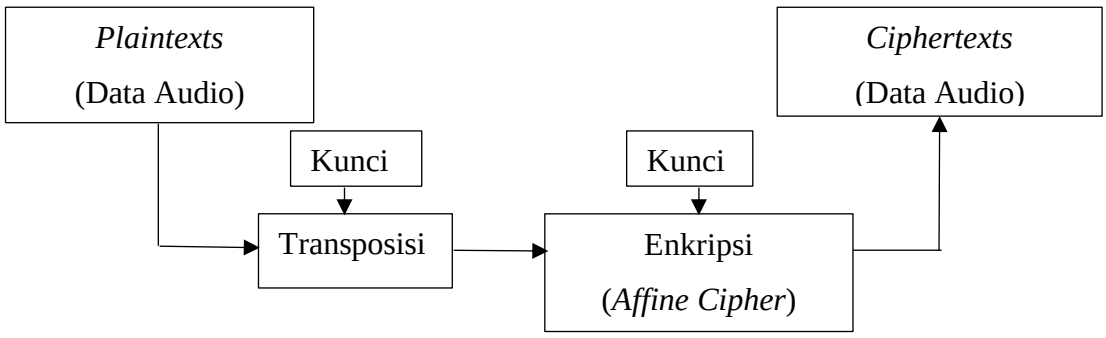
BAB III
METODOLOGI PENELITIAN

Metode yang digunakan dalam penulisan penelitian ini adalah studi literatur dan pengembangan model, serta pengujiannya dalam program aplikasi yang akan diuraikan secara rinci dalam langkah-langkah sebagai berikut:

3.1. Identifikasi Masalah

Keamanan serta kerahasiaan komunikasi sangatlah penting dalam era informasi ini. Kriptografi hadir untuk membantu menyelesaikan hal tersebut menjadi mungkin. Dengan kriptografi ini, pengirim dapat menjamin keamanan dan kerahasiaan pesan yang akan dikirim kepada penerima tanpa diketahui oleh penyadap. Selama bertahun-tahun kriptografi telah diterapkan namun sebagian besar hanya diterapkan pada file teks saja, Sebagian kecilnya diterapkan pada file multimedia lainnya seperti file audio. Dengan mengamankan file audio menggunakan kriptografi, membuat suara yang dihasilkan setelah proses enkripsi menjadi tidak dapat dikenali oleh penyadap. File audio asli dapat diperoleh dengan melakukan proses dekripsi pada file audio yang telah dienkripsi menggunakan kunci yang telah disepakati. Salah satu cara mengamankan file audio yaitu dengan melakukan transposisi pada data audio menggunakan pembangkit bilangan acak semu *Blum Blum Shub*, tetapi untuk meningkatkan keamanan diperlukan teknik enkripsi substitusi pada data audio tersebut. Salah satu algoritma kriptografi yaitu *Affine Cipher* dapat diterapkan pada kriptografi audio ini untuk melakukan proses enkripsi substitusi. Namun *Affine Cipher* merupakan kriptografi klasik yang keamanannya kurang baik, maka diperlukan pengembangan pada *Affine Cipher* untuk meningkatkan keamanan serta kerahasiaan file audio. Pembangkit barisan bilangan acak dapat digunakan untuk mengembangkan *Affine Cipher,* salah satunya yaitu algoritma pembangkit bilangan acak semu *Blum Blum Shub.* Proses mengamankan file audio dapat diilustrasikan sebagai berikut:



