Enkripsi Citra dengan Algoritma *One-Time Pad* Menggunakan *Pseudorandom Number Generator* (PRNG) sebagai Kunci pada *Embedded System*

SKRIPSI

KEMINATAN TEKNIK KOMPUTER

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Muhammad Dhiya' Ainul Labib
NIM: 115060900111023

TENIK INFORMATIKA
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

PENGESAHAN

Enkripsi Citra dengan Algoritma *One-Time Pad* Menggunakan *Pseudorandom Number Generator* (PRNG) sebagai Kunci pada *Embedded System*

SKRIPSI

KEMINATAN TEKNIK KOMPUTER

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Muhammad Dhiya' Ainul Labib
NIM: 115060900111023

Skripsi ini telah diuji dan dinyatakan lulus pada
14 Januari 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Eko Setiawan, S.T., M.Eng	Barlian Henryranu P., S.T., M.T.
NIK: 201102 870610 1 001	NIK: 821024 06 1 1 0254

Mengetahui
Ketua Program Studi Teknik Informatika

Drs. Marji, M.T.
NIP: 19670801 199203 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 25 Januari 2016

Muhammad Dhiya' Ainul Labib

NIM: 115060900111023

KATA PENGANTAR

Puji syukur kehadirat Allah SWT yang telah melimpahkan rahmat, taufik dan hidayah-Nya sehingga laporan skripsi yang berjudul **"Sistem Kriptografi Citra dengan Algoritma *One-Time Pad* Menggunakan *Pseudorandom Number Generator* (PRNG) sebagai Kunci"** ini dapat terselesaikan. Skripsi ini untuk memenuhi salah satu syarat menyelesaikan studi serta dalam rangka memperoleh gelar Sarjana Pendidikan Strata Satu pada Program Studi Sistem Komputer Fakultas Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya. Penghargaan dan terima kasih yang setulus-tulusnya kepada Ayahanda tercinta Drs. Masrukin, M.A. dan Ibunda yang kusayangi Dra. Maftuchah, S.Pd. serta adik-adik yang kusayangi semoga Allah SWT melimpahkan Rahmat, Kesehatan, Karunia dan keberkahan di dunia dan di akhirat atas budi baik yang telah diberikan kepada penulis.

Penulis menyadari bahwa skripsi ini tidak akan berhasil tanpa bantuan dari beberapa pihak. Oleh karena itu, penulis ingin menyampaikan rasa hormat dan terima kasih kepada:

1. Bapak Prof. Dr. Ir. Mohammad Bisri, MS selaku Rektor Universitas Brawijaya
2. Bapak Ir Sutrisno, M.T. selaku Dekan Fakultas Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
3. Bapak Eko Setiawan, S.T., M.T. M.Eng selaku pembimbing I yang telah banyak membantu dalam pengerjaan skripsi ini.
4. Bapak Barlian Henryranu P., S.T., M.T. selaku pembimbing II yang telah banyak membantu dalam pengerjaan skripsi ini.
5. Terima kasih kepada teman-teman kontrakan, Ahonk, Bagus, Kholis, Kojek, yang membantu dan mengganggu dalam pengerjaan skripsi.
6. Seluruh teman-teman Teknik Komputer 2011 yang telah menemani selama perkuliahan dan seluruh pihak yang tidak dapat disebutkan namanya satu persatu.

Malang, Januari 2016

Penulis

Labib.amd@gmail.com

ABSTRAK

Perkembangan teknologi yang semakin cepat, membuat pertukaran data menjadi semakin cepat dan mudah. Salah satu tipe data yang sering digunakan adalah citra. User sering kali melupakan keamanan untuk melindungi data dari pihak ketiga. Pengamanan yang sering digunakan untuk melindungi data dari pihak ketiga adalah kriptografi. *One-Time Pad* (OTP) merupakan *stream cipher* yang melakukan enkripsi dengan perubahan kunci tiap satu kali proses, sehingga sesuai untuk enkripsi citra yang prosesnya memerlukan enkripsi untuk tiap piksel. Keamanan OTP terletak pada keacakan barisan kuncinya.

Pada penelitian ini, barisan kunci dibangkitkan menggnukanan *Pseudorandom Number Generator* (PRNG) Mersenne Twister yang memiliki karakteristik chaos, yang berarti perubahan kecil pada nilai awal (*seed*) menghasilkan barisan bilangan berbeda. Sistem kriptografi OTP menggunakan seed sebagai masukan untuk proses enkripsi, sehingga dekripsi menggunakan *seed* yang sama menghasilkan gambar tepat seperti semula. Dari hasil pengujian penulis, pengimplementasian enkripsi OTP dengan kunci PRNG pada Raspberry Pi membutuhkan waktu 6,4 detik untuk citra 300 x 300 piksel dan 8,6 detik untuk citra 400 x 300 piksel. PRNG Mersenne Twister menghasilkan *key* yang benar-benar acak dan telah diverifikasi dengan pengujian autokorelasi menggunakan metode Durbin Watson.

Kata kunci: Enkripsi, Citra, *One-Time Pad*, PRNG, Raspberry Pi

ABSTRACT

The increasingly rapid development of technology making data exchange becomes faster and easier. One type of data that is often used is image data. User is often forget about security to protect data from third parties. Most common practice to protect data from third parties is by using cryptography. One-Time Pad (OTP) is a stream cipher that encrypts by changing the key every one-time process, so it is suitable for image encryption wich is done by encrypting each pixel. OTP security lies in the randomness of the key row.

In this study, key rows generated by Pseudorandom Number Generator (PRNG) Mersenne Twister which has the characteristics of chaos, which means that a small change in the initial value (seed) produce a different sequence number. OTP Cryptographic system used seed as input for encryption, thus decryption using the same seed produces images as precise as before. Based on the testing by the writer, OTP encryption implementation with PRNG key on Raspberry Pi take 6.4 seconds for 300 x 300 pixels image and 8.6 seconds for 400 x 300 pixels image. Mersenne Twister's PRNG used in this study produced true random keys wich is verified with Durbin Watson's autocorrelation method.

Keyword: *Encryption, Image, One-Time Pad, PRNG, Raspberry Pi*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

Bab ini akan menjelaskan latar belakang masalah yang ada pada topik, permasalahan dari topik, batasan-batasan masalah yang akan dibahas dalam topik, tujuan, manfaat, dan disertai dengan sistematika penulisan.

1.1 Latar Belakang

Perkembangan teknologi yang semakin cepat, terjadi dalam beberapa tahun terakhir. Dengan menggunakan internet, berbagai pihak dapat bertukar data dengan cepat dan mudah. Kemudahan berbagi data, pihak yang bersangkutan sering kali melupakan keamanan untuk melindungi dari pihak ketiga.

Pengamanan yang sering digunakan untuk melindungi berkas dari pihak ketiga adalah dengan kriptografi. Kriptografi memiliki dua bagian, yaitu enkripsi dan dekripsi. Enkripsi adalah proses pengamanan berkas asli (plainteks), dengan mengubah menjadi berkas yang tidak dapat dibaca (cipherteks) tanpa bantuan kunci. Sedangkan dekripsi merupakan proses mengembalikan cipherteks menjadi plainteks.

Salah satu tipe data yang sering digunakan adalah citra. Penyimpanan dan pengiriman citra melalui saluran komunikasi publik rentan terhadap pengaksesan dan penyadapan oleh pihak-pihak yang tidak berhak (Munir, 2011). Sehingga enkripsi untuk citra perlu dilakukan.

Berbagai algoritma enkripsi yang umum digunakan adalah RSA, DES, AES, dan lain sebagainya. RSA adalah algoritma enkripsi yang memiliki 2 macam kunci, *private* untuk pemilik dan *public* untuk disebar kepada publik. Sedangkan algoritma DES (*Data Encryption Standard*) menggunakan sandi blok kunci simetrik, dengan ukuran blok 64-bit dan ukuran kunci 56-bit. AES memiliki algoritma perpindahan byte, pertukaran baris, pergesaran baris, dan penambahan subkunci. AES merupakan salah satu algoritma enkripsi yang menggunakan *block cipher*.

Citra pada umumnya memiliki ukuran data yang besar dibanding dengan data teks, sehingga kebanyakan algoritma tersebut kurang cocok digunakan untuk enkripsi citra (Munir, 2011). Citra memiliki karakter dua dimensi yang berisi koordinat letak piksel. Setiap piksel citra dengan format PNG memiliki kedalaman warna 3 bytes atau 24 bit. Sehingga ukuran asli gambar dapat diketahui dari lebar dan panjang gambar. Teknik kompresi *lossless* yang digunakan PNG menghasilkan ukuran file yang lebih kecil dengan tidak mengurangi penurunan kualitas gambar. Proses enrkipsi citra dengan algoritma umum memerlukan waktu yang lebih lama (Krikor, 2009).

Stream cipher relatif lebih efektif dalam mengenkripsi citra, jika dibanding dengan algoritma *block cipher*. Kelebihan yang dimiliki *stream cipher* adalah tidak melakukan pengulangan, sebagaimana yang dilakukan *block cipher*. Selain itu,

stream cipher menggunakan operasi sederhana seperti konjungsi eksklusif atau biasa disebut XOR.

