MAKALAH KRIPTOGRAFI
APLIKASI DIFFIE-HELLMAN DAN AUTOKEY VIGENERE

Diajukan untuk memenuhi salah satu syarat kelulusan Mata Kuliah Keahlian Pilihan Program Studi Kriptografi (MT542) yang diampu oleh Dra. Hj. Rini Marwati, M.S.

Disusun Oleh:

Fahmi Ismail Ramantoko	2205676
Muhammad Rizqi Winnel Adnin	2209397
Muhammad Thoriq Atallah	2202991
Shafira Mayora Handriyudha	2202576

PROGRAM STUDI MATEMATIKA
FAKULTAS PENDIDIKAN MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS PENDIDIKAN INDONESIA
BANDUNG

TAHUN 2024/2025

DAFTAR ISI

BAB I
PENDAHULUAN

1.1. Latar Belakang

Dalam era digital yang terus berkembang, kebutuhan akan komunikasi yang aman menjadi semakin penting, terutama dalam pengiriman data sensitif melalui jaringan yang tidak aman. Salah satu inovasi penting dalam bidang ini adalah protokol Diffie-Hellman, yang diperkenalkan oleh Whitfield Diffie dan Martin Hellman pada tahun 1976. Protokol ini menandai dimulainya era baru dalam kriptografi modern dengan memperkenalkan konsep pertukaran kunci publik, sebuah metode revolusioner yang memungkinkan dua pihak untuk menghasilkan kunci rahasia bersama tanpa perlu bertemu secara langsung (Diffie & Hellman, 1976).

Sejarah penemuan protokol ini berawal dari penelitian Whitfield Diffie dan Martin Hellman di Stanford University, di mana mereka berfokus pada pengembangan metode baru untuk mengamankan komunikasi. Sebelum protokol ini ditemukan, pengamanan data mengandalkan sistem simetris, di mana pengirim dan penerima harus berbagi kunci rahasia yang sama, yang sering kali menimbulkan tantangan logistik. Dengan memperkenalkan prinsip dasar masalah logaritma diskrit, protokol Diffie-Hellman menjadi dasar bagi berbagai algoritma kriptografi modern yang digunakan hingga saat ini.

Maka dari itu, kami menggunakan Algoritma Pertukaran Kunci Diffie-Hellman untuk metode pertukaran kunci yang aman. Dalam kasus ini juga kami menggunakan kriptografi Autokey Vigenère yang merupakan pengembangan dari Vigenère Cipher, yaitu metode enkripsi yang diperkenalkan oleh Giovan Battista Bellaso pada tahun 1553 dan kemudian dipopulerkan oleh Blaise de Vigenère pada tahun 1586. Metode ini dirancang untuk mengatasi kelemahan enkripsi substitusi sederhana dengan menggunakan kunci berbasis urutan huruf. Pada abad ke-19, François-Joseph Saucier memperkenalkan metode Autokey, yang menggunakan plainteks itu sendiri sebagai bagian dari kunci setelah beberapa huruf pertama dienkripsi menggunakan kunci awal. Inovasi ini menciptakan pola enkripsi yang lebih acak dan sulit dianalisis, sehingga meningkatkan keamanannya (Kahn, 1996; Singh, 1999).

Meskipun lebih aman dibandingkan Vigenère klasik, Autokey Vigenère tetap memiliki kelemahan. Jika sebagian kunci diketahui, metode ini masih rentan terhadap serangan teks-tertebak. Namun, memadukan Pertukaran Kunci Diffie-Hellman dengan Autokey Vigenère akan meningkatkan tingkat keamanannya menjadi lebih optimal.

BAB II
KAJIAN TEORI

2.1. Teori Dasar Matematika

1. Faktor Persekutuan Terbesar (FPB)
   Menurut Burton (2011), jika a dan b adalah bilangan bulat dan salah satunya tidak nol, maka $FPB(a, b)$ adalah d suatu bilangan bulat positif yang memenuhi dua kondisi:
   - d membagi a
   - d membagi b Jika c juga memenuhi kondisi tersebut, maka $c \leq d$.
2. Bilangan Prima
   Suatu bilangan bulat positif p, dengan $p > 1$, disebut bilangan prima jika pembaginya hanya 1 dan dirinya sendiri (Munir, 2016).
3. Modulo
   Misalkan a adalah bilangan bulat dan b adalah bilangan bulat yang lebih besar dari 1. Operasi $a \, mod \, m$ memberi hasil sisa pembagian a dengan m, di mana $0 \leq r < m$ (Munir, 2016).

2.2. Teori Kriptografi

Pertukaran Kunci Diffie-Hellman :

1. Alice membangkitan bilangan bulat acak yang besar x dan mengirim hasil perhitungan berikut kepada Bob:
   $X = g^x mod \, n$
2. Bob membangkitkan bilangan bulat acak yang besar y dan mengirim hasil perhitungan berikut kepada Alice:
   $Y = g^y mod \, n$
3. Alice menghitung
   $K = Y^x mod \, n$
4. Bob menghitung
   $K' = X^y mod \, n$
5. Jika perhitungan dilakukan dengan benar, maka
   $K = K'$

Vigenere Cipher :

$P = C = K = Z_{26}$

Misalkan m, n bilangan bulat positif

Untuk plainteks $P = (x_1, x_2, \ldots, x_n)$ dan kunci $K = (k_1, k_2, \ldots, k_m)$

Enkripsi

$e_k(x_i) = (x_1 + k_i) \, mod \, 26$

Dekripsi

$d_k(y_i) = (y_1 - k_i) \, mod \, 26$

Semua operator dikenakan dalam Z_{26}

Autokey Cipher :

$P = C = K = Z_{26}$

Misalkan untuk suatu $a \in Z_{26}$

definisikan $k_1 = a$ dan $k_i = x_{i-1}$ untuk semua $i \geq 2$

Enkripsi

$e_K(x_i) = (x_1 + k_i) \; mod \; 26$

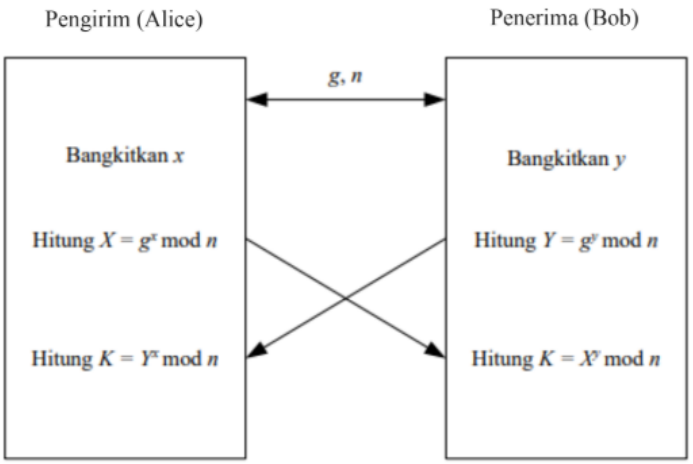
Dekripsi

$d_K(y_i) = (y_1 - k_i) \; mod \; 26$

BAB III
PEMBAHASAN

3.1. Algoritma/Skema/Protokol

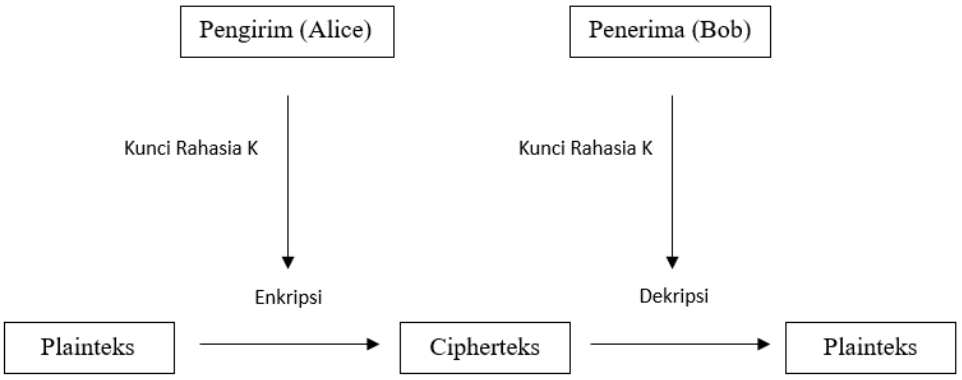
Skema pertukaran kunci Diffie-Hellman adalah sebagai berikut:



Gambar 1.1 Skema Pertukaran Kunci Diffie-Hellman

Skema pada **Gambar 1.1** tersebut memperlihatkan bahwa Alice dan Bob menyepakati bilangan *g* dan *n*. Alice kemudian membangkitkan *x*, menghitung nilai *X* dan mengirimkan nilai *X* itu ke Bob, lalu menghitung kunci *K* dengan nilai *Y* yang diterima dari Bob. Sedangkan Bob membangkitkan *y*, menghitung nilai *Y* kemudian mengirimkan nilai *Y* itu ke Alice, dan menghitung kunci *K* dengan nilai *X* yang diterimanya dari Alice. Jika perhitungan dengan benar, nilai kunci *K* milik Alice akan sama dengan nilai kunci *K* milik Bob.

Adapun untuk skema proses enkripsi dan dekripsi pesan menggunakan Autokey Vigenere Cipher adalah sebagai berikut:

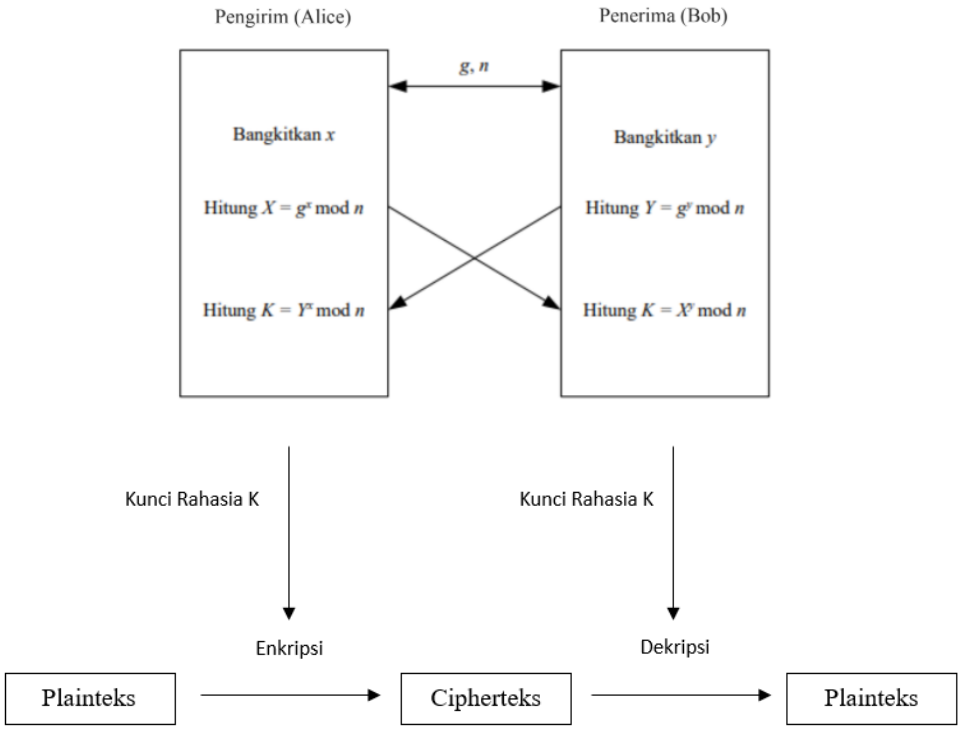


Gambar 1.2 Skema Kriptografi Autokey Vigenere Cipher

Skema pada **Gambar 1.2** memperlihatkan bahwa Alice dan Bob menggunakan kunci

rahasia K yang telah ditentukan untuk melakukan enkripsi dan dekripsi menggunakan Autokey Vigenere Cipher. Autokey Vigenere Cipher sendiri merupakan algoritma kriptografi klasik yang merupakan perluasan dari Vigenere Cipher, dimana Autokey Vigenere Cipher menggunakan sebagian plainteks sebagai kunci yang digunakan untuk enkripsi serta dekripsinya.