Gambar 3. *1* Proses Enkripsi File Audio

3.2. Transposisi

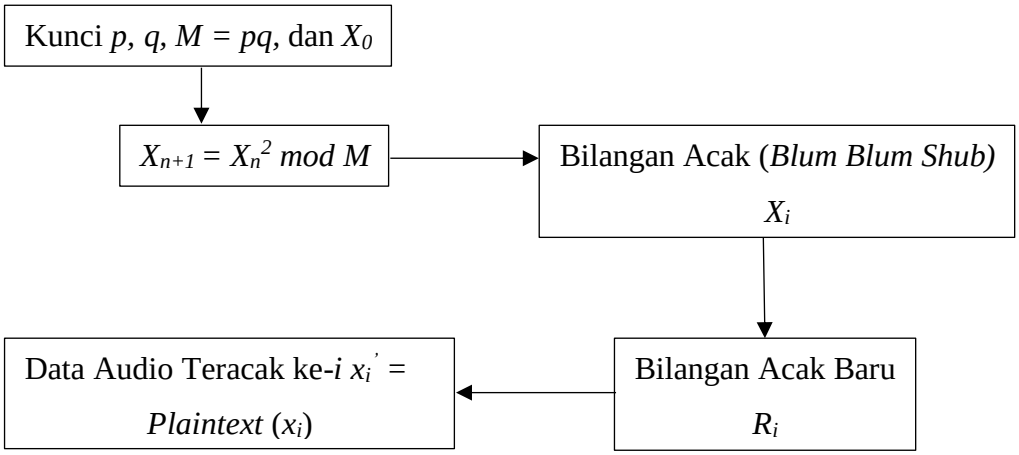
Teknik transposisi dilakukan untuk mengacak data audio pada file audio sehingga suara yang dihasilkan tidak dapat dimengerti oleh penyadap. Dalam proses ini penulis menggunakan pembangkit bilangan acak semu *Blum Blum Shub* untuk mengacak urutan data audio. Rumus untuk membentuk bilangan acak dari algoritma *Blum Blum Shub* ini adalah $X_{n+1} = X_n^2 \ mod \ M$ dengan M merupakan perkalian dua bilangan prima yang besar p dan q yang memenuhi $p \equiv q \equiv 3 \ mod \ 4$. Algoritma ini memerlukan nilai awal yaitu X_0 yang relatif prima dengan M sehingga menghasilkan nilai yang beragam.

Transposisi yang dilakukan yaitu dengan menukar nilai-nilai data audio sehingga urutannya teracak, maka dari itu dibutuhkan barisan bilangan acak sebanyak n dengan n yaitu banyaknya data audio. Misalkan bilangan acak yang dihasilkan oleh algoritma *Blum Blum Shub* yaitu $X_1, X_2, X_3, \ldots, X_n$ yang memiliki nilai berbeda satu sama lain dalam modulus M. Penulis menggunakan metode ranking untuk mendapatkan nilai yang memenuhi $0 < X_i \leq n$, $i = 1,2,3,\ldots,n$ dari bilangan acak yang diperoleh. Misalkan R_i merupakan urutan nilai X_i dari yang terkecil hingga yang terbesar, maka diperoleh bilangan acak $R_1, R_2, R_3, \ldots, R_n$ yang memenuhi $0 < R_i \leq n$, $i = 1,2,3,\ldots,n$ dan R_i merupakan nilai yang berbeda satu sama lainnya. Setelah itu, barulah proses transposisi dapat dilakukan dengan mengurutkan data audio sesuai bilangan acak tersebut. Sehingga diperoleh data audio baru yang telah diacak sebagai berikut:

Tabel 3. 1
Proses Transposisi Data Audio

Data Audio ke	**Data Audio Teracak**
$x_1{}'$	x_{R_1}
$x_2{}'$	x_{R_2}
$x_3{}'$	x_{R_3}
⋮	⋮
$x_n{}'$	x_{R_n}

Sehingga dari pemaparan di atas, dapat ditentukan banyaknya kemungkinan urutan data audio yang mungkin sebesar *n!* dengan *n* banyaknya data audio. Hal ini merupakan keamanan yang cukup tinggi karena file audio memiliki sebesar 44100 data audio per detik. Proses transposisi dapat diilustrasikan sebagai berikut:



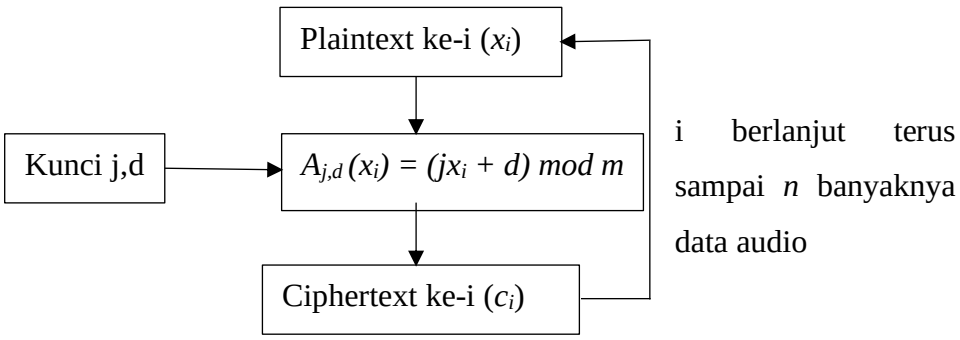
Gambar 3. *2* Skema Transposisi Menggunakan Algoritma *Blum Blum Shub*