One-Tim Pad (OTP) adalah salah satu *stream cipher* klasik yang secara matematis terbukti sempurna aman (Stalling, 2003). OTP melakukan kriptografi dengan perubahan kunci pada setiap satu kali proses enkripsi. Algoritma OTP sesuai untuk diterapkan pada kriptografi citra. Dengan melakukan perubahan kunci untuk enkripsi pada setiap piksel, sehingga cipherteksnya susah dipecahkan. Keamanan algoritma OTP terletak pada keacakan barisan kunci acak.

Pseudorandom number generator (PRNG) adalah algortima yang digunakan untuk membangkitkan berisan angka semi acak. PRNG tidak ada yang benar-benar menghasilkan bilangan acak, karena untuk membangkitkan bilangan acak ada algoritma yang digunakan. Algoritma PRNG dengan periode pengulangan tinggi adalah Mersenne Twister, yang memiliki 2^{19937}-1 kemungkinan nilai sebelum melakukan periode pengulangan. Mersenne Twister memiliki karakteristik *chaos*, yang berarti perubahan kecil pada nilai awal (*seed*) menghasilkan barisan angka yang berbeda. Mersenne Twister dengan bersifat deterministik, yang artinya nilai-nilai acak yang dihasilkan dapat dibangkitkan asalkan seed yang digunakan tetap sama.

Penelitian ini mendeskripsikan sebuah algoritma kriptografi sederhana untuk citra digital dengan algoritma OTP dengan kunci acak yang dibangkitkan dari PRNG Mersenne Twister. PRNG Mersenne Twister dipilih karena periode pengulangan yang tinggi, sehingga hampir tidak mungkin mengalami periode pengulangan. PRNG Mersenne Twister berperan untuk membangkitkan barisan bilangan acak sebagai kunci OTP. Barisan bilangan acak sebagai kunci OTP yang kemudian dienkripsikan dengan piksel-piksel citra.

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang, maka dapat ditarik perumusan masalah sebagai berikut:

1. Seberapa besar tingkat kecocokan antara gambar asli dengan hasil dekripsi?
2. Berapa waktu yang dibutuhkan melakukan enkripsi dan dekripsi citra dengan algoritma *One-Time Pad* dengan kunci dari PRNG Mersenne Twister?
3. Seberapa tinggi tingkat keacakan dari barisan bilangan yang dihasilkan PRNG Mersenne Twister?

1.3 Tujuan

Berdasarkan rumusan masalah, maka tujuan dari penelitian ini diuraikan pada poin-poin berikut:

1. Untuk mengetahui tingkat kecocokan antara gambar asli dengan hasil dekripsi.

2. Untuk mengetahui waktu yang dibutuhkan melakukan enkripsi dan dekripsi citra dengan algoritma *One-Time Pad* dengan kunci dari PRNG Mersenne Twister.
3. Untuk mengetahui tingkat keacakan dari barisan bilangan yang dihasilkan PRNG Mersenne Twister.

1.4 Manfaat

Penulisan penelitian ini diharapkan mempunyai manfaat yang baik dan berguna bagi penulis dan pembaca. Adapun manfaat yang diharapkan adalah sebagai berikut:

1. Bagi Penulis

   Sebagai sarana untuk menimplementasikan ilmu pengetahuan bidang teknologi, yang telah didapat selama mengikuti perkuliahan di Teknik Informatika Universitas Brawijaya. Juga memperdalam wawasan dalam bidang kriptografi.

2. Bagi Pembaca

   Sebagai salah satu sarana untuk menambah wawasan terhadap bidang kriptografi, dan menjadi bahan informasi ilmiah untuk diadakan penelitian lebih lanjut.

1.5 Batasan Masalah

Batasan permasalahan dirumuskan agar penelitian yang dilakukan dapat lebih terfokus, sehingga dibatasi beberapa hal berikut:

1. Kriptografi citra diimplementasikan pada Raspberry Pi.
2. Bahasa pemrograman yang digunakan adalah Python.
3. Algoritma kriptografi yang digunakan adalah *One-Time Pad*, dengan barisan bilangan dari PRNG Mersenne twister sebagai kuncinya.
4. Tipe file citra yang digunakan adalah PRNG.
5. Penelitian ini difokuskan pada proses enkripsi dan dekripsi dari citra.

1.6 Sistematika Pembahasan

Sistematika pembahasan dalam penelitian ini sebagai berikut:

BAB I Pendahuluan

Memuat latar belakang permasalahan, rumusan masalah, tujuan, manfaat, batasan masalah, dan sistematika pembahasan dari penelitian yang dilaksanakan.

BAB II Landasan Kepustakaan

Memuat landasan kepustakaan mengenai penelitian-penelitian sebelumnya dan menjelaskan teori dasar tentang kriptografi, PRNG, Raspberry Pi, dan Python.

BAB III Metodologi

Membahas metode yang digunakan dalam penelitian yang terdiri dari studi literatur, analisis kebutuhan sistem, perancangan, implementasi, pengujian dan analisis Sistem Kriptografi Citra dengan Algoritma *One-Time Pad* Menggunakan *Pseudorandom Number Generator* (PRNG) sebagai Kunci.

BAB IV Perancangan dan Implementasi

Membahas tentang perancangan dan implementasi kriptografi citra dengan algoritma *One-Time Pad* pada Raspberry Pi menggunakan bahasa pemrograman Python.

BAB V Pengujian dan Analisis

Menjelaskan proses pengujian beserta analisis hasil terhadap kecocokan antara gambar asli dengan hasil dekripsi, waktu yang dibutuhkan untuk melakukan dekripsi, dan tinggak keacakan barisan bilangan PRNG.

BAB VI Penutup

Memuat kesimpulan yang diperoleh dari pengujian dan hasil analisis, serta saran-saran untuk penngembangan lebih lanjut.

BAB 2 LANDASAN KEPUSTAKAAN

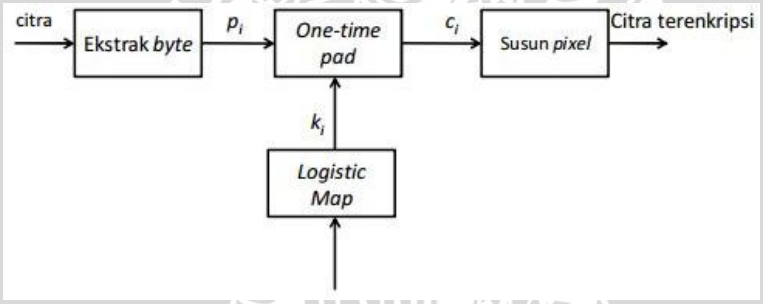
Bab ini berisi tentang kajian pustaka dan dasar teori yang diperlukan dalam penelitian. Tinjauan pustaka diambil dari beberapa penelitian relevan yang pernah dilakukan. Dasar teori membahas tentang pengetahuan dasar yang diperlukan untuk menyusun penelitian yang diusulkan.

2.1 Tinjauan Pustaka

Kajian pustaka membahas ringkasan dari beberapa penelitian yang telah dilakukan oleh peneliti sebelumnya.

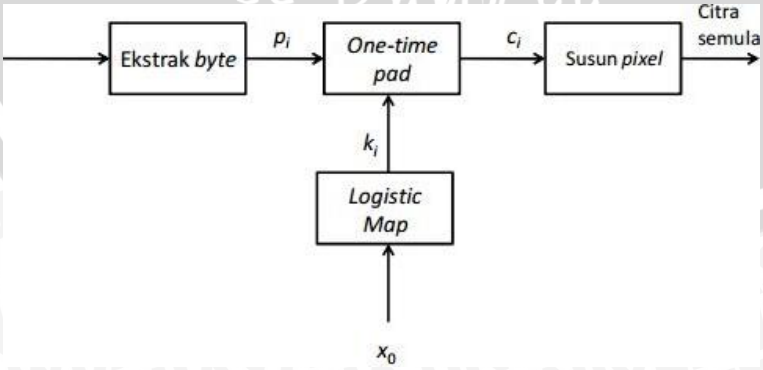
2.1.1 Penelitian Algoritma Enkripsi Citra dengan *Pseud One-Time Pad* yang Menggunakan Sistem *Chaos*. Oleh Rinaldi Munir.

Rinaldi Munir pernah melakukan riset mengenai enkripsi citra digital. Pada riset dengan judul Algoritma Enkripsi Citra Digital dengan Pseudo One-Time Pad yang Menggunakan Sistem Chaos. Penelitian ini membahas algoritma sederhana untuk mengenkripsi citra digital dengan mememanfaatkan sistem chaos. Enkripsi dilakukan dengan algortima One-Time Pad, yang dalam hal ini berisan kunci dibangkitkan dengan sistem Logistic Map. Proses kerja dari penelitian Rinaldi Munir di ilustrasikan pada Gambar 2.1 dan Gambar 2.2.



