BAB II

METODE PENELITIAN

2.1 Obyek penelitian

Obyek penelitian ini yang digunakan dalam penelitian ini adalah penerapan algoritma RSA pada citra digital. Penerapan ini mencangkup proses enkripsi dan dekripsi pada citra digital menggunakan algoritma RSA. Citra digital yang digunakan ini dapat berupa berbagai jenis gambar dalam format seperti JPEG, PNG, atau BMP. Karena penelitian ini bertujuan untuk menguji aktifitas dan keamanan penerapan algoritma RSA, pada citra digital serta untuk mengevaluasi kualitas dari citra yang dihasilkan setelah di enkripsi dan dekripsi.

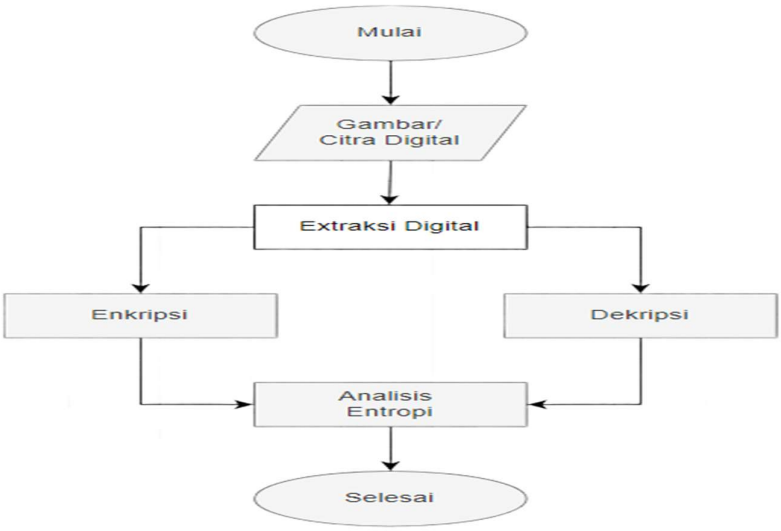
Penelitian ini dilakukan agar tau pentingnya keamanan gambar dalam konteks citra digital sering kali terdapat yang tidak baik. Dengan menerapkan algoritma RSA ini maka citra digital dapat diamankan tanpa memikir ada resiko yang berbahaya, sehingga dapat memberikan kenyamanan yang baik dalam menjaga kerahasiaan pada citra itu sendiri.

2.2 Alat dan Bahan

Untuk membantu proses penelitian dibutuhkan peralatan yang mendukung, Adapun peralatan yang digunakan pada penelitian ini yaitu perangkat keras (Hardware) yang di pakai adalah Laptop Lenovo Ideapad Slim 1, Processor AMD Ryzen 3 7000, dan RAM 8 GB. Perangkat lunak (Software) yang dipakai OS Windows 11, dan Python 3 atau google colab.

2.3 Prosedur penelitian

Agar penelitian dapat dilaksanakan secara terstruktur, diperlukan panduan atau acuan berupa prosedur penelitian. Diagram prosedur penelitian dapat dilihat dalam Gambar 2.1.



Gambar 2.1 Prosedur Penelitian.

Berikut adalah penjelasan langkah-langkah pada gambar diatas:

2.3.1 Gambar/Citra Digital

Tahap awal ini adalah pengambilan gambar yang akan di enkripsi dan didekripsi. Sebelum melakukan proses tahap selanjutnya, citra maka yang dilakukan adalah gambar dibagi menjadi tiga buah matriks, yaitu matriks warna merah (red), matriks warna hijau (green), dan matriks warna biru (blue). Gambar inj dapat berbentuk gambar digital dalam format seperti JPEG atau PNG. Pengambilan gambar ini dapat dilakukan dengan menggunakan kamera atau memanfaatkan data yang sudah tersedia di media internet. Selanjutnya, tahap dimana melakukan gambar yang diperoleh dengan melakukan proses enkripsi dan dekripsi ke dalam media ekstraksi gambar dan menerapkan kombinasi algoritma RSA. Dibawah ini kode uploud gambar yang akan ditunjukkan pada gambar 2.2.

