BAB III
METODE PENELITIAN

3.1 Identifikasi Masalah

Dalam penelitian ini, pengamanan akan dilakukan terhadap pesan teks dengan menggunakan *Present Cipher*. Pada *Present Cipher*, plainteks dan kunci dapat berupa angka, huruf, maupun tanda baca. Sedangkan cipherteks berupa karakter heksadesimal. Oleh karena itu, kriptosistem dari kriptografi *Present Cipher* untuk pengamanan pesan teks adalah sebagai berikut:

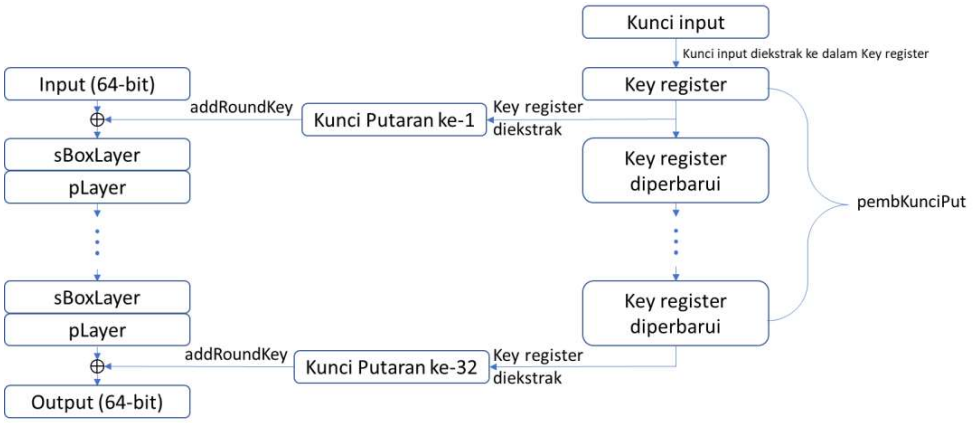
1. $\mathcal{P}$ himpunan plainteks berupa angka, huruf, atau tanda baca (*printable character*)
2. $\mathcal{C}$ himpunan cipherteks berupa karakter heksadesimal
3. $\mathcal{K}$ himpunan kunci berupa angka, huruf, atau tanda baca
4. Untuk setiap $K \in \mathcal{K}$, terdapat sebuah aturan enkripsi $e_K \in \mathcal{E}$ dan
5. sebuah aturan dekripsi yang sesuai $d_k \in \mathcal{D}$. Setiap $e_K: \mathcal{P} \rightarrow \mathcal{C}$ dan $d_k: \mathcal{C} \rightarrow \mathcal{P}$ adalah fungsi-fungsi sedemikian sehingga $d_k(e_K(p)) = p$ untuk setiap anggota plainteks $p \in \mathcal{P}$.

3.2 Model Dasar (Bogdanov, dkk., 2007)

Algoritma enkripsi dan dekripsi *Present Cipher* terdiri dari pembangkitan kunci putaran ($generateRoundKey$), operasi XOR ($addRoundKey$), substitusi ke dalam sbox ($sBoxLayer$), dan permutasi ($pLayer$). Berikut skema enkripsi dan dekripsi *Present Cipher*:

3.2.1 Skema Enkripsi *Present Cipher*

Enkripsi *Present Cipher* bertujuan untuk mengamankan pesan berupa plainbit sepanjang 64-bit dengan kuncibit 80 atau 128-bit. Hasil dari enkripsi ini merupakan cipherbit sepanjang 64-bit yang berbeda dari pesan asalnya. Skema enkripsi *Present Cipher* ini menjadi pondasi dari skema enkripsi pesan teks. Enkripsi *Present Cipher* ini dapat lebih mudah dipahami jika diilustrasikan dengan skema berikut:

