BAB III
METODOLOGI PENELITIAN

3.1 Analisis Masalah

Dalam proses pertukaran informasi, keamanan merupakan hal yang sangat penting karena seringkali terjadi penyadapan atau pencurian informasi oleh pihak yang tidak berhak Oleh karena itu, dalam proses pengiriman atau penerimaan informasi dibutuhkan suatu sistem keamanan yang dapat menjaga kerahasiaan pesan atau informasi agar tidak disalahgunakan oleh pihak lain. Salah satu cara dalam menjaga keamanan pesan adalah dengan melakukan enkripsi atau penyandian pesan agar informasi yang disampaikan tidak dapat dipahami oleh orang lain.

Penyandian pesan dilakukan dengan beberapa algoritma, salah satunya dengan algoritma *Playfair Cipher*. Pada algoritma *Playfair Cipher* yang biasa, *plaintext* dan kunci hanya terdiri dari 25 huruf alfabet dengan penghilangan huruf J.

Seiring berjalannya waktu, sudah dilakukan beberapa modifikasi pada algoritma *Playfair Cipher*. Salah satunya dengan memodifikasi tabel kunci menjadi bujur sangkar berukuran 6×6 yang berisi semua huruf alfabet dan angka. Tetapi proses penyandian pesan masih dilakukan per karakter sehingga terdapat kendala jika terdapat dua karakter yang bersebelahan maka akan dienkripsi ke *ciphertext* yang sama.

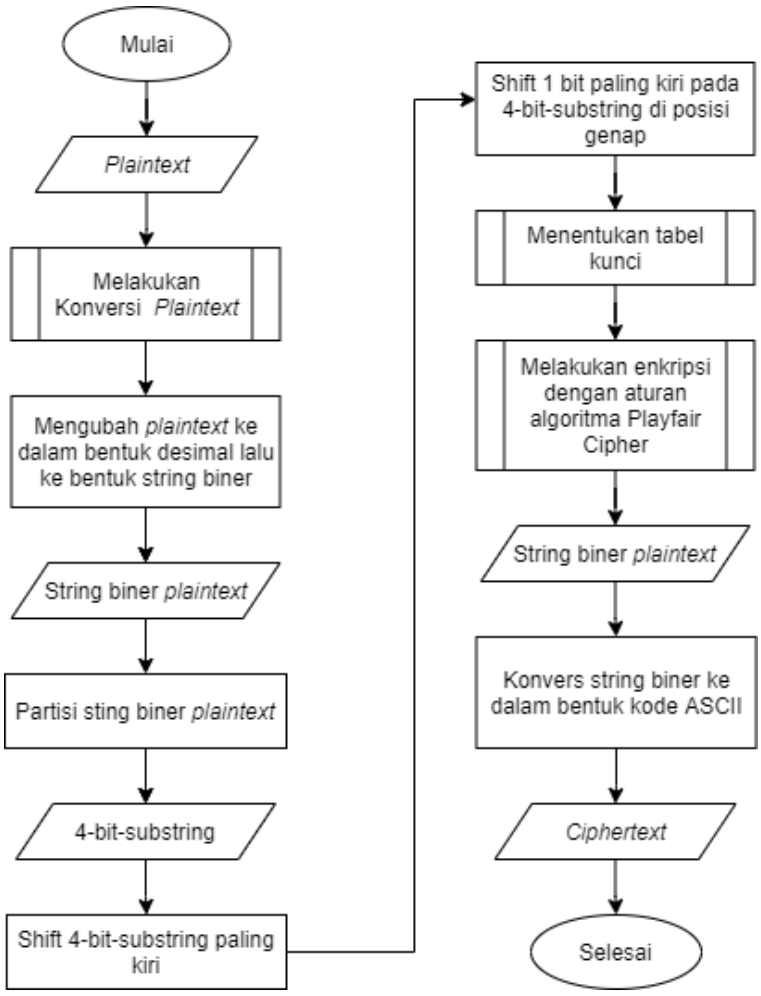
Pada penelitian ini dilakukan enkripsi pesan menggunakan algoritma Playfair Cipher dengan konversi kode ASCII – biner. Proses enkripsi dilakukan pada string biner setiap karakter sehingga menghasilkan *ciphertext* yang berbeda-beda untuk setiap karakter yang bersebelahan. Pengunaan kode ASCII menghasilkan kombinasi karakter pada *ciphertext* lebih beragam.

