Aplikasi *Random Number Generator* untuk Menghasilkan *Key* dalam Enkripsi Pesan Sederhana

Ilham Prasetyo Wibowo - 13520013[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13520013@stei.itb.ac.id

***Abstract*—Enkripsi adalah sebuah sebuah proses untuk mengamankan informasi tertentu. Caesar Cipher adalah sebuah algoritma enkripsi sederhana yang dulu digunakan oleh Julius Caesar untuk mengirim pesan. Caesar Cipher dapat diperkuat dengan menggunakan *random number generator* yang akan menghasilkan key untuk dekripsi pesan Caesar Cipher.**

***Keywords*—Enkripsi, Dekripsi, Cipher, Random Number.**

I. PENDAHULUAN

Pada abad ke-21, perkembangan teknologi semakin cepat. Jika tidak berhati-hati, informasi yang tersebar di internet bisa diakses oleh siapa saja.

Untuk menghindari data atau pesan terlihat oleh orang lain, bisa digunakan enkripsi pesan. Menurut Cloudfare, enkripsi adalah sebuah proses atau cara mengacak pesan agar bisa hanya bisa dibaca oleh orang yang memiliki aksesnya saja. Proses enkripsi membuat pesan (*plainteks)* menjadi sebuah tulisan acak. Kemudian tulisan acak tersebut bisa dikembalikan menjadi semula menggunakan algoritma tertentu dan kunci yang digunakan.

Caesar Cipher merupakan salah satu algoritma enkripsi populer yang dulu digunakan pada zaman Kaisar Julius Caesar dalam masa peperangan. Sekarang, kemanan algoritma Caesar Cipher sudah tidak bisa dijamin lagi. Oleh karena itu, menggunakan teori Pembangkit Angka Acak (*Random Number Generator)* untuk membuat sebuah kunci untuk memperkuat algoritma Caesar Cipher.

II. TEORI DASAR

2.1 Teori Bilangan

Teori bilangan adalah salah satu cabang matematika yang mempelajari tentang bilangan bulat dan sifat sifatnya. Bilangan bulat adalah bilangan yang tidak memiliki pecahan. Dalam teori bilangan, dipelajari pembagan, bilangan prima, FPB, algoritma untuk menentukan FPB (algoritma Euclidean), kekongruenan.

2.1.1 Bilangan Bulat

Bilangan bulat adalah bilangan yang tidak memiliki pecahan desimal. Bilangan bulat memiliki sifat "habis membagi" jika terdapat dua bilangan bulat yang jika dibagi tetap menghasilkan bilangan bulat. Sifat habis membagi dinotasikan

$$a \mid b$$

yang berarti bilangan bulat a membagi habis bilangan bulat b.

Teorema Euclidean menyatakan bahwa jika bilangan a dibagi bilangan b, maka terdapat sebuah bilangan q dan r dengan m adalah hasil pembagian dan r adalah sisa pembagian. Syarat sisa pembagian dalam hal ini adalah r lebih dari sama dengan nol dan kurang dari b.

2.1.2 Pembagi Bersama Terbesar

Pembagi bersama terbesar (PBB) atau biasanya disebut faktor persekutuan terbesar (FPB) adalah sebuah bilangan terbesar yang dapat membagi bilangan a dan b. Dengan demikian sebuah PBB d dengan d | a dan d | b.

Untuk mencari PBB dari dua buah bilangan bulat dapat menggunakan algoritma Euclidean. Algoritma Euclidean untuk mencari PBB dari a dan b diawali dengan mencari sisa pembagian kedua bilangan tersebut. Kemudian, dicari sisa pembagian dari b dan r(sisa pembagian pertama). Langkah tersebut diulang sampai menemukan sisa pembagian nol. Ketika mencapai sisa pembagian nol, sisa pembagian sebelumnya adalah PBB dari kedua bilangan. Secara garis besar, algoritma Euclidean dapat digambarkan sebagai berikut.

$$A.B = B.Q + R$$
$$B.R = R.Q1 + R1$$
$$R.R1 = R1.Q2 + 0$$
$$PBB(A,B) = R1$$

2.1.3 Kombinasi Linear

Sebuah nilai PBB dari a dan b yang diperoleh dari algortima Euclidean dapat dinyatakan sebagai kombinasi linear a dan b.Kombinasi linear diperoleh dengan melakukan substitusi mudur dari algoritma Euclidean.