Gambar 2.1 Skema enkripsi dengan *Pseudo One-Time Pad*

Sumber : Munir, 2011



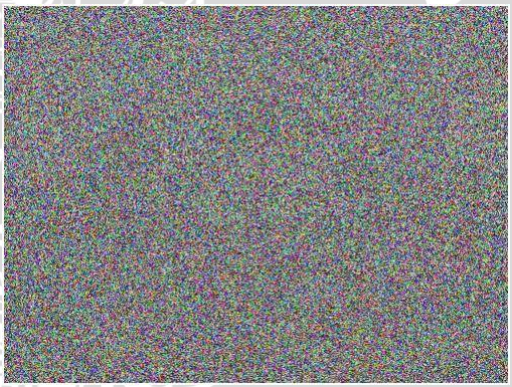
Gambar 2.2 Skema dekripsi dengan *Pseudo One-Time Pad*

Sumber : Munir, 2011

Algoritma yang diusulkan dalam penelitian ini disebut *pseudo One-Time Pad,* karena tidak menggunakan barisan yang benar-benar acak, melainkan bilangan semiacak yang dibangkitkan menggunakan *logistic map*. Algoritma enkipsi dapat digunakan untuk semua citra *bitmap* (*grayscale* dan berwarna) dan nilai awal *logistic map*. Citra *bitmap grayscale* memiliki nilai tiap piksel sebesar 8 bit, sedangkan yang berwarna memiliki 24 bit. Pada citra berwarna disusun oleh bagian warna *Red* (R) *, Green* (G)*, Blue* (B) yang masing-masing memiliki nilai 1 byte.

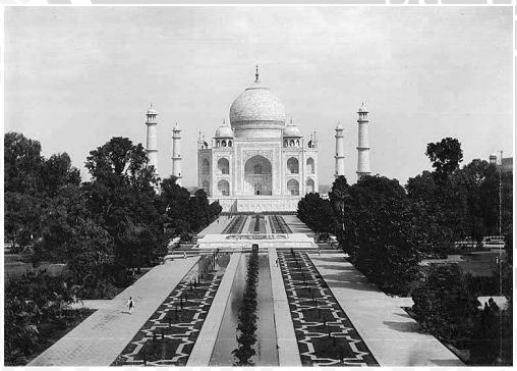
Implementasi algoritma *Pseudo One-Time Pad* dilakukan pada program Matlab. Pengujian dilakukan pada citra grayscale dan berwarna. Semua citra dapat dienkripsi dan didekripsi menjadi citra tepat seperti semula. Kunci enkripsi adalah parameter nilai awal *logistic map*, pada penelitian ini kunci yang digunakan adalah 0,625.

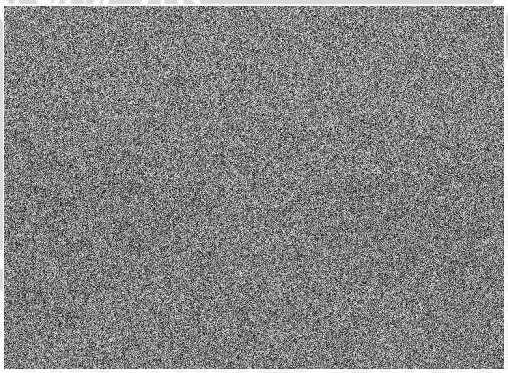
Gambar 2.3 memperlihatkan hasil pengujian enkripsi untuk citra berwarna dengan ukuran 500 × 374 dan kedalaman warna 24-bit. Dekripsi menghasilkan citra yang tepat sama dengan citra semula. Gambar 2.4 memperlihatkan contoh hasil enkripsi untuk citra grayscale dengan ukuran 768 × 573 dan kedalaman warna 8-bit. Dekripsi menghasilkan citra yang tepat sama dengan citra semula.



(a) Citra ‘Gedung Sate’ (b) Citra ‘Gedung Sate’ Terenkripsi

Gambar 2.3 Enkripsi citra berwarna (‘Gedung Sate’)





(a) Citra ‘Taj Mahal’ (b) Citra ‘Taj Mahal’ terenkripsi

Gambar 2.4 Enkripsi citra gayscale (‘Taj Mahal’)

Sumber : Munir, 201

Hasil pengujian didapatkan bahwa algoritma enkripsi ini dapat mengenkripsi citra dengan baik dan mendekripsinya kembali sama seperti citra semula. Pengujian perubahan parameter nilai chaos memperlihatkan bahwa algoritma ini aman terhadap serangan exhaustive attack.

2.1.2 Penelitian Implementasi Enkripsi Gambar dengan Algoritma *One Time Pad* Menggunakan Metode Logistic Map pada FPGA. Oleh Darmawan Lahru Riatma.

Pada penelitian yang dilakukan oleh Darmawan lahru Riatma, peneliti melakukan perancangan dan implementasi enkripsi data gambar yang diimplementasikan pada FPGA Spartan 3e. Hasil pengujian menunjukkan implemantasi algoritma One-Time Pad berhasil dilakukan pada FPGA Spartan 3e, kriptografi yang diimplementasikan berhasil melakukan enkripsi dan dekripsi pada citra. Namun pada penelitian ini masih menggunakan citra buatan sendiri yang tidak lebih dari 400 pixel dengan kedalaman warna 3 bit.

Algoritma One-time pad menggunakan persamaan yang berbeda pada enkripsi dan dekripsi. Persamaan pada enkripsi ditunjukkan oleh persamaan 1. Sedangkan dekripsi ditunjukkan oleh Persamaan 2.

$$C_1 = P_1 \; XOR \; K_1 \quad (1)$$

$$P_1 = C_1 \; XOR \; K_1 \quad (2)$$

Pada Persamaan (1) digunakan untuk enkripsi, disjungsi eksklusif atau XOR dari plaintext (P_1) dengan secret key (K_1) akan menghasilkan ciphertext (C_1). Sedangkan Persamaan (2) adalah untuk menghasilkan plaintext dari ciphertext menggunakan kunci yang sama.

Kunci pada penelitian ini dibangkitkan menggunakan persamaan *logistic map*. Logistic map merupakan salah satu pseudo-random number generator (PRNG) atau pembangkit bilangan semi acak yang memiliki nilai awal X_i dengan nilai antara 0 sampai 1. Berikut ini adalah persamaan logistic map:

$$X_{i+1} = R.X_i \; (1 - X_i) \quad (3)$$

Perubahan nilai dari desimal ke biner 3 bit dilakukan untuk memenuhi persaratan XOR, yaitu panjang kunci harus sama panjang dengan plaintext. Pada Tabel 2.1 akan dijelaskan perubahan nilai dari hasil logistic map menjadi biner 3 bit.

Tabel 2.1 Algoritma konversi nilai desimal ke biner

Konversi nilai desimal menjadi biner	
Desimal	**Biner**
0-0.125	000
0.126-0.251	001
0.252-0.377	010
0.378-0.503	011
0.504-0.629	100
0.630-0.755	101
0.756-0.881	110
0.882-1.0	111

Hasil pengujian validitas yang didapatkan dari penelitian ini dimasukkan dalam tabel validitas, seperti yang ditunjukkan pada Tabel 2.2.

Tabel 2.2 Hasil pengujian validitas

NO	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan	Status Validitas
1	Dekripsi dengan kunci sama	Gambar dekripsi dapat berubah kembali seperti gambar asli	Gambar dekripsi dapat berubah kembali seperti gambar asli	Valid
2	Dekripsi dengan kunci berbeda	Gambar dekripsi tetap acak dan tidak kembali seperti gambar asli	Gambar dekripsi tetapacak dan tidak kembali seperti gambar asli	Valid

2.2 Dasar Teori

Dasar teori membahas tentang teori-teori dasar yang diperlukan untuk menyusun penelitian yang diusulkan. Dasar teori yang digunakan oleh penulis antara lain: kriptografi, algoritma *One-Time Pad*, PRNG, Raspberry Pi, dan Python.

2.2.1 Kriptografi

Kriptografi adalah teknik pengamanan pesan dengan mengubahnya menjadi pesan yang isinya berbeda berbada. Pengertian kriptografi dalam kamus bahasa Inggris OxFord adalah sebuah teknik rahasia dalam penulisan, dengan karakter khusus, dengan menggunakan huruf dan karakter diluar bentuk aslinya, atau dengan metode-metode lain yang hanya dapat dipahami oleh pihak-pihak yang memproses kunci, juga semua hal yang ditulis dengan cara seperti ini. Tujuan kiptografi adalah kerahasiaan, integritas data, otentikasi, dan nirpenyalahgunaan (Munir, 2006).

Di dalam kriptografi akan sering ditemukan berbagai istilah atau terminologi. Beberapa istilah yang penting untuk diketahui diberikan penjelasan dibawah ini.

