Enkripsi dan Dekripsi Citra Digital dengan Teknik Permutasi Piksel sebagai Upaya Pengamanan Citra

Valentino Daniel Kusumo - 13524104
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: danielkusumo550@gmail.com , 13524104@std.stei.itb.ac.id

***Abstrak*—Enkripsi citra adalah teknik yang bertujuan untuk mengamankan data gambar. Hal ini dilakukan dengan mengubah tampilan visual citra menggunakan kunci enkripsi tertentu, menjadikannya tidak dapat dikenali oleh pihak lain. Untuk memulihkan gambar ke bentuk aslinya, proses dekripsi memerlukan kunci dekripsi yang spesifik. Salah satu pendekatan yang digunakan dalam proses ini adalah teknik permutasi piksel. Proses ini dapat dibangun menggunakan konsep teori bilangan, khususnya aritmetika modulo dan invers modulo, untuk membentuk fungsi permutasi yang bijektif. Dengan demikian, transformasi posisi piksel dapat dilakukan secara aman dan dapat dikembalikkan pada saat dekripsi, sehingga citra asli dapat dipulihkan secara utuh.**

***Kata kunci*—Enkripsi Citra, Dekripsi, Kunci Enkripsi, Kunci Dekripsi, Permutasi Piksel, Aritmetika Modulo, Invers Modulo.**

I. PENDAHULUAN

Perkembangan teknologi informasi telah berkembang pesat, khususnya dalam penggunaan data digital yang meningkat signifikan, yaitu citra atau gambar. Citra digital kini tidak hanya digunakan untuk penyimpanan visual, tetapi juga untuk keperluan komunikasi, identifikasi, hingga pemrosesan kecerdasan buatan. Seiring dengan berkembangnya teknologi, kebutuhan akan keamanan data menjadi semakin penting, termasuk perlindungan terhadap citra digital dari pihak lain yang tidak berhak mengaksesnya.

Salah satu metode utama untuk menjaga kerahasiaan citra digital adalah melalui proses enkripsi, yaitu teknik mengubah posisi piksel sehingga gambar asli tidak dapat dikenali tanpa kunci khusus. Teknik permutasi piksel menjadi salah satu pendekatan yang efektif, dimana posisi piksel dalam citra diacak menurut sebuah kunci enkripsi sehingga citra menjadi tidak beraturan. Untuk memulihkan citra asli, diperlukan proses dekripsi yang membalikkan permutasi tersebut dengan kunci dekripsi

Pemanfaatan konsep teori bilangan seperti aritmetika modulo dan invers modulo memungkinkan penyusunan ulang posisi piksel dalam citra digital secara aman. Permutasi dilakukan melalui fungsi bijektif yang menjamin setiap piksel berpindah ke posisi unik tanpa duplikasi. Dalam pendekatan ini, kunci enkripsi dan kunci dekripsi yang digunakan berbeda, tetapi tetap terkait secara matematis. Kunci enkripsi berupa fungsi permutasi acak yang digunakan untuk mengacak posisi piksel saat proses enkripsi, dihasilkan menggunakan *generator* bilangan acak sehingga setiap proses enkripsi menghasilkan urutan acak yang berbeda – beda. Proses dekripsi dilakukan dengan menggunakan kunci invers yang mampu mengembalikan posisi piksel ke susunan semula secara tepat, sehingga citra asli dapat dipulihkan tanpa kehilangan informasi.

II. LANDASAN TEORI

A. *Teori Bilangan*

Teori bilangan adalah cabang matematika murni yang ditujukan untuk mempelajari bilangan bulat (*integer*) atau fungsi bernilai bilangan bulat. Bilangan bulat (*integer*) adalah bilangan yang tidak memiliki komponen pecahan atau desimal, misalnya 3, -7, dan 0. Berbeda dengan bilangan bulat, bilangan rill adalah bilangan yang memiliki desimal, seperti 3.3, 2.1, dan 4.5.

Gambar 1. Tabel Bilangan Bulat.
Sumber: https://www.gramedia.com/literasi/bilangan-bulat-operasi-hitung-dan-contoh-soalnya/

Salah satu konsep fundamental dalam teori bilangan adalah sifat pembagian pada bilangan bulat. Dalam hal ini, sebuah bilangan bulat a dikatakan habis membagi b jika terdapat bilangan bulat c sedemikian sehingga $b = ac$, dengan a tidak bernilai 0. Dalam penulisannya, hal ini dinotasikan dengan $a \mid b$, yang berarti b habis dibagi oleh a tanpa menghasilkan sisa. Sebagai contoh, $5 \mid 20$ karena terdapat bilangan bulat 4 yang memenuhi $20 = 5 \times 4$. Namun, $5 \nmid 22$ karena hasil pembagiannya menghasilkan pecahan, yaitu 4.4 yang bukan bilangan bulat.

