Laporan Proyek 1: Keamanan Komputer

Oleh:

Charis Hulu - 191900129
Moody Asyer - 191900154
Yosia Farianto - 191900478

Program Studi IT and Big Data Analytics
Calvin Institute of Technology
Jakarta
2022

1. Buatlah sebuah program yang dapat mengenkripsi dan dekripsi menggunakan affine cipher seperti yang dijelaskan di Problem 3.1 di buku Cryptography and Network Security seventh edition (William Stalling).

AffineCipher.py > ...

```python
import sys


def egcd(a, b):
  x,y, u,v = 0,1, 1,0
  while a != 0:
    q, r = b//a, b%a
    m, n = x-u*q, y-v*q
    b,a, x,y, u,v = a,r, u,v, m,n
  gcd = b
  return gcd, x, y

def modinv(a, m):
  gcd, x, y = egcd(a, m)
  if gcd != 1:
    return None
  else:
    return x % m

def encrypt(text, key):
  return ''.join([ chr((( key[0]*(ord(t) - ord('A')) + key[1] ) % 26) + ord('A')) for t in text.upper().replace(' ', '') ])


def decrypt(cipher, key):
  return ''.join([ chr((( modinv(key[0], 26)*(ord(c) - ord('A') - key[1])) % 26) + ord('A')) for c in cipher ])


if len(sys.argv) <= 8:
    key = [17,20]
    if sys.argv[1] == '-a' and sys.argv[3] == '-b' and sys.argv[5] == '-e' :
        key[0] = int(sys.argv[2])
        key[1] = int(sys.argv[4])
        text = str(sys.argv[6])
        enc_text = encrypt(text, key)
        print('Encrypted sentence: {}'.format(enc_text))

    if  sys.argv[1] == '-a' and sys.argv[3] == '-b'and sys.argv[5] == '-d' :
        text = str(sys.argv[6])
        key[0] = int(sys.argv[2])
        key[1] = int(sys.argv[4])
        dec_text = decrypt(text, key)
        print('Decrypted sentence: {}'.format(dec_text))
else:
    print('Format Error')
```

```
C:\Users\yosia\keamanan komputer>python AffineCipher.py -a 17 -b 20 -e HelloWorld
Encrypted sentence: JKZZYEYXZT

C:\Users\yosia\keamanan_komputer>python AffineCipher.py -a 17 -b 20 -d JKZZYEYXZT
Decrypted sentence: HELLOWORLD
```

2. Implementasikan program enkripsi dan dekripsi untuk mengenkripsi input plaintext menggunakan AES-128 dengan operasi mode CBC.

   Kode yang digunakan :

AESCipher.py > ...

```python
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad,unpad
import binascii
import sys

key = pad(b"mykey", AES.block_size)
iv = pad(b"myiv", AES.block_size)


def encrypt(plaintext):
    data_bytes = bytes(plaintext, 'utf-8')
    padded_bytes=pad(data_bytes, AES.block_size)
    AES_obj = AES.new(key, AES.MODE_CBC,iv)
    ciphertext = AES_obj.encrypt(padded_bytes)
    return ciphertext

def decrypt(ciphertext):
    AES_obj = AES.new(key, AES.MODE_CBC, iv)
    raw_bytes=AES_obj.decrypt(ciphertext)
    extracted_bytes = unpad(raw_bytes, AES.block_size)
    return extracted_bytes

def read_txt(file):
    file = open(file, 'r')
    return file.read()

def save_to_txt(file, result):
    f = open(file, 'w+')
    f.write(result.decode('utf-8'))
    f.close()
    return f

```

```python
if __name__ == '__main__':

    plaintext = "foeiafjeifajoeiafjo"
    f = open('test.txt', 'w+')
    f.write(plaintext)
    f.close()



    if len(sys.argv) == 3:
        if sys.argv[1] == '-e':
            file = str(sys.argv[2])
            plaintext = read_txt(file)
            ciphertext = encrypt(plaintext)
            binascii_ = binascii.hexlify(ciphertext)
            plaintext_enc = save_to_txt('encrypted.txt', binascii_)
            print("encrypted.txt added")

        if sys.argv[1] == '-d':
            file = sys.argv[2]
            binascii_data  = read_txt(bytes(file, 'utf-8'))
            ciphertext = binascii.unhexlify(binascii_data)
            plaintext_dec = decrypt(ciphertext)
            plaintext = save_to_txt('plaintext.txt', plaintext_dec)
            print("plaintext.txt added")
    else:
        print('Format Error')
```

Ketika Program dijalankan:

```
C:\Users\yosia\keamanan_komputer>python AESCipher.py -e test.txt
encrypted.txt added
```

test.txt

```
1  foeiafjeifajoeiafjo
```

encrypted.txt

```
cf61000d8d048e655b0f42e5a9bd019d18b74c71cea8437008e4eb9bfa95cf6e
```

```
C:\Users\yosia\keamanan komputer>python AESCipher.py -d encrypted.txt
plaintext.txt added

C:\Users\yosia\keamanan_komputer>
```

plaintext.txt

```
foeiafjeifajoeiafjo
```

3. Tulislah sebuah program fungsi perkalian (perkalian) dan pembagian (pembagian) untuk menghitung operasi aritmatika di Bidang Galois GF(2^N) - (AES GF(2^8) diharapkan).