1. Plaintext : pesan asli yang dapat terbaca. Plaintext adalah masukan bagi algoritma enkripsi. Untuk selanjutnya digunakan istilah teks asli sebagai padanan kata plaintext.
2. Secret key : sekret key juga salah satu masukan bagi algoritma yang merupakan nilai yang bebas terhadap teks asli dan menentukan hasil keluaran algoritma enkripsi.
3. Ciphertext : Ciphertext dapat dianggap sebagai pesan dalam bentuk tersembunyi. Ciphertext merupakan keluaran atau hasil enkripsi. Algoritma enkripsi yang baik menghasilkan ciphertext yang terlihat acak.
4. Algoritma : langkah-langkah atau cara yang dilakukan untuk enkripsi, entah dengan persamaan matematis atau tradisional.
5. Enkripsi : proses merubah paintext dengan secret key menggunakan algoritma, sehingga menghasilkan ciphertext.
6. Dekripsi : proses mengembalikan ciphertext menjadi plaintext seperti semula menggunakan algoritma dan *secretkey*.

Berdasarkan jenis kuncinya algoritma kriptografi terbagi menjadi dua janis, yaitu algoritma simetri atau algoritma kunci privat dan algritma asimetris yang sering disebut algoritma kunci publik.

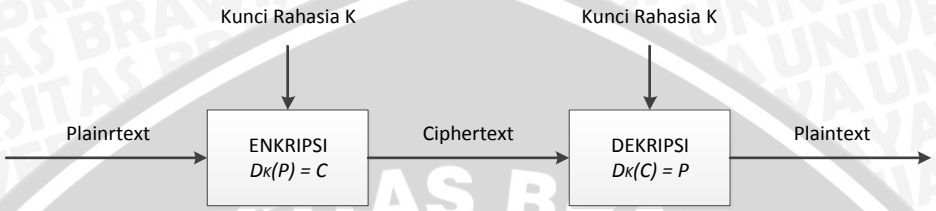
1. Algoritma Simetris

   Algoritma simetris termasuk salah satu jenis kunci dalam kriptografi yang mana menggunakan kunci sama pada saat proses enkripsi dan dekripsinya. Kriptografi semetri diasumsikan sebagai pengirim dan penerima pesan yang telah berbagi kunci yang sama sebelum bertukar pesan. Keamanan kriptografi simetri terdapat pada kerahasiaan kuncinya.

   Secara umum, cipher yang termasuk ke dalam kriptografi simetris beroperasi dalam mode blok (blok cipher), yaitu setiap kali enkripsi/dekripsi dilakukan terhadap satu blok data (yang berukuran tertentu), atau beroperasi dalam mode aliran (stream cipher), yaitu setiap

kali enkripsi/dekripsi dilakukan terhadap satu satuan atau beberapa blok data.

Salah satu kelebihan pada algoritma simetris yaitu proses enkripsi dan dekripsinya jauh lebih cepat dibandingkan dengan algoritma asimetris. Sedangkan kelemahannya yaitu pada permasalahan ditribusi kunci (key distribution) [ZEL-12]. Skema kriptografi simetri ditunjukan pada Gambar 2.5.



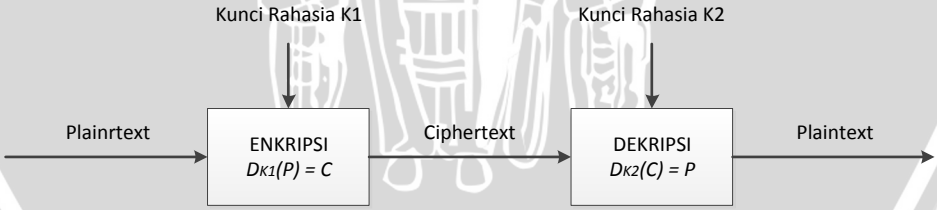
Gambar 2.5 Skema kriptografi simetris

Sumber : Munir, 2006

2. Algoritma Asimetris

Algoritma asimetris atau yang sering disebut dengan algoritma kunci publik. Dimana kunci untuk enkripsi tidak rahasia (diumumkan ke publik), sementara kunci dekripsinya bersifat rahasia (hanya diketahui oleh penerima pesan).

Sistem ini memiliki dua keuntungan, yang pertama yaitu tidak ada kebutuhan untuk menditribusikan kunci, sebagaimana kunci pada sistem kriptografi simetri. Kedua kunci publik dapat dikirim kepenerima melalui saluran yang sama dengan saluran yang digunakan untuk mengirim pesan. Saluran untuk mengirim pesan umumnya tidak aman. Skema kriptoografi asimetris ditunjukkan pada Gambar 2.6.



Gambar 2.6 Skema proses kriptografi asimetris

Sumber : Munir, 2006

2.2.2 Algoritma *One Time Pad*

One-time pad (OTP) adalah *stream cipher* yang melakukan enkripsi dan dekripsi setiap satu karekter. Algoritma ini ditemukan oleh Major Joseph Mauborgne pada tahun 1917, sebagai pengembangan dari Verman cipher untuk menghasilkan keamanan yang lebih baik. Mauborgne mengusulkan penggunaan *one-time pad* (pad = ketas kotak-kotak) yang berisi deretan karakter-karekter

kunci yang dibangkitkan secara acak. Satu pad hanya digunakan sekali (one-time) saja untuk mengenkripsi pesan, setelah itu pad yang telah digunakan dihancurkan supaya tidak dipakai kembali untuk mengenkripsi pesan yang lain.

Algoritma *One-Time Pad* merupakan algoritma yang mencapai kerahasiaan sempurna yaitu menghasilkan teks sandi yang tidak memiliki hubungan statistik terhadap teks asli, sehingga analisa statistik sulit untuk dilakukan (Sadikin, 2012). Persamaan yang digunakan dalam OTP adalah disjungsi eksklusif atau XOR. XOR merupakan operasi Boolean yang hanya memiliki dua nilai, yaitu true atau false. Dalam bahasa matematika true akan diwakili oleh angka 1, sedangkan false diwakili oleh angka 0. Dalam matematika operasi XOR dinotasikan dengan $\oplus$. Berikut adalah tebel kebenaran dari logika XOR.

Tabel 2.3 Logika disjungsi eksklusif

A	B	A $\oplus$ B
0	0	1
0	1	0
1	0	0
1	1	1

2.2.3 PRNG

Pseudorandom Number Generator (PRNG) adalah algoritma yang digunakan untuk membangkitkan barisan bilangan semi-acak. Barisan bulangan yang dihasilkan tidak dapat disebut sebagai barisan bilangan yang benar-benar acak, karena untuk membangkitkannya menggunakan proses matematis. Ada beberapa algoritma yang dapat membangkitkan barisan bilangan acak, salah satunya adalah Mersenne Twister.

Mersenne Twister dikembangkan oleh Makoto Matsumoto dan Takuji Nishimura. Mersenne Twister telah dijadikan sebagai PRNG bawaan pada beberapa perangkat lunak, beberapa diantaranya adalah MATLAB, PHP, dan Python, dikarenakan lebih dapat diandalkan daripada algoritma yang lebih tua atau usang (IBM, 2011). Para penulis mengklaim algoritma ini memiliki 1,5 sampai 2 kali lebih cepat dari *Advanced Encryption Standard* dalam mode counter (Matsumoto, 2005).

Uji autokorelasi merupakan salah satu uji statistik yang digunakan untuk mengetahui ada atau tidaknya korelasi antara satu pengamatan dengan pengamatan lain dengan model regresi. Pada penelitian ini, autokorelasi digunakan untuk menguji ada atau tidaknya korelasi antara bilangan yang dibangkitkan dengan Mersenne Twister. Metode pengujian yang sering digunakan adalah dengan uji Durbin-Watson (uji DW), dengan ketentuan sebagai berikut:

1. Deteksi Autokorelasi Positif

- Jika d < dL maka terdapat autokorelasi positif.
- Jika d > dU maka tidak terdapat autokorelasi positif.
- Jika dL < d < dU maka pengujian tidak meyakinkan atau tidak ada kesimpulan pasti.

2. Deteksi Autokorelasi Negatif
   - Jika (4 – d) < dL maka terdapat autokorelasi negatif.
   - Jika (4 – d) > dU maka tidak terdapat autokorelasi negatif.
   - Jika dL < (4 – d) < dU maka pengujian tidak meyakinkan atau tidak ada kesimpulan pasti.

Nilai dL dan dU dapat diperoleh dari tabel statistik Durbin Watson yang bergantung banyaknya observasi serta variabel yang digunakan. Salah satu program yang dapat menghitung nilai Durbin Watson adalah SPSS.

