p59	p60	p61	p62	p63	p64

Gambar 3.4 Alur penyusunan bit *input* plainteks setelah ditambahkan *input* kunci 2

Sedangkan pada Gambar 3.5, *input* bit plainteks nantinya diambil dengan pola menurun (*column*).

p1	p9	p17	p25	p33	p41	p49	p57
p2	p10	p18	p26	p34	p42	p50	p58
p3	p11	p19	p27	p35	p43	p51	p59
p4	p12	p20	p28	p36	p44	p52	p60
p5	p13	p21	p29	p37	p45	p53	p61
p6	p14	p22	p30	p38	p46	p54	p62
p7	p15	p23	p31	p39	p47	p55	p63
p8	p16	p24	p32	p40	p48	p56	p64

Gambar 3.5 Alur pengambilan bit *input* plainteks

3.) Metode operasi XOR. Metode ini menggunakan hasil dari plainteks dan kunci yang telah diproses sebelumnya. Hasil tersebut kemudian disubstitusikan menggunakan logika operasi XOR untuk menghasilkan *output* berupa bilangan biner baru. Tabel operasi XOR dapat dilihat pada Tabel 3.1.

Tabel 3.1 Operasi logika XOR

Input		*Output*
A	*B*	*AB*
0	0	0
0	1	1
1	0	1
1	1	0

4.) Metode Transposisi Bit. Metode ini merupakan metode terakhir yang akan digunakan dalam penelitian pembuatan algoritma kriptografi ini. Metode ini merupakan metode terakhir yang digunakan dalam penelitian pembuatan algoritma kriptografi ini. Metode transposisi bit melibatkan perubahan urutan bit dalam blok data tanpa mengubah nilai bit tersebut. Sebagai contoh, terdapat blok data 01000110. Jika ditransposisikan sebanyak satu kali ke arah kiri,

maka bit yang paling kiri akan dipindah ke belakang menjadi 10001100. Dalam penelitian ini, metode transposisi bit menggunakan pendekatan di mana *input* kunci 1 dilibatkan dalam menentukan seberapa banyak transposisi bit yang akan dilakukan. Metode ini mengubah *input* kunci 1 menjadi bilangan desimal terlebih dahulu. Setelah itu, semua bilangan desimal tersebut dijumlahkan dan di modulus dengan 64, lalu ditambahkan 1 agar nilai akhirnya tidak bernilai nol. Berikut adalah persamaan lengkap untuk menghitung seberapa banyak bit yang digeser pada metode transposisi bit:

$$Nt = \big((ck1 + ck2 + ck3 + ck4 + ck5 + ck6 + ck7 + ck8) mod\ 64\big) + 1 \quad (3.1)$$

di mana N_t adalah jumlah transposisi bit, dan $ck_1, ck_2, \ldots, ck_8$ adalah karakter kunci pada posisi masing-masing.

CONTOH KUNCI 1
SECURITY

ASCII	DECIMAL
S	83
E	69
C	67
U	85
R	82
I	73
T	84
Y	89
Total	**632**

RUMUS PENGHITUNG JUMLAH TRANSPOSISI BIT
Nt = (ck1+ck2+ck3+ck4+ck5+ck6+ck7+ck8 mod 64) + 1
Note: ck adalah *Char Key*

JUMLAH TRANSPOSISI BIT
Nt = (83+69+67+85+82+73+84+89 mod 64) + 1
Nt = **57**

Gambar 3.6 Contoh proses kalkulasi jumlah transposisi bit

Gambar 3.6 menunjukkan bahwa dengan *input* kunci 1 berupa "SECURITY" menghasilkan jumlah transposisi bit sebesar 57 kali. Selain digunakan untuk menghasilkan jumlah transposisi bit, kunci 1 juga digunakan untuk menentukan berapa banyak putaran enkripsi yang akan dilakukan. Meskipun serupa dengan penentuan jumlah transposisi bit, terdapat sedikit perbedaan dalam penerapannya. Setelah mendapatkan nilai desimal dari masing-masing karakter, dilakukan operasi penjumlahan, pengurangan, dan modulus dengan persamaan lengkap sebagai berikut:

$$Nr = \big((ck1 + ck2 - ck3 + ck4 - ck5 + ck6 - ck7 + ck8) mod\ 10\big) + 1 \quad (3.2)$$

di mana N_r adalah jumlah putaran, dan ck_n adalah karakter kunci ke-n (di mana n menunjukkan posisi karakter).