2.1.4 Bilangan Relatif Prima

Bilangan Prima adalah bilangan yang hanya bisa dibagi 1 dan dirinya sendiri. Dua buah bilangan bulat a dan b disebut relatif prima jika PBB dari kedua bilangan tersebut adalah satu.

2.1.5 Modulo

Dalam matematika, operasi modulo memberikan sisa pembagian a dan m. Modulo dinotasikan sebagai berikut.

$$a \bmod b = c$$

Dari notasi diatas, c merupakaan sisa pembagian a dibagi b. Syarat c adalah bilangan bulat tak negatif dan kurang dari a. Jika

c adalah bilangan negatif, maka operasi modulo tersebut salah.

2.1.6 Kekongruenan

Sebuah bilangan bulat a disebut kongruen dengan bilangan b dalam mod m jika m membagi habis a-b.

$$a \equiv b \ (mod\ m)$$
$$a = b + km$$

Yang berarti a dan b kongruen dan modulus m.

Sifat-sifat kongruen :

A. a ≡ b (mod m)
   1. (a + c) ≡ (b + c) (mod m)
   2. ac ≡ bc (mod m)
   3. $a^p \equiv b^p$ (mod m), p ≥ 0

B. a ≡ b (mod m) dan c ≡ d (mod m)
   1. (a + c) ≡ (b+d) (mod m)
   2. ac = bd (mod m)

2.1.7 Invers Modulo

Sebuah bilangan a disebut invers dari bilangan b jika perkalian antara a dan b sama dengan 1. Syarat sebuah modulo memiliki invers adalah jika a mod b maka a dan b harus relatif prima dan m lebih dari 1. Sehingga invers dari a mod b dapat dinyatakan dengan

$$xa = 1 \ (mod\ b)$$

2.1.8 Kekongruenan Linear

Kekongruenan linear berbentuk ax ≡ b (mod m) dapat dicari nilai xnya, dengan

$$ax \equiv b \ (mod\ m) \rightarrow \ x = \frac{b+km}{a}$$

Nilai k dicoba mulai dari 0,1,2 sampai menghasilkan x bilangan bulat. Ketika diperoleh k yang menghasilkan x bilangan bulat, maka solusi dari kekongruenan linear tersebut adalah x (mod m) dan seluruh nilai x ditambah/dikurang m.

Sistem kekongruenan linear terdiri dari lebih satu kekongruenan. Dengan

$$x \equiv a \ (mod\ m)$$
$$x \equiv b \ (mod\ n)$$

Untuk mencari penyelesaian dari sistem kekongruenan linear tersebut dapat menggunakan cara diubah menjadi bentuk sama dengan, dan dilakukan substitusi sampai ditemukan nilai x yang memenuhi seluruh kekongruenan linear. Cara lainnya adalah menggunakan "Chinese Remainder Theorem".

Chinese Remainder Theorem

Jika terdapat sebuah sistem kekongruenan linear

$$x \equiv a1 \ (mod\ m1)$$
$$x \equiv a2 \ (mod\ m2)$$
$$x \equiv an \ (mod\ mn)$$

dan seluruh m adalah bilangan bulat positif memiliki PBB = 1, maka sistem kekongruenan linear memiliki solusi unik dalam modulus m = m1.m2.mn.

2.1.8 Bilangan Prima

Bilangan prima adalah bilangan yang lebih dari satu dan hanya bisa dibagi 1 dan dirinya sendiri. Seluruh bilangan prima yang ada adalah ganjil keculai bilangan 2. Bilangan yang bukan bilangan prima disebut bilangan komposit.

Fundamental Theorem of Arithmetic

Bilangan bulat positif yang lebih besar atau sama dengan dua dapat dinyatakan sebagai perkalian satu atau lebih bilangan prima.