2.2.4 Raspberry Pi

Raspberry merupakan salah satu mini *Personal Computer* (PC) yang dikembangkan oleh Raspberry Pi Foundation di UK. Raspberry Pi 2 merupakan generasi kedua yang menggantikan model B+ pada Februari 2015. Dengan ARMv7, Rarpberry Pi dapat menjalankan *full ARM GNU/Linux Distribution*, Snappy Ubuntu Core, dan bahkan Windows 10. Rapberry Pi 2 menggunakan Micro SD Card untuk melakukan booting dan storage, dengan spesifikasi berikut:

1. Chipset Broadcom BCM2836 SoC
2. Prosesor Quad-core ARM Cortex-A7
3. Dual Core VideoCore IV® Multimedia co-Prosesor
4. Memori RAM 1 GB LPDDR2
5. Catu daya Micro USB socket 5V, 2A
6. Port Ethernet 10/100 BaseT Ethernet socket
7. Output Video HDMI (rev 1.3 dan 1.4)
8. Output Audio 3.5mm jack, HDMI
9. 4 x USB 2.0 Connector
10. Pin GPIO 40-pin 2.54 mm
11. 15-pin MIPI Camera Serial Interface (CSI-2)

Gambar 2.7 Rapberry Pi 2 model B

Sumber : Wikipedia.com

Sistem operasi yang sering digunakan untuk Raspberry Pi adalah Raspbian. Raspbian adalah *free Operating System* berbasis Debian yang telah dioptimasi oleh Raspberry. Berbagai macam bahasa pemrograman dapat digunakan dalam sistem operasi Raspbian ini, salah satunya adalah Python.

2.2.5 Python

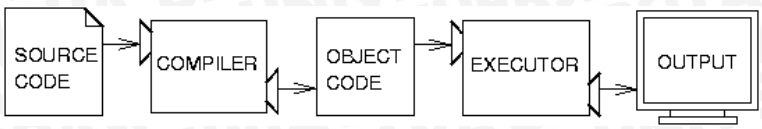
Python merupakan salah satu bahasa pemrograman tingkat tinggi (*High Level language*) yang bersifat interpreter, interaktif, dan berorientasi objek. Pada dasarnya semua perangkat komputer menggunakan bahasa tinkat rendah (*Low Level Language*), yang sering disebut bahasa mesin atau bahasa assembly. Dengan demikian, suatu program yang menggunakan *High Level language* harus diproses dengan menerjemahkannya ke dalam *Low Level Language* sebelum program tersebut dapat dijalankan.

Terdapat dua proses pemrograman agar suatu program dapat dijalankan oleh komputer, yaitu: *Interpreter* dan *Compiler*.

Gambar 2.8 Proses eksekusi program

Sumber : greenteapress.com

Sedangkan *compiler* membaca program secara keseluruhan, yaitu menerjemahkan seluruh instruksi dalam program sekaligus.

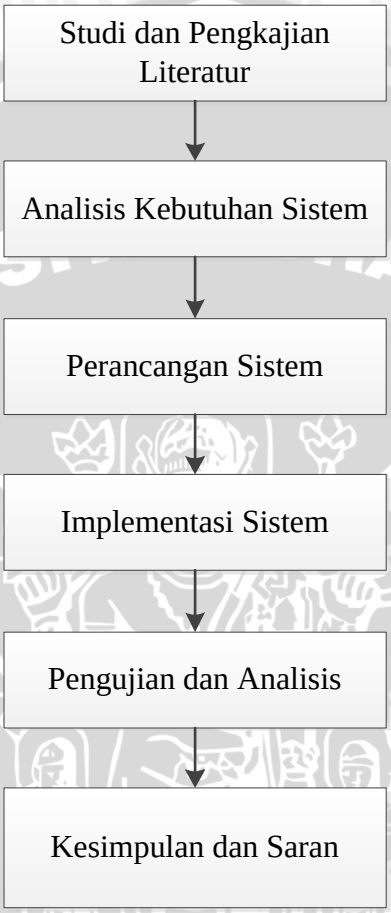


Gambar 2.9 Proses kompilasi

Sumber : greenteapress.com

BAB 3 METODOLOGI

Bab ini menjelaskan langkah-langkah yang akan ditempuh dalam penyusunan penelitian, meliputi studi dan pengkajian litarature, analisis kebutuhan sistem, perancangan sistem, implementasi sistem, pengujian dan analisis, kesimpulan dan saran.



Gambar 3.1 Flowchart metode penelitian

3.1 Studi dan Pengkajian Literatur

Dalam perancangan dan implementasi penelitian ini, perlu diadakan studi literatur. Literatur digunakan sebagai teori penguat dan landasan dasar dalam penelitian. Teori pendukung tersebut didapat dari buku, jurnal, paper, dan internet. Literatur yang digunakan meliputi:

1. Kriptografi
2. Algoritma *One Time Pad*
3. *Pseudorandom Number Generator (PRNG)*
4. Rapberry Pi
5. Python

3.2 Analisis Kebutuhan Sistem

Analisis kebutuhan bertujuan untuk menganalisa semua kebutuhan yang diperlukan oleh sistem yang akan dibangun. Analisa kebutuhan dilakukan dengan mengidentifikasi kebutuhan sistem. Analisi kebutuhan tersebut meliputi, kebutuhan *software* dan kebutuhan *hardware*.

3.3 Perancangan Sistem

Perancangan sistem dilakukan setelah semua kebutuhan sistem didapatkan melalui tahap analisis kebutuhan yang meliputi kebutuhan *software* dan kebutuhan *hardware*. Setelah semua sistem terpenuhi selanjutnya adalah perancangan *software* dan perancangan *hardware*.

3.4 Implementasi

Implementasi merupakan perancangan program kriptografi gambar menggunakan bahasa pemrograman Python. Python memiliki *library* yang dapat membangkitkan PRNG dengan fungsi *random*.

Proses kriptografi dilakukan pada Raspberry Pi dengan sistem operasi Raspbian. Raspberry Pi dikendalikan dengan komputer menggunakan remote terminal. Komputer dapat mengendalikan Raspberry Pi dengan menghubungkan keduanya dengan kabel ethernet. Pengiriman file antara Rapberry Pi dengan komputer menggunakan program WinSCP.

3.5 Pengujian dan Analisis

Pengujian sistem bertujuan untuk mengetahui seberapa baik sistem dapat berjalan. Lebih detailnya pengujian meliputi:

1. Pengujian tingkat kecocokan gambar asli dengan hasil dekripsi.
2. Pengujian waktu yang dibutuhkan untuk melakukan enkripsi dan dekripsi.
3. Pengujian tingkat keacakan barisan bilangan yang dihasilkan PRNG Mersenne Twister.

3.6 Kesimpulan

Pengambilan kesimpulan dilakukan setelah semua tahapan perancangan, implementasi, pengujian dan analisis telah selesai dilakukan. Kesimpulan diambil dari hasil pengujian dan analisis terhadap sistem yang dibuat.

BAB 4 PERANCANGAN DAN IMPLEMENTASI

Bab ini menjelaskan tentang perancangan sistem kriptografi citra dengan algoritma *One-Time Pad* menggunakan *Pseudorandom Number Generator* (PRNG) sebagai kunci. Tahap awal yaitu tahap analisis kebutuhan yang meliputi kebutuhan *software* dan kebutuhan *hardware*. Setelah kebutuhan analisis sistem terpenuhi, selanjutnya adalah perancangan *software* dan perancangan *hardware*.

4.1 Perancangan

Perancangan berisi tentang pembuatan alur kinerja sistem, baik secara bagian maupun keseluruhan. Bagian ini meliputi analisa kebutuhan sistem, diagram blok sistem untuk perancangan enkripsi serta dekripsi, dan algoritma sistem untuk perancangan perangkat lunak.

4.1.1 Analisa Kebutuhan Sistem

Analisis kebutuhan merupakan langkah awal sebelum dilakukan perancangan *software* dan perancangan *hardware*. Analisa ini terdiri dari 2 bagian, yaitu analisa kebutuhan perangkat keras dan analisa perangkat lunak. Penguraian sistem secara sederhana, sehingga dapat diambil poin-poin tentang kebutuhan sistem.

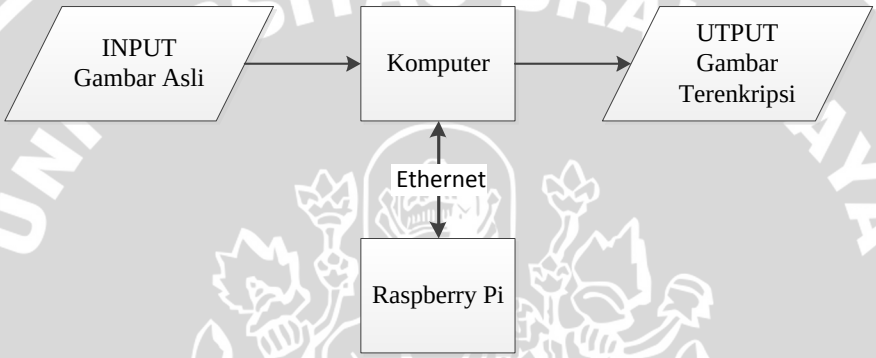
Pada penelitian ini, sistem kriptografi dijalankan pada Raspberry Pi dengan sistem operasi Raspbian. Raspberry Pi dapat dijalankan secara individu dengan menambahkan perangkat tambahan berupa monitor dan keyboard, atau dengan menggunakan komputer sebagai *remote*. Komputer memerlukan port ethernet untuk dapat terhubung dan mengontrol Rapberry Pi, dan kabel ethernet sebagai sambungan. Komputer mengontrol Raspberry Pi untuk mendapatkan akses direktori dan terminal, sehingga dapat memasukkan dan menjalankan program. Berdasarkan uraian singkat mengenai cara kerja Raspberry Pi ini, sehingga didapatkan tentang kebutuhan sistem sebagai berikut:

1. Raspberry Pi, mini komputer sebagai perangkat utama untuk menjalankan sistem. Pada penelitian ini menggunakan Raspberry Pi 2 B.
2. Komputer, sebagai perangkat tambahan untuk dapat menjalakan Raspberry menggunakan *remote* direktori dan terminal.
3. Kabel ethernet, sebagai perantara untuk menghubungkan antara Raspberry Pi dan komputer.
4. Raspbian, sistem operasi yang digunakan untuk menjalankan Raspberry Pi. Pada penelitian ini menggunakan minimal Raspbian, yang beroperasi tanpa GUI.
5. WinSCP, perangkat lunak yang digunakan mengakses direktori Raspberry Pi dari komputer.