CONTOH KUNCI 1
SECURITY

ASCII	DECIMAL
S	83
E	69
C	67
U	85
R	82
I	73
T	84
Y	89
Total	**632**

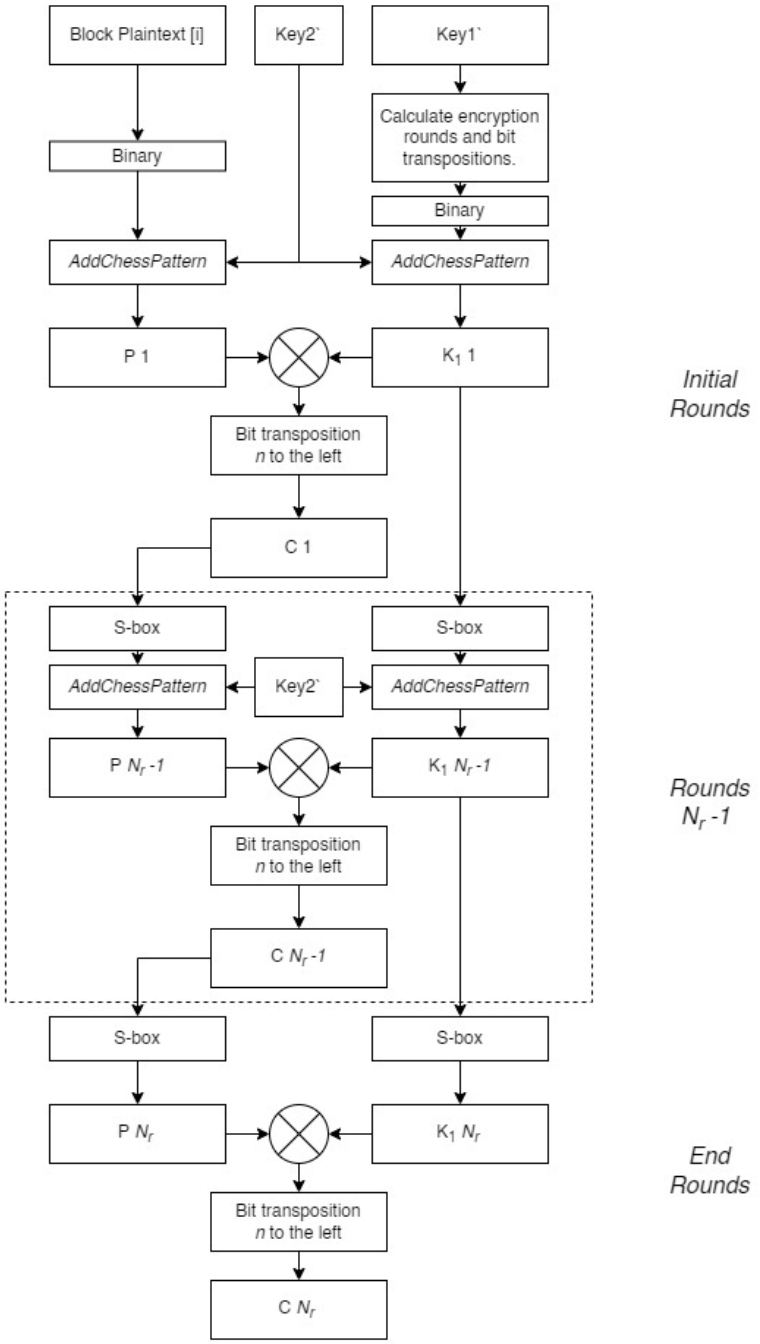
RUMUS PENGHITUNG JUMLAH PUTARAN ENKRIPSI
Nr = (ck1+ck2-ck3+ck4-ck5+ck6-ck7+ck8 mod 10) + 1
Note: ck adalah *Char Key*

JUMLAH PUTARAN ENKRIPSI
Nr = (83+69-67+85-82+73-84+89 mod 10) + 1
Nr = **7**

Gambar 3.7 Contoh proses kalkulasi jumlah putaran enkripsi

Gambar 3.7 menunjukkan bahwa dengan *input* kunci 1 berupa "SECURITY" menghasilkan jumlah putaran enkripsi sebesar 7 kali.