Dalam ruang lingkup teori bilangan, terdapat berbagai teorema penting seperti teorema fermat kecil, teorema kongruensi linear, teorema *Chinese Remainder*, dan teorema-teorema lainnya. Namun, untuk keperluan enkripsi citra digital pada makalah ini, fokus akan ditujukan pada konsep ***teorema dan algoritma Euclidean***, ***pembagi bersama terbesar, kongruen, kekongruenan linier, relatif prima, aritmetika modulo*** dan ***invers modulo***.

Teorema Euclidean memisalkan nilai *m* dan *n* adalah bilangan bulat dengan $n > 0$. Jika *m* dibagi dengan *n*, maka hasil pembagiannya dapat dinyatakan dalam bentuk kuosien *q* dan sisa *r*, sedemikian sehingga

$$m = nq + r$$

dengan $0 \leq r < n$.

Sebagai contoh, misalkan $m = 25$ dan $n = 3$. Pembagian 25/3 menghasilkan kuosien $q = 8$ dan sisa $r = 1$ sehingga

$$25 = 3 \times 8 + 1$$

Ini memenuhi kondisi $0 \leq r < 3$.

Pembagi Bersama Terbesar (PBB) dari dua bilangan bulat *a* dan *b* (*a* dan *b* tidak bernilai 0) adalah bilangan bulat terbesar *d* sedemikian hingga $d \mid a$ dan $d \mid b$. Dalam hal ini, dinyatakan bahwa PBB(*a*, *b*) = *d*.

Salah satu metode paling efisien untuk menentukan PBB dari 2 bilangan bulat adalah **algoritma Euclidean**. Algoritma ini bekerja berdasarkan prinsip bahwa PBB dari dua bilangan *a* dan *b* dapat diperoleh secara rekursif melalui pembagian berulang, yaitu

$$\text{PBB}(a, b) = \text{PBB}(b, a \bmod b)$$

Proses ini dilanjutkan hingga sisa pembagian menjadi nol, dan bilangan terakhir yang bukan nol merupakan nilai PBB.

```
procedure Euclidean(input m, n : integer,
                    output PBB : integer)
{ Mencari PBB(m, n) dengan syarat m dan n bilangan tak-
  negatif dan m ≥ n
  Masukan: m dan n, m ≥ n dan m, n ≥ 0
  Keluaran: PBB(m, n)
}
Kamus
   r : integer

Algoritma:
   while n ≠ 0 do
      r ← m mod n
      m ← n
      n ← r
   endwhile
   { n = 0, maka PBB(m,n) = m }
   PBB ← m
```

Gambar 2. Algoritma Euclidean (*pseudocode*).
Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Teori-Bilangan-2020-Bagian1.pdf

Langkah – Langkah algoritma Euclidean :

1. Jika $n = 0$, maka *m* adalah PBB(*m*, *n*) dan berhenti, tetapi jika $n \neq 0$, lanjutkan ke langkah 2.
2. Bagilah *m* dengan *n* dan misalkan *r* adalah sisanya.
3. Ganti nilai *m* dengan nilai *n* dan nilai *n* dengan nilai *r*, lalu ulang kembali ke langkah 1.

Sebagai contoh, terdapat $m = 80$, $n = 12$ ($m \geq n$)

$$80 = 12 \times 6 + 8$$

$$12 = 8 \times 1 + 4$$

$$8 = 4 \times 2 + 0$$

PBB(80,12) = 4 karena sisa pembagian terakhir sebelum 0 adalah 4.

Dua bilangan bulat *a* dan *b* dikatakan **relatif prima** jika PBB (*a*, *b*) = 1. Sebagai contoh, terdapat $m = 35$, $n = 12$

$$35 = 12 \times 2 + 11$$

$$12 = 11 \times 1 + 1$$

$$11 = 1 \times 11 + 0$$

Karena PBB(35,12) = 1, 35 dan 12 adalah relatif prima.