Sehingga berdasarkan teorema tersebut, sebuah bilangan dapat dites bilangan prima atau bukan dengan algoritma

1. Cari akar n
2. Cari seluruh bilangan prima yang lebih kecil daripada akar n
3. Bagi akar n dengan bilangan prima yang lebih kecil. Jika terdapat bilangan prima yang membagi habis akar n, maka n bukanlah bilangan prima, melainkan bilangan komposit .

Fermat's Theorem

Jika a bilangan bulat, dan p adalah bilangan prima yang tidak membagi habis a sehingga PBB dari a dan p adalah 1, maka berlaku

$$a^{p-1} \equiv 1 \ (mod\ p)$$

III. KRIPTOGRAFI

A. Definisi

Kriptografi adalah sebuah metode untuk melindungi informasi dengan menggunakan algoritma tertentu. Kriptografi bertujuan agar yang bisa membaca dan memahami sebuah pesan hanya orang tertentu yang memiliki akses terhadap pesan tersebut. Kriptografi merupakan gabungan dari ilmu matematika, informatika, fisika, ilmu komunikasi.

B. Kriptografi Sederhana

Agar sebuah pesan hanya bisa dilihat oleh orang yang bisa mengaksesnya saja, pesan tersebut harus diubah. Pesan yang sebelumnya bisa dibaca (plainteks) akan dibuah menjadi sebuah pesan yang tidak bisa dipahami sama sekali. Proses perubahan ini disebut enkripsi. Kemudian ketika pesan sudah mencapai tujuan, pesan bisa dikembalikan ke bentuk semulanya. Proses pengembalian pesan ke bentuk semula ini dinamakan dekripsi.

Terdapat beberapa jenis kriptografi sederhana, yaitu :

1. *Shift Cipher*
   Shift cipher didasarkan dengan aritmetika modulo, Secara sederhana, algoritma enkripsi *shift cipher* dilakukan dengan menggeser huruf dengan sebuah bilangan tertentu.

   Misal p adalah huruf yang akan digeser, dan K adalah konstanta penggeseran, maka berlaku enkripsi

   $$E(p) = (p + K)\ mod\ 26$$

   untuk dekripsi berlaku

   $$D(p) = (p - K)\ mod\ 26$$

   angka 26 menyatakan jumlah huruf alfabet.

2. *Substitution Cipher*
   Algoritma enkripsi menggunakan *substitution cipher* sudah digunakan sejak lama. Jika *shift cipher* menggunakan aritmetika modulo, *substitution cipher* menggunakan permutasi dari huruf alfabet. Menggunakan fungsi permutasi, semua huruf diacak dan disubstitusikan kedalam pesan plainteks yang kemudian menghasilkan pesan terenkripsi. Dekripsi menggunakan invers dari permutasi.

3. *Affine Cipher*
   Affine cipher adalah sebuah enkripsi cabang dari *substitution cipher*, sehingga affine cipher lebih sederhana.

   Misal p adalah huruf yang akan digeser, dan K adalah konstanta penggeseran, a adalah bilangan prima dengan PBB(a,26) = 1, maka berlaku enkripsi

   $$E(p) = (ap + K)\ mod\ 26$$

   untuk dekripsi berlaku

   $$D(p) = a^{-1}(p - K)\ mod\ 26$$

   angka 26 menyatakan jumlah huruf alfabet.

4. *The Vigenere Cipher*
   Vigenere cipher memiliki proses yang mirip dengan shift cipher. vigenere cipher menerima sebuah pesan dan sebuah key, key tersebut diubah menjadi sebuah urutan angka. Kemudian urutan angka tersebut ditambahkan ke pesan sampai pesan selesai.

5. *Permutation Cipher*
   Permutation cipher mengubah posisi dari pesan yang akan dienkripsi menggunakan sebuah fungsi permutasi. *Permutation cipher* hampir mirip dengan *substitution cipher*, hanya saja *substitution cipher* menggunakan modulo 26 sedangkan *permutation cipher* tidak.