Pengembangan model yang dibuat pada aplikasi adalah dengan melakukan pertukaran kunci Diffie-Hellman terlebih dahulu, lalu menggunakan kunci rahasia *K* yang disesuaikan agar enkripsi dan dekripsi Autokey Vigenere Cipher bekerja dalam modulo 26.
Cara kerja pengembangan model tersebut dapat ditunjukkan pada skema berikut ini:

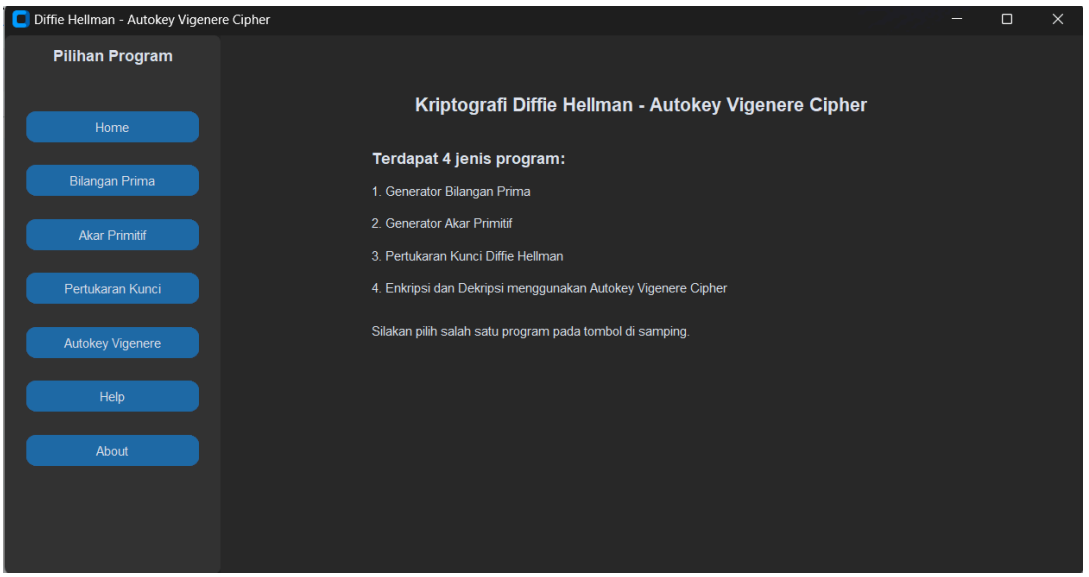


Gambar 1.3 Skema Pengembangan Model

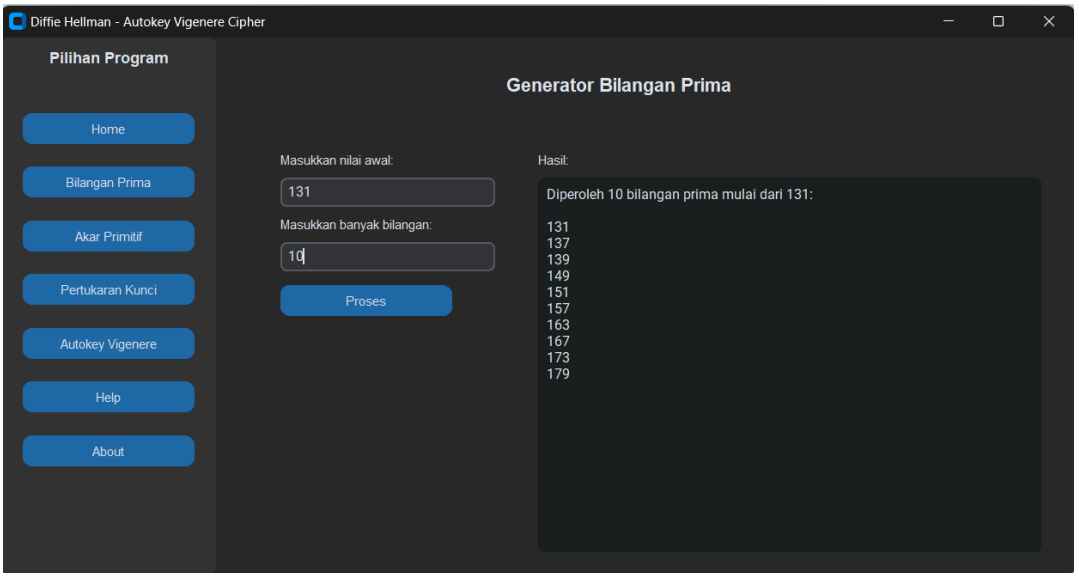
3.2. Contoh Kasus

Alice dan Bob ingin saling berbagi pesan tetapi Alice dan Bob tidak bisa langsung berbagi kunci enkripsi melalui saluran komunikasi biasa karena takut disadap oleh Eve. Mereka perlu metode untuk menghasilkan kunci bersama tanpa mengungkapkan kunci rahasia mereka. Alice ingin memberikan pesan kepada Bob berisikan "hellobobhowareyoutoday" secara rahasia. Alice dan Bob bersepakat menghasilkan kunci bersama menggunakan **Diffie-Hellman**, lalu mengenkripsi pesan tersebut dengan **Autokey Vigenere Cipher** sehingga Eve sulit ataupun tidak bisa membaca pesan tersebut meskipun dia menyadap komunikasi.

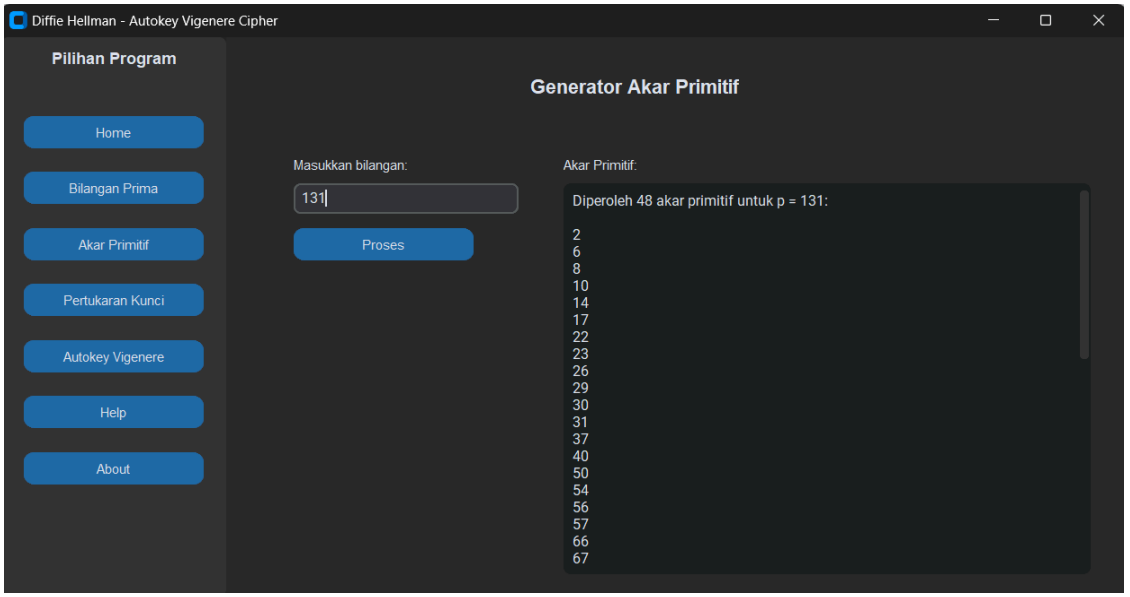
3.3. Contoh Hasil Aplikasi Bahasa Pemrograman



