BAB III
METODOLOGI PENELITIAN

3.1 Identifikasi Masalah

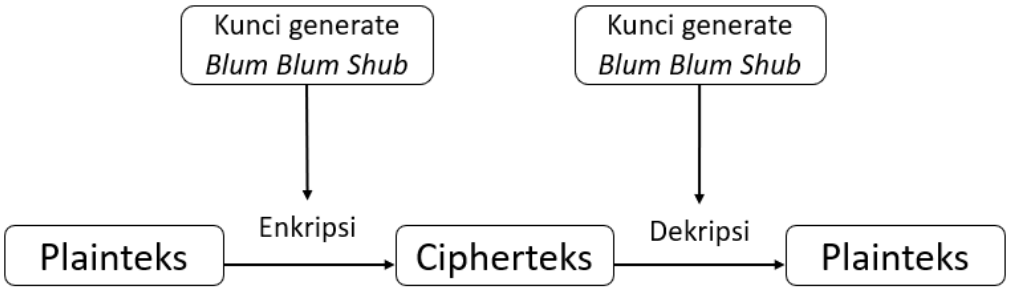
Keamanan pertukaran informasi rahasia melalui jaringan komputer menjadi kebutuhan yang tak terhindarkan. Untuk menjaga kerahasiaan data, diperlukan sistem kriptografi hibrida yang menggabungkan keunggulan dari berbagai algoritma. Vernam *Cipher* dan Merkle-Hellman *Knapsack* adalah dua algoritma yang dapat digunakan untuk mengamankan data. Namun, Vernam *Cipher* memiliki kekurangan yang cukup signifikan, yaitu ketergantungan pada panjang kunci yang harus sama dengan panjang pesan, serta tantangan dalam distribusi kunci yang aman. Oleh karena itu, penggabungan kedua algoritma ini dalam suatu sistem hibrida menjadi penting untuk meningkatkan keamanan data.

Dalam penelitian ini plainteks yang digunakan berupa pesan teks. Kemudian pesan teks dienkripsi menggunakan algoritma Vernam *Cipher* dengan kunci Vernam yang merupakan bilangan acak hasil dari algoritma *Blum Blum Shub* dan menghasilkan cipherteks berupa bilangan. Selanjutnya kunci Vernam dienkripsi dengan kunci publik menggunakan algoritma Merkle-Hellman *Knapsack* dan menghasilkan *cipherkeys* berupa bilangan. Dalam proses dekripsi, Bob dapat menggunakan kunci privat untuk mendekripsi *cipherkeys* dan mendapatkan kunci Vernam. Selanjutnya, cipherteks didekripsi menggunakan kunci Vernam untuk mendapatkan kembali pesan asli. Sementara itu, digunakan dua buah plainteks yaitu plainteks 1 dan plainteks 2 untuk pengujian *Avalanche Effect* dengan plainteks 2 merupakan plainteks 1 yang salah satu hurufnya telah dirubah. Kemudian dilakukan pengujian menggunakan kunci yang sesuai dan menghasilkan nilai persentase dari *Avalanche Effect*.

3.2 Model Dasar

Model dasar yang akan digunakan pada penelitian adalah algoritma Vernam *Cipher* dan Merkle-Hellman *Knapsack*. Vernam *Cipher* merupakan algoritma

kunci simetris di mana dalam proses enkripsi dan dekripsinya menggunakan kunci yang sama. Vernam *Cipher* menggunakan kunci yang dihasilkan oleh generator bilangan acak *Blum Blum Shub*.