C. Random Number Generator

Sebuah pembangkit bilangan acak (*random number generator)* adalah sebuah urutan bilangan yang tidak bisa diprediksi. Menurut Marsaglia, sebuah urutan bilangan acak adalah penyebaran secara uniform ke seluruh nilai dan setiap nilai adalah independen dari nilai yang dihasilkan sebelumnya. Terdapat beberapa jenis pembangkit bilangan acak. Bilangan acak yang dihasilkan dari sistem atau menggunakan algoritma tertentu adalah bilangan acak-semu (*pseudo-random generator).*

1. *True Random Number Generators*

True random number generator menggunakan sumber entropi yang sudah ada. Dalam termodinamika, entropi adalah ukuran derajat ketidakteraturan suatu sistem. Dalam dunia nyata seperti melempar koin, ketidak teraturannya sangat tinggi, karena tidak dipengaruhi algoritma apapun dan tidak bisa diprediksi hasilnya. Tahun 2021, website yang menggunakan *true random number generator* adalah random.org.

2. *Pseudo-random Number Generator*

Pseudo random number generator tidak bergantung pada kejadian di dunia nyata untuk menhasilkan bilangan acak. *Pseudo-random number generator* akan terlihat acak oleh siapa saja jika tidak mengetahui nilai awalnya (umpan). Ketidak teraturan dalam generator ini dimulai dari masukan nilai awal.

Jika nilai awal dari generator semu diketahui, maka setiap nilai pada urutan akan mudah diketahui juga. Karena generator semu menggunakan sebuah algoritma tertentu yang hanya menggunakan nilai awal sebagai acuannya. Hal ini menyebabkan generator bilangan acak semu harus diberi perhatian lebih terhadap sistem keamanannya. Apalagi jika pembangkit bilangan acak semu digunakan untuk menhasilkan kata sandi.

Pembangkit bilangan acak semu harus dicek terlebih dahulu oleh ahli. Agar algoritma yang digunakan benar aman untuk digunakan dalam skala yang lebih besar. Pembangkit bialngan acak biasanya digunakan untuk komunikasi internet yang aman.

i. *Linear Conruential Generator*

Linear congruential generator (LCG) merupakan sebuah contoh sederhana dari pembangkit bilangan acak-semu. Algoritma yang digunakan dalam LCG adalah

$$X = (a.X_0 + b)\ mod\ m$$

Algoritma ini memerlukan sebuah nilai X awal atau umpan, dan tiga konstanta lain (a,b, dan m). Algoritma ini kurang aman karena ketika nilai X awal ditentukan, seluruh urutan bilangan dari algoritma sudah diketahui.

ii. *Lagged Fibonacci Generator*

Menggunakan *Lagged Fibonacci Generator (LFG)* sebuah siklus urutan bilangan dapat menjadi lebih besar dibandingkan menggunakan LCG. Bentuk dari LFG adalah

$$X_{i+1} = X_{i-p} \pm X_{i-q}\ mod\ m$$

Karena banyak nilai yang diguanakan untuk menentukan urutan bilangan acak selanjutnya, nilai dapat lebih bervariasi

IV. APLIKASI PEMBANGKIT BILANGAN ACAK PADA SHIFT CIPHER

Pembangkit bilangan acak digunakan untuk menghasilkan kunci shift pada pesan. Aplikasi pembangkait bilangan acak pada *shift cipher* akan dibuat sebuah program menggunakan bahasa python. Alasan dibuat program adalah mempermudah untuk mengecek hasil dari algortima yang telah dibuat.