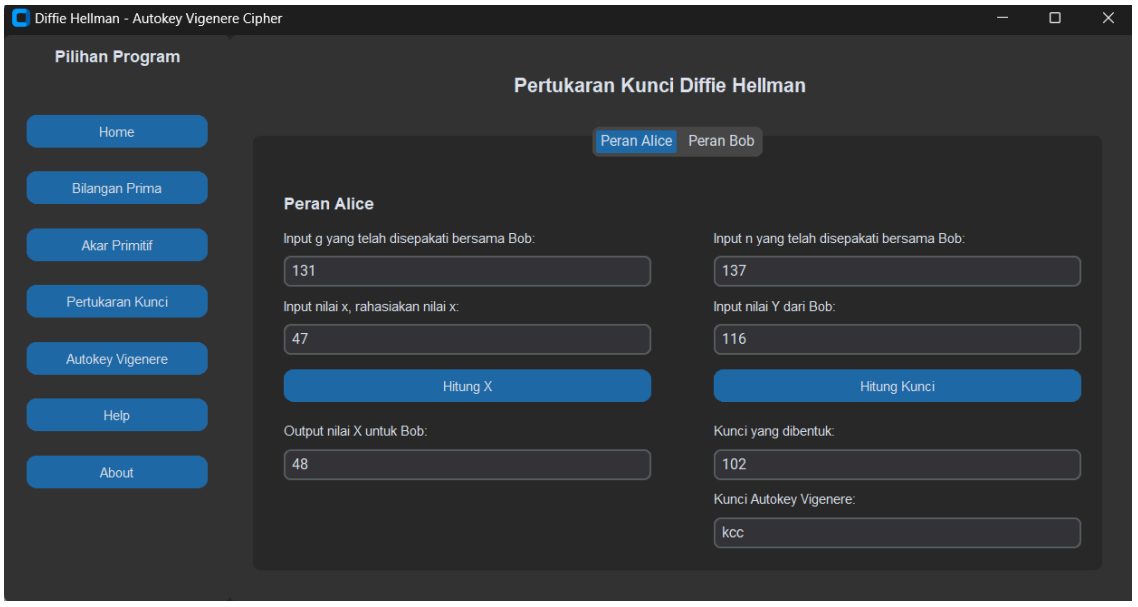
Gambar 3.1 Home atau Menu Utama



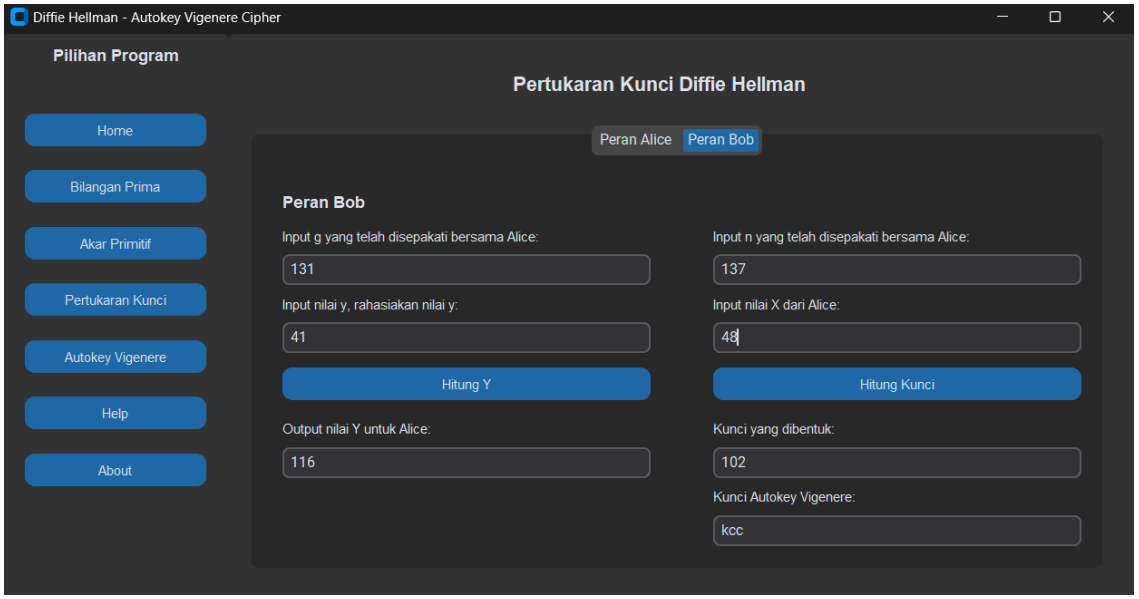
Gambar 3.2 Generator Bilangan Prima



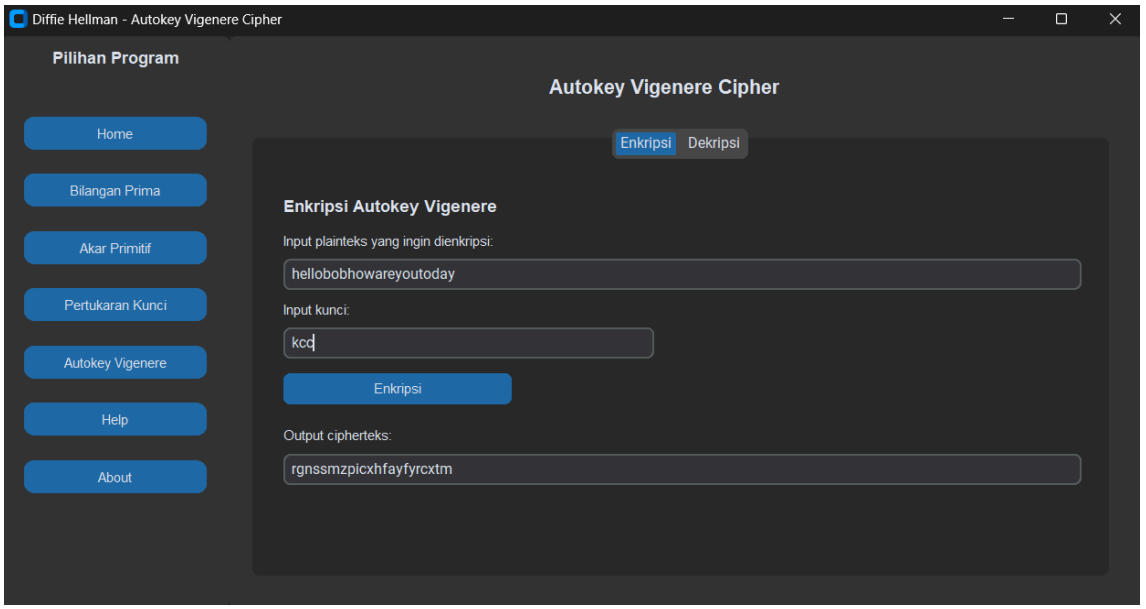
Gambar 3.3 Generator Akar Primitif



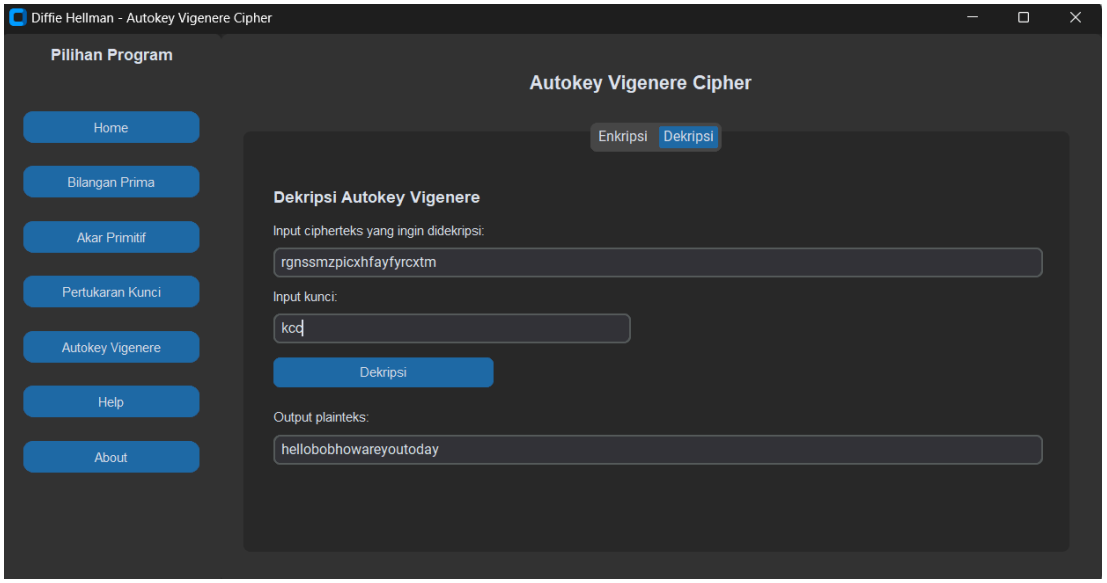
Gambar 3.4 Pertukaran Kunci Diffie Hellman Peran Alice



Gambar 3.5 Pertukaran Kunci Diffie Hellman Peran Bob



Gambar 3.6 Autokey Vigenere Cipher Enkripsi



Gambar 3.7 Autokey Vigenere Cipher Dekripsi

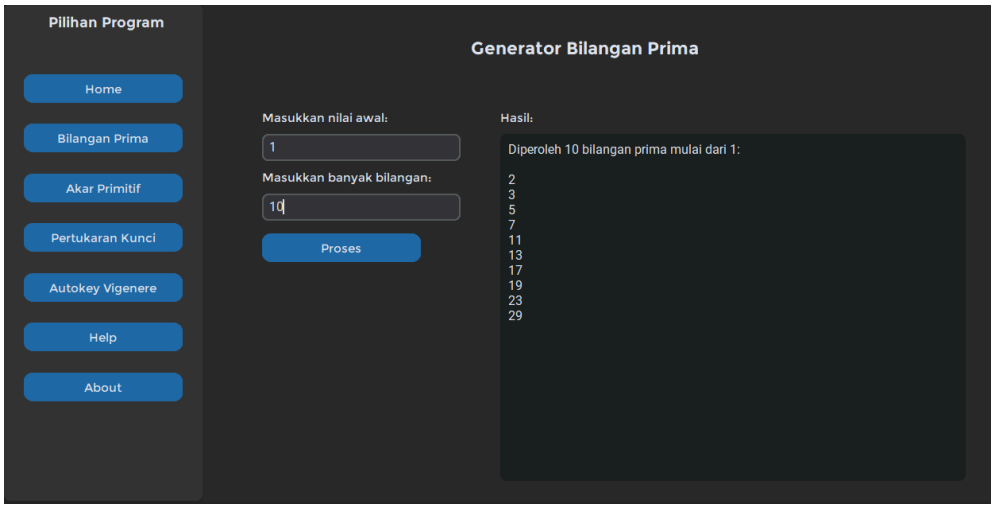
3.4. Validasi dengan Aplikasi Lainnya

Validasi untuk aplikasi ini akan menggunakan bantuan Excel. Untuk praktiknya, kami akan memvalidasi bagian bilangan prima, akar primitif, pertukaran kunci Diffie Hellman dan Autokey Vigenere.

1. Bilangan Prima

   Aplikasi Diffie Hellman - Autokey Vigenere

   Pada generator bilangan prima, *user* akan diminta untuk memasukkan nilai awal dana banyaknya bilangan yang akan dicari bilangan prima. Dalam hal ini nilai awalnya 1 dan banyak bilangannya 10, diperoleh 10 bilangan prima seperti pada **Gambar 4.1**. Selanjutnya pilih salah satu bilangan prima pada *textbox* (misal 11) yang nantinya akan divalidasi kebenarannya menggunakan Excel.

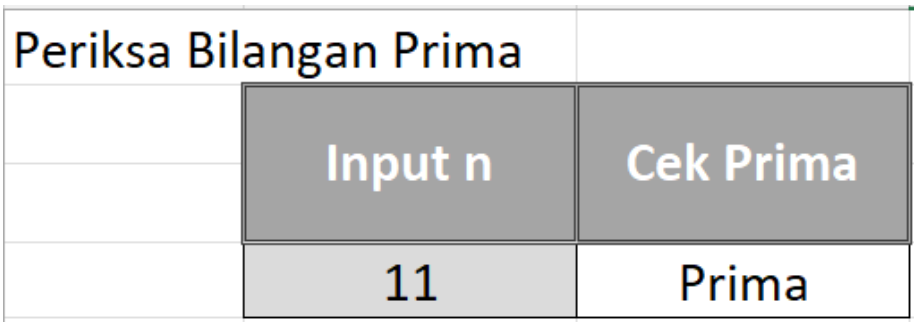


Gambar 4.1 Generator Bilangan Prima

Excel

Untuk periksa apakah bilangan tersebut merupakan bilangan prima atau bukan,

akan menggunakan *syntax* berikut {=IF(B4=2,"Prima",IF(AND(MOD(B4,ROW(INDIRECT("2:"&ROUNDUP(SQRT(B4),0))))<>0),"Prima","Bukan Prima"))}. Misalkan akan diperiksa apakah 11 (salah satu bilangan pada *textbox*) merupakan bilangan prima atau bukan. Dan berdasarkan **Gambar 4.2**, 11 merupakan bilangan prima.

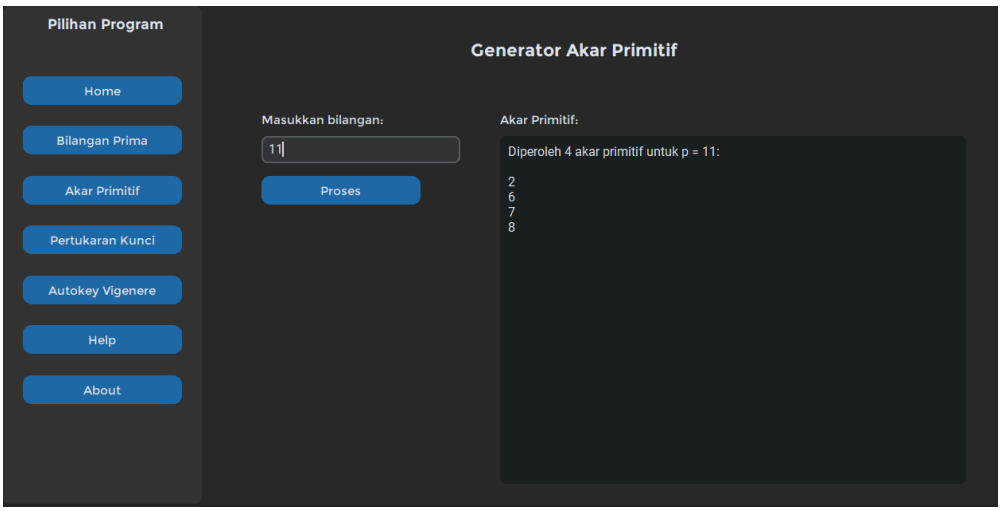


Gambar 4.2 Periksa Bilangan Prima

2. Akar Primitif

Aplikasi Diffie Hellman - Autokey Vigenere

Setelah memilih bilangan prima pada generator bilangan prima, selanjutnya akan dicari akar primitif dari bilangan prima tersebut pada generator akar primitif. Dalam hal ini berarti akan dicari akar primitif dari 11. Dan berdasarkan **Gambar 4.3** diperoleh akar primitif dari 11 adalah 2, 7, 6 dan 8. Sebagai validasi pada Excel akan dipilih salah satu bilangan pada *textbox* (misal 2) apakah merupakan akar primitif dari 11 atau bukan.



Gambar 4.3 Periksa Bilangan Prima

Excel

Untuk memeriksa akar primitif akan melibatkan $\phi(n) = n - 1$ dengan n merupakan bilangan prima, dimana akar primitif dari bilangan prima akan menghasilkan nilai yang berbeda. Dimana *syntax* masing-masing kolom bagian $\phi(n)$ (Lihat **Gambar4.4**) diantaranya, {=MOD(C10^1,B10)}; {=MOD(C10^2,B10)}; {=MOD(C10^3,B10)}; … {=MOD(C10^10,B10)}. Sehingga untuk memeriksa apakah bilangan 2

merupakan akar primitif dari 11, haruslah diperiksa dari masing-masing kolom menghasilkan nilai yang berbeda. Dan ternyata diperoleh bahwa 2 merupakan akar primitif dari 11.

Periksa Akar Primitif

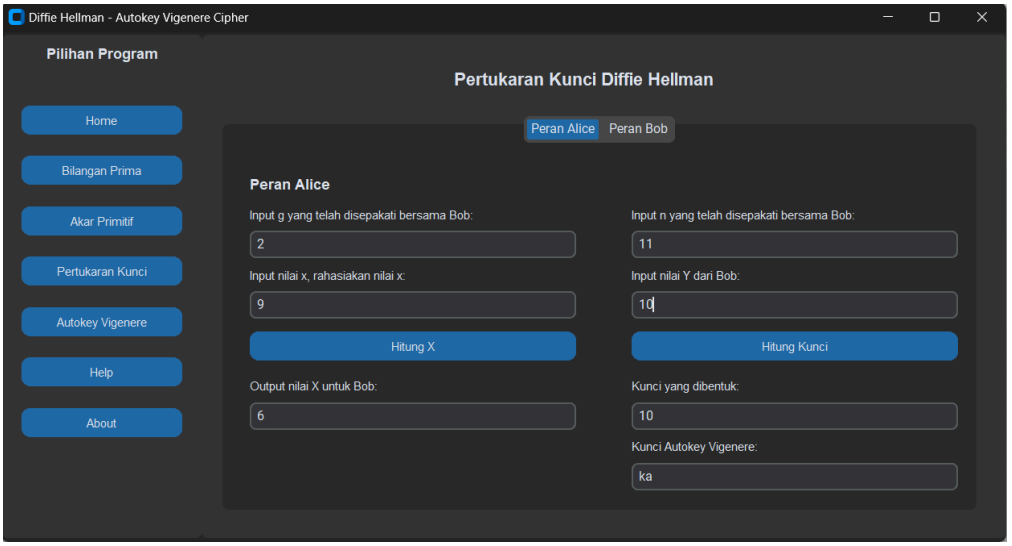
Input n	Input a	$\emptyset(n)$										Cek Akar Primitif
		1	2	3	4	5	6	7	8	9	10	
11	2	2	4	8	5	10	9	7	3	6	1	Akar Primitif

Gambar 4.4 Periksa Akar Primitif

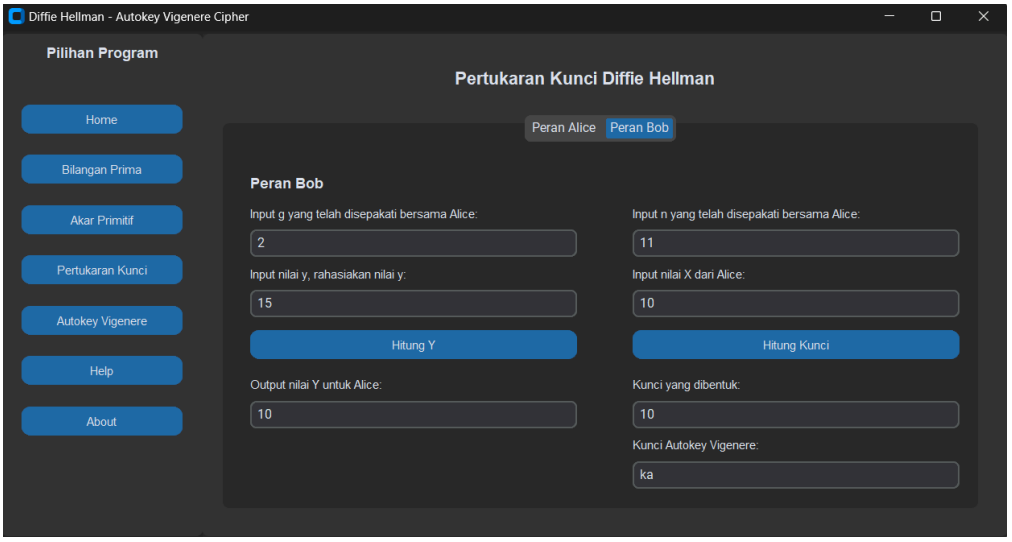
3. Pertukaran kunci

Aplikasi Diffie Hellman - Autokey Vigenere

Lalu pada bagian pertukaran kunci ini akan dibagi menjadi 2 peran dimana masing-masing peran haruslah menyepakati bilangan g dan n dengan n merupakan bilangan prima dan g merupakan salah satu akar primitif dari n. Dalam hal ini g adalah 2 sedangkan n adalah 11. Dengan x yang hanya diketahui oleh Alice (yaitu $x = 9$) dan dengan X merupakan output untuk Bob. Sedangkan y yang hanya diketahui oleh Bob (yaitu $y = 15$) dan dengan Y merupakan output untuk Alice. Sehingga diperoleh kunci yang dibentuk merupakan kunci yang sama (yaitu 10), baik pihak Bob maupun Alice seperti pada **Gambar 4.5** dan **Gambar 4.6**.