6. PuTTY, perangkat lunak untuk mengakses terminal Rapberry Pi dari komputer.
7. Python, sebagai bahasa pemrograman yang digunakan dalam sistem.
8. File editor, yang digunakan untuk menuliskan program Python.

4.1.2 Perancangan Hardware

Pada perancangan ini, dijelaskan hubungan antar *hardware* yang digunakan pada penelitian ini. Perangkat keras yang ada pada penelitian ini terdiri dari komputer, Raspberry Pi, dan kabel ethernet. Komputer dihubungkan dengan Raspberry Pi menggunakan kabel ethernet, seperti pada Gambar 4. 1. Komputer memerlukan alamat ip, *username* dan *password* untuk dapat mengakses Raspberry Pi menggunakan *remote* terminal.



Gambar 4. 1 Perancangan hardware

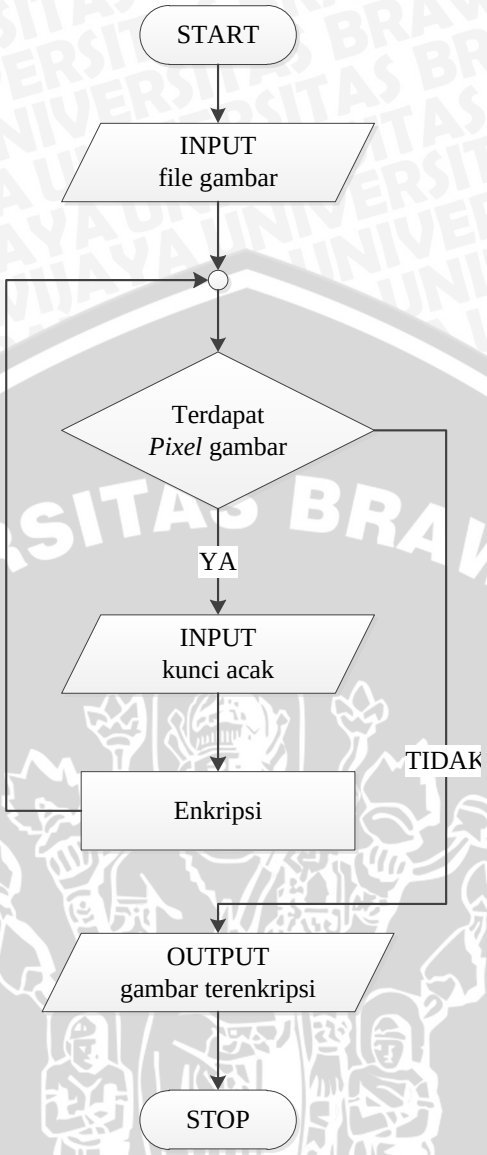
Komputer terhubung dengan Raspberry Pi memelalui kabel ethernet, menggunakan koneksi *Secure Shell* (SSH). Program WinSCP dan PuTTY dijalankan pada komputer untuk dapat mengakses direktori dan terminal Raspberry Pi. Raspberry Pi menerima semua masukan melalui komputer, yang diteruskan melalui program PuTTY maupun WinSCP. Salah satunya adalah gambar yang disiapkan di komputer, dengan menggunakan aplikasi WinSCP dikirim ke Raspberry Pi. Gambar kemudian dienkripsi di Raspberry Pi menggunakan program yang dijalankan melalui *remote* terminal pada komputer, yang hasilnya dapat dilihat pada direktori Raspberry Pi melalui WinSCP.

4.1.3 Perancangan Algoritma Sistem

Bagian ini menjelaskan tentang perancangan algoritma untuk *source code* yang akan digunakan untuk mengolah gambar. perancangan yang dilakukan pada bab ini merupakan gambaran yang kemudian akan diimplementasikan ke dalam bahasa pemrograman. Bahasa pemrograman yang digunakan adalah bahasa Python.

4.1.3.1 Perancangan Algoritma Sistem Secara Garis Besar

Untuk mempermudah dalam proses peancangan algoritma sistem, maka langkah awal yang dilakukan adalah membuat diagram alir / *flowchart* secara umum, seperti yang ditunjukkan pada Gambar 4. 2.

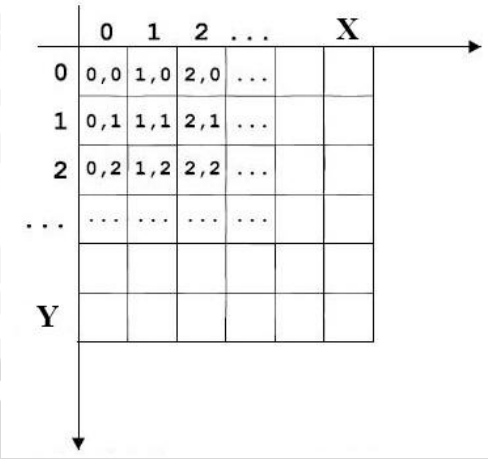


Gambar 4. 2 Diagram alir program secara umum

Langkah pertama yang dilakukan pada bagian ini adalah memberikan masukan berupa file gambar. Enkripsi dilakukan pada tiap pixel, dengan melakukan pengulangan proses selama pixel masih terdapat pada gambar. Setelah didapatkan nilai pada pixel, didapatkan dari program nilai acak sebagai kunci. Nilai pada pixel dienkripsi dengan kunci dari nilai acak.

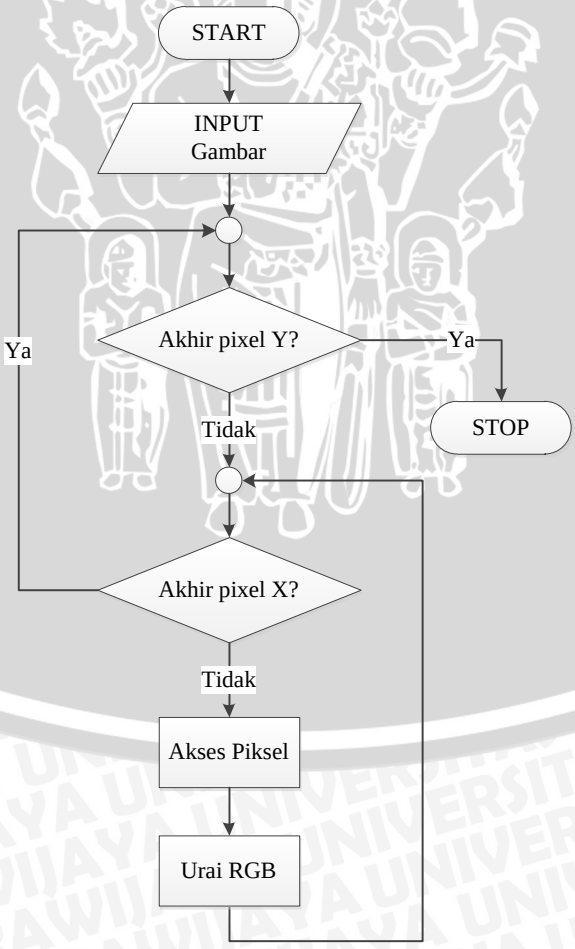
4.1.3.2 Algoritma Penguraian Piksel Gambar

Gambar memiliki total piksel yang dihasilkan dari perkalian panjang dan lebar. Koordinat gambar dimulai dengan titik 0,0 yang terdapat pada ujung kiri atas, seperti pada Gambar 4. 3. untuk mendapatkan nilai dari piksel. Pengulangan dilakukan untuk sumbu Y terlebih dahulu, diawali dengan angka awal dengan nilai 0. Selanjutnya ditentukan nilai sumbu X, yang diawali juga dengan nilai 0. Seperti yang ditampilkan *flowchart* pada Gambar 4. 4.



