Group 5:

1. Deny Wahyudi Asaloei
2. A. Alfian Tenggara Putra
3. Juan Daniel Titarsole

Sistem Enkripsi XOR Berbasis MD5

Sistem kriptografi ini menggunakan metode enkripsi berbasis operasi XOR berlapis-lapis dengan empat kunci biner (B1, B2, B3, dan B4) yang dihasilkan dari satu *secret key* melalui fungsi hash MD5. Setiap kunci digunakan secara berurutan untuk melakukan operasi XOR terhadap data plaintext, menciptakan ciphertext.

Proses enkripsi dimulai dengan plaintext yang dikonversi menjadi bentuk integer. Selanjutnya, integer ini dioperasikan dengan XOR bersama empat kunci biner secara berurutan. Dekripsi membalikkan proses tersebut dengan urutan yang sama, sehingga berubah kembali menjadi plaintext asli dari ciphertext.

Sistem ini sederhana namun cukup aman karena menggunakan operasi XOR dengan beberapa lapisan kunci yang diperoleh dari satu secret key.

Cara Kerja Code:

1. Fungsi generate_keys
   Fungsi ini bertugas menghasilkan empat kunci dari kunci rahasia yang di-hash menggunakan MD5. Code:

```
def generate_keys(secret_key):
    hashed_key = hashlib.md5(secret_key.encode()).digest()

    B1 = hashed_key[0]
    B2 = hashed_key[1]
    B3 = hashed_key[2]
    B4 = hashed_key[3]

    return B1, B2, B3, B4
```

   Pada code ini secret_key adalah kunci rahasia yang diberikan sebagai input. Fungsi hashlib.md5(secret_key.encode()).digest() akan menghasilkan hash MD5 dalam bentuk byte array. Byte pertama hingga keempat dari hash ini digunakan sebagai empat kunci: B1, B2, B3, dan B4.

   Contoh:
   secret_key = “kunci” , maka hash MD5-nya adalah:
   b'\xfek_\x11\xf0i\xa5a\xc5\x11\xbb\x17\x14q\xc9\xae'

Byte pertama (B1) = ‘\xfe’ → 0xfe (254 dalam desimal)
Byte kedua (B2) = ‘k’ → ‘\x6b’ → 0x6b (107 dalam desimal)
Byte ketiga (B3) = ‘_’ → ‘\x5f’ → 0x5f (95 dalam desimal)
Byte keempat (B4) = ‘\x11’ → 0x11 (17 dalam desimal)

2. Fungsi encrypt
   Fungsi ini melakukan proses enkripsi dengan menggunakan operasi XOR antara teks biasa dan keempat kunci yang dihasilkan. Code:

```
def encrypt(plaintext, secret_key):
    B1, B2, B3, B4 = generate_keys(secret_key)

    A = int.from_bytes(plaintext.encode(), 'big')
    C = (((A ^ B1) ^ B2) ^ B3) ^ B4

    return C
```

   Pada code ini, plaintext adalah teks yang ingin dienkripsi (misalnya, "Hello"). Teks tersebut diubah menjadi bilangan bulat menggunakan int.from_bytes(plaintext.encode(), 'big'). Sebagai contoh, "Hello" dalam bentuk byte adalah b'Hello', yang dikonversi menjadi bilangan bulat 310939249775.

- Byte 1 (H) = 0x48 → Desimal 72 → 72×256^4
  $72 \times 4294967296 = 3087002425$
- Byte 2 (e) = 0x65 → Desimal 101 → 101×256^3
  $101 \times 16777216 = 1694498816$
- Byte 3 (l) = 0x6c → Desimal 108 → 108×256^2
  $108 \times 65536 = 7077888$
- Byte 4 (l) = 0x6c → Desimal 108 → 108×256^1
  $108 \times 256 = 27648$
- Byte 5 (o) = 0x6f → Desimal 111 → 111×256^0
  $111 \times 1 = 111$

3087002425 + 1694498816 + 7077888 + 27648 + 111 = 310939249775 → A

Selanjutnya operasi XOR dilakukan secara berantai antara nilai bilangan bulat tersebut (A) dan empat kunci biner (B1, B2, B3, B4):

C = (((A ^ B1) ^ B2) ^ B3) ^ B4

C = ((((310939249775 ^ 87) ^ 135) ^ 223) ^ 208) = 310939249464
Hasil ciphertext adalah 310939249464.

3. Fungsi decrypt
   Fungsi ini melakukan dekripsi dengan membalikkan proses XOR yang digunakan saat enkripsi. Code:

```
def decrypt(ciphertext, secret_key):
    B1, B2, B3, B4 = generate_keys(secret_key)

    A = (((ciphertext ^ B4) ^ B3) ^ B2) ^ B1
    plaintext = A.to_bytes((A.bit_length() + 7) // 8, 'big').decode()

    return plaintext
```

Pada code ini, ciphertext adalah hasil enkripsi yang ingin didekripsi. Ciphertext didekripsi dengan melakukan operasi XOR dalam urutan terbalik.

A = (((ciphertext ^ B4) ^ B3) ^ B2) ^ B1

A = ((((310939249464 ^ 208) ^ 223) ^ 135) ^ 87) = 310939249775

Hasilnya adalah bilangan bulat yang sama seperti sebelum enkripsi (310939249775). Bilangan bulat tersebut kemudian dikonversi kembali menjadi teks biasa menggunakan A.to_bytes(...).decode(), yang mengembalikan teks "Hello".

310939249775 → 'Hello'

4. Contoh Penggunaan

```
secret_key = "key"
plaintext = "Hello"

ciphertext = encrypt(plaintext, secret_key)
print(f"Ciphertext: {ciphertext}")

decrypted_text = decrypt(ciphertext, secret_key)
print(f"Decrypted Text: {decrypted_text}")
```

Pada contoh di atas, kita menggunakan secret_key "kunci" untuk mengenkripsi plaintext "Hello". Berikut hasil enkripsi dan dekripsi dari code tersebut:

Ciphertext: 310939249852
Decrypted Text: Hello