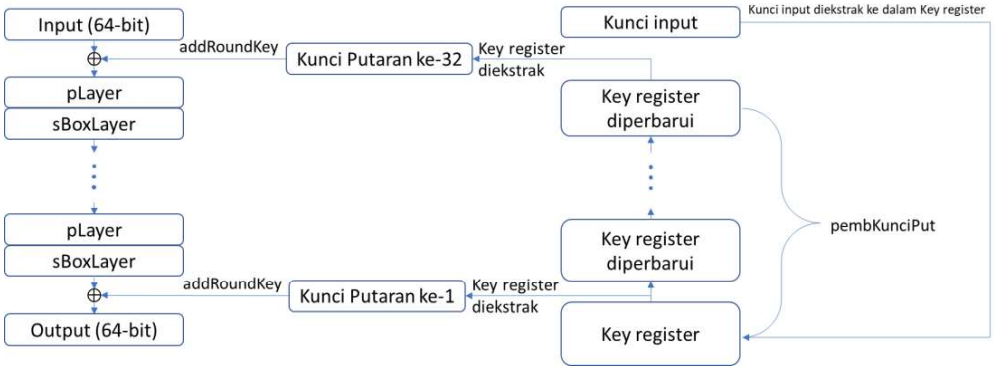


Gambar 3.1 Skema Enkripsi *Present Cipher*

Enkripsi *Present Cipher* dimulai dengan fungsi *Add Round Key* antara state dengan kunci putaran pertama. Setelah itu, state melalui fungsi *S-Box* dan *Permutation Layer.* Proses state melalui 3 fungsi tersebut diulangi sebanyak 31 satu kali. Setelah itu, state dan kunci putaran ke-32 melalui fungsi *Add Round Key* sehingga menghasilkan output.

3.2.2 Skema Dekripsi *Present Cipher*

Dekripsi *Present Cipher* bertujuan untuk mendapatkan pesan berupa plainbit dengan mengembalikan cipherbit ke bentuk plainbitnya menggunakan kuncibit sepanjang 80 atau 128-bit. Skema dekripsi *Present Cipher* ini menjadi pondasi dari skema dekripsi pesan teks. Dekripsi *Present Cipher* dapat deperoleh dengan membalik proses enkripsi *Present Cipher*. Hal ini dapat terjadi karena *Present Cipher* termasuk kedalam blok *cipher* skema SPN. Dekripsi *Present Cipher* ini dapat lebih mudah dipahami jika diilustrasikan dengan skema berikut:



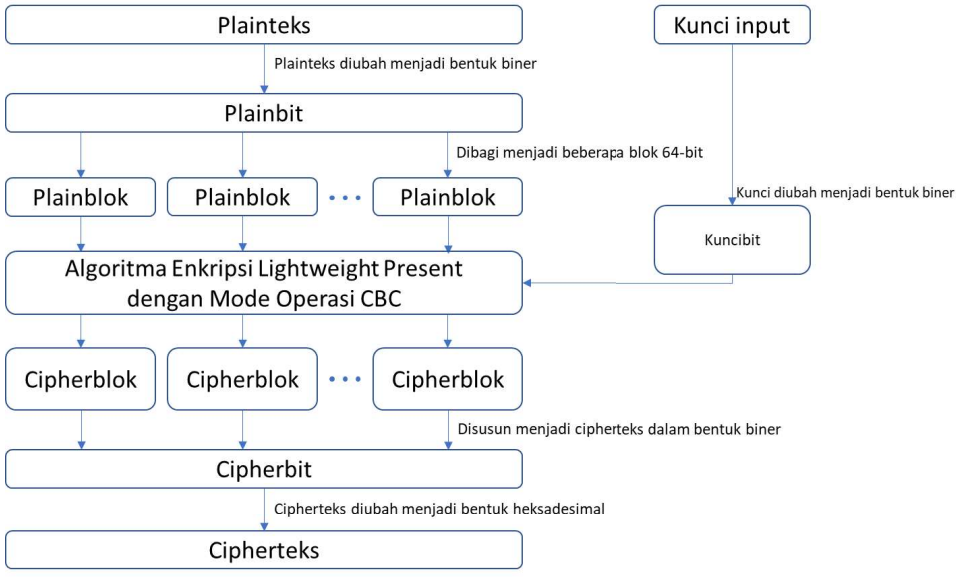
Gambar 3.2 Skema Dekripsi *Present Cipher*

3.3 Pengembangan Model

Present Cipher membutuhkan *input* dan *output* berupa blok bit sepanjang 64-bit. Sedangkan pesan teks yang umumnya digunakan merupakan huruf alfabet, angka, simbol atau tanda baca. Oleh karena itu, suatu plainteks dan kunci *input* harus diubah terlebih dahulu menjadi beberapa blok byte agar dapat dienkripsi dengan *Present Cipher*. Berikut skema enkripsi dan dekripsi *Present Cipher* untuk pengamanan pesan teks.