Gambar 4.5 Kunci Diffie Hellman Peran Alice



Gambar 4.6 Kunci Diffie Hellman Peran Bob

Excel

Sama seperti pada aplikasi tersebut, baik Alice dan Bob harus memasukkan bilangan g, n dan kunci privat (x dan y). Dimana nantinya akan digunakan untuk mencari kunci publiknya (X dan Y), dengan $X = g^x mod\ n$ dan $Y = g^y mod\ n$. Sedangkan *syntax* dari X adalah {=MOD(2^9,11)} dan *syntax* Y adalah {=MOD(2^15,11). Atau $X = 6$ dan $Y = 10$. Selanjutnya dari pihak Alice dan Bob akan membentuk suatu kunci (K untuk Alice dan K' untuk Bob), dimana *syntax* dari K adalah {=MOD(10^9,11)} dan *syntax* dari K' adalah {=MOD(6^15,11)} yang diperoleh baik nilai K ataupun K' bernilai 10. Berdasarkan **Gambar 4.7** diperoleh keluaran yang sama dengan aplikasi.

	Alice	Bob
Input g	2	2
Input n	11	11
Input x	9	
Input y		15
Output X	6	
Output Y		10
Kunci K	10	
Kunci K'		10

Gambar 4.7 Periksa Kunci Diffie Hellman

Sedangkan untuk membangkitkan kunci Autokey Vigenere dari hasil pertukaran kunci Diffie-Hellman, akan dibagi menjadi 2 bagian yaitu 10 dan 0. Dari kedua bagian tersebut akan diterjemahkan ke dalam ASCII dengan menggunakan *syntax* {=MOD(10,26)+97} untuk bagian 10, sedangkan bagian 0 menggunakan *syntax* {=MOD(0,26)+97}. Selanjutnya diterjemahkan dengan

bantuan tabel ASCII. Jadi kunci Autokey Vigenere yang diperoleh adalah *ka* (Lihat **Gambar 4.8)**, sama halnya dengan kunci Autokey Vigenere yang diperoleh menggunakan aplikasi pada **Gambar 4.6**.

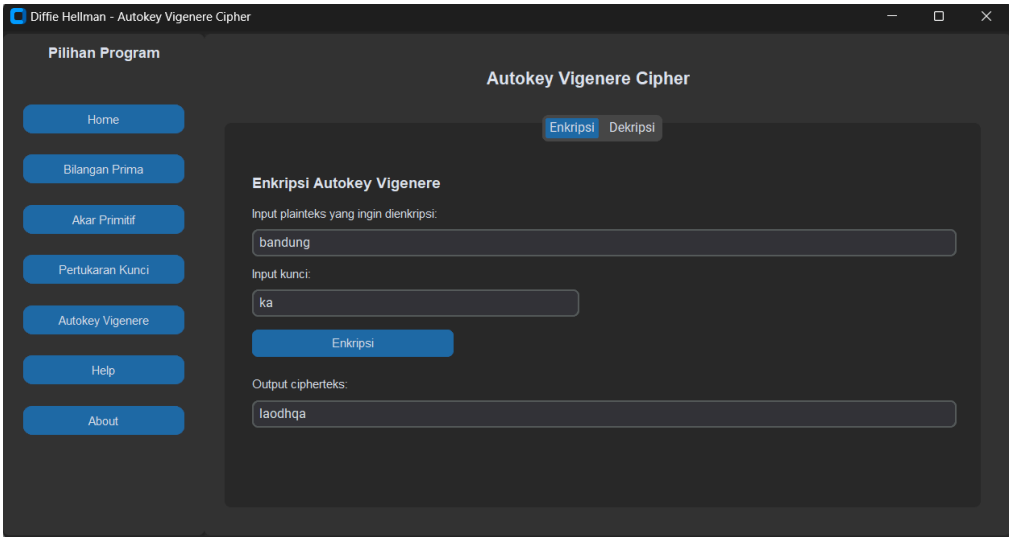
K	10		
Membangun Kunci Autokey Vigenere dari nilai K			
Block	Chunk	ASCII	KEY
1	10	107	k
2	0	97	a
3			
4			
5			

Gambar 4.8 Periksa Kunci Autokey Vigenere

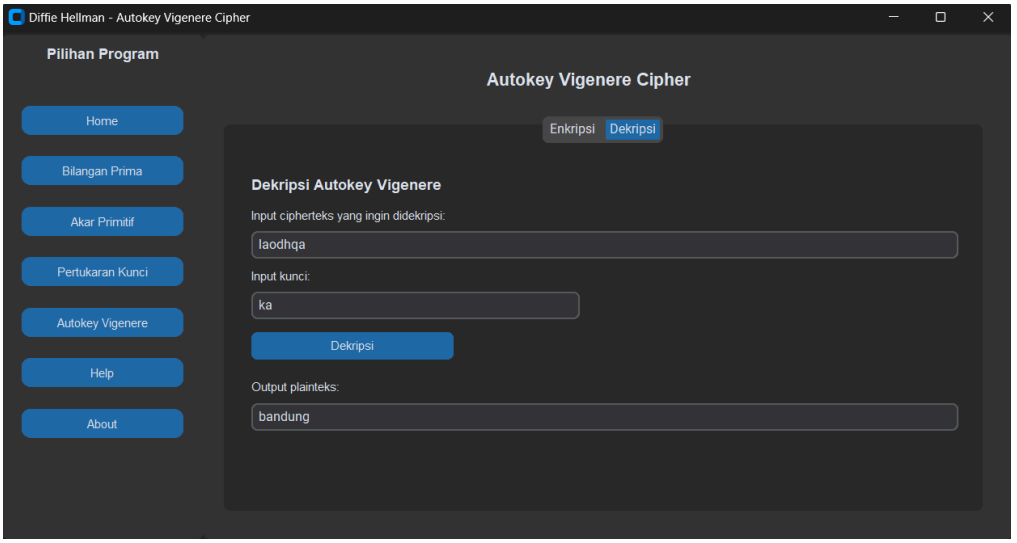
4. Autokey Vigenere

Aplikasi Diffie Hellman - Autokey Vigenere

Selanjutnya untuk mengenkripsi pesan dengan kunci Autokey Vigenere, dengan memasukkan plainteksnya yaitu ‘bandung’ dan kuncinya ‘ka’, berdasarkan aplikasi diperoleh cipherteksnya adalah ‘laodhqa’ seperti pada **Gambar 4.9**. Begitupun sebaliknya pada saat mendekripsi pesan dengan menggunakan kunci yang sama dan cipherteksnya ‘laodhqa’ maka diperoleh plainteksnya adalah ‘bandung’ seperti pada **Gambar 4.10**.



Gambar 4.9 Enkripsi Autokey Vigenere Cipher



Gambar 4.10 Dekripsi Autokey Vigenere Cipher

Excel

Sama halnya pada aplikasi tersebut untuk memvalidasi bagian enkripsi, perlu memasukkan plainteknya dan kuncinya terlebih dahulu (misal plainteks:bandung; kunci:ka). Pada Autokey Vigenere, jika panjang kunci lebih kecil dari panjang plainteks, maka kunci disambung dengan plainteks tersebut. Dimana panjang kunci yang baru akan sama dengan panjang plainteks, sehingga diperoleh kunci yang baru adalah ‘kabandu’ (Lihat **Gambar 4.12**). Dari kunci baru tersebut selanjutnya dienkripsi dengan plainteksnya sehingga diperoleh cipherteks yang sama dengan aplikasi diatas yaitu ‘laodhqa’.

Enkripsi		Length
Plainteks	bandung	7
KEY	ka	2
key	kabandu	7
Cipherteks	laodhqa	7

Gambar 4.11 Periksa Enkripsi Autokey Vigenere Cipher

Sedangkan untuk mendekripsi cipherteks perlu memasukkan cipherteksnya yaitu ‘laodhqa’ dan dengan kunci baru yang sama yaitu ‘kabandu’, sehingga diperoleh plainteksnya yaitu ‘bandung’ (Lihat **Gambar 4.12**).

Dekripsi		Length
Cipherteks	laodhqa	7
KEY	ka	2
key	kabandu	7
Plainteks	bandung	7

Gambar 4.12 Periksa Dekripsi Autokey Vigenere Cipher

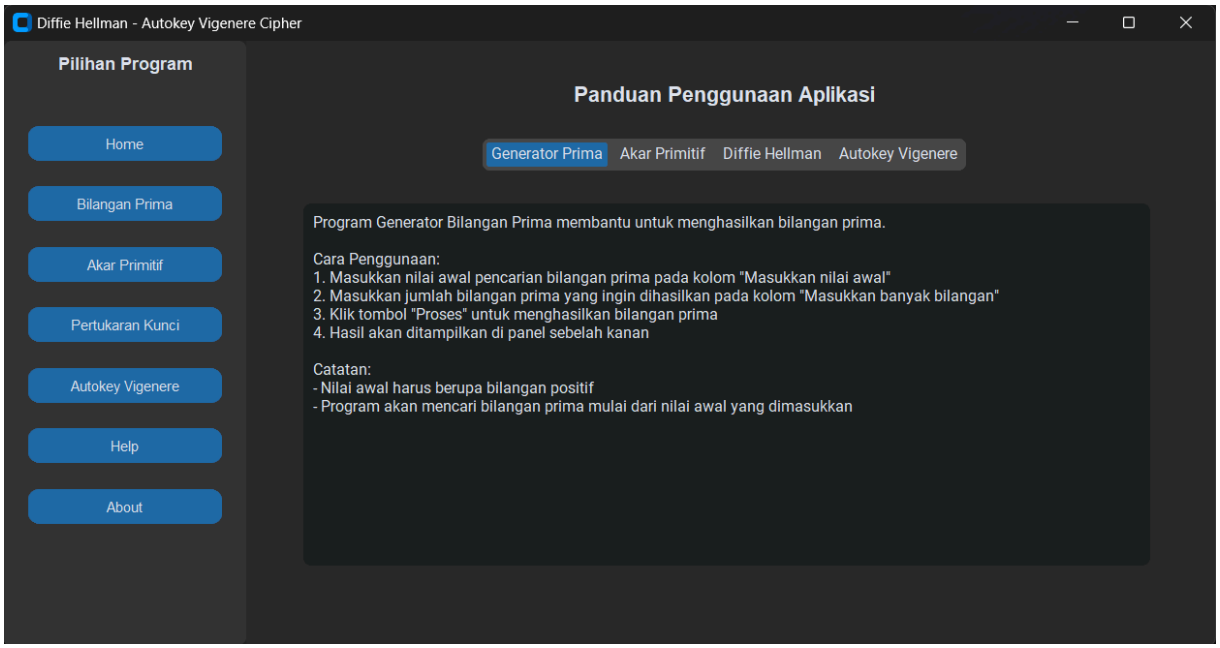
3.5. Petunjuk Penggunaan

Petunjuk penggunan terdapat pada bagian Help sebagai berikut:

Program Generator Bilangan Prima membantu untuk menghasilkan bilangan prima.
Cara Penggunaan:

1. Masukkan nilai awal pencarian bilangan prima pada kolom "Masukkan nilai awal"
2. Masukkan jumlah bilangan prima yang ingin dihasilkan pada kolom "Masukkan banyak bilangan"
3. Klik tombol "Proses" untuk menghasilkan bilangan prima
4. Hasil akan ditampilkan di panel sebelah kanan

Catatan:
- Nilai awal harus berupa bilangan positif
- Program akan mencari bilangan prima mulai dari nilai awal yang dimasukkan