3. Setelah menentukan metode apa saja yang akan digunakan, langkah selanjutnya adalah mengintegrasikan semua metode tersebut menjadi satu kesatuan yang utuh untuk digunakan dalam proses algoritma enkripsi dan dekripsi kriptografi *block cipher* yang baru. Proses enkripsi algoritma baru tersebut dapat dilihat pada Gambar 3.8.



Gambar 3.8 Diagram blok proses enkripsi (sebut saja *main algorithm*)

Gambar 3.8 menunjukkan alur proses enkripsi. Proses enkripsi akan dijelaskan sebagai berikut:

1.) Plainteks yang diperoleh dari *input* pengguna dikonversi ke bentuk biner.

2.) Bilangan biner tersebut akan diproses di dalam blok *AddChessPattern* sesuai dengan metode yang telah dipaparkan sebelumnya.

3.) Blok P1 merupakan hasil keluaran dari proses *AddChessPattern*.

4.) Kunci 1 yang diperoleh dari *input* pengguna sebelum dikonversi menjadi biner, juga digunakan untuk menghitung jumlah putaran enkripsi dan jumlah transposisi bit.

5.) Proses perhitungan jumlah putaran enkripsi dan jumlah transposisi bit mengikuti yang telah dijelaskan sebelumnya.

6.) Setelah itu, Kunci 1 dikonversi ke bentuk biner dan dimasukkan ke dalam blok *AddChessPattern*, di mana prosesnya sama seperti pada langkah nomor 2.

7.) Blok K_1 1 merupakan hasil keluaran dari proses *AddChessPattern*.

8.) P1 dan K_1 1 kemudian dioperasikan dengan operasi logika XOR.

9.) Hasil dari operasi XOR tersebut ditransposisikan ke kiri sesuai dengan perhitungan transposisi bit.

10.) Putaran awal/putaran nol selesai.

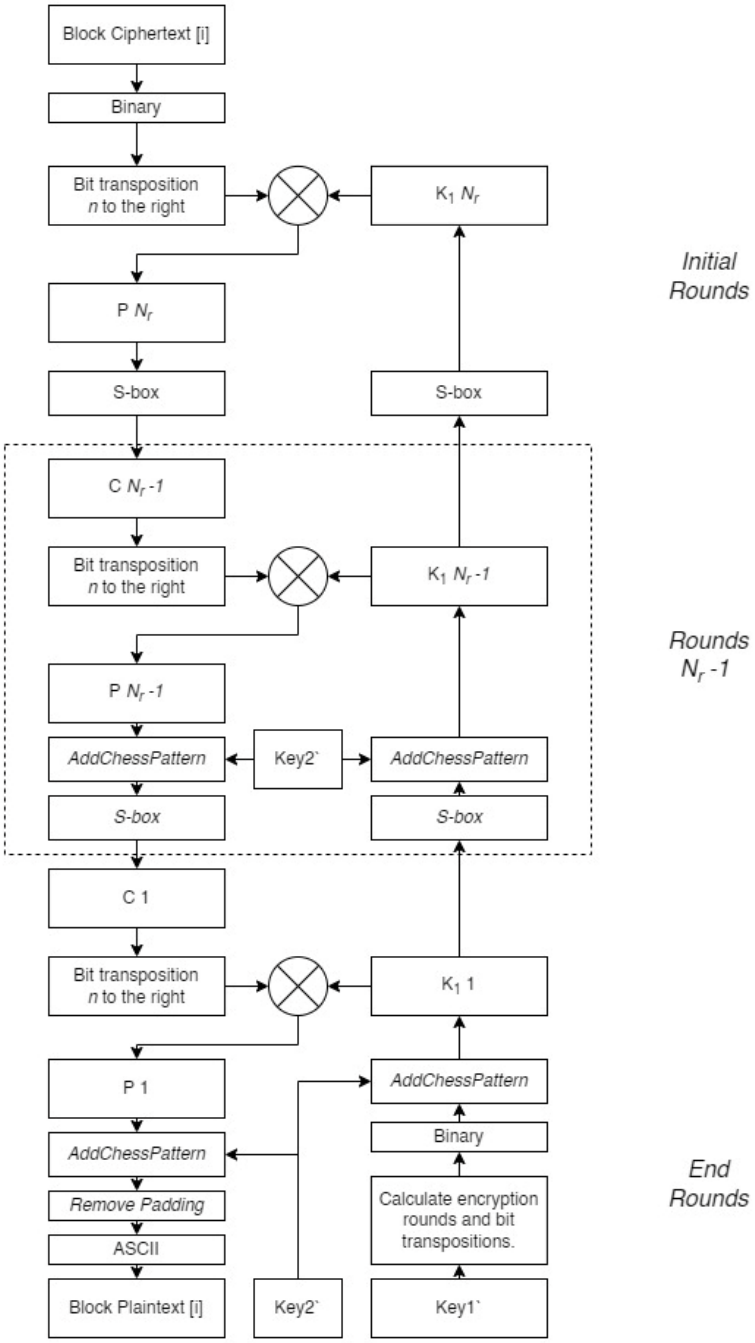
11.) Untuk putaran satu hingga ke-*n*, prosesnya sama seperti langkah 2-9, dengan tambahan integrasi metode S-box sebelum proses *AddChessPattern*.