3.3.1 Skema Enkripsi *Present Cipher* untuk Pengamanan Pesan Teks

Setelah plainteks dan kunci input diubah menjadi plainbit dan kuncibit, plainbit tersebut dibagi menjadi beberapa plainblok yang kemudian setiap plainbloknya dienkripsi oleh algoritma *Present Cipher* dengan Mode Operasi CBC. Oleh karena itu, algoritma *Present Cipher* akan diulangi sebanyak plainbloknya dengan kunci input yang sama. Cipherblok yang sudah diperoleh disusun sehingga membetuk suatu bitstring yang disebut dengan cipherbit. Setelah itu, cipherbit diubah ke bentuk heksadesimal. Enkripsi *Present Cipher* untuk Pengamanan Pesan Teks ini dapat lebih mudah dipahami jika diilustrasikan dengan skema berikut:

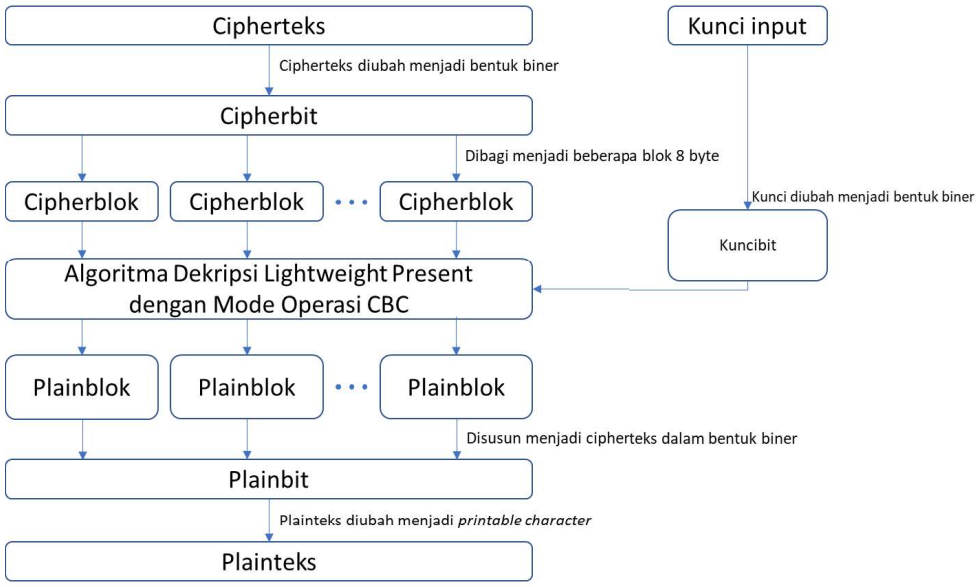


Gambar 3.3 Skema Enkripsi *Present Cipher* untuk Pengamanan Pesan Teks

3.3.2 Skema Dekripsi *Present Cipher* untuk Pengamanan Pesan Teks

Dekripsi *Present Cipher* untuk pengamanan pesan teks dapat dipandang sebagai enkripsi *Present Cipher* untuk pengamanan pesan teks yang prosesnya

"dibalik". Oleh karena itu, dekripsi *Present Cipher* untuk pengamanan pesan teks dimulai dengan mengubah cipherteks menjadi cipherbit dan diakhiri dengan plainbit yang diubah menjadi *printable character*. Dekripsi *Present Cipher* untuk pengamanan pesan teks ini dapat lebih mudah dipahami jika diilustrasikan dengan skema berikut:



Gambar 3.4 Skema Dekripsi *Present Cipher* untuk Pengamanan Pesan Teks

3.4 Konstruksi Program

Pada bagian ini menjelaskan mengenai algoritma yang berjalan pada program yang akan dibuat

3.4.1 Algoritma Deskriptif

a. Enkripsi Pesan

1. Pesan dan kunci *input* masing-masing diubah menjadi plainbit dan kuncibit.
2. Dilakukan pembangkitan kunci putaran
   - Pembangkitan kunci putaran dimulai dengan mengekstrak kunci *input* ke dalam kunci register *K*
   - Lalu, kunci register *K* akan diperbarui oleh suatu algoritma dan diekstrak menjadi kunci putaran dengan mengambil 64-bit paling

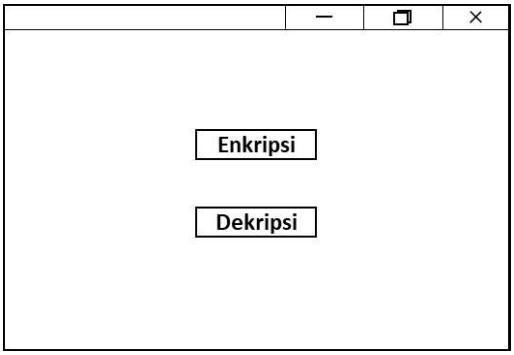
kiri dari kunci register *K* di putaran tersebut. Langkah ini diulangi sebanyak 32 kali sehingga menghasilkan 32 kunci putaran