Aritmetika modulo adalah operasi matematika yang menghitung sisa pembagian bulat dengan bilangan bulat lainnya. Misalkan, *a* dan *b* adalah bilangan bulat dengan $b > 0$. Operasi *a mod b* (dibaca "*a modulo b*") menghasilkan sisa dari pembagian *a* oleh *b*. Artinya, ada bilangan bulat *q* dan sisa *r* sehingga $a = bq + r$, dengan *r* memenuhi $0 \leq r < b$. Di sini, *b* disebut modulus, dan hasil dari operasi modulo selalu berada dalam rentang nilai $\{0,1,2, \ldots, b-1\}$.

Dalam konteks aritmetika modulo, 2 bilangan bulat *a* dan *b* dikatakan **kongruen** dalam modulus *m* jika selisih keduanya habis dibagi oleh *m*. Notasi formalnya ditulis seperti berikut

$$a \equiv b \bmod m$$

Artinya, $m \mid (a - b)$, yang dapat dituliskan sebagai $a - b = km$ untuk suatu bilangan bulat k. Dengan kata lain, *a* dan *b* memiliki sisa yang sama saat dibagi dengan *m*. Jika *a* tidak kongruen dengan *b* dalam modulus *m.* notasi ditulis $a \not\equiv b \pmod m$. Sebagai contoh, misalkan $a = 23$, $b = 8$, dan $m = 5$. Karena $5 \mid (23 - 8)$, 23 kongruen dengan 8 dalam modulus 5 atau dapat ditulis sebagai $23 \equiv 8 \bmod 5$

Dalam aritmetika bilangan rill, balikan suatu bilangan bukan nol adalah bentuk pecahan yang jika dikalikan dengan bilangan tersebut menghasilkan 1. Dalam aritmetika modulo, konsep balikan atau **invers modulo** juga ada, tetapi bentuknya berbeda karena tidak melibatkan pecahan. Invers dari suatu bilangan *a mod m* adalah bilangan bulat a^{-1} yang memenuhi:

$$a \cdot a^{-1} \equiv 1 \bmod m$$

Namun, tidak semua bilangan memiliki **invers modulo**. Syaratnya, *a* dan *m* harus relatif prima dan $m > 1$. Jika syarat ini terpenuhi, invers dari *a mod m* pasti ada.

Salah satu metode untuk menyelesaikan invers modulo adalah menggunakan **kekongruenan linier.** Persamaan ini berbentuk :

$$ax \equiv b \bmod m$$

dengan $m > 0$, *a* dan *b* sembarang bilangan bulat, dan *x* adalah peubah bilangan bulat. Persamaan tersebut ekuivalen dengan :

$$ax = b + km$$

$$x = (b + km) / a$$

Sebagai contoh, terdapat persamaan $3x \equiv 4 \mod 7$. Dari definisi kekongruenan, persamaan ini ekuivalen dengan :

$$3x \equiv 4 + 7k$$

$$x = (4 + 7k) / 3$$

Mencari nilai k sehingga pembilang $4 + 7k$ habis dibagi 3 :

$k = 0 \rightarrow x = 4/3$ – *bukan bilangan bulat*

$k = 1 \rightarrow x = 11/3$ – *bukan bilangan bulat*

$k = 2 \rightarrow x = 6$ – *salah satu solusi*

Semua solusi dari persamaan dapat ditulis sebagai :

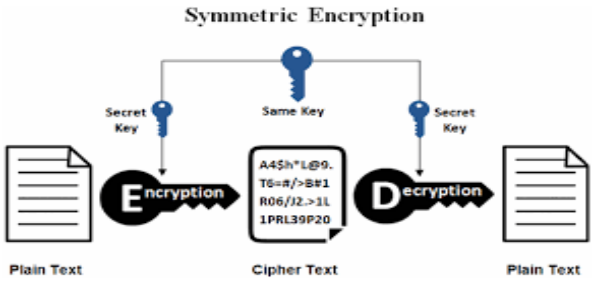
$x \equiv 6 \mod 7$ atau $x = 6 + 7k$

B. *Kriptografi*

Kriptografi berasal dari 2 kata dalam Bahasa Yunani, yaitu *kryptos* yang berarti tersembunyi, dan *graphia* yang berarti tulisan sehingga secara harafiah diartikan sebagai "tulisan tersembunyi" atau "seni menulis secara rahasia". Dalam konteks modern, kriptografi berkembang menjadi ilmu dan seni yang digunakan untuk menjaga kerahasiaan, keutuhan, dan keaslian informasi dengan memanfaatkan teknik-teknik pengkodean yang kompleks. Kriptografi memiliki peran yang penting dalam menjaga keamanan informasi di era digital.