3.3. Model Dasar

Teknik enkripsi diperlukan untuk mengamankan nilai-nilai data audio yang sebenarnya namun selain itu enkripsi pada file audio juga akan memberikan *noise* atau gangguan suara sehingga akan menyamarkan suara aslinya agar terdengar tidak jelas. Bermacam-macam kriptografi dapat diterapkan pada file audio, disini

penulis menggunakan kriptografi klasik *Affine Cipher* sebagai model dasar. Algoritma *Affine Cipher* merupakan jenis sandi substitusi *monoalphabetic*, dimana setiap huruf dalam alfabet dipetakan ke padanan numeriknya, dienkripsi menggunakan fungsi matematika sederahana dan diubah kembali menjadi huruf. Namun pada file audio, data audio sudah berbentuk nilai numerik sehingga cukup melakukan proses enkripsi pada nilai-nilai data audio. Rumus yang digunakan adalah $A_{j,d}(x) = (jx + d) \bmod m$ dimana setiap data audio *x* dienkripsi dengan menggunakan kunci *j* dan *d* serta menggunakan modulus m. Jika dalam alfabet nilai *m* yaitu 26, maka dalam audio memiliki dua nilai tergantung *bit* yang digunakan pada file audio.

Analisis keamanan dapat dilakukan dengan menghitung banyaknya pasangan kunci yang mungkin untuk mengenkripsi. Pada file audio 8 *bit* maka nilai *m* yaitu $2^0 + 2^1 + 2^2 + 2^3 + 2^4 + 2^5 + 2^6 + 2^7 = 255$. Karena nilai *j* harus relatif prima dengan *m* maka banyaknya nilai yang relatif prima dengan 255 adalah 127. Dengan banyaknya pergeseran atau nilai *d* yang mungkin pada modulus 255 adalah 255, maka banyaknya pasangan kunci *j,d* yang mungkin pada file audio 8 *bit* adalah 127 x 255 = 32385. Namun pada file audio 16 *bit* nilai *m* yaitu $2^0 + 2^1 + 2^2 + 2^3 + 2^4 + 2^5 + 2^6 + 2^7 + 2^8 + 2^9 + 2^{10} + 2^{11} + 2^{12} + 2^{13} + 2^{14} + 2^{15} = 65.535$. Maka terdapat sebanyak 32.767 kunci *j* yang relatif prima dengan 65.535. Dengan banyaknya pergeseran atau nilai *d* yang mungkin pada modulus 65.535 adalah 65.535, maka banyaknya pasangan kunci *j,d* yang mungkin pada file audio 16 *bit* adalah 32.767 x 65.535 = 2.147.385.345. Dapat terlihat perbedaan keamanan dari banyaknya kemungkinan kunci yang ada pada penggunaan file audio 8 *bit* dan 16 *bit* berbeda sangat jauh, namun tentu ukuran file yang dihasilkan pada file audio 8 *bit* lebih kecil dibandingkan ukuran file audio 16 *bit.*