Gambar 2. 2 Kode Uploud Gambar

```
# Fuction to upload image
def upload_and_process_image():
    upload = FileUpload(accept='image/*', multiple=False)
    container = VBox([upload])

    def on_upload_change(change):
        for filename, file_info in upload.value.items():
            img = Image.open(BytesIO(file_info['content']))
            display(img)
```

```
Upload an image to scramble and encrypt using RSA:
        Upload (0)
```

Berikut ini adalah gambar/citra yang akan digunakan dalam enkripsi ditunjukkan pada gambar 2.3.

Gambar 2. 3 Gambar/Citra Digital

2.3.2 Data extraction

Tahap ini melakukan pada gambar yang merujuk kepada proses pengambilan nilai-nilai numerik merepresensasikan warna atau intensitas dari setiap piksel dalm sebuah gambar. Pada gambar digital pixel adalah unit-unit terkecil dari informasi warna yang secara kolektif membentuk gambar tersebut. Pada proses ekstraksi data piksel ini melibatkan akases terhadap nilai-nilai, biasanya direpresentasikan dalam bentuk larik numerik, dan dapat mencangkup berbagai aspek seperti (i) Ukuran gambar pada data extraction melibatkan pengambilan informasi mengenai ukuran gambar, yang dinyatakan dalam jumlah piksel. Ukuran gambar mempengaruhi jumlah total data yang akan dienkripsi, yang penting untuk perencanaan penyimpanan dan pengiriman data terenkripsi. (ii) Ruang warna: Identifikasi ruang warna yang digunakan dalam gambar RGB. (iii) Mengubah data pixel ke format yang cocok untuk analisis lebih lanjut, seperti menggunakan larik numerik untuk algoritma pembelajaran mesin atau teknik pemrosesan gambar, Tahap ini dilakukan untuk memastikan bahwa gambar yang akan dienkripsi telah dipersiapkan dengan baik dan sesuai dengan kebutuhan sebelum dilakukan langkah-langkah keamanan lebih lanjut. Berikut ini adalah informasi coding data extraction yang diperoleh dapat dilihat pada tabel 2.4.

```
# Fungsi untuk ekstraksi data gambar
def extract_image_data(image):
    image_data = {
        "Ukuran": image.size,
        "Mode Warna": image.mode,
        "Format": image.format,
    }
    return image_data
```

Gambar 2. 4 *Coding* Extraction Data

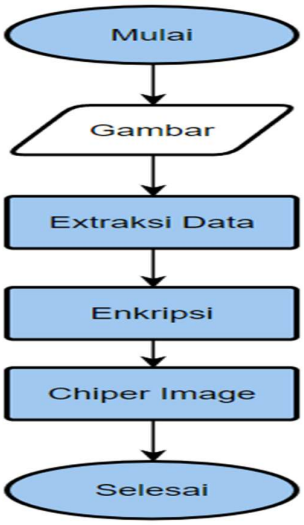
Berikut ini adalah informasi data extraction yang diperoleh dapat dilihat pada tabel 2.1.

Tabel 2.1 Informasi Data Extraction

Tahapan	Langkah	Keterangan
1	Definisikan fungsi	Fungsi *extract_image_data* didefinisikan dengan satu parameter yaitu image
2	Ambil ukuran gambar	*image.size* memberikan ukuran (width, height) dari gambar.
3	Ambil mode warna gambar	*image.mode* memberikan informasi tentang mode warna gambar (misalnya "RGB", "L", dll.).
4	Ambil format gambar	*image.format* memberikan informasi tentang format gambar (misalnya "JPEG", "PNG", dll.).
5	Simpan data dalam dictionary	Data yang diekstrak (ukuran, mode warna, dan format) disimpan dalam dictionary bernama *image_data.*
6	Kembalikan data gambar	Fungsi mengembalikan dictionary *image_data* yang berisi data gambar.

2.3.3 Enkripsi

Dari tahap ini yaitu alur dari proses enkripsi ditujukkan dimna gambar yang telah disiapkan akan dienkripsi menggunakan algoritma yang telah ditentukan yaitu algoritma RSA. Proses ini memerlukan kunci rahasia untuk gambar yang akan dinekripsi. Dibawah ini alur yang akan dienkripsi ditunjukkan pada gambar 2.4.