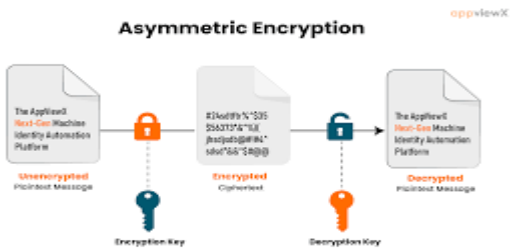
Kriptografi terbagi menjadi tiga jenis utama, yaitu:

1. **Kriptografi simetris (*symmetric cryptography*)**
   Kriptografi simetris menggunakan satu kunci yang sama untuk proses enkripsi dan dekripsi. Karena hanya ada satu kunci yang digunakan, kunci ini harus diajaga kerahasiaannya oleh kedua belah pihak.



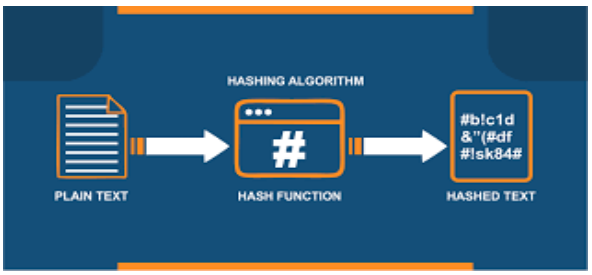
Gambar 3. Symmetric Encryption.
Sumber: https://l1nq.com/sFOne

2. **Kriptografi Public Key (*asymmetric cryptography*)**
   Kriptografi public key menggunakan dua kunci yang berbeda, tetapi saling berkaitan, yaitu kunci publik dan kunci pribadi. Kunci publik digunakan untuk enkripsi, sedangkan kunci pribadi digunakan untuk dekripsi. Kunci publik dapat dibagikan siapa pun, tetapi kunci pribadi harus dijaga kerahasiannya.



Gambar 4. Asymmetric Encryption.
Sumber: https://encr.pw/bGNLA

3. **Kriptografi Hash (*hash function*)**
   Kriptografi hash tidak memerlukan kunci untuk enkripsi dan dekripsi, tetapi menggunakan persamaan matematika untuk menghasilkan nilai tetap (hash) dari data yang diberikan.



Gambar 5. Hash Function.
Sumber: https://l1nq.com/8VDJh

Kriptografi mengandalkan konsep matematika dan algoritma untuk menjalankan fungsinya. Pada dasarnya, algoritma kriptografi dibagi menjadi tiga fungsi utama, yaitu:

1. **Enkripsi**
   Enkripsi adalah proses awal dalam kriptografi yang mengubah data dalam bentuk yang dapat dibaca (*plaintext*) menjadi bentuk yang tidak dapat dibaca atau dimengerti (*ciphertext*) menggunakan kunci enkripsi. Enkripsi bertujuan untuk menjaga kerahasiaan informasi dari pihak yang tidak berwenang.

2. **Dekripsi**
   Dekripsi adalah proses yang dilakukan untuk mengubah *ciphertext* kembali menjadi *plaintext*. Proses ini hanya dapat dilakukan oleh pihak yang memiliki kunci dekripsi yang sesuai.

3. **Key**
   Key dalam kriptografi merupakan parameter penting yang digunakan dalam proses enkripsi dan dekripsi. Terdapat dua jenis utama kunci yang daapat digunakan, yaitu **kunci simetris** yang sama untuk enkripsi dan dekripsi, atau **public key dan private key** dalam kriptograsi asimetris. Keamanan sistem kriptografi sangat bergantung pada kerahasiaan dan kekuatan kunci yang digunakan.

C. *Permutasi*

Permutasi adalah susunan atau pengurutan elemen – elemen yang urutan elemennya sangat diperhatikan. Dalam konteks matematika, permutasi dari *n* objek adalah jumlah seluruh kemungkinan penyusunan objek – objek tersebut dalam urutan yang berbeda.