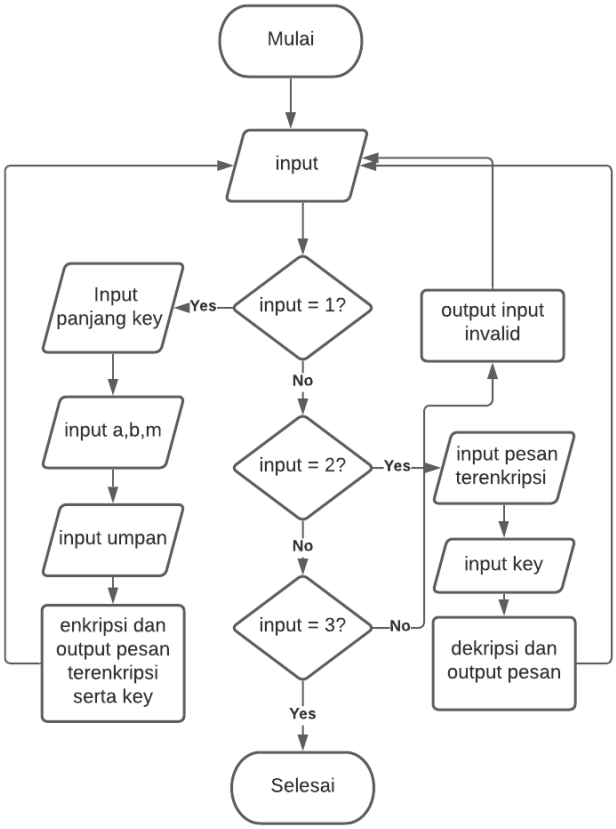
Program akan dibagi menjadi beberapa fungsi. Fungsi fungsi tersebut mengatasi berbagai masalah berbeda pada program, seperti enkripsi kata, dekripsi, dan mencari urutan bilangan acak yang akan dijadikan key.

Algortima pembangkit bilangan acak yang digunakan adalah pembangkit bilangan acak-semu LCG (*Linear Congruence Generator).* Nilai nilai konstanta a,b dan m akan diperoleh dari masukan pengguna. Kemudian masukan umpan akan diproses sehingga menghasilkan sebuah urutan angka yang akan dijadikan acuan dalam pergeseran pesan.

Algortima enkripsi yang digunakan adalah *shift cipher* dengan memanfaatkan aritmetika modulo. Tiap huruf digeser sesuai dengan key untuk menghasilkan pesan baru.

4.1 Flowchart

Implementasi program menggunakan bahasa python. Flowchart program dari awal sampai terminasi program adalah sebagai berikut.



Gambar 4.1.1 flowchart program

4.2 Implementasi Fungsi dan Prosedur

i. Fungsi Enkripsi

```
#fungsi enkripsi
def encrypt(text,s):
    result = ""
    for i in range(len(text)):
        char = text[i]
        if (char.isupper()):
            result += chr((ord(char) + s - 65) % 26 + 65)
        elif(31< ord(char) <65):
            result += chr((ord(char) + s - 32) % 32 + 32)
        else:
            result += chr((ord(char) + s - 97) % 26 + 97)
    return result
```

Gambar 4.2.1 Implementasi Fungsi Enkripsi

Implementasi fungsi enkripsi dibagi menjadi tiga kasus, yaitu ketika huruf kapital, huruf kecil, dan simbol. Angka 65 merupakan huruf kapital pertama pada kode ASCII. Sama halnya dengan angka 32 (simbol) dan 97 (huruf kecil).

Fungsi ini merupakan implementasi dari *cipher* yang berbasis aritmetika modulo. Tiap kasus dicari modulo, seperti huruf besar modulo 26 dan seterusnya.

ii. Fungsi Dekripsi

```
#fungsi dekripsi
def decrypt(text,s):
    result = ""
    for i in range(len(text)):
        char = text[i]
        if (char.isupper()):
            result += chr((ord(char) - s - 65) % 26 + 65)
        elif(31< ord(char) <65):
            result += chr((ord(char) - s - 32) % 32 + 32)
        else:
            result += chr((ord(char) - s - 97) % 26 + 97)
    return result
```

Gambar 4.2.2 Implementasi Fungsi Dekripsi

Implementasi fungsi dekripsi sama dengan fungsi dekripsi. Hanya saja jika fungsi enrkipsi tiap karakter digeser kekanan (ditambah), pada fungsi dekripsi tiap karakter digeser kekiri (dikurang).

iii. Fungsi Random Number

```
def get_random_num(str_num,a,b,m,umpan):
    result = []
    number = umpan
    for i in range(str_num):
        number = (a*umpan + b) % m
        result.append(number+32)
        umpan = number

    return result
```