3.2 Rancangan Sistem

Penelitian ini memiliki rancangan algoritma yang dibagi menjadi dua yaitu proses enkripsi dan proses dekripsi.

3.2.1 Proses Enkripsi

Pada penelitian ini, rancangan algoritma untuk proses enkripsi digambarkan menggunakan diagram alir (*flowchart*). Seperti pada Gambar 3.1.



Gambar 3.1 *Flowchart* proses enkripsi

Dari *flowchart* di atas dapat dilihat bahwa tahapan proses enkripsi adalah sebagai berikut :

i) Menentukan *Plaintext*

Plaintext berupa pesan teks.

ii) Enkripsi *Plaintext*

Tahapan proses enkripsi *plaintext* adalah sebagai berikut :

1. *Plaintext* di terjemahkan dalam kode ASCII.
2. Kode ASCII dikonversi kedalam string biner dengan ukuran 8 bit.
3. Partisi 8 bit biner hasil konversi menjadi dua bagian (substring), masing-masing terdiri dari 4 bit biner.
4. *Shift* 4 bit paling kiri pada *plaintext*.
5. *Shift* 1 bit paling kiri pada tiap 4-bit-substring di posisi genap.
6. Penentuan tabel kunci *Playfair* yang digunakan.
7. Lakukan enkripsi dengan tabel *Playfair Cipher*.
8. Konversi string biner menjadi karakter ASCII. Untuk karakter yang tidak tercetak akan ditampilkan dalam bentuk heksadesimal dari karakter tersebut dengan penambahan spasi pada awal dan akhir nama karakter.

   Contoh : [spasi]0x1[spasi], [spasi]0xFF[spasi].

Berikut merupakan contoh penyelesaian pada proses enkripsi.

Plaintext : **IF ITERA**

Konversi *Plaintext* kedalam bentuk desimal kode ASCII. Lalu kode ASCII dikonversi kedalam string biner dengan ukuran 8 bit seperti pada Tabel 3.1

Tabel 3.1 Konversi *plaintext*

Huruf *plaintext*	**Desimal**	**Biner**
I	73	01001001
F	70	01000110
spasi	32	00100000
I	73	01001001
T	84	01010100
E	69	01000101
R	82	01010010
A	65	01000001

Partisi biner menjadi dua bagian sehingga dihasilkan 4-bit-substring pada setiap karakter seperti pada Tabel 3.2.

Tabel 3.2 Partisi string biner *plaintext*

I	F	Spasi	I	T
0100 1001	0100 0110	0010 0000	0100 1001	0101 0100

E	R	A
0100 0101	0101 0010	0100 0001

Susunan angka biner dari *plaintext* :

0100 1001 0100 0110 0010 0000 0100 1001 0101 0100 0100 0101 0101 0010 0100 0001

Shift 4-bit-substring paling kiri ke ujung paling kanan *plaintext* :

1001 0100 0110 0010 0000 0100 1001 0101 0100 0100 0101 0101 0010 0100 0001 0100

Pada Tabel 3.3 dilakukan *Shift* 1 bit paling kiri pada 4-bit-substring yang berada di posisi genap (digaris bawah). Posisi genap dihitung dari sisi kiri susunan 4-bit-substring *plaintext*.

1001 0100 0110 0010 0000 0100 1001 0101 0100 0100 0101 0101 0010 0100 0001 0100

Tabel 3.3 *Shift* pada 4-bit-substring posisi genap *plaintext*

4-bit-substring posisi genap	**4-bit-substring setelah pemindahan bit paling kiri**
0100	1000
0010	0100
0100	1000
0101	1010
0100	1000

4-bit-substring posisi genap	4-bit-substring setelah pemindahan bit paling kiri
0101	1010
0100	1000
0100	1000

Susun kembali *plaintext* dengan 4-bit-substring posisi genap yang baru. Sebagai berikut :

1001 1000 0110 0100 0000 1000 1001 1010 0100 1000 0101 1010 0010 1000 0001 1000

Lakukan enkripsi dengan *Playfair Cipher*. Tabel kunci merupakan bujur sangkar berukuran 4×4 sehingga terdapat 16 blok. Isi setiap blok berupa kode biner sebanyak 4 bit yang disusun berdasarkan permutasi dari 16 angka tersebut (0 sampai 15)

Dipilih : tabel kunci dengan susunan permutasi = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15}. Dengan kode biner setiap angka seperti pada Tabel 3.4.