Permutasi dapat dipahami sebagai aplikasi langsung dari kaidah perkalian. Jika terdapat *n* objek berbeda, maka :

- Posisi pertama dipilih dari *n* objek,
- Posisi kedua dipilih dari *n* – 1 objek yang tersisa,
- Posisi ketiga dipilih dari *n* – 2 objek yang tersisa,
- ….
- Posisi terakhir dipilih oleh 1 objek yang tersisa.

Dengan demikian, jumlah permutasi dari *n* objek adalah :

$$n \times (n - 1) \times (n - 2) \times \ldots \times 2 \times 1 = n!$$

III. IMPLEMENTASI

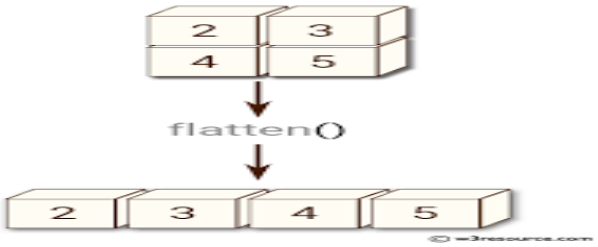
A. *Proses Enkripsi*

Proses enkripsi pada program ini bertujuan untuk menyamarkan informasi visual dalam sebuah citra digital berwarna (RGB), yaitu citra yang merepresentasikan setiap piksel sebagai kombinasi tiga komponen warna: merah (Red), hijau (Green), dan biru (Blue), masing-masing dengan nilai intensitas antara 0 hingga 255. Dalam implementasinya, gambar dibuka menggunakan pustaka PIL (Python Imaging Library, atau sekarang dikenal sebagai Pillow), dan bila mode warna gambar bukan RGB, maka secara eksplisit dikonversi terlebih dahulu menggunakan *convert("RGB")* untuk memastikan formatnya sesuai dengan tiga kanal warna yang dibutuhkan.

Kemudian, data citra diubah dalam bentuk ***Numpy array*** untuk memudahkan manipulasi matematis. Numpy array adalah struktur data berbentuk array multidimensi yang efisien dan dioptimalkan untuk komputasi numerik – numerik di Python. Dengan menggunakan Numpy, citra dalam format matriks piksel dapat diakses, dimodifikasi, dan diproses secara cepat menggunakan operasi vektorisasi. Dalam program ini, citra dibaca dan dikonversi menjadi array menggunakan *np.array()* dari library PIL (*Pyhton Imaging Library* atau sekarang lebih dikenal *pillow*). Setelah citra dikonversi ke dalam array (*arr = np.array()*), terdapat dua atribut penting yang digunakan :

- *arr.shape()* adalah atribut yang memberikan **dimensi array**, yaitu tinggi, lebar, jumlah kanal warna. Misalnya, jika citra berukuran 300 *x* 300 piksel, maka *arr.shape* akan bernilai (300, 300, 3).
- *arr.size()* adalah atribut yang memberikan **jumlah total elemen dalam array**, yaitu hasil perkalian seluruh dimensi array. Untuk array dengan shape = (300, 300, 3), maka *arr.size* = 270000, yang berarti ada 270.000 nilai intensitas warna.

Selanjutnya, setiap kanal warna dipisahkan dan diproses secara terpisah, dan array dua dimensi dari masing-masing kanal akan diratakan (*flattened*) menjadi satu dimensi untuk memudahkan proses permutasi acak piksel pada langkah enkripsi berikutnya.



Gambar 6. Ilustrasi Flattened
Sumber: https://encr.pw/jyOOn

```
filepath = askopenfilename(
    title="Pilih gambar",
    filetypes=[("Image files", "*.png *.jpg *.jpeg")])

img = Image.open(filepath)
if img.mode != 'RGB':
    img = img.convert('RGB')

arr = np.array(img)
shape = arr.shape
height, width, channels = shape
size = height * width
```

Gambar 7. Konversi Citra
Sumber: Dokumen Penulis

Proses enkripsi dilakukan dengan menggabungkan dua teknik utama, yaitu **perkalian modular** (menggunakan *aritmetika modulo*) dan **permutasi posisi piksel**.

1. **Perkalian Modular**

   Langkah awal dalam proses enkripsi adalah mengacak nilai intensitas (*brightness*) dengan melakukan **perkalian modular**. Setiap elemen array dari masing-masing kanal dikalikan dengan bilangan kunci yang dipilih secara acak. Hasil perkalian kemudian diambil dalam modulo 256, agar tetap berada dalam rentang valid 8-bit (0–255) yang digunakan dalam format citra digital.