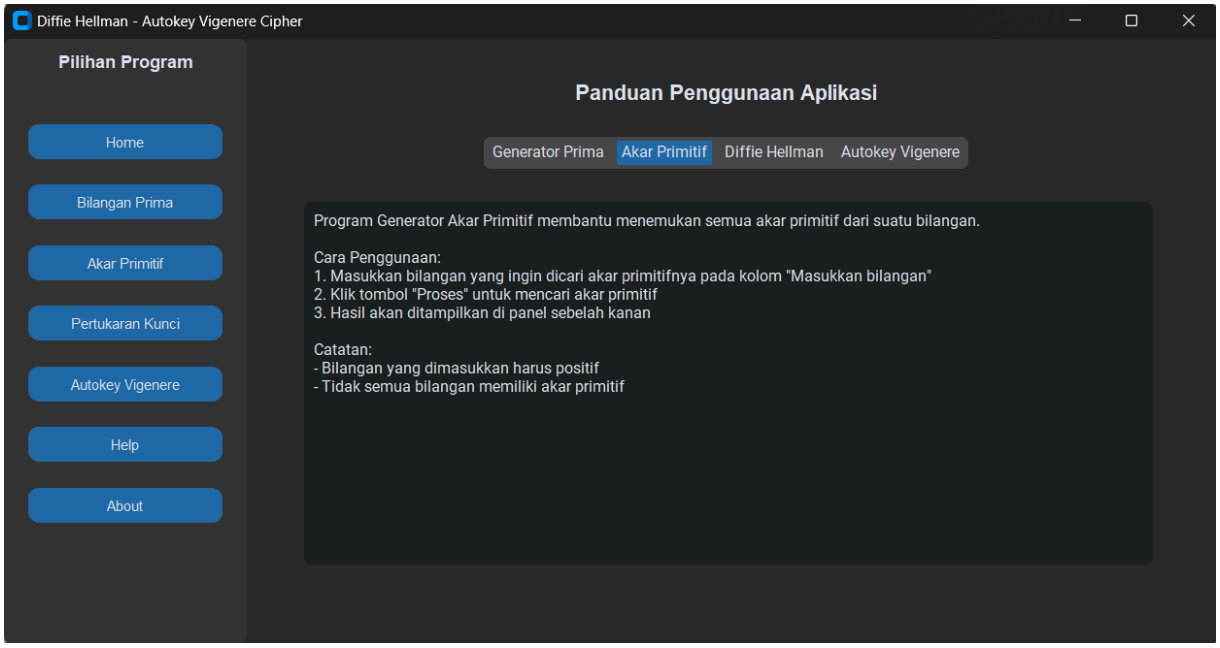


Gambar 5.1 Panduan Penggunaan Aplikasi Generator Prima

Program Generator Akar Primitif membantu menemukan semua akar primitif dari suatu bilangan.
Cara Penggunaan:

1. Masukkan bilangan yang ingin dicari akar primitifnya pada kolom "Masukkan bilangan"
2. Klik tombol "Proses" untuk mencari akar primitif
3. Hasil akan ditampilkan di panel sebelah kanan

Catatan:
- Bilangan yang dimasukkan harus positif
- Tidak semua bilangan memiliki akar primitif

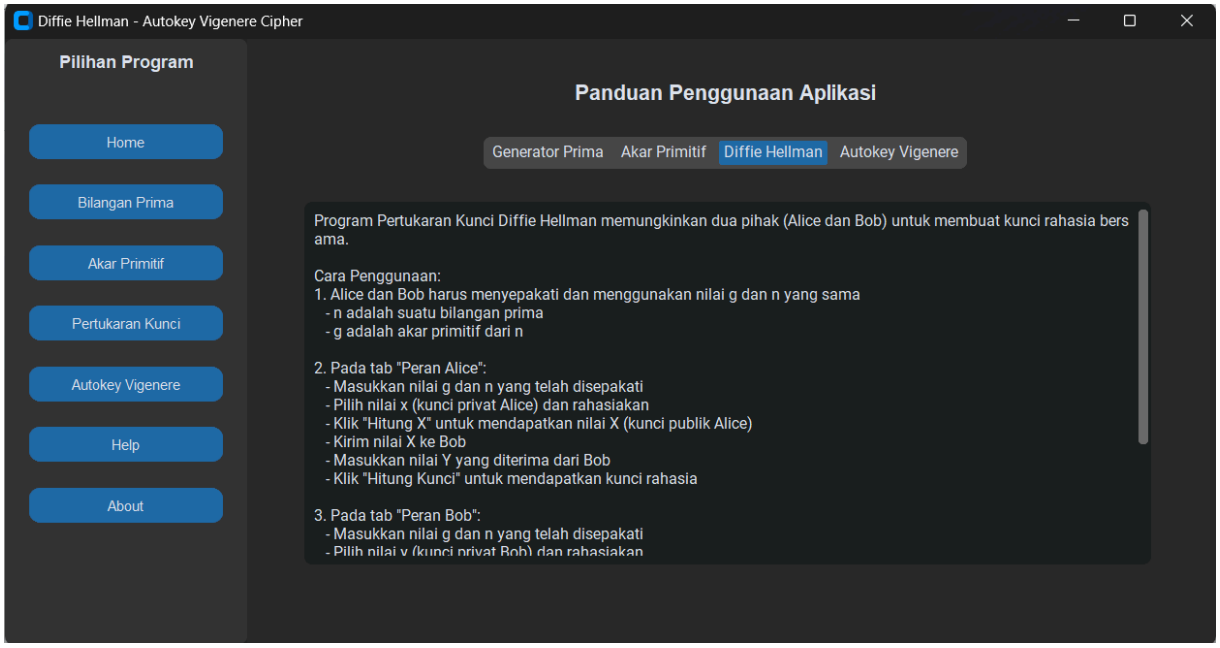


Gambar 5.2 Panduan Penggunaan Aplikasi Akar Primitif

Program Pertukaran Kunci Diffie Hellman memungkinkan dua pihak (Alice dan Bob) untuk membuat kunci rahasia bersama.
Cara Penggunaan:

1. Alice dan Bob harus menyepakati dan menggunakan nilai g dan n yang sama
   - n adalah suatu bilangan prima
   - g adalah akar primitif dari n
2. Pada tab "Peran Alice":
   - Masukkan nilai g dan n yang telah disepakati
   - Pilih nilai x (kunci privat Alice) dan rahasiakan
   - Klik "Hitung X" untuk mendapatkan nilai X (kunci publik Alice)
   - Kirim nilai X ke Bob
   - Masukkan nilai Y yang diterima dari Bob
   - Klik "Hitung Kunci" untuk mendapatkan kunci rahasia
3. Pada tab "Peran Bob":
   - Masukkan nilai g dan n yang telah disepakati
   - Pilih nilai y (kunci privat Bob) dan rahasiakan
   - Klik "Hitung Y" untuk mendapatkan nilai Y (kunci publik Bob)
   - Kirim nilai Y ke Alice
   - Masukkan nilai X yang diterima dari Alice
   - Klik "Hitung Kunci" untuk mendapatkan kunci rahasia

Catatan:
- Kunci privat (x dan y) harus dirahasiakan
- Kunci publik (X dan Y) aman untuk dikirim
- Jika prosesnya benar, Alice dan Bob akan mendapatkan kunci rahasia yang sama



Gambar 5.3 Panduan Penggunaan Aplikasi Diffie Hellman

Program Autokey Vigenere Cipher digunakan untuk mengenkripsi dan mendekripsi pesan menggunakan kunci dari hasil pertukaran Diffie Hellman.

Cara Penggunaan Enkripsi:

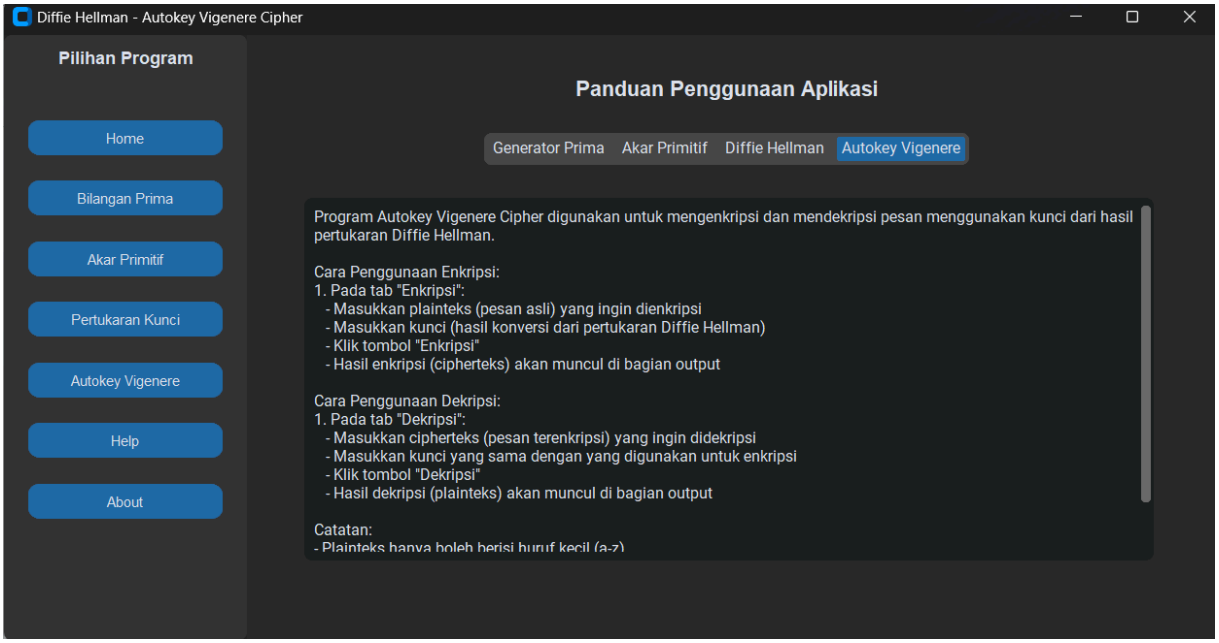
1. Pada tab "Enkripsi":
   - Masukkan plainteks (pesan asli) yang ingin dienkripsi
   - Masukkan kunci (hasil konversi dari pertukaran Diffie Hellman)
   - Klik tombol "Enkripsi"
   - Hasil enkripsi (cipherteks) akan muncul di bagian output

Cara Penggunaan Dekripsi:

1. Pada tab "Dekripsi":
   - Masukkan cipherteks (pesan terenkripsi) yang ingin didekripsi
   - Masukkan kunci yang sama dengan yang digunakan untuk enkripsi
   - Klik tombol "Dekripsi"
   - Hasil dekripsi (plainteks) akan muncul di bagian output

Catatan:

- Plainteks hanya boleh berisi huruf kecil (a-z)
- Spasi dan karakter khusus tidak diperbolehkan
- Kunci harus sama untuk enkripsi dan dekripsi
- Metode Autokey Vigenere menggunakan plainteks sebagai bagian dari kunci



Gambar 5.4 Panduan Penggunaan Aplikasi Autokey Vigenere

DAFTAR PUSTAKA

Diffie, W., & Hellman, M. (1976). New directions in cryptography. IEEE Transactions on Information Theory, 22(6), 644–654. https://doi.org/10.1109/TIT.1976.1055638

Kahn, D. (1996). The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet. Scribner.

Singh, S. (1999). The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography. Doubleday.

Munir, R. (2016). Kriptografi. Bandung: Informatika.

Burton, D. M. (2011). Elementary number theory (7th ed.). McGraw-Hill.