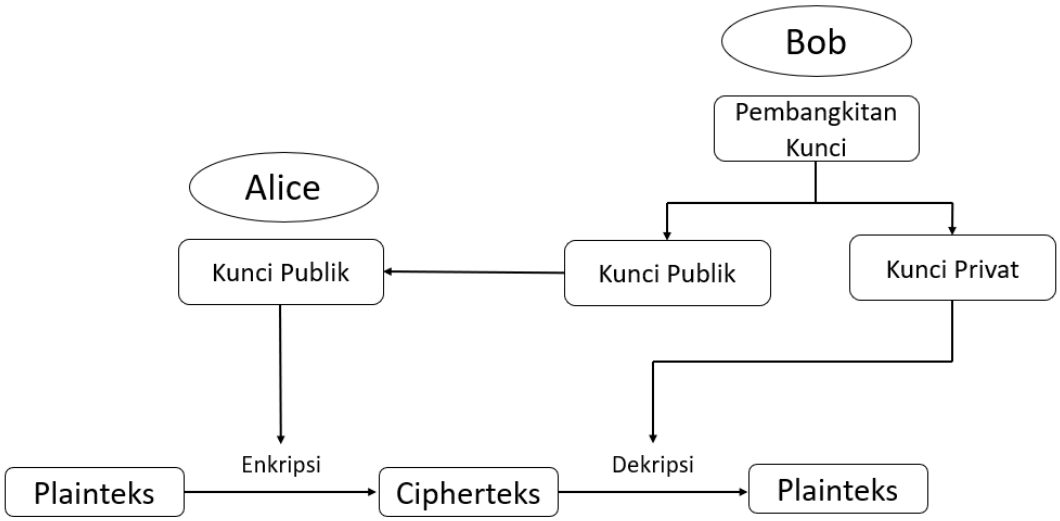


Gambar 3.1 Skema Algoritma Vernam *Cipher*

Skema pada Gambar 3.1 dapat dijelaskan sebagai berikut:

1. Alice menentukan plainteks yang akan dikirimkan kepada Bob. Kunci Vernam dibangkitkan dengan algoritma *Blum Blum Shub*.
2. Alice melakukan enkripsi terhadap plainteks menggunakan kunci Vernam dan menghasilkan cipherteks.
3. Alice mengirimkan cipherteks kepada Bob.
4. Bob menggunakan kunci Vernam yang sama untuk melakukan dekripsi terhadap cipherteks yang diterima dari Alice.
5. Setelah proses dekripsi, Bob memperoleh plainteks yang asli.

Algoritma Merkle-Hellman *Knapsack* adalah algoritma yang menggunakan barisan *superincreasing knapsack* untuk menyelesaikan *knapsack problem*. *Superincreasing knapsack* adalah barisan dengan setiap elemen dalam barisan tersebut memiliki nilai yang lebih besar dari jumlah nilai elemen-elemen sebelumnya. Karena *superincreasing knapsack* adalah algoritma yang lemah, maka *superincreasing knapsack* dimodifikasi menjadi *nonsuperincreasing knapsack* dengan *superincreasing knapsack* sebagai kunci *privat* dan *nonsuperincreasing knapsack* sebagai kunci publik.



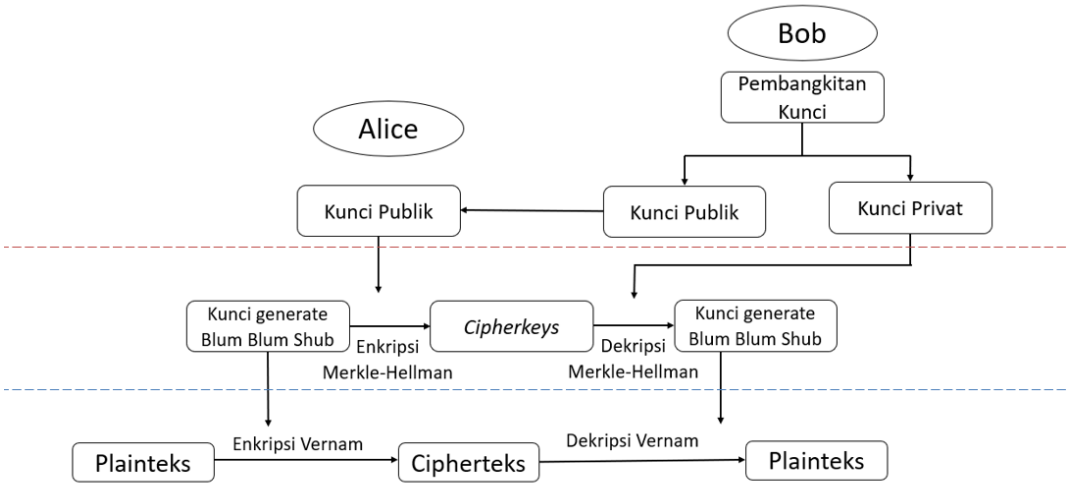
Gambar 3.2 Skema Algoritma Merkle-Hellman *Knapsack*

Skema pada Gambar 3.2 dapat dijelaskan sebagai berikut:

1. Bob membangkitkan kunci publik dan kunci privat.
2. Bob memberikan kunci publik kepada Alice.
3. Alice menentukan plainteks yang akan dikirimkan kepada Bob.
4. Alice mengenkripsi plainteks dengan menggunakan kunci publik dan menghasilkan cipherteks.
5. Alice mengirimkan cipherteks kepada Bob.
6. Bob menggunakan kunci privat untuk mendekripsi cipherteks yang telah diterima dari Alice.
7. Setelah proses dekripsi, Bob memperoleh plainteks yang asli.