Gambar 3. *3* Skema *Affine Cipher*

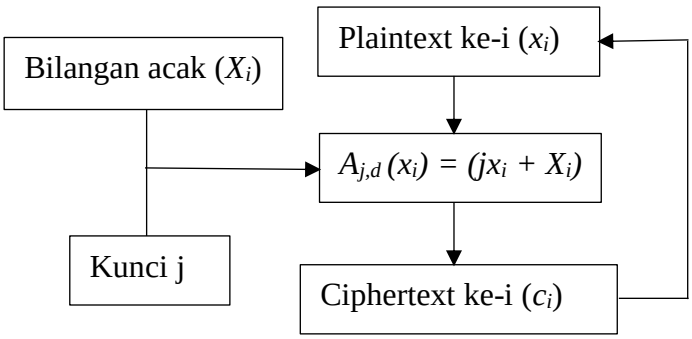
3.4. Pengembangan Model Dasar

Pengembangan algoritma *Affine Cipher* dalam penelitian ini adalah penggunaan algoritma pembangkit bilangan acak semu *Blum Blum Shub* dengan tujuan meningkatkan keamanan pada file audio. Algoritma *Blum Blum Shub* dapat meningkatkan keamanan algoritma *Affine Cipher* dengan memanfaatkan barisan bilangan acak yang tidak dapat diprediksinya. Dengan rumus $X_{n+1} = X_n^2 \ mod \ M$, barisan bilangan acak yang dapat dibentuk sesuai banyaknya data audio pada file audio yang akan dienkripsi. Jika pada awalnya rumus *Affine Cipher* yaitu $A_{j,d}(x) = (jx + d) \ mod \ m$ dengan nilai *j* dan *d* konstan yang telah ditentukan sebelum proses enkripsi dilakukan, maka pada pengembangan *Affine Cipher* ini dilakukan pergantian nilai *d* atau nilai pergeseran dengan barisan bilangan acak yang telah dibentuk dengan algoritma *Blum Blum Shub*. Sehingga rumus yang digunakan untuk setiap data audio ini berbeda yang mengakibatkan keberagaman data audio yang dihasilkan setelah dienkripsi.

Pengembangan algoritma *Affine Cipher* dengan algoritma *Blum Blum Shub* ini dapat diilustrasikan sebagai berikut:

Tabel 3. 2
Rumus Modifikasi Affine Cipher

Data audio	**Rumus *Affine Cipher***
x_1	$(j \times x_1 + X_1) \ mod \ m$
x_2	$(j \times x_2 + X_2) \ mod \ m$
x_3	$(j \times x_3 + X_3) \ mod \ m$
⋮	⋮
x_n	$(j \times x_n + X_n) \ mod \ m$



Gambar 3. *4* Skema Modifikasi *Affine Cipher*

3.5. Konstruksi Program Aplikasi

Adapun prosedur konstruksi program aplikasi dalam penelitian ini yaitu:

3.5.1. Perancangan Program Aplikasi

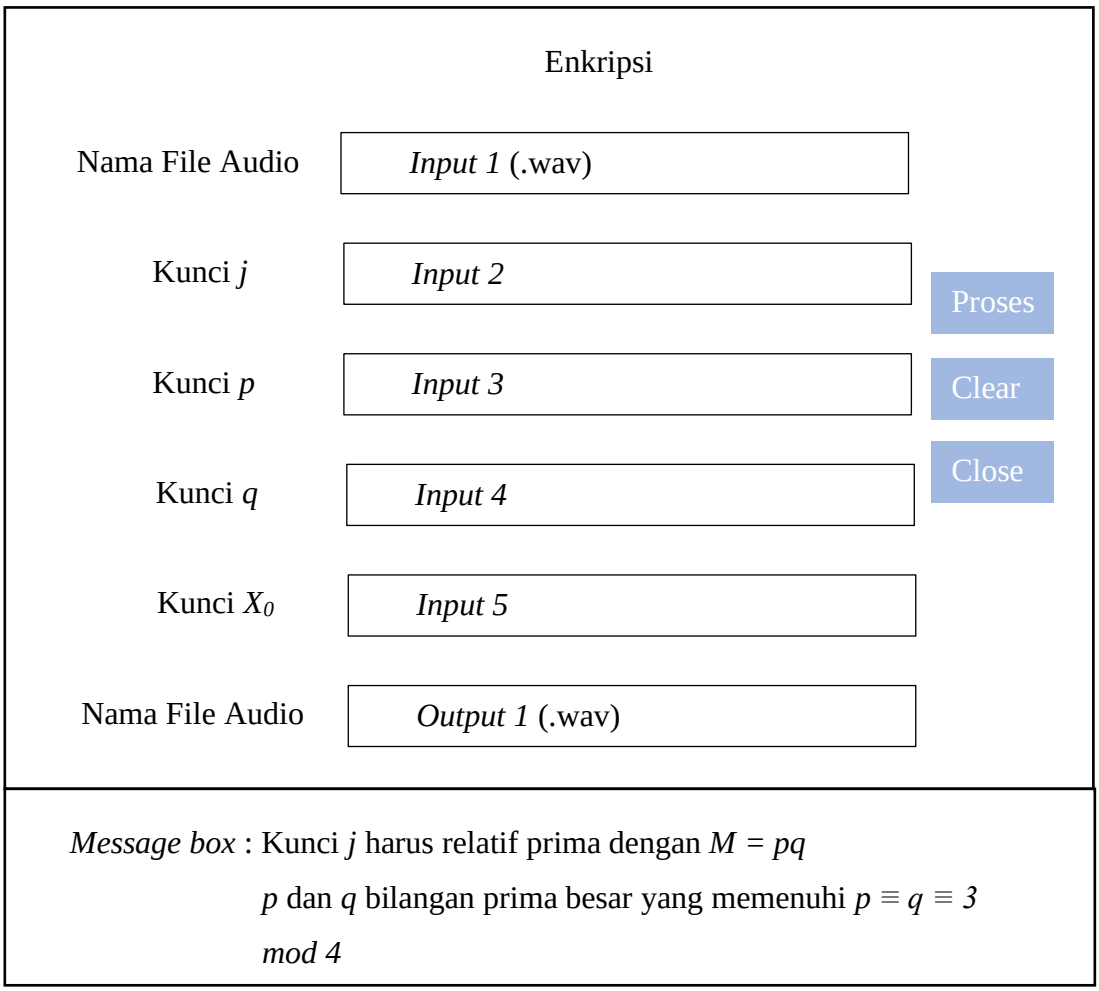
Pada tahap ini dilakukan perancangan *input* apa saja yang diperlukan dan *output* apa yang diinginkan untuk program aplikasi enkripsi dan dekripsi. Input dari program enkripsi adalah file audio (data audio *plaintext)*, kunci *j, p, q,* dan X_0 dengan *output* file audio (data audio *ciphertext*). Untuk *input* program dekripsi adalah file audio (data audio *ciphertext*), kunci *j, p, q,* dan X_0 dengan output file audio (data audio *plaintext*).