Tabel 3.4 Konversi desimal kunci

Desimal	Kode biner
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010

Desimal	Kode biner
11	1011
12	1100
13	1101
14	1110
15	1111

Tabel 3.5 merupakan tabel *Playfair Cipher* yang di akan digunakan. Pada contoh ini permutasi yang digunakan yaitu = {1, 3, 2, 12, 6, 7, 9, 11, 0, 8, 5, 13, 4, 15, 10, 14}

Tabel 3.5 Tabel kunci

0001	0011	0010	1100
0110	0111	1001	1011
0000	1000	0101	1101
0100	1111	1010	1110

Lakukan enkripsi dengan tabel *Playfair cipher* yang sudah dibuat. Dengan aturan sebagai berikut :

1. Jika dua buah 4-bit-substring berada pada baris bujursangkar yang sama maka tiap 4-bit-substring diganti dengan 4-bit-substring di kanannya.
2. Jika dua buah 4-bit-substring berada pada kolom bujursangkar yang sama maka tiap 4-bit-substring diganti dengan 4-bit-substring di bawahnya.
3. Jika dua buah 4-bit-substring tidak berada pada baris yang sama atau kolom yang sama, maka 4-bit-substring pertama diganti dengan 4-bit-substring pada perpotongan baris 4-bit-substring pertama dengan kolom 4-bit-substring kedua. 4-bit-substring kedua diganti dengan 4-bit-substring pada titik sudut keempat dari persegi panjang yang dibentuk dari tiga buah 4-bit-substring yang digunakan.
4. Jika 4-bit-substring pertama dan 4-bit-substring kedua sama yaitu yang memiliki string biner 00000000 dan 11111111 hasil enkripsi akan tetap/tidak berubah.

Sehingga susunan bit-substring *ciphertext* adalah sebagai berikut :

0111 0101 0000 0001 1000 0101 0101 0010 1111 0000 1010 0010 0011 0101 0011 0000

Gabung kembali dua buah 4-bit-substring biner menjadi string biner dengan panjang 8 bit. Lalu lakukan konversi pada 8 bit string biner ke bentuk karakter ASCII seperti pada Tabel 3.6.

Untuk karakter ASCII yang tidak tercetak maka digunakan spasi sebagai pemisah antara karakter yang dapat tercetak dengan karakter yang tidak dapat dicetak. Hal tersebut bertujuan agar saat *ciphertext* didekripsi kembali, tidak terjadi kesalahan saat proses konversi karakter ke string biner.