3.3 Pengembangan Model Dasar

Dalam pengembangan model dasar, dengan memanfaatkan algoritma bilangan acak semu *Blum Blum Shub* sebagai pembangkit kunci pada algoritma Vernam *Cipher* dapat meningkatkan keamanan serta mempermudah proses pembuatan kunci. Kemudian digabungkan dengan algoritma Merkle-Hellman *Knapsack* agar pertukaran kunci Vernam lebih aman. Implementasi kombinasi algoritma-algoritma tersebut ditunjukkan dalam skema pada Gambar 3.3.



Gambar 3.3 Skema Pengembangan Model Dasar

Skema pada Gambar 3.2 dapat dijelaskan sebagai berikut:

1. Bob membangkitkan kunci publik dan kunci privat.
2. Bob memberikan kunci publik kepada Alice.
3. Alice menentukan plainteks yang akan dikirimkan kepada Bob. Kunci Vernam dibangkitkan dengan algoritma *Blum Blum Shub*.
4. Alice melakukan enkripsi terhadap plainteks menggunakan kunci Vernam dan menghasilkan cipherteks.
5. Alice melakukan enkripsi terhadap kunci Vernam menggunakan kunci publik dan menghasilkan *cipherkeys*.
6. Alice mengirimkan cipherteks dan *cipherkeys* kepada Bob.
7. Bob menggunakan kunci privat untuk mendekripsi *cipherkeys* dan menghasilkan kunci Vernam.
8. Bob menggunakan kunci Vernam untuk mendekripsi cipherteks dan diperoleh kembali plainteks yang asli.

3.4 Konstruksi Program

Program akan dikonstruksi menggunakan bahasa pemograman *Python*. Pada bagian ini akan dibahas tentang input dan output, serta rancangan tampilan dari aplikasi yang akan dibuat.

3.4.1 Input dan Output

Program aplikasi dari kriptografi hibrida *Blum Blum Shub* pada Vernam *Cipher* dengan Merkle-Hellman *Knapsack* dan pengujian *Avalanche Effect* akan dibuat menggunakan bahasa pemograman *Python*. Berikut adalah input dan output dari rancangan program yang akan dibuat.

Tabel 3.1 Tabel Input dan Output

Keterangan	Input	Output
Pembakitan Kunci	• Kunci privat (s) • Bilangan bulat P, a	• Kunci publik (t)
Enkripsi	• Plainteks • Bilangan prima b, c • Kunci Publik (t)	• Kunci Vernam • Cipherteks • *Chiperkeys*
Dekripsi	• Chiperteks • *Chiperkeys* • Kunci privat (s) • Bilangan bulat P, a	• Plainteks • Kunci Vernam
Uji AE pada Vernam *Chiper*	• Plainteks 1 • Plainteks 2 • Kunci	• Persentase hasil pengujian
Uji AE pada Vernam *Chiper* dan BBS	• Plainteks 1 • Plainteks 2 • Bilangan prima b, c	• Persentase hasil pengujian
Uji AE pada Merkle-Hellman *Knapsack*	• Kunci Vernam 1 • Kunci Vernam 2 • Kunci privat (s) • Bilangan bulat P, a	• Persentase hasil pengujian

3.4.2 Algoritma Deskriptif

Terdapat enam algoritma utama yang akan digunakan dalam program yang akan dibuat dalam penyandian pesan. Berikut adalah algoritma deskriptif dari program yang akan dikonstruksi.

A. Algoritma Pembangkitan Kunci:
   1. Bob memasukkan nilai dari kunci privat yang merupakan barisan *superincreasing*.
   2. Bob memasukkan bilangan bulat $P > \sum_{i=1}^{n} S_i$.
   3. Bob memasukkan bilangan bulat a dengan $1 \leq a \leq p - 1$.
   4. Bob membangkitkan kunci publik dan dikirmkan kepada Alice.