Gambar 4.2.3 Implementasi Fungsi Random Number Generator

Fungsi memiliki lima parameter yaitu str_num (panjang key) a,b, m dan umpan untuk membentuk sebuah algoritma random number generator. Mengikuti persamaan LCG yaitu

$$X_n = (a.X_{n-1} + b)\ mod\ m$$

Program akan melakukan iterasi sebanyak jumlah key yang diinginkan. Dalam program Xn dihitung dengan variabel number, yang kemudian akan dimasukkan kedalam array result. setelah perhitungan dilakukan, umpan akan “maju” untuk menghitung variabel number selanjutnya.

iv. Fungsi Mencari Key

```
#mencari key
def get_key(str_num,sequence):
    final_key = ""
    for i in range(str_num):
        if (sequence[i] < 33):
            final_key += str(sequence[i])
        else :
            final_key += chr(sequence[i])
    return final_key
```

Gambar 4.2.4 Implementasi fungsi mencari key

Fungsi bertujuan untuk mengubah urutan bilangan acak yang sudah dihasilkan sebelumnya, kemudian dikonversikan menjadi karakter.

v. Fungsi Mencari Urutan Bilangan dari Key

```
#fungsi mencari sequence
def get_seq(key):
    seq = []
    for i in range(len(key)):
        seq.append(ord(key[i]))
    return seq
```

Gambar 4.2.5 Implementasi fungsi mencari urutan dari key

Fungsi ini merupakan invers dari fungsi sebelumnya. Fungsi akan mencari urutan bilangan dari *key*. Setiap karakter pada kunci akan dicari kode ASCII dan diubah menjadi sebuah array.

vi. Main Program

```
#main program
print("WELCOME TO MY PROGRAM")
end = False
while not (end) :
    print("Which you wanna choose? Pick 1,2 or 3")
    print("1. Encrypt")
    print("2. Decrypt")
    print("3. Exit")
    choice = int(input())
    if (choice == 1) :
        print("Please input your sentence")
        sentence = input()
        print("Please enter your key length, key must not exceeds " + str(len(sentence)))
        key_len = int(input())
        a = int(input("A = "))
        b = int(input("B = "))
        m = int(input("M = "))
        umpan = int(input("Umpan = "))
        seq = get_random_num(key_len,a,b,m,umpan)
        print(seq)
        key = get_key(key_len,seq)
        encrypted = get_encrypted(sentence,seq)
        print("Here is your encrypted text : ")
        print(encrypted)
        print("Here is your key : ")
        print(key)

    elif (choice == 2):
        print("Please input your encrypted sentence")
        sentence = input()
        print("Please enter your key")
        key = input()
        seq2 = get_seq(key)
        decrypted = get_decrypted(sentence,seq2)
        print("Here is your decrypted text : ")
        print(decrypted)
    elif (choice == 3):
        print("YOU QUIT THE PROGRAM :( ")
        end = True
    else :
        print("Input invalid!")
```

Gambar 4.2.6 Main Program

4.3 Hasil Uji Coba

4.3.1 Enkripsi

Akan dilakukan enkripsi terhadap pesan “This is the message, don’t share this to anyone else. It is highly confidential, the enemy is coming. I hope we can prepare ourselves. If you can read it after decryption, then the program is complete. “ dengan panjang key 10 karakter.

```
Please input your sentence
This is the message, don’t share this to anyone else. It is hig
ion, then the program is complete.
Please enter your key length, key must not exceeds 202
10
A = 7
B = 11
M = 17
Umpan = 0
[43, 35, 47, 46, 39, 41, 38, 34, 40, 48]
Here is your encrypted text :
Kqdm'xe"hdv#hyfhmos<+mjhvi&avwin/nuxe"hk+jisbcq"shjn=.Vi&qg0yrb
qcj7#obrc&bva+ymitgmu(ej#xizexmha9#
Here is your key :
+#/.')&"(0
```