12.) Putaran berulang selesai.

13.) Pada putaran ke-*n* + 1, yang merupakan putaran terakhir, prosesnya tetap mirip dengan langkah 2-9, namun proses blok *AddChessPattern* digantikan dengan S-box.

14.) Proses enkripsi selesai.

Sedangkan untuk proses dekripsi dapat dilihat pada Gambar 3.9.



Gambar 3.9 Diagram blok proses dekripsi (sebut saja *main algorithm*)

Gambar 3.9 menunjukkan alur proses dekripsi. Proses dekripsi akan dijelaskan sebagai berikut:

1.) Proses dekripsi dimulai dengan memproses kunci 1 hingga mendapatkan K_1 *Nr*, dengan proses yang sama seperti pada tahap kunci dalam proses enkripsi.

2.) Cipherteks yang diperoleh dari *input* pengguna dikonversi menjadi biner, setelah itu dilakukan transposisi ke kanan sesuai dengan perhitungan transposisi bit.

3.) Hasil transposisi bit cipherteks dan K_1 *Nr* kemudian dioperasikan dengan operasi logika XOR.

4.) Hasil dari operasi XOR tersebut masuk ke proses pengolahan dengan S-box.

5.) Putaran awal/putaran nol selesai.

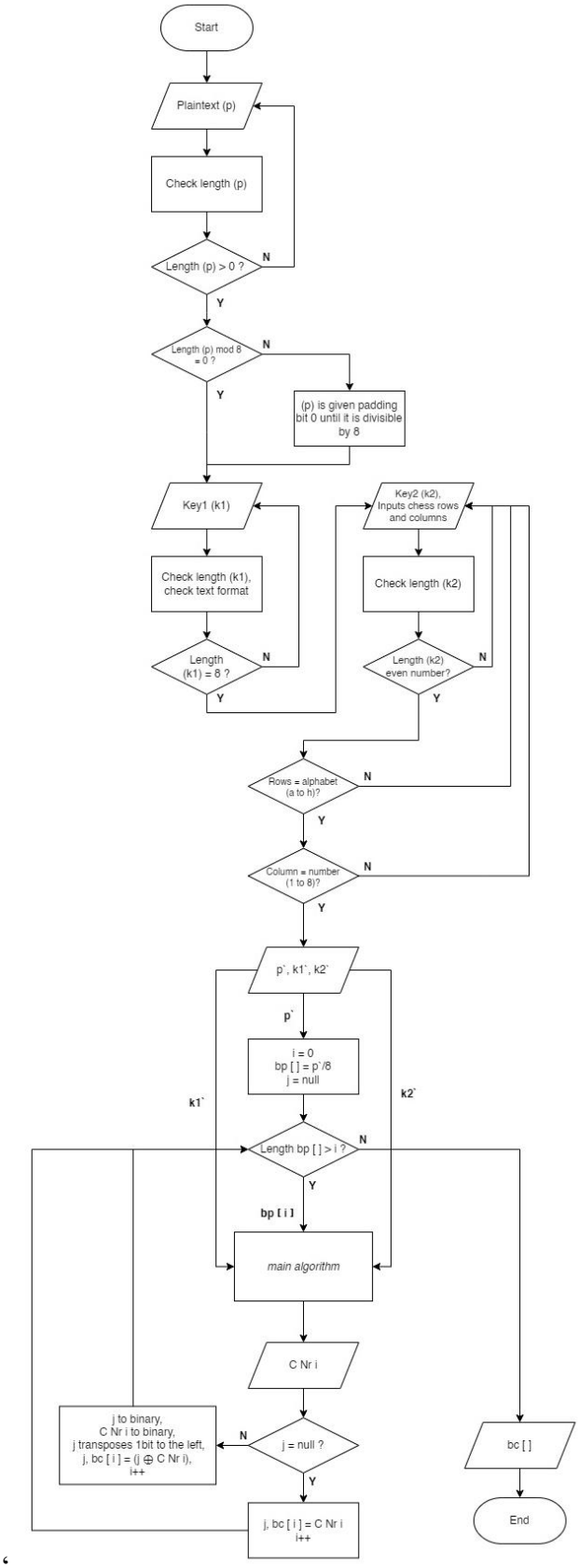
6.) Untuk putaran satu hingga ke-*n*, prosesnya sama seperti langkah 2-4, dengan tambahan integrasi metode *AddChessPattern* sebelum proses S-box.