   Bilangan kunci tersebut dipilih secara acak dari himpunan bilangan ganjil antara 1 – 255. Bilangan yang dipilih adalah ganjil karena semuanya relatif prima terhadap 256, sehingga memiliki *invers modulo*. Kunci yang memiliki invers ini penting untuk menjamin bahwa nilai piksel dapat dikembalikan ke semula saat proses dekripsi.

2. **Permutasi Posisi Piksel**

   Setelah nilai intensitas piksel dienkripsi, langkah berikutnya adalah mengacak posisi piksel. Proses ini dilakukan dengan menghasilkan indeks secara acak

menggunakan *np.random.permutation(size)*, di mana *size* adalah jumlah total piksel (height × width). Nilai-nilai hasil enkripsi pada setiap kanal warna disusun ulang berdasarkan indeks acak tersebut, kemudian dikembalikan ke bentuk dua dimensi menggunakan *reshape(shape)* agar struktur array sesuai dengan dimensi citra semula.

Dalam implementasinya, program menghasilkan **array kunci** untuk masing-masing kanal warna (R, G, dan B), yaitu *key_R*, *key_G*, dan *key_B*, yang masing-masing berisi bilangan ganjil acak antara 1 hingga 255. Bilangan-bilangan ganjil ini dipilih karena semuanya memiliki **invers modulo 256**, sebuah syarat penting untuk menjamin keberhasilan proses dekripsi nantinya. Pemilihan nilai kunci dilakukan dengan menggunakan fungsi *np.random.choice()*, dan hasilnya disesuaikan dengan dimensi gambar (tinggi × lebar) menggunakan *reshape((height, width))*, sehingga setiap piksel memiliki pasangan kunci tersendiri.

Setiap elemen dalam kanal warna kemudian dikalikan dengan elemen kunci yang bersesuaian menggunakan operasi aritmetika modular: *(channel * key) % 256*. Di sini, kanal warna dapat berupa matriks piksel merah, hijau, atau biru. Untuk memastikan tidak terjadi **overflow** saat melakukan perkalian dua nilai 8-bit, baik piksel maupun kunci dikonversi terlebih dahulu ke tipe *np.uint16* sebelum perkalian. Setelah operasi selesai, hasilnya dikembalikan ke tipe *np.uint8* agar sesuai dengan format standar penyimpanan gambar RGB.

Setelah nilai piksel terenkripsi, dilakukan permutasi posisi piksel secara acak dengan *np.random.permutation(size)*, di mana *size = height × width* yang menghasilkan indeks acak sebanyak jumlah piksel. Hasil permutasi kemudian dikembalikan ke bentuk dua dimensi *(reshape((height, width)))* agar dapat digabungkan kembali menjadi gambar RGB. Proses penggabungan ini dilakukan menggunakan fungsi *np.stack()*, yang berfungsi untuk menyusun tiga buah array dua dimensi (masing-masing channel R, G, dan B) ke dalam satu array berdimensi tiga, yaitu *(height, width, 3)*.

```
def generate_keys():
    return np.random.choice(
    np.arange(1, 256, 2), size=size)
    .astype(np.uint8).reshape((height, width))

def encrypt_channel(channel, key):
    encrypted = (channel.astype(np.uint16) * key.astype(np.uint16)) % 256
    return encrypted.astype(np.uint8)

key_R = generate_keys()
key_G = generate_keys()
key_B = generate_keys()
perm_indices = np.random.permutation(size)

R, G, B = arr[:,:,0], arr[:,:,1], arr[:,:,2]

R_enc = encrypt_channel(R, key_R).flatten()[perm_indices].reshape((height,
G_enc = encrypt_channel(G, key_G).flatten()[perm_indices].reshape((height,
B_enc = encrypt_channel(B, key_B).flatten()[perm_indices].reshape((height,
encrypted_image = np.stack((R_enc, G_enc, B_enc), axis=2)
```