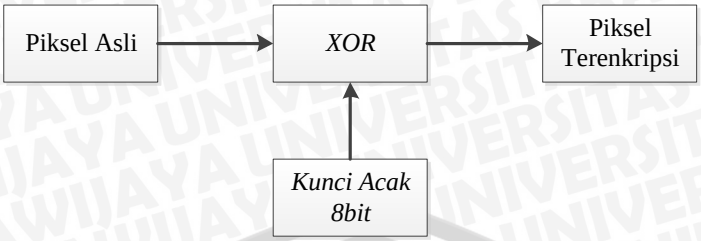
Gambar 4. 3 Koordinat piksel gambar

Dari teori koordinat piksel di atas, maka dapat digambarkan rancangan program yang akan dibuat. Dengan melakukan pengulangan (*looping)* untuk sumbu Y, dari nilai 0 sampai Y maka lakukan pengulangan untuk sumbu X. Sedangkan pada sumbu X dilakukan juga pengulangan dari nilai 0 sampai X, maka ambil piksel pada posisi (sumbu x, sumbu y) sekarang. Piksel dengan nilai RGB disimpan dalam variable rgb_im.



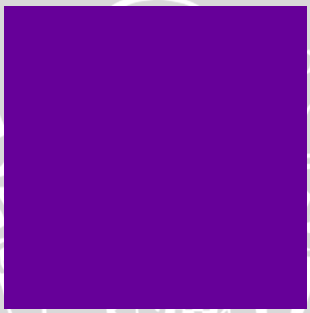
Gambar 4. 4 *Flowchart* penguraian piksel

4.1.3.3 Algoritma Enkripsi



Gambar 4. 5 Diagram blok enkripsi

Pada Gambar 4. 5 merupakan diagram blok khusus untuk proses enkripsi. Dari penguraian pada gambar 4.5 yang yang menghasilkan piksel dengan bit *Red*, *Green*, dan *Blue* (RGB) akan dienkripsi dengan kunci dengan nilai 8 bit. Operasi enkripsi menggunakan logika XOR, sehingga didapatkan piksel terenkripsi atau dengan kata lain memiliki nilai RGB baru.



Gambar 4. 6 Citra dengan RGB (103, 0, 153)

Pada Gambar 4. 6, yang memiliki nilai RGB dengan nilai biner (103, 0, 153) pada piksel pertama akan di-XOR-kan dengan nilai 153. Proses XOR dilakukan pada setiap satuan nilai yang terdapat pada RGB.

R (103) = 01100111

G (0) = 00000000

B (153) = 10011001

Proses XOR pada nilai R:

01100111

10011001 XOR

11111110 (254)

Proses XOR pada nilai G:

00000000

10011001 XOR

10011001 (153)

Proses XOR pada nilai B:

10011001

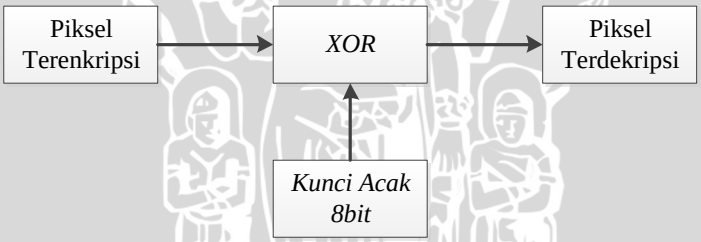
10011001 XOR

00000000 (0)

Nilai 103 di-XOR-kan dengan 153 menjadi 254, 0 menjadi 153, dan 153 menjadi 0. Sehingga nilai RGB baru yang didapatkan pada piksel pertama adalah (254, 153, 0). Proses enkripsi dilakukan terus hingga seluruh piksel pada citra telah di-XOR-kan dengan kunci. Sehingga citra baru hasil enkripsi yang didapatkan ditunjukkan pada Gambar 4. 7.

Gambar 4. 7 Citra dengan RGB (254, 153, 0)

4.1.3.4 Algoritma Dekripsi



Gambar 4. 8 Diagram blok dekripsi

Gambar 4. 8 merupakan diagram blok untuk mendekripsi piksel yang telah terenkripsi. Nilai piksel pada citra dapat kembali seperti semula dengan menggunakan persamaan semula dengan menggunakan kunci yang sama.

Gambar 4. 9 Citra dengan RGB (254, 153, 0)

Pada Gambar 4. 9, yang memiliki nilai RGB (254, 153, 0) pada piksel pertama akan di-XOR-kan dengan nilai 153. Proses XOR dilakukan pada setiap satuan nilai yang terdapat pada RGB.

R (254) = 11111110

G (153) = 10011001

B (0) = 00000000

Proses XOR pada nilai R:

11111110

10011001 XOR

01100111 (103)

Proses XOR pada nilai G:

10011001

10011001 XOR

00000000 (0)

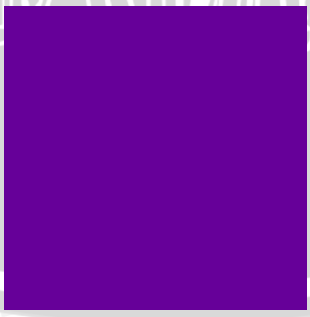
Proses XOR pada nilai B:

00000000

10011001 XOR

10011001 (153)

Nilai 254 di-XOR-kan dengan 153 menjadi 103, 153 menjadi 0, dan 0 menjadi 153. Sehingga nilai RGB baru yang didapatkan pada piksel pertama adalah (103, 0, 153). Proses enkripsi dilakukan terus hingga seluruh piksel pada citra telah di-XOR-kan dengan kunci. Sehingga citra baru hasil dekripsi yang didapatkan ditunjukkan pada Gambar 4. 10.



Gambar 4. 10 Citra dengan RGB (103, 0, 153)

4.2 Implementasi

4.2.1 Lingkungan Implementasi

Lingkungan implementasi dari sistem kriptografi citra dengan algoritma *One-Time Pad* menggunakan *Pseudorandom Number Generator* (PRNG) sebagai kunci meliputi lingkungan perangkat lunak (software) dan lingkungan perangkat keras (Hardware).

4.2.1.1 Lingkungan Perangkat Keras

Lingkungan perangkat keras yang digunakan dalam mengimplementasikan sistem kriptografi citra dengan algoritma *One-Time Pad* menggunakan *Pseudorandom Number Generator* (PRNG) sebagai kunci adalah sebagai berikut :

1. Raspberry Pi
2. Komputer
3. Kabel Ethernet

4.2.1.2 Lingkungan Perangkat Lunak

Lingkungan perangkat lunak yang digunakan dalam mengimplementasikan sistem kriptografi citra dengan algoritma *One-Time Pad* menggunakan *Pseudorandom Number Generator* (PRNG) sebagai kunci adalah sebagai berikut :

1. Raspbian, sistem operasi yang digunakan untuk menjalankan Raspberry Pi. Pada penelitian ini menggunakan minimal Raspbian, yang beroperasi tanpa GUI.
2. WinSCP, perangkat lunak yang digunakan mengakses direktori Raspberry Pi dari komputer.
3. PuTTY, perangkat lunak untuk mengakses terminal Rapberry Pi dari komputer.
4. Python, sebagai bahasa pemrograman yang digunakan dalam sistem.
5. File editor, yang digunakan untuk menuliskan program Python.

4.2.2 Batasan Implementasi

Beberapa batasan dalam implementasi sistem kriptografi citra dengan algoritma *One-Time Pad* menggunakan *Pseudorandom Number Generator* (PRNG) sebagai kunci sebagai berikut:

1. Program enkripsi gambar diimplementasikan di Raspberry Pi menggunakan bahasa pemrograman Python.
2. Gambar yang digunakan memiliki kedalaman warna 3bytes.
3. Kunci yang dihasilkan memiliki nilai 8bit.

4.2.3 Implementasi Perangkat Keras

Implementasi sistem kriptografi citra dengan algoritma *One-Time Pad* menggunakan *Pseudorandom Number Generator* (PRNG) sebagai kunci berdasarkan perancangan pada subbab 4.1. Implementasi perangkat keras pada penelitian ini adalah menghubungkan Raspberry Pi untuk dapat diakses menggunakan komputer.

4.2.4 Implementasi Perangkat Lunak

Pada proses implementasi enkripsi terdapat beberapa proses yaitu inisialisasi gambar, pembentukan kunci acak dengan Mersenne twister, enkripsi gambar dengan kunci menggunakan metode XOR.

4.2.4.1 Inisialisasi Gambar

Tahap awal dalam proses enkripsi adalah inisialisasi gambar, inisialisasi gambar adalah memasukkan gambar pada program, sehingga dapat dikenali warna dan pixelnya. *Source code* inisialisasi gambar akan dijelaskan pada Kode Program 4. 1.

```
im = Image.open("/home/asli1.png")
pixdata = im.load()
rgb_im = im.convert('RGB')
width = im.size[0]
height = im.size[1]
```

Kode Program 4. 1 Program inisialisasi gambar

Program diatas digunakan untuk memasukan gambar yang akan dienripsi. Dengan menggunakan inisialisasi standar agar dapat diakses per piksel, sehingga dapat digunakan untuk enkripsi menggunakan metode *One-Time Pad*.