7.) Putaran berulang selesai.

8.) Pada putaran ke-*n* + 1, yang merupakan putaran terakhir, prosesnya tetap mirip dengan langkah 2-4, namun proses blok S-box digantikan dengan *AddChessPattern*.

9.) Proses dekripsi selesai.

4. Poin c yang telah dijelaskan sebelumnya menjelaskan alur enkripsi dan dekripsi untuk ukuran plainteks atau cipherteks yang bernilai 64-bit. Sekarang akan dijelaskan algoritma lengkap atau *flowchart* tanpa batasan jumlah bit untuk plainteks atau cipherteks. *Flowchart* proses enkripsi dapat dilihat pada Gambar 3.10.



Gambar 3.10 *Flowchart* proses enkripsi

Gambar 3.10 menunjukkan diagram alir proses enkripsi secara penuh. Proses enkripsi secara penuh akan dijelaskan sebagai berikut:

1.) Mulai

2.) Pengguna memasukkan plainteks

3.) Sistem memeriksa panjang plainteks. Jika panjang plainteks lebih dari nol dan habis dibagi 8, sistem lanjut ke langkah berikutnya. Jika tidak, plainteks akan di *padding* dengan bit 0.

4.) Pengguna memasukkan Kunci 1

5.) Sistem memeriksa panjang dan format Kunci 1. Jika panjang Kunci 1 adalah 8 karakter, pengguna melanjutkan memasukkan Kunci 2. Jika tidak, pengguna harus memasukkan Kunci 1 kembali.

6.) Pengguna memasukkan Kunci 2

7.) Sistem memeriksa panjang dan menetapkan baris dan kolom untuk Kunci 2. Jika panjang Kunci 2 adalah kelipatan genap (tanda koma akan dikecualikan), sistem memeriksa apakah baris berupa huruf a hingga h dan kolom berupa angka 1 hingga 8. Jika sesuai, sistem akan mengolah ketiga *input* tersebut (plainteks, kunci 1, dan kunci 2). Jika tidak, pengguna harus memasukkan Kunci 2 kembali.

8.) Sistem menghasilkan *output* p', k1', k2', di mana variabel yang memiliki aksen (`) sudah diolah oleh sistem.

9.) Sistem menetapkan variabel i sebagai nol, variabel bp (blok plainteks) sebagai kumpulan dari p` yang sudah dibagi ke blok yang berisikan 8 karakter, dan variabel j sebagai *null*.

10.) Jika panjang bp lebih dari i, sistem memproses semua *input* pengguna (bp[i], k1`, dan k2`) ke dalam algoritma utama. Jika ya, lanjut ke langkah 12. Tapi jika tidak lanjut ke langkah 13.

11.) Sistem menghasilkan *output* berupa C Nr i.

12.) Jika j adalah *null*, sistem menetapkan variabel j dan bc[i] sebagai C Nr i, kemudian sistem meng *increment* variabel i dan kembali ke langkah 10. Jika tidak, sistem mengonversi variabel j dan

C Nr i menjadi biner. Variabel j kemudian ditransposisikan satu bit ke kiri, kedua biner tersebut di-XOR-kan, dan sistem menetapkan variabel j dan bc[i] sebagai hasil XOR tersebut. Terakhir, sistem meng *increment* variabel i dan kembali ke langkah 10.

13.) Jika i lebih besar dari panjang bp, sistem menghasilkan *output* berupa variabel bc dalam bentuk *array*.

14.) Selesai

Sedangkan untuk diagram alir proses dekripsi, sama persis dengan diagram alir proses enkripsi, namun variabel plainteks diubah menjadi cipherteks dan bp diubah menjadi bc.