Tabel 3. 3
Rancangan Program Enkripsi dan Dekripsi

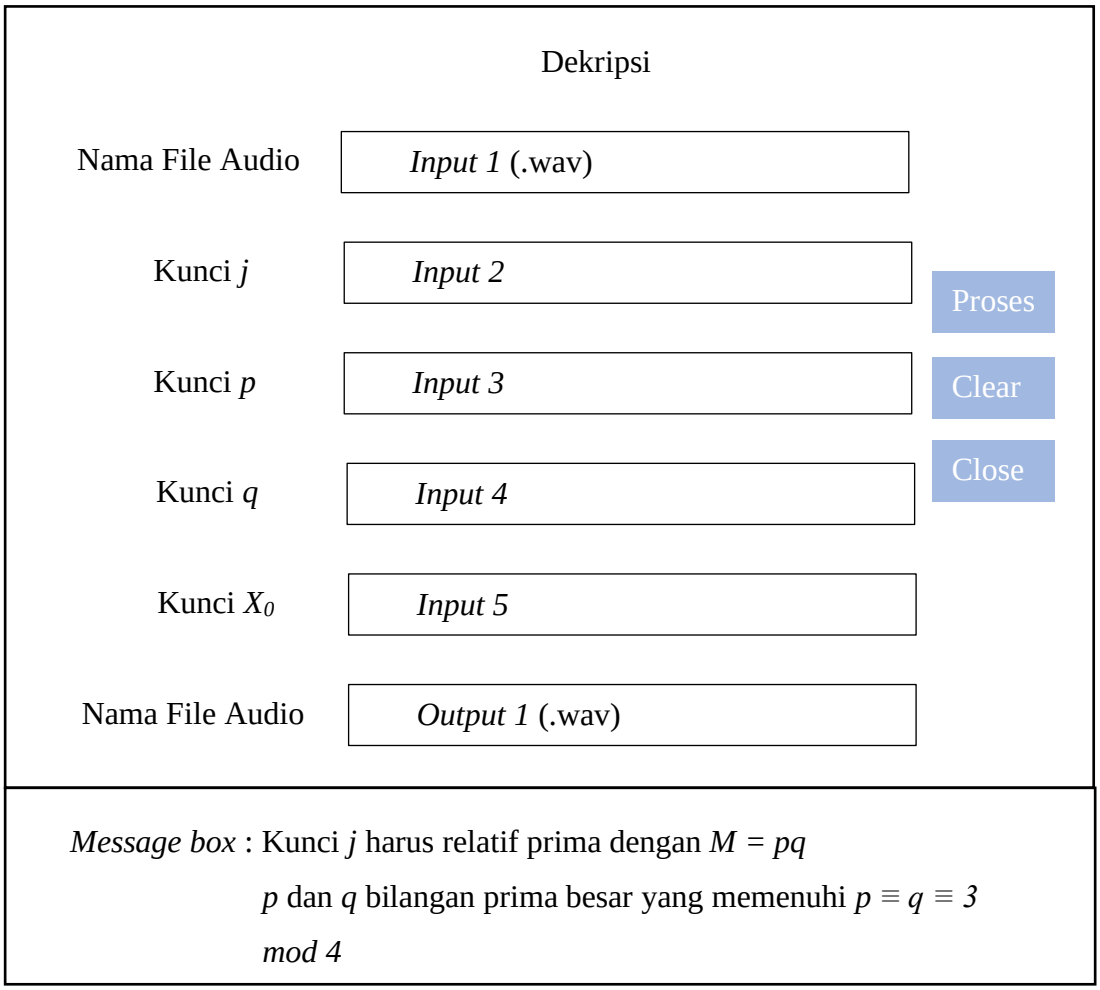
	Enkripsi	**Dekripsi**
Input	file audio (data audio *plaintext)* kunci *j* kunci *p* kunci *q* kunci X_0	file audio (data audio *ciphertext)* kunci *j* kunci *p* kunci *q* kunci X_0
Output	file audio (data audio *ciphertext)*	file audio (data audio *plaintext)*