GaloisField.py > ...

```python
def degree(digit):
    for i in range(len(digit)):
        if digit[i] == 1:
            return len(digit) - i - 1
    return -1


def index_of_power(power, len_list):
    return len_list - power - 1

def multiply(a, b):
    res = [0] * (degree(a) + degree(b) + 1)
    len_res = len(res)
    len_a = len(a)
    len_b = len(b)
    for power_a in range(degree(a), -1, -1):
        for power_b in range(degree(b), -1, -1):
            res_index = index_of_power(power_a+ power_b, len_res)
            a_index = index_of_power(power_a, len_a)
            b_index = index_of_power(power_b, len_b)
            res[res_index] += a[a_index] * b[b_index]
    for i in range(len(res)):
        res[i] = res[i]%2
    return res


```

```python
def xor(a, b):

    a_degree = degree(a)
    b_degree = degree(b)
    a_size = len(a)
    b_size = len(b)
    if(a_size > b_size):
        b = resize(b, a_size)
    elif(a_size < b_size):
        a = resize(a, b_size)
    res = []
    for i in range(a_size):
        res.append(a[i]^b[i])
    return res

def modulo(a, b):

    a_degree = degree(a)
    b_degree = degree(b)
    div_res_degree = a_degree
    div_res = [0] * div_res_degree
    remainder = a
    remainder_degree = degree(a)

    multi_res = []

    while remainder_degree >= b_degree:
        div_res_power = remainder_degree - b_degree
        div_res_power_index = index_of_power(div_res_power, div_res_degree)
        div_res[div_res_power_index] = 1
        multi_res = multiply(div_res[div_res_power_index:], b)
        remainder = xor(remainder, multi_res)
        remainder_degree = degree(remainder)
    return remainder

def resize(binary, n):
    binary_size = len(binary)
    if(binary_size < n):
        binary = [0]*(n - binary_size) + binary
    elif(binary_size > n):
        binary = binary[binary_size-n:]
    return binary

def gf_multiply_modular(a,b,mod,m):
    multi_res = multiply(a,b)
    res = modulo(multi_res, mod)

    return resize(res, m)

a_string = '00000111'
b_string = '00000011'
mod_string ='100011011'
m = 8
a = list(map(lambda i:int(i), a_string))
b = list(map(lambda i:int(i), b_string))
mod = list(map(lambda i:int(i), mod_string))

print(f"Multiplication of '{a_string}' and '{b_string}' :",gf_multiply_modular(a,b, mod, m))

```

```
87   from BitVector import BitVector
88
89   mod = BitVector(bitstring=mod_string)
90   a = BitVector(bitstring= a_string)
91   b = BitVector(bitstring= b_string)
92   quotient1, remainder1 = a.gf_divide_by_modulus(mod, m)
93   quotient2, remainder2 = b.gf_divide_by_modulus(mod, m)
94
95   print("Division by Modulos")
96   print('Quotient a:', quotient1)
97   print('Remainder a:',remainder1)
98   print('Quotient b:', quotient2)
99   print('Remainder b:',remainder2)
100
101
```

```
C:\Users\yosia\keamanan_komputer>python GaloisField.py
Multiplication of '00000111' and '00000011' : [0, 0, 0, 0, 1, 0, 0, 1]


Division by Modulos
Quotient a: 00000000
Remainder a: 00000111
Quotient b: 00000000
Remainder b: 00000011
```

4. Lakukan perhitungan enkripsi dan dekripsi menggunakan algoritma RSA, seperti pada Gambar 9.5 berikut ini:
   a. p = 3; q = 7, e = 5; M = 10
   b. p = 5; q = 13, e = 5; M = 8
   c. p = 7; q = 17, e = 11; M = 11

   a. Diketahui :
      p = 3
      q = 7
      e = 5
      M = 10
      - Menghitung nilai n
        - n = p x q
          n = 3 x 7
          n = 21
      - Menghitung nilai phi(n)
        - phi(n) = (p-1)(q-1)
          phi(n) = (3-1)(7-1)
          phi(n) = 12

- Menghitung nilai d
    - d = e^-1 (mod phi(n))
      d = 5^-1 (mod 12)
      d = 5

- Dari hasil di atas, didapatkan:
    - Public Key : 5,21
    - Private key : 5,21
- Nilai enkripsi menjadi Chipertext :
    - C = M^e mod n
      = 10^5 mod 21
      = 19
- Nilai dekripsi menjadi Plaintext
    - M = C^d mod n
      = 19^5 mod 21
      = 10

b. Diketahui :
   p = 5
   q = 13
   e = 5
   M = 8
- Menghitung nilai n
    - n = p x q
      n = 5 x 13
      n = 65
- Menghitung nilai phi(n)
    - phi(n) = (p-1)(q-1)
      phi(n) = (5-1)(13-1)
      phi(n) = 48
- Menghitung nilai d
    - d = e^-1 (mod phi(n))
      d = 5^-1 (mod 48)
      d = 29

- Dari hasil di atas, didapatkan:
    - Public key : 5,65
    - Private key : 29,65
- Nilai enkripsi menjadi Chipertext :
    - C = M^e mod n

= 8^5 mod 65
= 8

- Nilai dekripsi menjadi Plaintext
  - M = C^d mod n
    = 8^29 mod 65
    = 8

c. Diketahui :

p = 7
q = 17
e = 11
M = 11

- Menghitung nilai n
  - n = p x q
    n = 7 x 17
    n = 119
- Menghitung nilai phi(n)
  - phi(n) = (p-1)(q-1)
    phi(n) = (7-1)(17-1)
    phi(n) = 96
- Menghitung nilai d
  - d = e^-1 (mod phi(n))
    d = 11^-1 (mod 96)
    d = 35

- Dari hasil di atas, didapatkan:
  - Public key : 11,119
  - Private key : 35,119
- Nilai enkripsi menjadi Chipertext :
  - C = M^e mod n
    = 11^11 mod 119
    = 114
- Nilai dekripsi menjadi Plaintext
  - M = C^d mod n
    = 114^35 mod 119
    = 11