B. Algoritma Enkripsi:
   1. Alice memasukkan pesan yang akan dienkripsi.

2. Alice memasukkan nilai b dan c yang memenuhi $b \equiv c \equiv 3 \bmod 4$, dengan b dan c bilangan prima.
3. Alice membangkitkan kunci Vernam.
4. Alice memasukkan kunci publik yang telah diterima.
5. Alice melakukan proses enkripsi pada pesan dan kunci Vernam.
6. Pesan dan kunci Vernam yang telah dienkripsi akan menghasilkan cipherteks dan *cipherkeys*.

C. Algoritma Dekripsi:
1. Bob memasukkan cipherteks dan *cipherkeys* yang telah diterima.
2. Bob memasukkan kunci privat yang merupakan barisan *superincreasing* beserta bilangan bulat P dan a.
3. Bob melakukan dekripsi pada cipherteks dan *cipherkeys*.
4. Cipherteks dan *cipherkeys* yang telah didekripsi akan menghasilkan pesan asli dan kunci Vernam.

D. Algoritma Pengujian *Avalanche Effect* untuk Vernam *Cipher*:
1. Alice memasukkan plainteks 1 dan plainteks 2.
2. Alice memasukkan kunci dengan panjang kunci sama dengan panjang plainteks.
3. Alice melakukan pengujian *Avalanche Effect*.
4. Alice memperoleh persentase hasil pengujian *Avalanche Effect*.

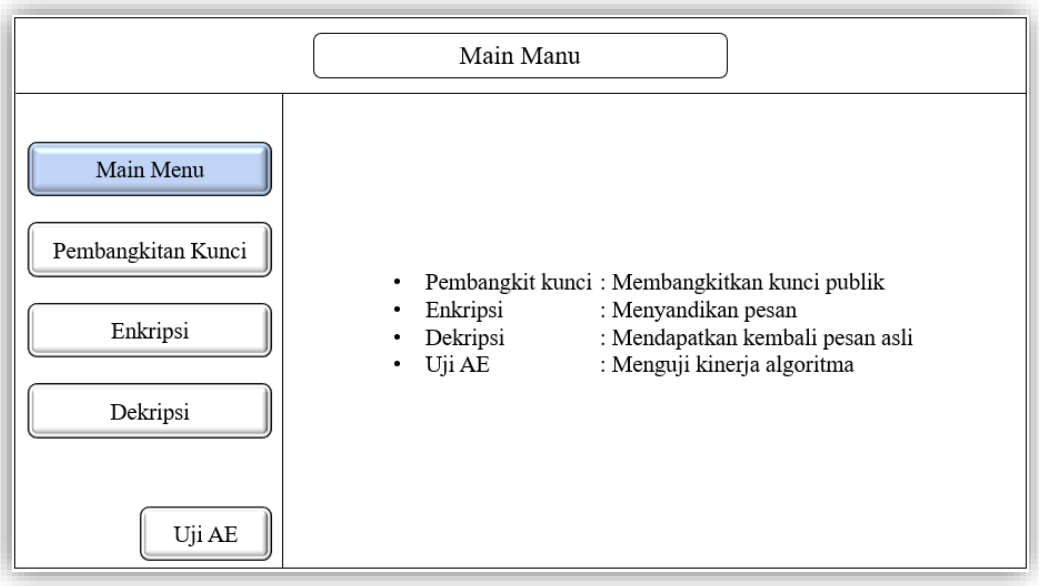
E. Algoritma Pengujian *Avalanche Effect* untuk Vernam *Cipher* dengan *Blum Blum Shub*:
1. Alice memasukkan plainteks 1 dan plainteks 2.
2. Alice memasukkan nilai b dan c yang memenuhi $b \equiv c \equiv 3 \bmod 4$, dengan b dan c bilangan prima.
3. Alice melakukan pengujian *Avalanche Effect*.
4. Alice memperoleh persentase hasil pengujian *Avalanche Effect*.

F. Algoritma Pengujian *Avalanche Effect* untuk Merkle-Hellman Knapsack:
1. Bob memasukkan nilai dari kunci Vernam 1 dan kunci Vernam 2.
2. Bob memasukkan kunci privat merupakan barisan *superincreasing*.
3. Bob memasukkan bilangan bulat $P > \sum_{i=1}^{n} S_i$.
4. Bob memasukkan bilangan bulat a dengan $1 \leq a \leq p - 1$.