Gambar 2.5 Alur Enkripsi Algoritma RSA

```
# Fungsi untuk enkripsi pesan menggunakan RSA
def rsa_encrypt(message, public_key):
    e, n = public_key
    cipher = pow(message, e, n)
    return cipher
```

Gambar 2.6 Coding Enkripsi.

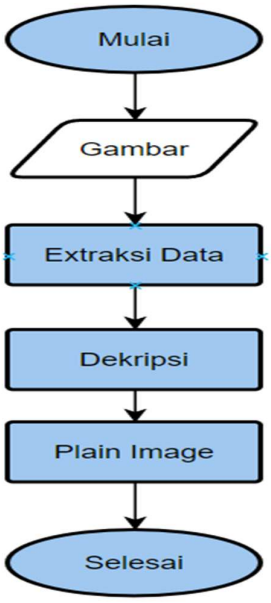
Berikut ini adalah table 2.2 yang berisi tentang input dan penjelasan yang digunakan dalam kode coding diatas tersebut:

Tabel 2. 2 Keterangan Coding Enkripsi

Fungsi	Langkah	Keterangan
Ekstraksi Kunci Publik	*e, n = public_key*	Mengambil komponen `e` (eksponen publik) dan `n` (modulus) dari pasangan kunci publik yang diberikan.
Enkripsi Pesan	*cipher = pow (message, e, n)*	Mengenkripsi pesan dengan menggunakan operasi modular eksponen. Fungsi `pow` menghitung *`message^e`*
Mengembalikan Cipher	*return cipher*	Mengembalikan nilai cipher yang merupakan hasil enkripsi pesan dengan kunci publik.

2.3.4 Dekripsi

Dari proses sebelumnya di enkripsi, langkah selanjutya adalah proses dekripsi. dalam proses ini, gambar yang telah terenkripsi akan didekripsi Kembali menjadi bentuk aslinya mengunakan kunci rahasia yang sama yang digunakan untuk enkripsi. Proses dekripsi ini hasilnya akan mendaptakan file citra yang semula sebelum dienkripsi.



Gambar 2. 7 Diagram alir dekripsi

Tahap pertama ini adalah proses memuat gambar yang sudah dienkripsi ke dalam folder di computer dan laptop. Setelah gambar di proses, langkah selanjutnya mengekstraksi gambar yang telah dienkripsi dari gambar tersebut. Gambar yang telah diekstraksi kemudian akan didekripsi menggunakan kunci privat. Proses ini akan mengembalikan gambar ke bentuk sebelum dienkripsi. Setelah gambar di

dekripsi, langkah terakhir adalah untuk mengembalikan gambar aslinya, harus melakukan proses menggabungkan kembali gambar yang telah didekripsi ke dalam format gambar yang sudah sesuai, seperti RGB, JPEG, PNG, dan gambar yang lainnya.

```
# Fungsi untuk dekripsi pesan menggunakan RSA
def rsa_decrypt(cipher, private_key):
    d, n = private_key
    message = pow(cipher, d, n)
    return message
```

Gambar 2. 8 Coding Dekripsi

Berikut ini adalah table 2.3 yang berisi tentang input dan penjelasan yang digunakan dalam kode coding diatas tersebut:

Tabel 2. 3 Keterangan coding dekripsi

Fungsi	Langkah	Keterangan
Ekstraksi Kunci Privat	*d, n = private_key*	Mengambil komponen `d` (eksponen privat) dan `n` (modulus) dari pasangan kunci privat yang diberikan.
Dekripsi Pesan	*message = pow (cipher, d, n)*	Mendekripsi pesan dengan menggunakan operasi modular eksponen. Fungsi `pow` menghitung `cipher^d % n`.
Mengembalikan Pesan	*return message*	Mengembalikan nilai pesan yang merupakan hasil dekripsi cipher dengan kunci privat.