LAMPIRAN

Coding

PROGRAM APLIKASI KRIPTOGRAFI DIFFIE-HELLMAN.py > laman_help

```python
#PROGRAM APLIKASI KRIPTOGRAFI DIFFIE HELLMAN - AUTOKEY VIGENERE CIPHER

from customtkinter import *
import math

#--- Form ---
root = CTk()
root.title('Diffie Hellman - Autokey Vigenere Cipher')
root.geometry('1000x500')
set_appearance_mode("dark")
set_default_color_theme("blue")

# --- Fungsi Cek Prima ---
def cek_prima(n):
    if n < 2:
        return False
    for i in range(2, int(math.sqrt(n) + 1)):
        if n % i == 0:
            return False
    return True

# --- Fungsi Generator Bilangan Prima ---
def generate_prima(awal, jumlah):
    prima = []
    n = awal
    while len(prima) < jumlah:
        if cek_prima(n):
            prima.append(n)
        n += 1
    return prima

# --- Fungsi Proses Generator Bilangan Prima ---
def proses_generator_prima(entry_inputawal, entry_jumlah, textbox_hasil):
    awal = int(entry_inputawal.get())
    jumlah = int(entry_jumlah.get())
    if jumlah < 1:
        textbox_hasil.delete("0.0", END)
        textbox_hasil.insert(END, 'Jumlah harus lebih dari 0.')
    else:
        prima = generate_prima(awal, jumlah)
        hasil = f"Diperoleh {len(prima)} bilangan prima mulai dari {awal}:\n\n"
        for i in range(len(prima)):
            hasil += f"{prima[i]}\n"
        textbox_hasil.delete("0.0", END)
        textbox_hasil.insert("0.0", hasil)

# --- Fungsi Totient ---
def totient(n):
    hasil = n
    i = 2
    while i * i <= n:
        if n % i == 0:
            while n % i == 0:
                n //= i
            hasil -= hasil // i
        i += 1
    if n > 1:
        hasil -= hasil // n
    return hasil

# --- Fungsi Faktor Prima ---
def faktor_prima(n):
    faktorprima = []
    i = 2
    while i * i <= n:
        if n % i == 0:
            count = 0
            while n % i == 0:
                n //= i
                count += 1
            faktorprima.append((i, count))
        i += 1
    if n > 1:
        faktorprima.append((n, 1))
    return faktorprima

# --- Fungsi Order ---
def order(a, n):
    if math.gcd(a, n) != 1:
        return 0
    hasil = totient(n)
    faktorprima = faktor_prima(hasil)
    for p, e in faktorprima:
        while hasil % p == 0 and pow(a, hasil // p, n) == 1:
            hasil //= p
    return hasil
```

```python

# --- Fungsi Faktor Prima ---
def faktor_prima(n):
    faktorprima = []
    i = 2
    while i * i <= n:
        if n % i == 0:
            count = 0
            while n % i == 0:
                n //= i
                count += 1
            faktorprima.append((i, count))
        i += 1
    if n > 1:
        faktorprima.append((n, 1))
    return faktorprima

# --- Fungsi Order ---
def order(a, n):
    if math.gcd(a, n) != 1:
        return 0
    hasil = totient(n)
    faktorprima = faktor_prima(hasil)
    for p, e in faktorprima:
        while hasil % p == 0 and pow(a, hasil // p, n) == 1:
            hasil //= p
    return hasil

# --- Fungsi Akar Primitif ---
def punya_akar_primitif(n):
    if n <= 0:
        return False
    if n == 1 or n == 2 or n == 4:
        return True
    if n % 4 == 0:
        return False
    faktor = faktor_prima(n)
    if len(faktor) == 1 and faktor[0][0] > 2:
        return True
    if len(faktor) == 2 and faktor[0] == (2, 1):
        return True
    return False
```

```python

# --- Fungsi Hitung Semua Akar Primitif ---
def hitung_akar_primitif(n: int) -> list:
    if not punya_akar_primitif(n):
        return []
    phi = totient(n)
    akarprimitif = []
    for a in range(1, n):
        if math.gcd(a, n) == 1 and order(a, n) == phi:
            akarprimitif.append(a)
    return akarprimitif

# --- Fungsi Proses hitung Akar Primitif ---
def proses_hitung_akar_primitif(entry_inputbilangan, textbox_hasil):
    p = int(entry_inputbilangan.get())
    if p < 2:
        textbox_hasil.delete("0.0", END)
        textbox_hasil.insert(END, 'Bilangan harus lebih dari 1.')
    else:
        akarprimitif = hitung_akar_primitif(p)
        hasil = f"Diperoleh {len(akarprimitif)} akar primitif untuk p = {p}:\n\n"
        for i in range(len(akarprimitif)):
            hasil += f"{akarprimitif[i]}\n"
        textbox_hasil.delete("0.0", END)
        textbox_hasil.insert("0.0", hasil)

# --- Fungsi Hitung Kunci Publik ---
def hitung_kunci_publik(KunciPrivat, g, n):
    KunciPublik = pow(g, KunciPrivat, n)
    return KunciPublik

# --- Fungsi Hitung Kunci Simetri ---
def hitung_kunci_simetri(KunciPublik, KunciPrivat, n):
    K = pow(KunciPublik, KunciPrivat, n)
    KunciRahasia = K
    return KunciRahasia

# --- Fungsi Proses Hitung Kunci Publik ---
def proses_hitung_kunci_publik(entry_inputkunciprivat, entry_inputg, entry_inputn, entry_outputkuncipublik):
    kunciprivat = int(entry_inputkunciprivat.get())
    g = int(entry_inputg.get())
    n = int(entry_inputn.get())
    if kunciprivat < 1 or g < 1 or n < 1:
        entry_outputkuncipublik.delete(0, END)
        entry_outputkuncipublik.insert(END, 'Kunci privat, g, dan n harus lebih dari 0.')
```

```
    else:
        kuncipublik = hitung_kunci_publik(kunciprivat, g, n)
        entry_outputkuncipublik.delete(0, END)
        entry_outputkuncipublik.insert(END, kuncipublik)

# --- Fungsi Proses Hitung Kunci Simetri ---
def proses_hitung_kunci_simetri(entry_inputkuncipublik, entry_inputkunciprivat, entry_inputn, entry_outputkuncisimetri):
    kuncipublik = int(entry_inputkuncipublik.get())
    kunciprivat = int(entry_inputkunciprivat.get())
    n = int(entry_inputn.get())
    if kuncipublik < 1 or kunciprivat < 1 or n < 1:
        entry_outputkuncisimetri.delete(0, END)
        entry_outputkuncisimetri.insert(END, 'Kunci publik, kunci privat, dan n harus lebih dari 0.')
    else:
        kuncisimetri = hitung_kunci_simetri(kuncipublik, kunciprivat, n)
        entry_outputkuncisimetri.delete(0, END)
        entry_outputkuncisimetri.insert(END, kuncisimetri)

# --- Fungsi Konversi Kunci Simetri ---
def konversi_kunci_simetri(kuncisimetri):
    kunci_str = str(kuncisimetri)
    blok = []
    for i in range(len(kunci_str)):
        if i == len(kunci_str) - 1:
            blok.append(int(kunci_str[i]))
        else:
            blok.append(int(kunci_str[i:i+2]))
    hasil = ''
    for n in blok:
        hasil = hasil + chr(n % 26 + 97)
    return hasil

# --- Fungsi Proses Konversi Kunci Simetri ---
def proses_konversi_kunci_simetri(entry_inputkuncisimetri, entry_outputkuncikonversi):
    kuncisimetri = int(entry_inputkuncisimetri.get())
    kunciAutokeyVigenere = konversi_kunci_simetri(kuncisimetri)
    entry_outputkuncikonversi.delete(0, END)
    entry_outputkuncikonversi.insert(END, kunciAutokeyVigenere)

# --- Fungsi Pembangkit Kunci Enkripsi ---
def pembangkit_kunci_enkripsi(plainteks, kunciAutokeyVigenere):
    kunci = kunciAutokeyVigenere
    for i in range(len(plainteks) - len(kunciAutokeyVigenere)):
        kunci = kunci + plainteks[i]
    return kunci
```