3.5.2. Rancangan Tampilan Program Aplikasi

Program aplikasi dibuat menggunakan aplikasi *Python* 3.8 dengan tujuan untuk memudahkan proses enkripsi dan dekripsi serta validasi pada kriptografi audio menggunakan transposisi dan *Affine Cipher* yang dikembangkan dengan algoritma *Blum Blum Shub*. Tampilan program aplikasi dapat dilihat seperti berikut:



Gambar 3. *5* Rancangan Tampilan Program Aplikasi Enkripsi



Gambar 3. *6* Rancangan Tampilan Program Aplikasi Enkripsi

3.5.3. Algoritma

Algoritma-algoritma yang berkaitan dalam proses kriptografi audio dalam penelitian ini adalah sebagai berikut:

1. Algoritma Transposisi *Blum Blum Shub*

   Terdapat proses enkripsi dan dekripsi file audio pada algoritma transposisi *Blum Blum Shub* ini, serta terdapat pembangkitan bilangan acak menggunakan *Blum Blum Shub* dan juga pembentukan bilangan acak baru untuk digunakan pada proses transposisi.

2. Algoritma *Affine Cipher*

   Terdapat proses enkripsi dan dekripsi file audio pada algoritma *Affine Cipher* ini, yang bertujuan untuk mengetahui dengan jelas letak pengembangan model.

3. Algoritma Pengembangan *Affine Cipher*

Terdapat proses enkripsi dan dekripsi file audio pada algoritma pengembangan *Affine Cipher* ini, serta menggunakan bilangan acak *Blum Blum Shub* yang telah dibangkitkan sebelumnya pada proses transposisi.

3.5.4. Coding Python

Terdapat modul-modul yang dapat diinstall dalam *Python* untuk memudahkan proses pemograman. Modul *Tkinter* digunakan untuk membuat tampilan yang sudah dirancang sebelumnya, modul *Scipy* digunakan untuk menerjemahkan file audio menjadi data audio, dan modul *Pandas* digunakan untuk mengolah data audio. *Coding Python* yang telah dibuat sesuai *pseudocode* sebelumnya terlampir.

3.5.5. Validasi

Pada tahap ini dilakukan validasi menggunakan aplikasi *Python* untuk memperoleh data audio dari file audio lalu melakukan proses enkripsi pada setiap data audio. Kemudian akan diperiksa apakah data audio yang telah dienkripsi dapat didekripsi sehingga memperoleh file audio asli yang dirahasiakan.