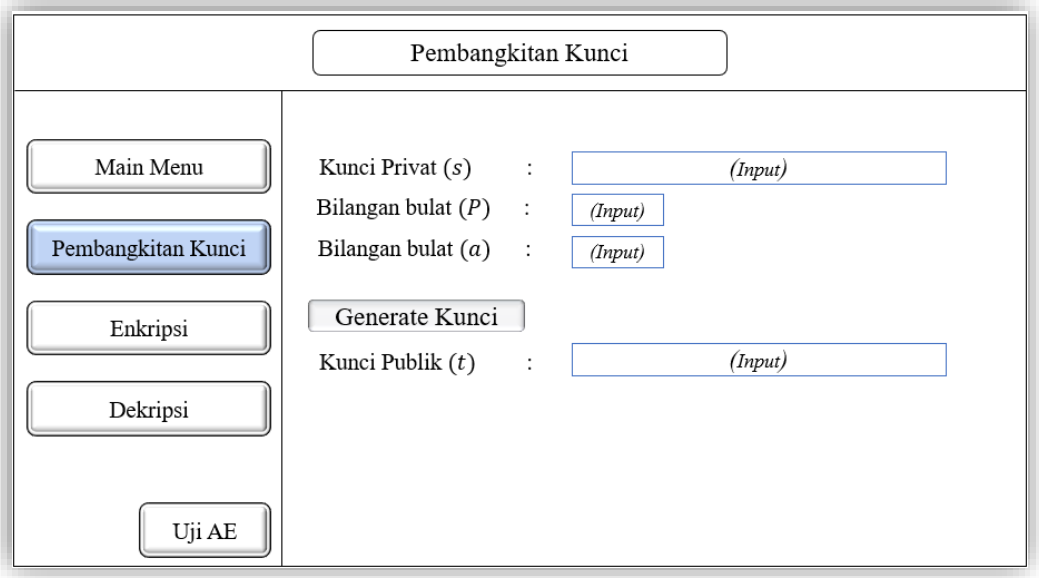
5. Bob melakukan pengujian *Avalanche Effect*.
6. Bob memperoleh persentasi hasil pengujian *Avalanche Effect*.

3.4.3 Rancangan Tampilan Program

Tampilan utama program ini dirancang mempunyai 5 *Button*, yaitu *Button Main Menu*, *Button* Pembangkitan Kunci, *Button* Enkripsi, *Button* Dekripsi dan *Button* Uji AE. Berikut rancangan tampilan program yang akan dibuat.