2.3.5 Analisis entropi

Proses ini salah satu metode yang digunakan untuk memvalidasi tingkat pengecakan kriptografi. Entropi mengkur ketidak pastian atau keacakan informasi. Dalam Konteks kroptografi, entropi keluaran algoritma kriptografi yang menunjukkan seberapa sulitnya algoritma tersebut untuk diprediksi. (i) Entropi ini dapat diukur dalam bit per symbol. Semakin tinggi entropi kelaran algoritma, maka semakin banyak bit per symbol yang dibutuhkan untuk diprediksi dan karenya lebih diacak. (ii) Rumusan Entropi bisa di lihat bahwa entropi (H) dari sebuah variable acak X yang dapat mengambil n nilai yang berbeda dengan probabilitas masing-masing P(x_i) dapat dihitung menggunakan rumus berikut:

$$H(x) = -\sum_{i=1}^{n} \mathrm{P}(xi) log_2 \mathrm{P}(xi) \quad (1)$$

Keterangan:

H*(x)*	: Entropi dari variabel acak *x*
P*(xi)*	: Probabilitas dari kejadian *xi*
Log_2	: Logaritma basis 2
n	: Jumlah total kemungkinan kejadian
∑	:Simbol sigma (jumlah), yang menunjukkan bahwa kita menjumlahkan seluruh nilai yang dihitung di dalamnya.

(iii) Penerapan dalam validasi disebut juga dalam validasi tingkat pengacakan kriptografi, entropi keluaran algoritma dihitung dan dibandingkan dengan entropi maksimum yang mungkin. Entropi maksimum adalah $\log_2(n)$, di mana n adalah jumlah nilai yang mungkin diambil oleh keluaran. (iv) Interpretasi hasil ini di bagi menjadi 2 bagian yaitu Entropi tinggi (dekat dengan entropi maksimum): Menunjukkan bahwa algoritma cukup acak dan sulit untuk diprediksi. Dan yang kedua Entropi rendah menunjukkan bahwa keluaran algoritma tidak cukup acak dan mungkin dapat diprediksi oleh penyerangan.

Jadi tahap ini bahwa entropi adalah sebagai alat yang berharga untuk memvalidasi tingkat pengacakan kriptografi pada citra digital itu sendiri. Dengan memahami konsep entropi ini dan cara menghitungnya, maka dapat menilai keacakan keluaran algoritma kriptografi dan memastkan keamanan sistem kriptografi pada saat ini.

```
# Fungsi untuk menghitung entropi dari distribusi nilai piksel dalam gambar
def calculate_entropy(image):
    pixels = np.array(image)
    histogram, _ = np.histogram(pixels, bins=256, range=(0, 256))
    histogram = histogram.astype(float) / pixels.size
    histogram = histogram[histogram != 0]
    entropy = -np.sum(histogram * np.log2(histogram))
    return entropy
```

Gambar 2. 9 Coding Entropi

Tabel 2. 4 Penjelasan Coding Entropi

Tahapan	Langkah	Keterangan
Konversi Citra ke Array Numpy	*pixels = np.array(image)*	Mengonversi citra menjadi array numpy yang berisi nilai-nilai piksel citra tersebut.
Menghitung Histogram	*histogram, _ = np.histogram(pixels, bins=256, range=(0, 256))*	Menghitung histogram dari nilai-nilai piksel dengan 256 bins, rentang dari 0 sampai 256. Histogram ini menunjukkan frekuensi kemunculan setiap nilai piksel dalam citra.
Normalisasi Histogram	*histogram = histogram.astype(float) / pixels.size*	Mengubah histogram menjadi tipe data float dan menormalisasinya dengan membagi setiap nilai histogram dengan jumlah total piksel dalam citra. Ini mengubah histogram menjadi distribusi probabilitas.
Menghilangkan Nilai Nol dari Histogram	*histogram = histogram [histogram ! = 0]*	Menghapus nilai-nilai nol dari histogram untuk menghindari kesalahan dalam perhitungan logaritma, karena logaritma dari nol tidak terdefinisi.
Menghitung Entropi	*entropy = -np.sum(histogram * np.log2(histogram))*	Menghitung entropi menggunakan rumus entropi Shannon. Rumus ini mengalikan setiap nilai dalam histogram dengan logaritma basis 2 dari nilai tersebut, lalu menjumlahkan hasilnya dan mengubah tanda menjadi negatif.
Mengembalikan Nilai Entropi	*return entropy*	Mengembalikan nilai entropi yang telah dihitung, yang menggambarkan tingkat ketidakpastian atau kerandoman dalam distribusi nilai piksel citra.