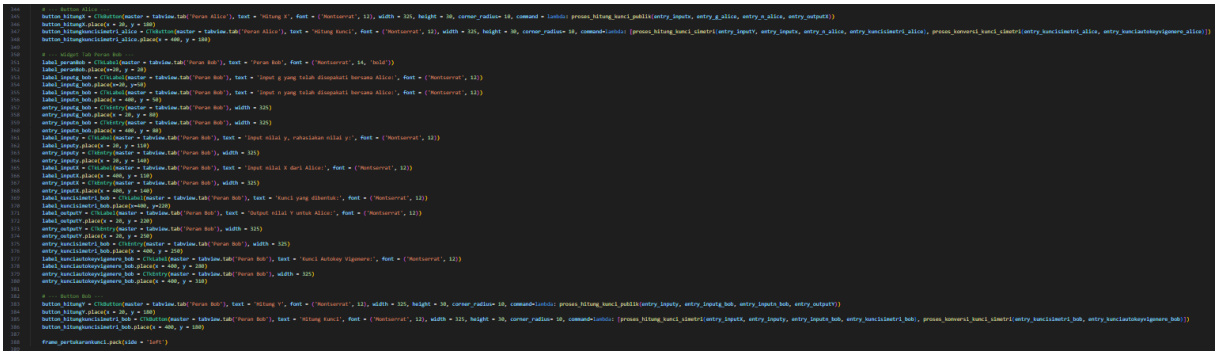
```

# --- Fungsi Pembangkit Kunci Dekripsi
def pembangkit_kunci_dekripsi(cipherteks, kunciAutokeyVigenere):
    kunci = kunciAutokeyVigenere
    for i in range(len(cipherteks) - len(kunciAutokeyVigenere)):
        kunci = kunci + chr(((ord(cipherteks[i]) - 97) - (ord(kunci[i]) - 97)) % 26 + 97)
    return kunci

# --- Fungsi Enkripsi Autokey Vigenere ---
def enkripsi_AutokeyVigenere(entry_plainteks, entry_kunciAutokeyVigenere, entry_cipherteks):
    plainteks = entry_plainteks.get()
    kunciAutokeyVigenere = entry_kunciAutokeyVigenere.get()
    kunci = pembangkit_kunci_enkripsi(plainteks, kunciAutokeyVigenere)
    cipherteks = ''
    for i in range(len(plainteks)):
        cipherteks = cipherteks + chr(((ord(plainteks[i]) - 97) + (ord(kunci[i]) - 97)) % 26 + 97)
    entry_cipherteks.delete(0, END)
    entry_cipherteks.insert(END, cipherteks)
    return cipherteks

def dekripsi_AutokeyVigenere(entry_cipherteks, entry_kunciAutokeyVigenere, entry_plainteks):
    cipherteks = entry_cipherteks.get()
    kunciAutokeyVigenere = entry_kunciAutokeyVigenere.get()
    kunci = pembangkit_kunci_dekripsi(cipherteks, kunciAutokeyVigenere)
    plainteks = ''
    for i in range(len(cipherteks)):
        plainteks = plainteks + chr(((ord(cipherteks[i]) - 97) - (ord(kunci[i]) - 97)) % 26 + 97)
    entry_plainteks.delete(0, END)
    entry_plainteks.insert(END, plainteks)
    return plainteks

# --- Frame Utama ---
frame_utama = CTkFrame(master = root, width = 800, height = 500)
frame_utama.propagate(False)

# --- Laman Home ---
def laman_home():
    frame_home = CTkFrame(master = frame_utama, width = 800, height = 500, fg_color="transparent")
    frame_home.propagate(False)
```

```
231
232     # --- Widget Home ---
233     label_judul = CTkLabel(master = frame_home, text = 'Kriptografi Diffie Hellman - Autokey Vigenere Cipher', font = ('Montserrat', 18, 'bold'))
234     label_judul.place(x=180, y=50)
235     label_keterangan = CTkLabel(master = frame_home, text = 'Terdapat 4 jenis program:', font = ('Montserrat', 14, 'bold'))
236     label_keterangan.place(x=140, y=100)
237     label_bilanganprima = CTkLabel(master = frame_home, text = '1. Generator Bilangan Prima', font = ('Montserrat', 12))
238     label_bilanganprima.place(x=140, y=130)
239     label_akarprimitif = CTkLabel(master = frame_home, text = '2. Generator Akar Primitif', font = ('Montserrat', 12))
240     label_akarprimitif.place(x=140, y=160)
241     label_pertukarankunci = CTkLabel(master = frame_home, text = '3. Pertukaran Kunci Diffie Hellman', font = ('Montserrat', 12))
242     label_pertukarankunci.place(x=140, y=190)
243     label_AutokeyVigenere = CTkLabel(master = frame_home, text = '4. Enkripsi dan Dekripsi menggunakan Autokey Vigenere Cipher', font = ('Montserrat', 12))
244     label_AutokeyVigenere.place(x=140, y=220)
245     label_penutup = CTkLabel(master = frame_home, text = 'Silakan pilih salah satu program pada tombol di samping.', font = ('Montserrat', 12))
246     label_penutup.place(x=140, y=260)
247
248     frame_home.pack(side = 'left')
249
250 # --- Laman Bilangan Prima ---
251 def laman_bilanganprima():
252     frame_bilanganprima = CTkFrame(master = frame_utama, width = 800, height = 500, fg_color="transparent")
253     frame_bilanganprima.propagate(False)
254
255     # --- Widget Bilangan Prima ---
256     label_judulbilanganprima = CTkLabel(master = frame_bilanganprima, text = 'Generator Bilangan Prima', font = ('Montserrat', 18, 'bold'))
257     label_judulbilanganprima.place(x=270, y=30)
258     label_inputawal = CTkLabel(master = frame_bilanganprima, text = 'Masukkan nilai awal:', font = ('Montserrat', 12))
259     label_inputawal.place(x=60, y=100)
260     entry_inputawal = CTkEntry(master = frame_bilanganprima, width = 200)
261     entry_inputawal.place(x=60, y=130)
262     label_inputbanyak = CTkLabel(master = frame_bilanganprima, text = 'Masukkan banyak bilangan:', font = ('Montserrat', 12))
263     label_inputbanyak.place(x=60, y=160)
264     entry_inputbanyak = CTkEntry(master = frame_bilanganprima, width = 200)
265     entry_inputbanyak.place(x=60, y=190)
266     label_hasil = CTkLabel(master = frame_bilanganprima, text = 'Hasil:', font = ('Montserrat', 12))
267     label_hasil.place(x=300, y=100)
268     textbox_hasil = CTkTextbox(master = frame_bilanganprima, width=470, height=350, scrollbar_button_color='#333333')
269     textbox_hasil.place(x=300, y=130)
270
```

```
271     # --- Button Proses ---
272     button_proses = CTkButton(master = frame_bilanganprima, text = 'Proses', font = ('Montserrat', 12), width = 160, height = 30, corner_radius= 10, command=lambda: proses_generator_prima(entry_inputawal, entry_inputbanyak, textbox_hasil))
273     button_proses.place(x = 60, y = 230)
274
275     frame_bilanganprima.pack(side = 'left')
```

```
282     # --- Widget Akar Primitif ---
283     label_judulakarprimitif = CTkLabel(master = frame_akarprimitif, text = 'Generator Akar Primitif', font = ('Montserrat', 18, 'bold'))
284     label_judulakarprimitif.place(x=270, y=30)
285     label_bilangan = CTkLabel(master = frame_akarprimitif, text = 'Masukkan bilangan:', font = ('Montserrat', 12))
286     label_bilangan.place(x=60, y=100)
287     entry_inputbilangan = CTkEntry(master = frame_akarprimitif, width = 200)
288     entry_inputbilangan.place(x=60, y=130)
289     label_hasil = CTkLabel(master = frame_akarprimitif, text = 'Akar Primitif:', font = ('Montserrat', 12))
290     label_hasil.place(x=300, y=100)
291     textbox_hasil = CTkTextbox(master = frame_akarprimitif, width=470, height=350, scrollbar_button_color='#333333')
292     textbox_hasil.place(x=300, y=130)
293
294     # --- Button Proses ---
295     button_proses = CTkButton(master = frame_akarprimitif, text = 'Proses', font = ('Montserrat', 12), width = 160, height = 30, corner_radius= 10, command=lambda: proses_hitung_akar_primitif(entry_inputbilangan, textbox_hasil))
296     button_proses.place(x = 60, y = 170)
297
298     frame_akarprimitif.pack(side = 'left')
```

```
300 # --- Laman Pertukaran Kunci ---
301 def laman_pertukarankunci():
302     frame_pertukarankunci = CTkFrame(master = frame_utama, width = 800, height = 500)
303     frame_pertukarankunci.propagate(False)
304
305     label_judulpertukarankunci = CTkLabel(master = frame_pertukarankunci, text = 'Pertukaran Kunci Diffie Hellman', font = ('Montserrat', 18, 'bold'))
306     label_judulpertukarankunci.place(x=250, y=30)
307     tabview = CTkTabview(master = frame_pertukarankunci, width = 750, height = 400)
308     tabview.place(x = 20, y = 70)
309     tabview.add('Peran Alice')
310     tabview.add('Peran Bob')
311
312     # --- Widget Tab Peran Alice ---
313     label_peranAlice = CTkLabel(master = tabview.tab('Peran Alice'), text = 'Peran Alice', font = ('Montserrat', 14, 'bold'))
314     label_peranAlice.place(x=20, y = 20)
315     label_inputg_alice = CTkLabel(master = tabview.tab('Peran Alice'), text = 'Input g yang telah disepakati bersama Bob:', font = ('Montserrat', 12))
316     label_inputg_alice.place(x=20, y=50)
317     label_inputn_alice = CTkLabel(master = tabview.tab('Peran Alice'), text = 'Input n yang telah disepakati bersama Bob:', font = ('Montserrat', 12))
318     label_inputn_alice.place(x = 400, y = 50)
319     entry_g_alice = CTkEntry(master = tabview.tab('Peran Alice'), width = 325)
320     entry_g_alice.place(x = 20, y = 80)
321     entry_n_alice = CTkEntry(master = tabview.tab('Peran Alice'), width = 325)
322     entry_n_alice.place(x = 400, y = 80)
323     label_inputx = CTkLabel(master = tabview.tab('Peran Alice'), text = 'Input nilai x, rahasiakan nilai x:', font = ('Montserrat', 12))
324     label_inputx.place(x = 20, y = 110)
325     entry_inputx = CTkEntry(master = tabview.tab('Peran Alice'), width = 325)
326     entry_inputx.place(x = 20, y = 140)
327     label_inputY = CTkLabel(master = tabview.tab('Peran Alice'), text = 'Input nilai Y dari Bob:', font = ('Montserrat', 12))
328     label_inputY.place(x = 400, y = 110)
329     entry_inputY = CTkEntry(master = tabview.tab('Peran Alice'), width = 325)
330     entry_inputY.place(x = 400, y = 140)
331     label_kuncisimetri_alice = CTkLabel(master = tabview.tab('Peran Alice'), text = 'Kunci yang dibentuk:', font = ('Montserrat', 12))
332     label_kuncisimetri_alice.place(x=400, y=220)
333     entry_kuncisimetri_alice = CTkEntry(master = tabview.tab('Peran Alice'), width = 325)
334     entry_kuncisimetri_alice.place(x = 400, y = 250)
335     label_outputX = CTkLabel(master = tabview.tab('Peran Alice'), text = 'Output nilai X untuk Bob:', font = ('Montserrat', 12))
336     label_outputX.place(x = 20, y = 220)
337     entry_outputX = CTkEntry(master = tabview.tab('Peran Alice'), width = 325)
338     entry_outputX.place(x = 20, y = 250)
339     label_kunciautokeyvigenere_alice = CTkLabel(master = tabview.tab('Peran Alice'), text = 'Kunci Autokey Vigenere:', font = ('Montserrat', 12))
340     label_kunciautokeyvigenere_alice.place(x = 400, y = 280)
341     entry_kunciautokeyvigenere_alice = CTkEntry(master = tabview.tab('Peran Alice'), width = 325)
342     entry_kunciautokeyvigenere_alice.place(x = 400, y = 310)
```



```
    # --- Button Alice ---
    button_hitungX = CTkButton(master = tabview.tab('Poran Alice'), text = 'Hitung X', font = ('Montserrat', 12), width = 325, height = 30, corner_radius= 10, command = lambda: proses_hitung_kunci_publik(entry_inputx, entry_g_alice, entry_n_alice, entry_outputX))
    button_hitungX.place(x = 20, y = 180)
    button_hitungkuncisimetri_alice = CTkButton(master = tabview.tab('Poran Alice'), text = 'Hitung Kunci', font = ('Montserrat', 12), width = 325, height = 30, corner_radius= 10, command=lambda: [proses_hitung_kunci_simetri(entry_inputY, entry_inputx, entry_n_alice, entry_kuncisimetri_alice), proses_konversi_kunci_simetri(entry_kuncisimetri_alice, entry_kunciautokeysvigenere_alice)])
    button_hitungkuncisimetri_alice.place(x = 400, y = 180)

    # --- Widget Tab Poran Bob ---
    label_poranBob = CTkLabel(master = tabview.tab('Poran Bob'), text = 'Poran Bob', font = ('Montserrat', 14, 'bold'))
    label_poranBob.place(x=20, y = 20)
    label_inputg_bob = CTkLabel(master = tabview.tab('Poran Bob'), text = 'Input g yang telah disepakati bersama Alice:', font = ('Montserrat', 12))
    label_inputg_bob.place(x=20, y=50)
    label_inputn_bob = CTkLabel(master = tabview.tab('Poran Bob'), text = 'Input n yang telah disepakati bersama Alice:', font = ('Montserrat', 12))
    label_inputn_bob.place(x = 400, y = 50)
    entry_inputg_bob = CTkEntry(master = tabview.tab('Poran Bob'), width = 325)
    entry_inputg_bob.place(x = 20, y = 80)
    entry_inputn_bob = CTkEntry(master = tabview.tab('Poran Bob'), width = 325)
    entry_inputn_bob.place(x = 400, y = 80)
    label_inputy = CTkLabel(master = tabview.tab('Poran Bob'), text = 'Input nilai y, rahasiakan nilai y:', font = ('Montserrat', 12))
    label_inputy.place(x = 20, y = 110)
    entry_inputy = CTkEntry(master = tabview.tab('Poran Bob'), width = 325)
    entry_inputy.place(x = 20, y = 140)
    label_inputX = CTkLabel(master = tabview.tab('Poran Bob'), text = 'Input nilai X dari Alice:', font = ('Montserrat', 12))
    label_inputX.place(x = 400, y = 110)
    entry_inputX = CTkEntry(master = tabview.tab('Poran Bob'), width = 325)
    entry_inputX.place(x = 400, y = 140)
    label_kuncisimetri_bob = CTkLabel(master = tabview.tab('Poran Bob'), text = 'Kunci yang didentuk:', font = ('Montserrat', 12))
    label_kuncisimetri_bob.place(x=400, y=210)
    label_outputY = CTkLabel(master = tabview.tab('Poran Bob'), text = 'Output nilai Y untuk Alice:', font = ('Montserrat', 12))
    label_outputY.place(x = 20, y = 220)
    entry_outputY = CTkEntry(master = tabview.tab('Poran Bob'), width = 325)
    entry_outputY.place(x = 20, y = 250)
    entry_kuncisimetri_bob = CTkEntry(master = tabview.tab('Poran Bob'), width = 325)
    entry_kuncisimetri_bob.place(x = 400, y = 250)
    label_kunciautokeysvigenere_bob = CTkLabel(master = tabview.tab('Poran Bob'), text = 'Kunci Autokey Vigenere:', font = ('Montserrat', 12))
    label_kunciautokeysvigenere_bob.place(x = 400, y = 280)
    entry_kunciautokeysvigenere_bob = CTkEntry(master = tabview.tab('Poran Bob'), width = 325)
    entry_kunciautokeysvigenere_bob.place(x = 400, y = 310)

    # --- Button Bob ---
    button_hitungY = CTkButton(master = tabview.tab('Poran Bob'), text = 'Hitung Y', font = ('Montserrat', 12), width = 325, height = 30, corner_radius= 10, command=lambda: proses_hitung_kunci_publik(entry_inputy, entry_inputg_bob, entry_inputn_bob, entry_outputY))
    button_hitungY.place(x = 20, y = 180)
    button_hitungkuncisimetri_bob = CTkButton(master = tabview.tab('Poran Bob'), text = 'Hitung Kunci', font = ('Montserrat', 12), width = 325, height = 30, corner_radius= 10, command=lambda: [proses_hitung_kunci_simetri(entry_inputX, entry_inputy, entry_inputn_bob, entry_kuncisimetri_bob), proses_konversi_kunci_simetri(entry_kuncisimetri_bob, entry_kunciautokeysvigenere_bob)])
    button_hitungkuncisimetri_bob.place(x = 400, y = 180)