Gambar 8. Enkripsi Citra
Sumber: Dokumen Penulis

B. *Proses Dekripsi*

Proses dekripsi bertujuan untuk memulihkan citra asli dari hasil enkripsi dengan membalik seluruh Langkah yang telah dilakukan sebelumnya. Tahap awal dekripsi dimulai dengan pembalikan urutan piksel yang telah dipermutasi secara acak dengan menggunakan fungsi *np.argsort(perm_indices)*, yang menghasilkan indeks posisi semula dari setiap piksel sebelum diacak. Dengan demikian, setiap channel citra terenkripsi dipulihkan ke urutan awalnya.

```
inverse_perm = np.argsort(perm_indices)
R_unshuffled = R_enc.flatten()[inverse_perm].reshape((height,
G_unshuffled = G_enc.flatten()[inverse_perm].reshape((height,
B_unshuffled = B_enc.flatten()[inverse_perm].reshape((height,
```

Gambar 9. Pembalikan Posisi Piksel
Sumber: Dokumen Penulis

Tahap selanjutnya adalah membalik operasi perkalian modular yang dilakukan sebelumnya pada proses enkripsi. Proses ini dilakukan dengan mengalikan kembali setiap elemen piksel dengan **invers modulo 256** dari kunci yang digunakan. Dengan kata lain, jika sebelumnya nilai piksel dikalikan dengan kunci *k*, maka untuk mendekripsinya, kita perlu mengalikannya dengan k^{-1} sehingga (piksel × k × k^{-1}) mod 256 = piksel.

Di awal program, telah disiapkan fungsi *inverse table* untuk mempercepat proses dekripsi. *Inverse table* adalah array yang menyimpan hasil perhitungan invers modulo 256 untuk setiap bilangan ganjil dari 1 hingga 255. *Inverse table* ini memungkinkan proses dekripsi dilakukan lebih cepat, karena nilai invers tidak perlu dihitung ulang satu per satu. Sebagai contoh, jika kunci *a* memiliki invers terhadap modulo 256, maka *inverse_table[a] = p*, dengan sifat $a \times p \equiv 1 \pmod{256}$.

```
def modinv(a, m):
    a = int(a)
    for x in range(1, m):
        if (a * x) % m == 1:
            return x
    raise ValueError(f"Tidak ada invers untuk {a} modulo {m}")

modulo = 256
inverse_table = np.zeros(modulo, dtype=np.uint8)
for a in range(1, modulo):
    try:
        inverse_table[a] = modinv(a, modulo)
    except ValueError:
        inverse_table[a] = 0
```

Gambar 10. Tabel Invers
Sumber: Dokumen Penulis

Sebelum dilakukan operasi perkalian, array dikonversi ke tipe *np.uint16* untuk mencegah overflow pada hasil perkalian dua nilai 8-bit. Setelah operasi selesai, hasil akhirnya dikembalikan dalam format 8-bit (*np.uint8*) agar sesuai dengan format standar citra RGB. Dengan menggabungkan ketiga channel hasil dekripsi (R, G, B) menggunakan *np.stack((R_dec, G_dec, B_dec), axis=2)*, maka citra RGB asli berhasil direkonstruksi secara utuh.

```
def decrypt_channel(channel, key):
    key_flat = key.flatten()
    key_inv = inverse_table[key_flat].reshape((height, width))
    decrypted = (channel.astype(np.uint16) * key_inv.astype(np.uint16)) % 256
    return decrypted.astype(np.uint8)

R_dec = decrypt_channel(R_unshuffled, key_R)
G_dec = decrypt_channel(G_unshuffled, key_G)
B_dec = decrypt_channel(B_unshuffled, key_B)
dec_image = np.stack((R_dec, G_dec, B_dec), axis=2)
```

Gambar 11. Dekripsi Citra
Sumber: Dokumen Penulis

IV. HASIL

Sebagai pengukur evaluasi keberhasilan proses enkripsi dan dekripsi yang telah dijalankan, citra asli, citra terenkripsi, dan citra terdekripsi divisualisasikan secara berdampingan menggunakan *library matplotlib.pyplot*. Dengan demikian, dapat diamati secara langsung apakah proses enkripsi-dekripsi telah berjalan dengan benar. Apabila seluruh langkah algoritma dijalankan dengan tepat, maka citra hasil dekripsi akan identik dengan citra aslinya, baik dari segi warna maupun struktur piksel.

```
plt.figure(figsize=(15, 5))
plt.suptitle("Proses Enkripsi dan Dekripsi Citra", fontsize=18,
fontweight='bold')

plt.subplot(1, 3, 1)
plt.imshow(arr)
plt.title("Gambar Asli", fontweight='bold')
plt.axis('off')

plt.subplot(1, 3, 2)
plt.imshow(enc_image)
plt.title("Terenkripsi", fontweight='bold')
plt.axis('off')

plt.subplot(1, 3, 3)
plt.imshow(dec_image)
plt.title("Terdekripsi", fontweight='bold')
plt.axis('off')

plt.tight_layout()
plt.show()
```