Setelah merancang algoritma kriptografi *block cipher* berbasis permainan pola catur sebagai kunci dinamis, langkah selanjutnya adalah mengimplementasikannya ke dalam kode program. Program aplikasi *website* dikembangkan menggunakan bahasa pemrograman *Python* dengan menggunakan *Integrated Development Environment* (IDE) Visual Studio Code. Pengembangan perangkat lunak menggunakan metode pendekatan *waterfall*. Pemilihan platform *website* dipertimbangkan agar dapat diakses dengan mudah oleh semua pengguna tanpa memerlukan instalasi aplikasi tambahan.

3.1.4 Studi Deskriptif II

Pengujian dan analisis hasil pengujian dilakukan setelah perancangan algoritma serta implementasi kode program selesai. Pengujian algoritma memanfaatkan platform permainan catur bernama chess.com sebagai media simulasi permainan catur. Pengujian dilakukan dengan melakukan proses enkripsi menggunakan berbagai kombinasi kunci 1 dan kunci 2. Kombinasi kunci 1 dan kunci 2 dapat dilihat pada Gambar 3.11.

```
plaintext_ascii = "Tekkom unggul, jaya, bermartabat"
plaintext_hex = 54656b6b6f6d20756e6767756c2c206a6179612c206265726d61727461626174
key1 [ ] = ["ch@Mp10N", "tekkom20", "SECURITY", "mrizkiw "]
key2 [ ] = ["d8,b8", "d1,d2,h2,h8", "e8,g8,f7,e7,g7", "e1,c1,d1,d2,e3,d3"]
```

Gambar 3.11 Daftar komponen untuk pengujian algoritma

Gambar 3.11 menampilkan berbagai sampel kunci 1 dan kunci 2. Dengan masing-masing terdapat 4 kunci 1 dan 4 kunci 2, total

pengujian akan mencakup 16 uji untuk setiap kombinasi. Oleh karena itu, keseluruhan pengujian akan mencakup total 32 uji, dengan 16 uji untuk uji korelasi dan 16 uji untuk uji *Avalanche Effect*. Selain melakukan uji korelasi dan uji *Avalanche Effect*, dilakukan juga uji ketahanan dari *brute force attack*.

3.2 Metode Penelitian

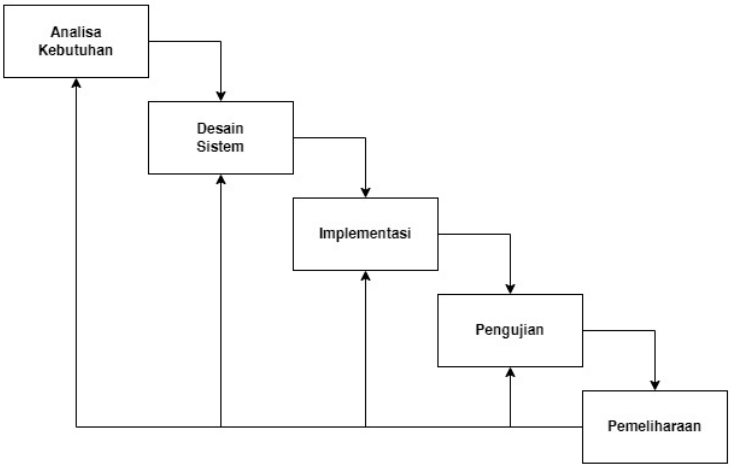
Metode penelitian ini dibagi menjadi dua bagian, yaitu metode pengumpulan data dan metode pengembangan perangkat lunak.

3.2.1 Metode Pengumpulan Data

Metode pengumpulan data dilakukan melalui eksplorasi studi literatur. Pada proses ini, peneliti mempelajari semua aspek yang terkait dengan pembuatan algoritma kriptografi *block cipher*. Beberapa aspek yang penting untuk dipelajari meliputi dasar-dasar kriptografi, prinsip Shannon seperti *confusion* dan *diffusion*, sistem kriptografi simetris, dan kriptografi *block cipher*. Selain itu, peneliti juga mempelajari beberapa metode penting dalam pembuatan algoritma kriptografi, seperti *substitution box* dan transposisi bit *cipher*. Tidak ketinggalan, peneliti mempelajari cara menguji hasil dari algoritma kriptografi yang dirancang menggunakan uji korelasi dan uji *Avalanche Effect*. Semua studi literatur tersebut dipelajari melalui berbagai literatur seperti buku, jurnal, skripsi, dan sumber ilmiah lainnya.