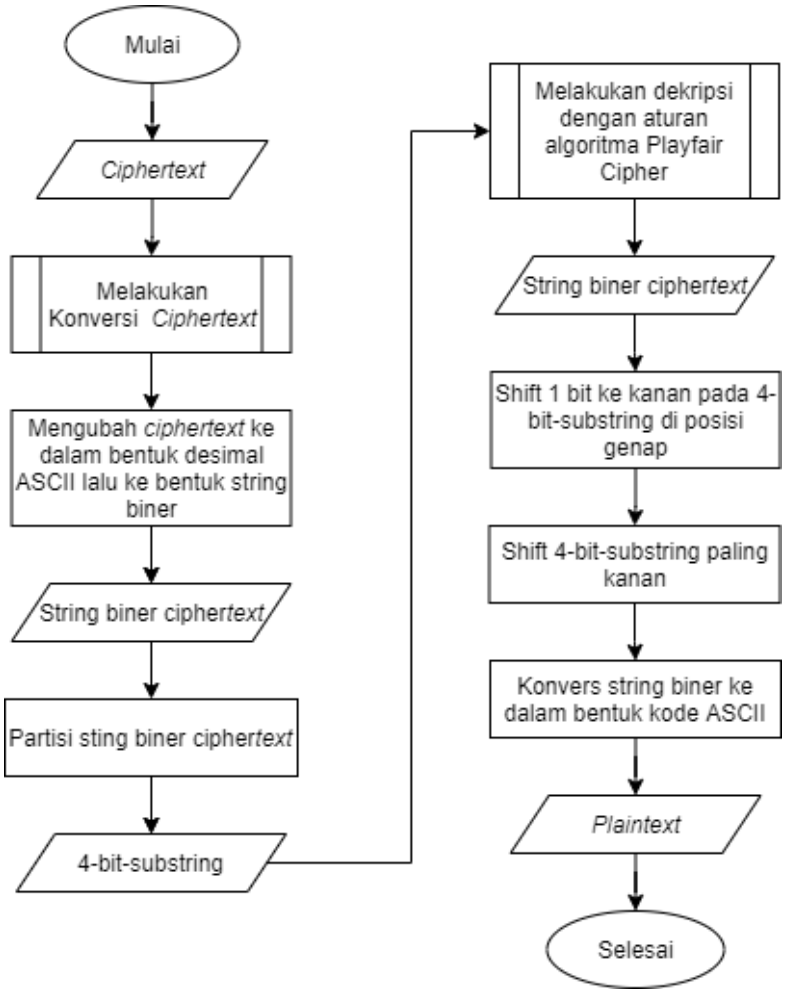
Tabel 3.6 Konversi *ciphertext*

Biner *Ciphertext*	**Desimal**	**Huruf *ciphertext***
01110101	117	U
00000001	1	0x1
10000101	133	0x85
01010010	82	R
11110000	240	Đ
10100010	162	¢
00110101	53	5
00110000	48	0

Ciphertext yang didapatkan : **u 0x1 0x85 Rð¢50**

3.2.2 Proses Dekripsi

Pada penelitian ini, rancangan algoritma untuk proses dekripsi digambarkan menggunakan diagram alir (*flowchart*). Seperti pada Gambar 3.2.



Gambar 3.2 *Flowchart* proses dekripsi

Dari *flowchart* dapat dilihat bahwa tahapan proses enkripsi sebagai berikut :

iii) Input *Ciphertext*

Ciphertext merupakan hasil enkripsi yang sebelumnya dilakukan.

iv) Dekripsi *Ciphertext*

Tahapan proses enkripsi *ciphertext* adalah sebagai berikut :

1. *Ciphertext* di terjemahkan dalam kode ASCII
2. Kode ASCII dikonversi kedalam string biner dengan ukuran 8 bit biner
3. Partisi 8 bit biner hasil konversi menjadi dua bagian (substring), masing-masing terdiri dari 4 bit biner

4. Dekripsi dengan algoritma *Playfair Cipher* dan kunci yang telah digunakan sebelumnya / saat proses enkripsi.
5. *Shift* 1 bit ke kanan pada tiap 4-bit-substring di posisi genap.
6. *Shift* 4 bit ke kanan pada *ciphertext*
7. Konversi string biner ke karakter ASCII

Berikut merupakan contoh penyelesaian pada proses enkripsi.

Ciphertext : **u 0x1 0x85 Rð¢50**

Konversi *Ciphertext* kedalam bentuk desimal kode ASCII. Lalu kode ASCII dikonversi kedalam string biner dengan ukuran 8 bit seperti pada Tabel 3.7

Tabel 3.7 Konversi *ciphertext*

Huruf *ciphertext*	**Desimal**	**Biner**
U	117	01110101
0x1	1	00000001
0x85	133	10000101
R	82	01010010
ð	240	11110000
¢	162	10100010
5	53	00110101
0	48	00110000

Partisi biner menjadi dua bagian sehingga dihasilkan 4-bit-substring pada setiap karakter seperti pada Tabel 3.8.

Tabel 3.8 Partisi string biner *ciphertext*

U	0x1	0x85	R	ð
0111 0101	0000 0001	1000 0101	0101 0010	1111 0000

¢	5	0
1010 0010	0011 0101	0011 0000

Sehingga didapat string biner *ciphertext* adalah sebagai berikut :
0111 0101 0000 0001 1000 0101 0101 0010 1111 0000 1010 0010 0011 0101 0011 0000

Tabel 3.9 merupakan kunci yang telah ditentukan dan digunakan saat proses enkripsi. Lakukan dekripsi dengan algoritma *Playfair Cipher* menggunakan tabel kunci tersebut.