    frame_pertukarankunci.pack(side = 'left')
```

```
# --- Laman Autokey Vigenere ---
def laman_AutokeyVigenere():
    frame_AutokeyVigenere = CTkFrame(master = frame_utama, width = 800, height = 500)
    frame_AutokeyVigenere.propagate(False)

    label_JudulAutokeyVigenere = CTkLabel(master = frame_AutokeyVigenere, text = 'Autokey Vigenere Cipher', font = ('Montserrat', 18, 'bold'))
    label_JudulAutokeyVigenere.place(x=280, y=30)
    tabview = CTkTabview(master = frame_AutokeyVigenere, width = 750, height = 400)
    tabview.place(x = 20, y = 70)
    tabview.add('Enkripsi')
    tabview.add('Dekripsi')

    # --- Widget Tab Enkripsi ---
    label_enkripsi = CTkLabel(master = tabview.tab('Enkripsi'), text = 'Enkripsi Autokey Vigenere', font = ('Montserrat', 14, 'bold'))
    label_enkripsi.place(x=20, y=20)
    label_inputplainteks = CTkLabel(master = tabview.tab('Enkripsi'), text = 'Input plainteks yang ingin dienkripsi:', font = ('Montserrat', 12))
    label_inputplainteks.place(x = 20, y = 50)
    entry_inputplainteks = CTkEntry(master = tabview.tab('Enkripsi'), width = 700)
    entry_inputplainteks.place(x = 20, y = 80)
    label_inputkunci_enkripsi = CTkLabel(master = tabview.tab('Enkripsi'), text = 'Input kunci:', font = ('Montserrat', 12))
    label_inputkunci_enkripsi.place(x = 20, y = 110)
    entry_inputkunci_enkripsi = CTkEntry(master = tabview.tab('Enkripsi'), width = 325)
    entry_inputkunci_enkripsi.place(x = 20, y = 140)
    label_outputcipherteks = CTkLabel(master = tabview.tab('Enkripsi'), text = 'Output cipherteks:', font = ('Montserrat', 12))
    label_outputcipherteks.place(x = 20, y = 220)
    entry_outputcipherteks = CTkEntry(master = tabview.tab('Enkripsi'), width = 700)
    entry_outputcipherteks.place(x = 20, y = 250)

    # --- Button Enkripsi ---
    button_enkripsi = CTkButton(master = tabview.tab('Enkripsi'), text = 'Enkripsi', font = ('Montserrat', 12), width = 200, command=lambda: enkripsi_AutokeyVigenere(entry_inputplainteks, entry_inputkunci_enkripsi, entry_outputcipherteks))
    button_enkripsi.place(x = 20, y = 180)

    # --- Widget Tab Dekripsi ---
    label_dekripsi = CTkLabel(master = tabview.tab('Dekripsi'), text = 'Dekripsi Autokey Vigenere', font = ('Montserrat', 14, 'bold'))
    label_dekripsi.place(x=20, y=20)
    label_inputcipherteks = CTkLabel(master = tabview.tab('Dekripsi'), text = 'Input cipherteks yang ingin didekripsi:', font = ('Montserrat', 12))
    label_inputcipherteks.place(x = 20, y = 50)
    entry_inputcipherteks = CTkEntry(master = tabview.tab('Dekripsi'), width = 700)
    entry_inputcipherteks.place(x = 20, y = 80)
    label_inputkunci_dekripsi = CTkLabel(master = tabview.tab('Dekripsi'), text = 'Input kunci:', font = ('Montserrat', 12))
    label_inputkunci_dekripsi.place(x = 20, y = 110)
    entry_inputkunci_dekripsi = CTkEntry(master = tabview.tab('Dekripsi'), width = 325)
    entry_inputkunci_dekripsi.place(x = 20, y = 140)
    label_outputplainteks = CTkLabel(master = tabview.tab('Dekripsi'), text = 'Output plainteks:', font = ('Montserrat', 12))
    label_outputplainteks.place(x = 20, y = 220)
    entry_outputplainteks = CTkEntry(master = tabview.tab('Dekripsi'), width = 700)
    entry_outputplainteks.place(x = 20, y = 250)

    # --- Button Dekripsi ---
    button_dekripsi = CTkButton(master = tabview.tab('Dekripsi'), text = 'Dekripsi', font = ('Montserrat', 12), width = 200, command=lambda: dekripsi_AutokeyVigenere(entry_inputcipherteks, entry_inputkunci_dekripsi, entry_outputplainteks))
    button_dekripsi.place(x = 20, y = 180)

    frame_AutokeyVigenere.pack(side = 'left')
```

```python
# --- Laman Help ---
def laman_help():
    frame_help = CTkFrame(master = frame_utama, width = 800, height = 500, fg_color="transparent")
    frame_help.propagate(False)

    # --- Widget Help ---
    label_judulhelp = CTkLabel(master = frame_help, text = 'Panduan Penggunaan Aplikasi', font = ('Montserrat', 18, 'bold'))
    label_judulhelp.place(x=270, y=30)
    tabview = CTkTabview(master = frame_help, width = 750, height = 400)
    tabview.place(x = 20, y = 70)
    tabview.add('Generator Prima')
    tabview.add('Akar Primitif')
    tabview.add('Diffie Hellman')
    tabview.add('Autokey Vigenere')

    # --- Panduan Generator Bilangan Prima ---
    panduan_prima = """Program Generator Bilangan Prima membantu untuk menghasilkan bilangan prima.

Cara Penggunaan:
1. Masukkan nilai awal pencarian bilangan prima pada kolom "Masukkan nilai awal"
2. Masukkan jumlah bilangan prima yang ingin dihasilkan pada kolom "Masukkan banyak bilangan"
3. Klik tombol "Proses" untuk menghasilkan bilangan prima
4. Hasil akan ditampilkan di panel sebelah kanan

Catatan:
- Nilai awal harus berupa bilangan positif
- Program akan mencari bilangan prima mulai dari nilai awal yang dimasukkan"""

    textbox_prima = CTkTextbox(master = tabview.tab('Generator Prima'), width=700, height=300)
    textbox_prima.insert("0.0", panduan_prima)
    textbox_prima.configure(state="disabled")
    textbox_prima.place(x=20, y=20)

    # --- Panduan Generator Akar Primitif ---
    panduan_akarprimitif = """Program Generator Akar Primitif membantu menemukan semua akar primitif dari suatu bilangan.

Cara Penggunaan:
1. Masukkan bilangan yang ingin dicari akar primitifnya pada kolom "Masukkan bilangan"
2. Klik tombol "Proses" untuk mencari akar primitif
3. Hasil akan ditampilkan di panel sebelah kanan

Catatan:
- Bilangan yang dimasukkan harus positif
- Tidak semua bilangan memiliki akar primitif"""

    textbox_akarprimitif = CTkTextbox(master = tabview.tab('Akar Primitif'), width=700, height=300)
    textbox_akarprimitif.insert("0.0", panduan_akarprimitif)
    textbox_akarprimitif.configure(state="disabled")
    textbox_akarprimitif.place(x=20, y=20)

    # --- Panduan Pertukaran Kunci Diffie Hellman ---
    panduan_diffiehellman = """Program Pertukaran Kunci Diffie Hellman memungkinkan dua pihak (Alice dan Bob) untuk membuat kunci rahasia bersama.
```

```
Cara Penggunaan:
1. Alice dan Bob harus menyepakati dan menggunakan nilai g dan n yang sama
   - n adalah suatu bilangan prima
   - g adalah akar primitif dari n

2. Pada tab "Peran Alice":
   - Masukkan nilai g dan n yang telah disepakati
   - Pilih nilai x (kunci privat Alice) dan rahasiakan
   - Klik "Hitung X" untuk mendapatkan nilai X (kunci publik Alice)
   - Kirim nilai X ke Bob
   - Masukkan nilai Y yang diterima dari Bob
   - Klik "Hitung Kunci" untuk mendapatkan kunci rahasia

3. Pada tab "Peran Bob":
   - Masukkan nilai g dan n yang telah disepakati
   - Pilih nilai y (kunci privat Bob) dan rahasiakan
   - Klik "Hitung Y" untuk mendapatkan nilai Y (kunci publik Bob)
   - Kirim nilai Y ke Alice
   - Masukkan nilai X yang diterima dari Alice
   - Klik "Hitung Kunci" untuk mendapatkan kunci rahasia

Catatan:
- Kunci privat (x dan y) harus dirahasiakan
- Kunci publik (X dan Y) aman untuk dikirim
- Jika prosesnya benar, Alice dan Bob akan mendapatkan kunci rahasia yang sama"""

    textbox_diffiehellman = CTkTextbox(master = tabview.tab('Diffie Hellman'), width=700, height=300)
    textbox_diffiehellman.insert("0.0", panduan_diffiehellman)
    textbox_diffiehellman.configure(state="disabled")
    textbox_diffiehellman.place(x=20, y=20)

    # --- Panduan Autokey Vigenere Cipher ---
    panduan_autokeyvigenere = """Program Autokey Vigenere Cipher digunakan untuk mengenkripsi dan mendekripsi pesan menggunakan kunci dari hasil pertukaran Diffie Hellman.

Cara Penggunaan Enkripsi:
1. Pada tab "Enkripsi":
   - Masukkan plainteks (pesan asli) yang ingin dienkripsi
   - Masukkan kunci (hasil konversi dari pertukaran Diffie Hellman)
   - Klik tombol "Enkripsi"
   - Hasil enkripsi (cipherteks) akan muncul di bagian output

Cara Penggunaan Dekripsi:
1. Pada tab "Dekripsi":
   - Masukkan cipherteks (pesan terenkripsi) yang ingin didekripsi
   - Masukkan kunci yang sama dengan yang digunakan untuk enkripsi
   - Klik tombol "Dekripsi"
   - Hasil dekripsi (plainteks) akan muncul di bagian output

Catatan:
- Plainteks hanya boleh berisi huruf kecil (a-z)
- Spasi dan karakter khusus tidak diperbolehkan
- Kunci harus sama untuk enkripsi dan dekripsi
- Metode Autokey Vigenere menggunakan plainteks sebagai bagian dari kunci"""
```

```
    textbox_autokeyvigenere = CTkTextbox(master = tabview.tab('Autokey Vigenere'), width=700, height=300)
    textbox_autokeyvigenere .insert("0.0", panduan_autokeyvigenere)
    textbox_autokeyvigenere .configure(state="disabled")
    textbox_autokeyvigenere .place(x=20, y=20)

    frame_help.pack(side = 'left')

# --- Laman About ---
def laman_about():
    frame_about = CTkFrame(master = frame_utama, width = 800, height = 500, fg_color="transparent")
    frame_about.propagate(False)

    # --- Widget About ---
    label_judulabout = CTkLabel(master = frame_about, text = 'Tentang Aplikasi', font = ('Montserrat', 18, 'bold'))
    label_judulabout.place(x=330, y=30)
    frame_info = CTkFrame(master = frame_about, width = 700, height = 400)
    frame_info.place(x = 50, y = 70)
    label_nama_aplikasi = CTkLabel(master = frame_info, text = 'Aplikasi Kriptografi\nDiffie Hellman - Autokey Vigenere Cipher', font = ('Montserrat', 16, 'bold'))
    label_nama_aplikasi.place(x=190, y=20)
    label_versi = CTkLabel(master = frame_info, text = 'Versi 1.0.0', font = ('Montserrat', 12))
    label_versi.place(x=315, y=70)

    # Deskripsi Aplikasi
    tentang_deskripsi = """Aplikasi ini merupakan implementasi dari algoritma pertukaran kunci Diffie-Hellman
yang dikombinasikan dengan metode enkripsi Autokey Vigenere Cipher. Aplikasi ini
bertujuan untuk memfasilitasi proses pertukaran kunci yang aman antara dua pihak
dan menggunakan kunci tersebut untuk enkripsi pesan menggunakan Autokey Vigenere Cipher.

Fitur Utama:
• Generator Bilangan Prima
• Generator Akar Primitif
• Pertukaran Kunci Diffie-Hellman
• Enkripsi dan Dekripsi menggunakan Autokey Vigenere Cipher"""

    textbox_deskripsi = CTkTextbox(master = frame_info, width=600, height=120)
    textbox_deskripsi.insert("0.0", tentang_deskripsi)
    textbox_deskripsi.configure(state="disabled")
    textbox_deskripsi.place(x=50, y=100)

    label_pengembang = CTkLabel(master = frame_info, text = 'Dikembangkan oleh:', font = ('Montserrat', 14, 'bold'))
    label_pengembang.place(x=50, y=240)
    tentang_pengembang = """Fahmi Ismail Ramantoko               (2205676)
Muhammad Rizqi Winnel Adnin      (2209397)
Muhammad Thoriq Atallah             (2202991)
Shafira Mayora Handriyudha          (2202576)"""

    textbox_pengembang = CTkTextbox(master = frame_info, width=600, height=80)
    textbox_pengembang.insert("0.0", tentang_pengembang)
    textbox_pengembang.configure(state="disabled")
    textbox_pengembang.place(x=50, y=270)

    frame_about.pack(side = 'left')
```

```
# --- Fungsi sembunyikan frame ---
def sembunyikan_frame():
    for frame in frame_utama.winfo_children():
        frame.destroy()

# --- Fungsi munculkan frame ---
def munculkan(laman):
    sembunyikan_frame()
    laman()

# --- frame pilihan program ---
frame_pilihan = CTkFrame(master = root, width = 200, height = 500, fg_color="#333333")
frame_pilihan.propagate(False)

# --- Pilihan Program ---
labelPilihanProgram = CTkLabel(master = frame_pilihan, text = 'Pilihan Program', font = ('Montserrat', 14, 'bold'))
labelPilihanProgram.place(x = 45, y = 5)
button_home = CTkButton(master = frame_pilihan, text = 'Home', font = ('Montserrat', 12), width = 160, height = 30, corner_radius= 10, command = lambda: munculkan(laman_home))
button_home.place(x = 20, y = 70)
button_bilanganprima = CTkButton(master = frame_pilihan, text = 'Bilangan Prima', font = ('Montserrat', 12), width = 160, height = 30, corner_radius= 10, command = lambda: munculkan(laman_bilanganprima))
button_bilanganprima.place(x = 20, y = 120)
button_akarprimitif = CTkButton(master = frame_pilihan, text = 'Akar Primitif', font = ('Montserrat', 12), width = 160, height = 30, corner_radius= 10, command = lambda: munculkan(laman_akarprimitif))
button_akarprimitif.place(x = 20, y = 170)
button_pembangkitkunci = CTkButton(master = frame_pilihan, text = 'Pertukaran Kunci', font = ('Montserrat', 12), width = 160, height = 30, corner_radius= 10, command = lambda: munculkan(laman_pertukarankunci))
button_pembangkitkunci.place(x = 20, y = 220)
button_AutokeyVigenere = CTkButton(master = frame_pilihan, text = 'Autokey Vigenere', font = ('Montserrat', 12), width = 160, height = 30, corner_radius= 10, command = lambda: munculkan(laman_AutokeyVigenere))
button_AutokeyVigenere.place(x = 20, y = 270)
button_help = CTkButton(master = frame_pilihan, text = 'Help', font = ('Montserrat', 12), width = 160, height = 30, corner_radius= 10, command = lambda: munculkan(laman_help))
button_help.place(x = 20, y = 320)
button_about = CTkButton(master = frame_pilihan, text = 'About', font = ('Montserrat', 12), width = 160, height = 30, corner_radius= 10, command = lambda: munculkan(laman_about))
button_about.place(x = 20, y = 370)

frame_pilihan.pack(side = 'left')
frame_utama.pack(side = 'left')
laman_home()

root.mainloop()
```