Gambar 12. Kode untuk Menampilkan Hasil
Sumber: Dokumen Penulis

Gambar 13. Hasil Enkripsi dan Dekirpsi
Sumber: Dokumen Penulis

V. KESIMPULAN DAN SARAN

Evaluasi terhadap proses enkripsi dan dekripsi citra digital RGB menggunakan kombinasi **perkalian modular dan permutasi piksel** menunjukkan bahwa metode ini mampu mengamankan informasi visual secara efektif, sekaligus tetap memungkinkan proses dekripsi yang akurat dan *lossless*. Proses enkripsi berhasil mengacak baik nilai intensitas piksel maupun posisinya, sehingga citra terenkripsi tidak lagi menyerupai gambar aslinya secara visual. Sementara itu, proses dekripsi mampu memulihkan citra asli secara utuh dengan mengandalkan invers modul0 dan pemulihan urutan piksel menggunakan indeks invers permutasi.

Namun, algoritma ini masih memiliki beberapa keterbatasan, terutama dari sisi keamanan. Salah satu isu utama adalah penggunaan kunci yang relatif sederhana dan seragam, yaitu berupa bilangan ganjil acak yang harus sesuai dengan dimensi citra. Ketergantungan terhadap kunci dengan pola yang dapat diprediksi ini membuka potensi risiko keamanan, terutama jika sebagian dari citra asli diketahui, karena dapat digunakan untuk menebak pola kunci. Selain itu, penyimpanan dan pengelolaan kunci per piksel memerlukan ruang memori yang besar, terutama pada citra beresolusi tinggi, sehingga kurang efisien untuk aplikasi berskala besar atau real-time. Di sisi lain, proses permutasi piksel dalam array berdimensi besar juga dapat memerlukan waktu dan memori yang signifikan, yang berdampak langsung pada performa sistem.

Demi meningkatkan keamanan, algoritma dapat dikembangkan menggunakan skema enkripsi asimetris. Dengan memisahkan kunci enkripsi dan dekripsi, distribusi kunci menjadi lebih aman dan pola piksel terenkripsi lebih sulit diprediksi.

VI. LAMPIRAN

Program Enkripsi dan Dekripsi Citra dapat diakses melalui link repository github berikut :
https://github.com/ValentinoDan/Enkripsi-Dekripsi-Citra.git
Video penjelasan makalah dapat diakses melalui link berikut :
https://youtu.be/8KxkdzOvOH0?si=ymWqDT99bUNlbPix

VII. UCAPAN TERIMA KASIH

Puji syukur kepada Tuhan Yang Maha Esa atas segala berkat dan rahmat-Nya sehingga makalah ini dapat diselesaikan dengan baik. Penulis juga menyampaikan terima kasih kepada dosen pengampu, teman-teman penulis yang telah memberikan dukungan, serta kedua orang tua penulis yang selalu memberikan semangat dan motivasi selama proses penyusunan makalah ini.

REFERENSI

[1] GeeksforGeeks. 2025. *NumPy Tutorial*. GeeksforGeeks.org. Diperoleh dari https://www.geeksforgeeks.org/numpy-tutorial/. Diakses pada 18 Juni 2025.

[2] Kariithi, Emmanuel, 2024. *Building a Simple Image Encryption Tool Using Python*. DEV Community. Diperoleh dari https://dev.to/immah/building-a-simple-image-encryption-tool-using-python-2439. Diakses pada 18 Juni 2025.

[3] Munir, Rinaldi. 2025. Teori Bilangan (Bagian 1) : "Bahan Kuliah IF2120 Matematika Diskrit." Bandung. Diakses pada 6 Juni 2025.

[4] Munir, Rinaldi. 2025. Kombinatorika (Bagian 1) : "Bahan Kuliah IF2120 Matematika Diskrit." Bandung. Diakses pada 14 Juni 2025.

[5] Sari, Rita Puspita. 2024. *Apa Itu Kriptografi?*. Cyberhub.id. Diperoleh dari https://cyberhub.id/pengetahuan-dasar/apa-itu-kriptografi. Diakses pada 14 Juni 2025.

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, 19 Juni 2025

Valentino Daniel Kusumo
13524104