Tabel 3.9 Tabel kunci

0001	0011	0010	1100
0110	0111	1001	1011
0000	1000	0101	1101
0100	1111	1010	1110

Lakukan enkripsi dengan tabel *Playfair cipher* yang sudah dibuat. Dengan aturan sebagai berikut :

1. Jika dua buah 4-bit-substring berada pada baris bujursangkar yang sama maka tiap 4-bit-substring diganti dengan 4-bit-substring di kirinya.
2. Jika dua buah 4-bit-substring berada pada kolom bujursangkar yang sama maka tiap 4-bit-substring diganti dengan 4-bit-substring di atasnya.
3. Jika dua buah 4-bit-substring tidak berada pada baris yang sama atau kolom yang sama, maka 4-bit-substring pertama diganti dengan 4-bit-substring pada perpotongan baris 4-bit-substring pertama dengan kolom 4-bit-substring kedua. 4-bit-substring kedua diganti dengan 4-bit-substring pada titik sudut keempat dari persegi panjang yang dibentuk dari tiga buah 4-bit-substring yang digunakan.
4. Jika 4-bit-substring pertama dan 4-bit-substring kedua sama yaitu yang memiliki string biner 00000000 dan 11111111 hasil enkripsi akan tetap/tidak berubah.

Sehingga susunan bit-substring *plaintext* adalah sebagai berikut :

1001 1000 0110 0100 0000 1000 1001 1010 0100 1000 0101 1010 0010 1000 0001 1000

Pada Tabel 3.10 dilakukan *Shift* 1 bit paling kanan pada 4-bit-substring yang berada di posisi genap (digaris bawah). Posisi genap dihitung dari sisi kiri susunan seluruh 4-bit-substring *ciphertext*.

1001 1000 0110 0100 0000 1000 1001 1010 0100 1000 0101 1010 0010 1000 0001 1000

Tabel 3.10 *Shift* pada 4-bit-substring posisi genap *ciphertext*

4-bit-substring posisi genap	**4-bit-substring setelah pemindahan bit paling kanan**
1000	0100
0100	0010
1000	0100
1010	0101
1000	0100
1010	0101
1000	0100
1000	0100

Susun kembali *plaintext* dengan 4-bit-substring posisi genap yang baru. Sebagai berikut :

1001 0100 0110 0010 0000 0100 1001 0101 0100 0100 0101 0101 0010 0100 0001 0100

Didapat string *plaintext* sebagai berikut :

1001 0100 0110 0010 0000 0100 1001 0101 0100 0100 0101 0101 0010 0100 0001 0100

Shift 4-bit-substring paling kanan ke ujung paling kiri *plaintext* :

0100 1001 0100 0110 0010 0000 0100 1001 0101 0100 0100 0101 0101 0010 0100 0001

Gabung kembali dua buah 4-bit-substring biner menjadi string biner dengan panjang 8 bit. Lalu lakukan konversi pada 8 bit string biner ke bentuk karakter ASCII seperti pada Gambar 3.11.

Tabel 3.11 Konversi *plaintext*

Biner *Plaintext*	**Desimal**	**Huruf *plaintext***
01001001	73	I
01000110	70	F
00100000	32	spasi
01001001	73	I
01010100	84	T
01000101	69	E
01010010	82	R
01000001	65	A

Didapat kembali *plaintext* : **IF ITERA**

3.3 Rancangan Pengujian

Rancangan pengujian yang akan dilakukan yaitu evaluasi *recovery*, analisis frekuensi dan perbandingan algoritma. Pada pengujian ini, *plaintext* yang digunakan berupa file *text* yang diperoleh dari *website* http://textfiles.com/stories/ dan http://textfiles.com/programming/.