3. Plainbit dibagi menjadi beberapa blok 64-bit.
    - Jika terdapat suatu blok dengan panjang kurang dari 64-bit maka dilakukan *padding* pada blok tersebut.
    - Setiap blok dienkripsi satu persatu menggunakan *Present Cipher* dengan kunci *input* yang sama sehingga menghasilkan cipherblok
4. Setelah setiap plainblok dienkripsi sehingga menghasilkan cipherblok, cipherblok-cipherblok tersebut disusun sesuai urutan enkripsi sehingga membentuk cipherbit yang berikutnya diubah menjadi heksadesimal. Rangkaian karakter heksadesimal ini disebut dengan cipherteks.

b. Dekripsi Pesan

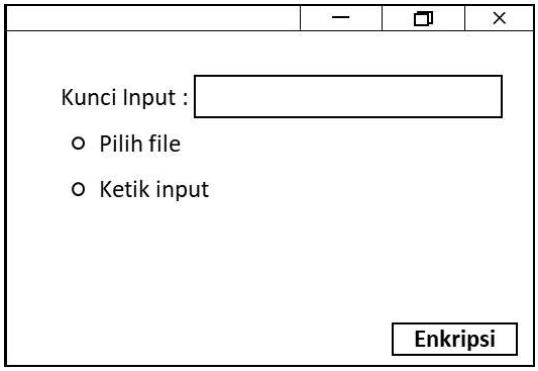
1. Cipherteks dan kunci *input* masing-masing diubah menjadi cipherbit dan kuncibit.
2. Dilakukan pembangkitan kunci putaran.
    - Pembangkitan kunci putaran dimulai dengan mengekstrak kunci *input* ke dalam kunci register *K*
    - Lalu, *register K* akan diperbarui oleh suatu algoritma dan diekstrak menjadi kunci putaran dengan mengambil 64 bit paling kiri dari kunci register *K* di putaran tersebut. Langkah ini diulangi sebanyak 32 kali sehingga menghasilkan 32 kunci putaran.
3. Cipherteks dibagi menjadi beberapa blok 64-bit.
    - Jika terdapat suatu blok dengan panjang kurang dari 64-bit maka dilakukan padding pada blok tersebut.
    - Setiap blok didekripsi satu persatu menggunakan *Present Cipher* dengan kunci *input* yang sama sehingga menghasilkan plainblok.
4. Setelah setiap cipherblok didekripsi sehingga menghasilkan plainblok, plainblok-plainblok tersebut disusun sesuai urutan dekripsi sehingga membentuk plainbit yang berikutnya diubah menjadi *printable character*. Rangkaian karakter ini disebut dengan plaintek.

3.3.3 Desain Tampilan

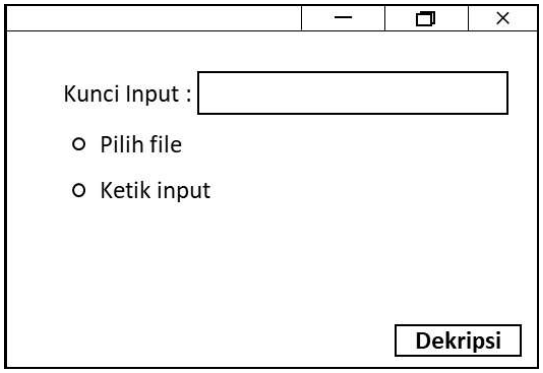


Gambar 3.5 Jendela Utama

Terdapat dua tombol pada jendela utama. Tombol ‘Enkripsi’ dan ‘Dekripsi’ berturut-turut berfungsi untuk menampilkan jendela berikut:

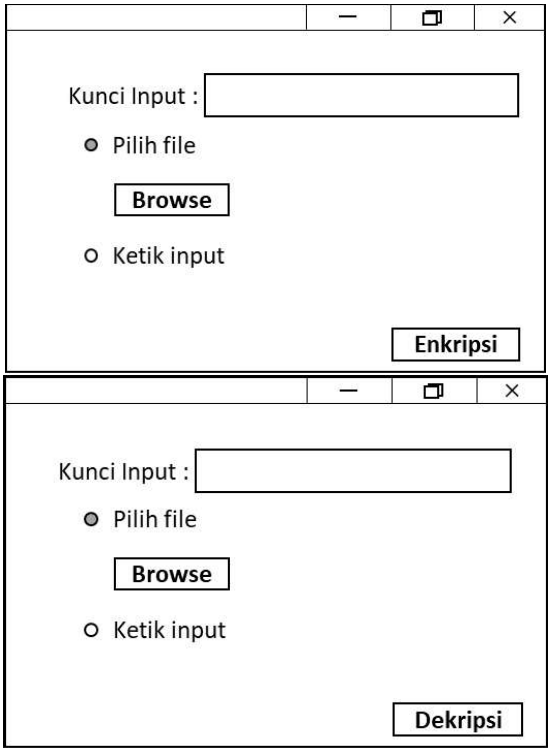


Gambar 3.6 Jendela Kedua Bagian Enkripsi



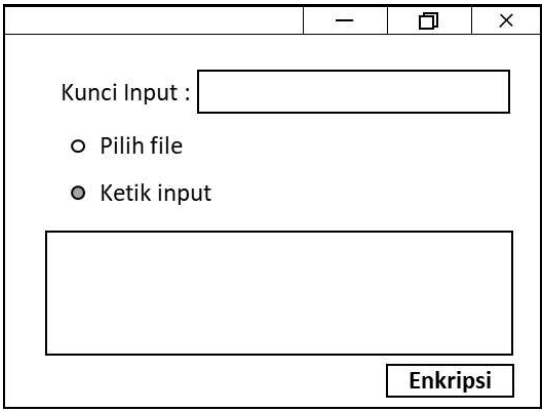
Gambar 3.7 Jendela Kedua Bagian Dekripsi

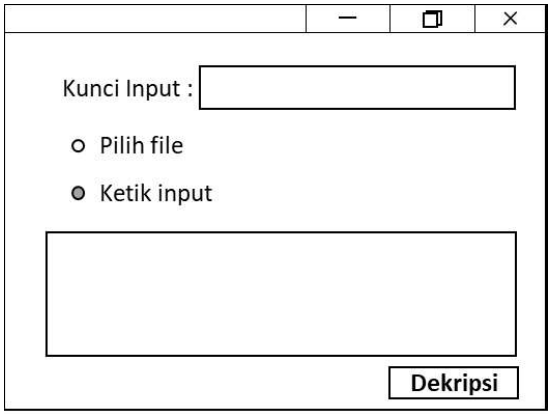
Pada jendela ke-2, dapat dipilih salah satu ‘Radio Button’ yaitu ‘Pilih file’ atau ‘Ketik *input*’. Jika dipilih ‘Pilih file’ maka jendela akan berubah menjadi sebagai berikut:



Gambar 3.8 Jendela Kedua 'Pilih file'

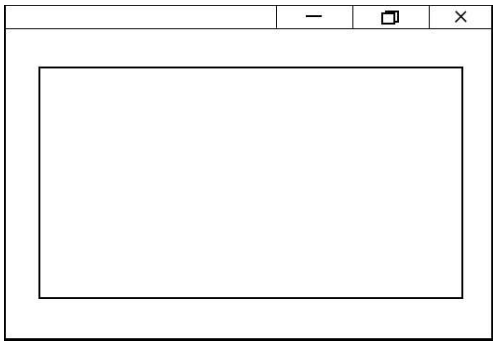
Sedangkan, jika dipilih 'Ketik *input*' maka jendela akan berubah sebagai berikut:



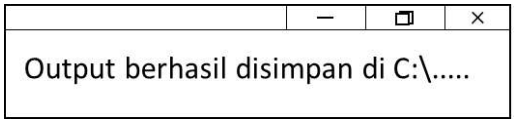


Gambar 3.9 Jendela Kedua 'Ketik *input*'

Jika tombol 'Enkripsi' atau 'Dekripsi' ditekan setelah memilih salah satu 'Radio Button' maka akan menghasilkan tampilan sebagai berikut:



Gambar 3.10 Jendela *Output*



Gambar 3.11 Jendela Notifikasi *Output*

3.3.4 Library Python

- Tkinter

  Kedua library tersebut akan digunakan untuk membuat User Interface program.

- Math

  Library ini akan digunakan untuk operasi pembulatan.

- Pyinstaller

  Pyinstaller digunakan untuk *compile* program sehingga dapat digunakan di komputer yang tidak memiliki program python.

3.5 Proses Validasi

Pada tahap ini dilakukan validasi menggunakan Excel terhadap *output* aplikasi yang dikembangkan. Validasi program dilakukan dengan membandingkan langkah demi langkah yang dilakukan oleh program dengan yang dilakukan oleh Excel. Jika hasilnya sama maka program dapat dikatakan sudah tervalidasi.

3.6 Pengambilan Kesimpulan

Jika *output* program sama dengan *output* Excel di setiap tahapnya maka program dapat dikatakan telah divalidasi. Setelah program divalidasi maka program dapat digunakan untuk mengamankan pesan teks.