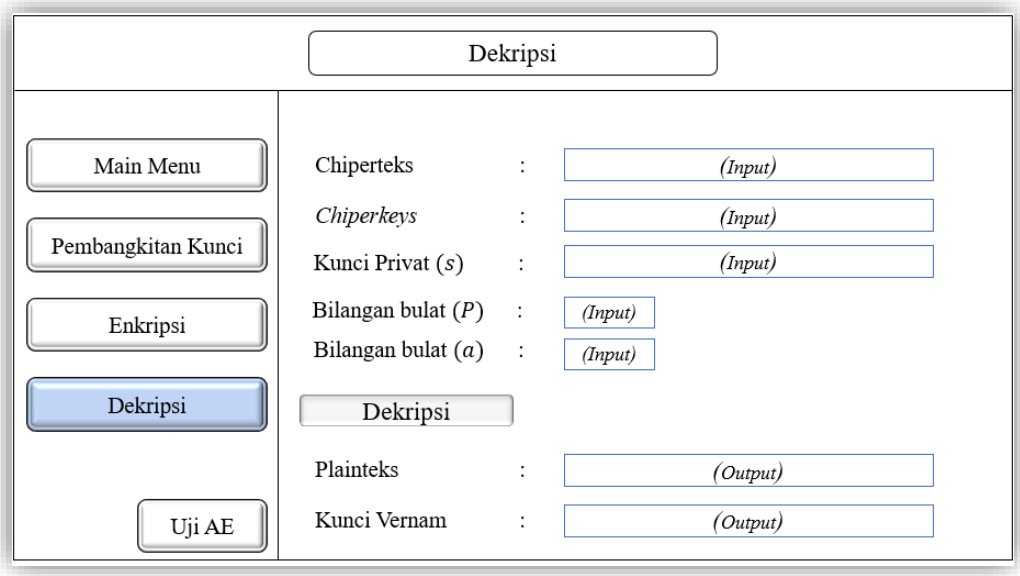


Gambar 3.4 Rancangan Tampilan *Main Menu*



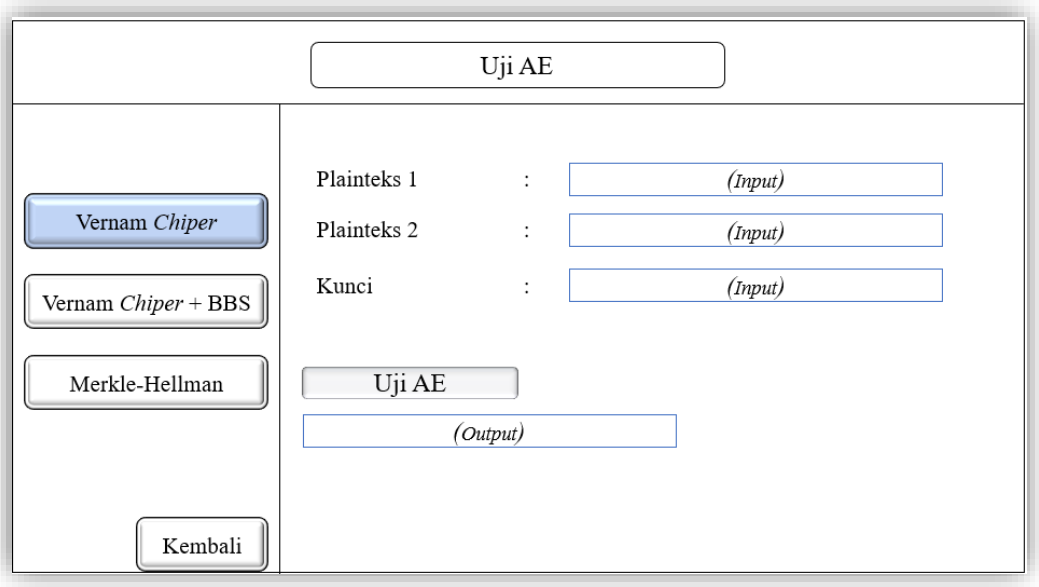
Gambar 3.5 Rancangan Tampilan Pembangkitan Kunci