3.3.1 Evaluasi *Recovery*

Evaluasi *recovery* dilakukan dengan mengevaluasi hasil pesan *ciphertext* yang didapat dari proses dekripsi. Evaluasi tersebut dilakukan untuk mengetahui apakah hasil dekripsi *ciphertext* dapat kembali ke pesan semula (*plaintext*). Pengujian ini dilakukan untuk mengetahui keefektifan kinerja sistem dalam melakukan proses enkripsi dan dekripsi.

Pada pengujian ini, *plaintext* berisi karakter berupa huruf, spasi dan karakter lainnya sebanyak 10 *plaintext* yang diperoleh dari website http://textfiles.com/programming/ dan http://textfiles.com/programming/. Kunci

yang digunakan yaitu permutasi biner dari angka 1, 3, 2, 12, 6, 7, 9, 11, 0, 8, 5, 13, 4, 15, 10, 14.

3.3.2 Analisis Frekuensi

Analisis frekuensi dilakukan untuk mengetahui karakter mana yang sering muncul pada *ciphertext* yang dihasilkan. Dalam pengujian ini akan diselidiki ada berapa frekuensi kemunculan tiap huruf.

Pada pengujian ini, digunakan plaintext yang diperoleh dari *website* http://textfiles.com/programming/ dengan judul "Fast Lane Modems (14.4k! YEAH!)". *Plaintext* tersebut berukuran 7948 berisi karakter berupa huruf, spasi dan karakter lainnya. Kunci yang digunakan yaitu permutasi biner dari angka 1, 3, 2, 12, 6, 7, 9, 11, 0, 8, 5, 13, 4, 15, 10, 14.

Dalam pengujian ini, dilakukan perhitungan nilai variansi kemunculan karakter pada *plaintext* dan *ciphertext*. Variansi digunakan untuk menganalisa hubungan secara statistik kemunculan karakter pada *plaintext* dan *ciphertext*. Karl Pearson merumuskan pengukuran varians sebagai [13] :

$$s^2 = \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2$$

Dengan :

s^2 = nilai varian

n = jumlah sampel

$\sum$ = jumlah

X_i = nilai data ke-i

$\bar{X}$ = nilai rata-rata data

3.3.3 Perbandingan Algoritma

Perbandingan algoritma dilakukan dengan membandingkan hasil analisis frekuensi pada *ciphertext* hasil enkripsi dengan algoritma *Playfair Cipher* biasa atau algoritma *Playfair Cipher* tanpa modifikasi dan algoritma *Playfair Cipher* yang telah dimodifikasi. Hal tersebut bertujuan untuk mengetahui apakah

algoritma yang sudah dimodifikasi menghasilkan hasil *ciphertext* yang lebih beragam daripada algoritma sebelumnya.

Pada pengujian ini, digunakan *plaintext* berukuran 9474 karakter. Plaintext diperoleh dari website http://textfiles.com/stories/ dengan judul “The Story of Jack and the Beanstalk”. Untuk proses enkripsi dengan Algoritma Playfair biasa *plaintext* akan diproses terlebih dahulu dengan menghilangkan karakter selain huruf alfabet seperti karakter spasi dan karakter lainnya (titik, koma, dll). Hal tersebut dilakukan karena algoritma *Playfair Cipher* yang belum dimodifikasi hanya dapat memproses *plaintext* berupa huruf alfabet yaitu A sampai Z.

3.3.4 Pengujian Komputasi

Pengujian komputasi dilakukan pada dua aspek yaitu komputasi waktu dan penggunaan *resource* dari segi ruang penyimpanan atau *hardisk.*

Pengujian komputasi waktu dilakukan dengan cara membandingkan waktu komputasi pada masing-masing proses enkripsi maupun dekripsi dengan algoritma *Playfair Cipher* biasa atau algoritma *Playfair Cipher* tanpa modifikasi dan algoritma *Playfair Cipher* ASCII – Biner. Sedangkan pengujian penggunaan *resource* dilakukan dengan cara membandingkan penggunaan *resource* dari segi ruang penyimpanan atau *hardisk* pada ukuran pesan *ciphertext* yang dihasilkan dari proses enkripsi dengan algoritma *Playfair Cipher* biasa atau algoritma *Playfair Cipher* tanpa modifikasi dan algoritma *Playfair Cipher* ASCII – Biner.