3.2.2 Metode Pengembangan Perangkat Lunak

Metode pengembangan perangkat lunak menggunakan pendekatan *waterfall* seperti yang dijelaskan oleh Sommerville (2011). Sommerville mengemukakan bahwa terdapat lima tahapan dalam metode *waterfall*, yaitu *Requirements Analysis and Definition*, *System and Software Design*, *Implementation and Unit Testing*, *Integration and System Testing*, dan *Operation and Maintenance*. Ilustrasi metode *waterfall* dapat dilihat pada Gambar 3.12.



Gambar 3.12 Ilustrasi metode *waterfall* oleh Sommerville (2011)

Tahapan-tahapan dari metode *waterfall* adalah sebagai berikut:

1. Analisa Kebutuhan. Tahap ini berfokus pada pemahaman mendalam terhadap kebutuhan sistem yang akan dikembangkan dari perspektif pengguna. Pada tahap ini, akan ditentukan fitur-fitur yang akan diimplementasikan dalam perangkat lunak tersebut.
2. Desain Sistem. Tahap ini mengubah spesifikasi sistem yang telah dikumpulkan menjadi desain sistem yang lengkap.
3. Implementasi. Tahap ini melibatkan penerapan seluruh desain sistem ke dalam program aplikasi *website* menggunakan bahasa pemrograman *Python*.
4. Pengujian. Tahap ini dilakukan setelah proses *coding* selesai, di mana aplikasi diuji secara menyeluruh untuk memastikan fungsionalitasnya dan meminimalkan kemungkinan terjadinya *bug* atau sistem eror.
5. Pemeliharaan. Tahap ini melibatkan pemeliharaan sistem setelah sistem diimplementasikan dan mulai beroperasi. Jika terjadi kesalahan atau perubahan kebutuhan, aplikasi dapat diperbaiki dan disesuaikan.

3.3 Alat dan Bahan Penelitian

Berdasarkan analisa kebutuhan di atas, alat dan bahan yang digunakan dalam penelitian ini adalah sebagai berikut.

3.3.1 Alat Penelitian

Dalam penelitian ini, peneliti menggunakan alat bantu penunjang baik berupa perangkat keras maupun perangkat lunak. Adapun perangkat keras yang digunakan adalah sebuah komputer dengan spesifikasi sebagai berikut:

1. Prosesor Intel i5-11400H 2.60 GHz
2. RAM 32GB
3. Penyimpanan SSD 1000 GB

Sementara itu, perangkat lunak yang digunakan untuk menunjang penelitian ini adalah sebagai berikut:

1. Sistem Operasi Windows 10 64-bit
2. Microsoft Visual Studio Code
3. Bahasa Pemrograman Python
4. Microsoft Excel
5. Notepad
6. Google Chrome
7. Google Colab
8. Google Cloud Platform
9. Cloud Digital Ocean
10. Diagrams.net
11. Canva
12. Chess.com

Selain itu, peneliti juga menggunakan sebuah *Virtual Private Server* (VPS) sebagai infrastruktur untuk menyediakan aplikasi *website* hasil penelitian ini. Berikut adalah spesifikasi dari VPS tersebut:

1. Prosesor Intel 1 vCPU
2. RAM 1GB
3. Penyimpanan SSD 25GB
4. Sistem Operasi Linux Ubuntu 20.04 LTS

3.3.2 Bahan Penelitian

Bahan penelitian yang digunakan dalam penelitian ini meliputi jurnal penelitian, buku, skripsi, dan dokumentasi lainnya yang diperoleh melalui *World Wide Web* dan perpustakaan. Bahan-bahan ini mencakup penelitian terdahulu mengenai perancangan algoritma kriptografi *block cipher*, dasar-dasar kriptografi, prinsip Shannon seperti *confusion* dan *diffusion*, sistem kriptografi simetris, dan kriptografi *block cipher*. Selain itu, peneliti mempelajari beberapa metode penting dalam pembuatan algoritma kriptografi, seperti *substitution box* dan transposisi bit *cipher*. Peneliti juga mempelajari cara menguji hasil dari algoritma kriptografi yang dirancang menggunakan uji korelasi dan uji *Avalanche Effect* serta uji ketahanan *brute force attack*.