Gambar 3.6 Rancangan Tampilan Enkripsi

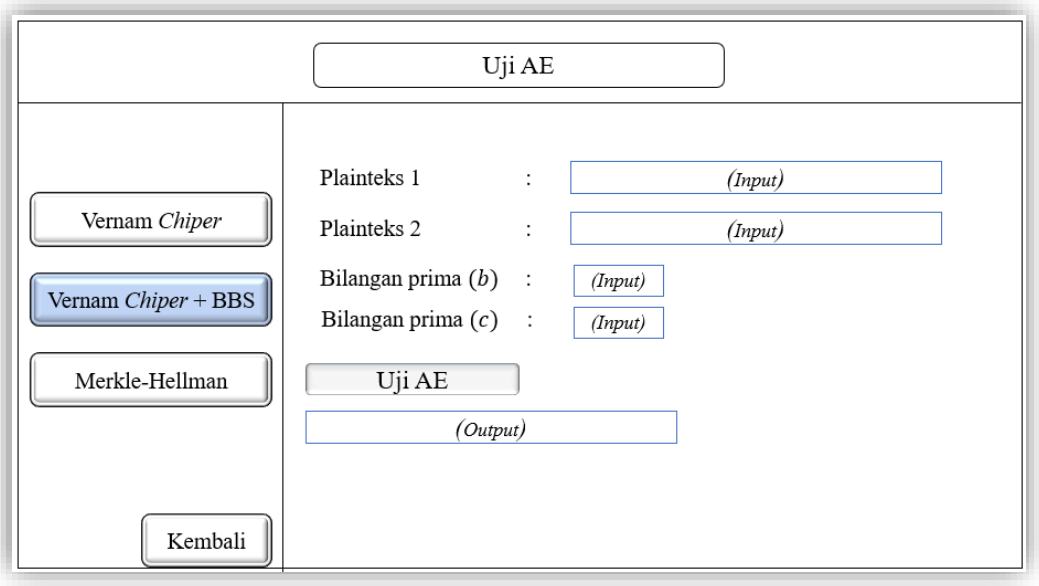


Gambar 3.7 Rancangan Tampilan Dekripsi

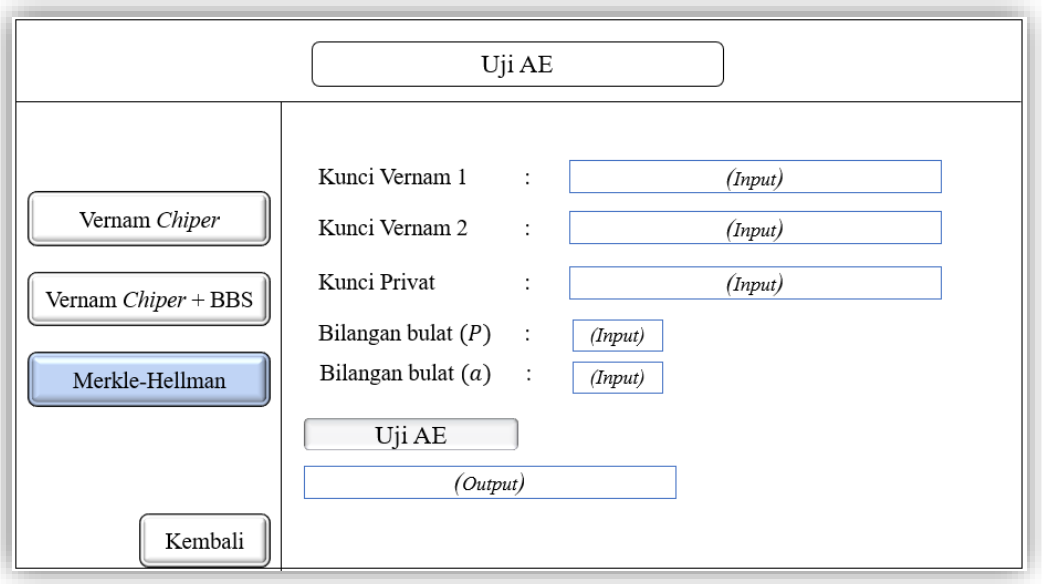
Pada tampilan awal *user* akan dimunculkan tampilan seperti pada Gambar 3.4. Selanjutnya jika *user* klik *button* **Pembangkitan Kunci** maka akan muncul tampilan seperti pada Gambar 3.5. Lalu jika *user* klik *button* **Enkripsi** maka akan muncul tampilan seperti pada Gambar 3.6 dan jika *user* klik *button* **Dekripsi** maka akan muncul tampilan seperti pada Gambar 3.7.



Gambar 3.8 Rancangan Tampilan Pengujian Vernam *Cipher*



Gambar 3.9 Rancangan Tampilan Pengujian Vernam *Cipher* dan *BBS*



Gambar 3.10 Rancangan Tampilan Pengujian Merkle-Hellman *Knapsack*

Jika *user* klik *button* **Uji AE**, maka akan muncul tampilan seperti Gambar 3.8. Selanjutnya jika *user* klik *button* **Vernam + BBS** maka akan muncul tampilan seperti pada Gambar 3.9 dan jika *user* klik *button* **Merkle-Hellman** maka akan muncul tampilan seperti pada Gambar 3.10.

3.4.4 *Library Python*

Library python yang akan digunakan pada pembuatan program aplikasi adalah sebagai berikut :

1. *Tkinter*.

   Tkinter adalah sebuah *library* yang digunakan untuk membangun GUI (*Grapical User Interface*) pada aplikasi *python*. Tkinter dapat digunakan untuk membuat jendela, tombol, kotak teks dan elemen-elemen lainnya sebagai tampilan aplikasi.

2. *Random*.

   Random adalah *library* untuk menghasilkan bilangan acak. Pada pembuatan program, *library* ini berfungsi memberikan bilangan acak dengan range tertentu sebagai koefisien.

3.5 Proses Validasi

Tahap validasi bertujuan untuk memastikan program hasil implementasi dapat berjalan dengan benar. Proses ini dilakukan dengan cara melakukan perhitungan manual untuk setiap proses enkripsi dan dekripsi. Program hasil implementasi dapat tervalidasi ketika hasil perhitungan manual dalam mengenkripsi dan mendekripsi pesan sama dengan hasil perhitungan oleh program.

3.6 Penarikan Kesimpulan

Setelah program hasil implementasi tervalidasi dilakukan penarikan kesimpulan dari hasil yang telah dilakukan dan memberikan saran serta rekomendasi kepada peneliti selanjutnya. Hal ini bertujuan untuk memperoleh hasil penelitian yang lebih baik.