Diperoleh pesan hasil enkripsi

Kqdm'xe"hdv#hyfhmos<+mjhvi&avwin/nuxe"hk+jisbcq"shjn=.Vi&qg0yrbbyn&kcjwryyaiuiz<+ccy'tzmau+rn.pdyqbc9#D.udbm(sv#xua)bzslraz.bjdashmnn<'Xr"mkl#xua)dmoz+ro.nufmf0unxllefqcj7#obrc&bva+ymitgmu(ej#xizexmha9#

dengan key : +#/.')&"(0

4.3.2 Dekripsi

Dekripsi pesan terenkripsi pada 4.3.1 diperoleh

```
Which you wanna choose? Pick 1,2 or 3
1. Encrypt
2. Decrypt
3. Exit
2
Please input your encrypted sentence
2
Please input your encrypted sentence
Kqdm'xe"hdv#hyfhmos<+mjhvi&avwin/nuxe"hk+jisbcq"shjn=.Vi&qg0yrb
byn&kcjwryyaiuiz<+ccy'tzmau+rn.pdyqbc9#D.udbm(sv#xua)bzslraz.bj
dashmnn<'Xr"mkl#xua)dmoz+ro.nufmf0unxllefqcj7#obrc&bva+ymitgmu(
ej#xizexmha9#
Please enter your key
+#/.')&"(0
Here is your decrypted text :
This is the message, donit share this to anyone else. It is hig
hly confidential, the enemy is coming. I hope we can prepare ou
rselves. If you can read it after decryption, then the program
is complete.
```

V. Conclusion

Terdapat berbagai macam algoritma untuk enkripsi pesan. Salah satunya dan yang paling sederhana adalah algoritma *shift cipher*. Algoritma ini menggeser karakter sesuai dengan konstanta pergeseran dan menghasilkan sebuah teks baru yang tidak bisa dibaca, kecuali dilakukan pergeseran lagi.

Pembangkit bilangan acak-semu merupakan salah satu pembangkit bilangan acak yang sederhana. Pembangkit bilangan acak-semu menggunakan aritmetika modulo.

Pembangkit bilangan acak-semu dengan modulo dapat diaplikasikan kedalam algortima enkripsi sederhana, untuk menhasilkan sebuah key agar pesan yang terenkripsi hanya bisa dibaca oleh orang yang memiliki key tersebut.

VII. Acknowledgment

Penulis mengucapkan terima kasih dan puji syukur terhadap Tuhan Yang Maha Esa atas rahmatnya, penulis bisa menyelesaikan makalah ini dengan tepat waktu. Penulis juga mengucapkan terima kasih kepada kedua orang tua yang sudah memberikan dukungan. Penulis juga mengucapkan terima kasih kepada Bapak Rinaldi Munir selaku dosen mata kuliah Matematika Diskrit semester 1 tahun ajaran 2021/2022.

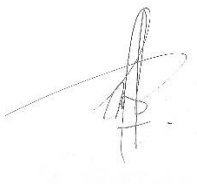
References

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Teori-Bilangan-2020-Bagian1.pdf. Diakses pada 13 Desember 2021.
[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Teori-Bilangan-2020-Bagian3.pdf Diakses pada 13 Desember 2021.
[3] https://core.ac.uk/download/pdf/58824567.pdf. Diakses pada 13 Desember 2021.
[4] https://wstein.org/ent/ent.pdf . Diakses pada 13 Desember 2021.
[5] https://simdos.unud.ac.id/uploads/file_penunjang_dir/118da0edfa7521509c0f38f25dcdacca.pdf. Diakses pada 13 Desember 2021.
[6] https://media.neliti.com/media/publications/176777-ID-peranan-sistem-modulo-dalam-penentuan-ha.pdf. Diakses pada 13 Desember 2021.
[7] Rivest, Ronald L. (1990). "Cryptography". In J. Van Leeuwen (ed.). Handbook of Theoretical Computer Science. 1. Elsevier.
[8] Elliptic Semiconductor Inc. and Author of the LibTom Project. “Cryptography for Developers” https://www.sciencedirect.com/book/9781597491044/cryptography-for-developers . Diakses pada 14 Desember 2021.
[9] Stinson, Douglas R. “Cryptography Theory and Practice”. University of Waterloo. Ontario, Canada.
[10]https://www.researchgate.net/publication/228034103_Random_Number_Generation/link/5a0491d1aca2726b4c704918/download . Diakses pada 14 Desember 2021.

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 13 Desember 2021.



Ilham Prasetyo Wibowo / 13520013