4.2.4.2 Pembentukan Kunci Acak

Pembentukan kunci acak dilakukan dengan metode Mersenne twister, yang merupakan *Pseudo-Number Randon Generator* (PNRG). Source code pembentukan kunci dengan Mersenne Twister sudah ada dalam *library* milik python. Sehingga untuk membentuk kunci acak, dilakukan dengan memanggil *library* tersebut. Berikut merupakan Kode Program 4. 2 yang digunakan untuk menggunakan *library* PRNG:

```
a = getpass.getpass("Masukkan password: ")
rnd = random.Random (a)
```

Kode Program 4. 2 Fungsi *Pseudoandom Number Generator*

Program di atas merupakan bagian dari program yang digunakan untuk menghasilkan nilai acak yang menggunakan *seed* sebagai pembangkit. Pada bahasa pemrograman Python sudah terdapat fungsi untuk menghasilkan acak. Dengan menggunakan `random.randint` maka fungsi akan menghasilkan nilai

acak. Dari program diatas nilai acak sudah ditentukan, yaitu antara nilai 1 sampai 255. Nilai acak yang dipanggil sebagai kunci yang akan digunakan untuk enkripsi.

4.2.4.3 Proses Enkripsi Citra dengan One-Time Pad

Tahap ketiga adalah proses enkripsi gambar dengan algoritma *One-Time Pad*, proses ini dilakukan sehingga dapat memanggil fungsi kunci acak yang telah dibentuk. Proses enkripsi akan dijelaskan pada Kode Program 4. 3.

```
for y in range(im.height):
    for x in range(im.width):
      R, G, B = rgb_im.getpixel((x, y))
      sandi = rnd.randint(1, 255)
      enR = R ^ sandi
      enG = G ^ sandi
      enB = B ^ sandi
      pixdata[x, y] = (enR, enG, enB)
```

Kode Program 4. 3 Program enkripsi

Baris pertama dan kedua dalam kode program 4.3 sebagai pengulangan untuk mengakses perpiksel dari citra. Pada baris ketiga kode `rgb_im.getpixel` digunakan untuk mendapatkan nilai piksel, yang selanjutnya disimpan pada variabel R untuk warna *red*, G untuk warna *green*, dan B untuk warna *blue*. Sandi didapatkan dengan memanggil fungsi `randint` yang mengacak nomor antara 1 sampai 255. Setiap bit warna pada piksel dienkripsi dengan sandi yang sama, sehingga disusun program pada baris lima sampai tujuh. Baris terakhir dari kode 4.3 digunakan untuk mengumpulkan bit yang telah dienkripsi.

4.2.4.4 Proses Dekripsi Citra dengan One-Time Pad

Tahap dekripsi digunakan untuk mengembalikan gambar yang telah terenkripsi. Dekripsi menggunakan persamaan dan kunci yang sama, sehingga dapat menghasilkan citra yang sama seperti semula. Kelebihan dari operasi XOR adalah dapat mengembalikan nilai yang telah dioperasikan menjadi seperti semula, asal menggunakan pembilang yang sama. Sehingga dapat disimpulkan bahwa kode untuk enkripsi sama persis untuk digunakan sebagai dekripsi, yang membedakan enkripsi dengan dekripsi adalah citra masukan.

BAB 5 PENGUJIAN DAN ANALISIS

Bab ini menjelaskan tentang hasil dari pengujian dan analisis secara keseluruhan.

5.1 Pengujian

Pada pengujian sistem ini akan membahas tingkat kecocokan antara citra asli dengan hasil dekripsi dan waktu yang dibutuhkan untuk melakukan kriptografi. Tingkat kecocokan citra dilakukan pengujian sebanyak sepuluh kali menggunakan citra yang berbeda. Citra yang digunakan pada pengujian ini terdapat pada LAMPIRAN B. Waktu yang dibutuhkan dihitung pada proses enkripsi pada setiap citra. Keacakan kunci untuk pengujian menggunakan *seed* "bekerja".

5.1.1 Pengujian Kriptografi

a. Tujuan :

   Menguji tingkat kecocokan antara citra asli sebelum dienkripsi dengan hasil dekripsi dan waktu yang diperlukan.

b. Peralatan :

   i. Raspberry Pi
   ii. WinSCP
   iii. PuTTY
   iv. Teks Editor
   v. Komputer
   vi. Kabel Ethernet
   vii. Google Chrome

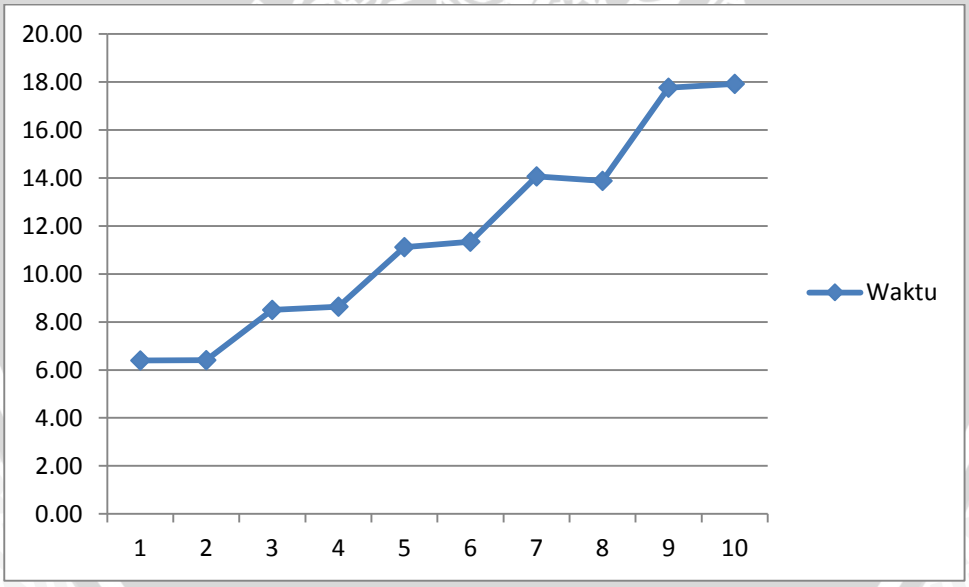
c. Langkah Pengujian :

   i. Nyalakan Raspberry dengan terhubung dengan komputer.
   ii. Edit program enkripsi untuk mengakses citra 1.
   iii. Jalankan program enkripsi.
   iv. Mencatat waktu yang dibutuhkan untuk enkripsi.
   v. Mengubah program dekripsi untuk mengakses citra 1 terenkripsi.
   vi. Menjalankan program dekripsi.
   vii. Buka website http://huddle.github.io/Resemble.js/ untuk uji kecocokan antara dua citra.

d. Hasil :

Tabel 5. 1 Hasil pengujian waktu enkripsi dan kecocokan

Pengujian	Ukuran (piksel)	Waktu (detik)	Persentase Kecocokan (%)
1	300 x 300	6,40	100
2	300 x 300	6,41	100
3	400 x 300	8,51	100
4	400 x 300	8,64	100
5	400 x 400	11,12	100
6	400 x 400	11,35	100
7	500 x 400	14,07	100
8	500 x 400	13,88	100
9	500 x 500	17,76	100
10	500 x 500	17,92	99,20
Rata-rata			99,92



Gambar 5. 1 Pengaruh ukuran piksel terhadap waktu pemrosesan

Berdasarkan grafik pada Gambar 5. 1 didapatkan bahwa semakin besar ukuran piksel pada citra, semakin tinggi pula waktu yang digunakan untuk enkripsi. Pada pengujian dihasilkan tingkat kecocokan citra asli dengan hasil dekripsi memiliki nilai sebesar 99,92 persen.

5.1.2 Pengujian Keacakan Kunci

Pengujian keacakan kunci bertujuan untuk mengetahui tingkat keacakan barisan bilangan yang dihasilkan PRNG. Metode yang digunakan adalah Durbin watson, dengan ketentuan sebagai berikut:

1. Deteksi Autokorelasi Positif
   - Jika d < dL maka terdapat autokorelasi positif.
   - Jika d > dU maka tidak terdapat autokorelasi positif.
   - Jika dL < d < dU maka pengujian tidak meyakinkan atau tidak ada kesimpulan pasti.
2. Deteksi Autokorelasi Negatif
   - Jika (4 – d) < dL maka terdapat autokorelasi negatif.
   - Jika (4 – d) > dU maka tidak terdapat autokorelasi negatif.
   - Jika dL < (4 – d) < dU maka pengujian tidak meyakinkan atau tidak ada kesimpulan pasti.

Barisan bilangan yang dilakukan pengujian menggunakan *seed* 'bekerja'. Pengujian dengan metode Durbin Watson menggunakan program SPSS. Durbin Watson memiliki tabel pegujian yang berisi batas bawah (dL) dan batas atas (dU) nilai, yang memiliki nilai berbeda berdasarkan jumlah sampel dan variabel bebas yang diujikan. Pada pengujian ini menggunakan satu variabel bebas, sehingga batas maksimal yang dapat diuji adalah 200 sampel. Pengujian dilakukan dengan mengambil 200 bilangan sebagai sampel, sehingga ditetapkan pada 200 bilangan pada awal, tengah, dan akhir dari barisan. Dari tabel Durbin Watson didapatkan nilai batas bawah (dL) = 1.7584 dan batas atas (dU) = 1.7785. Nilai dL dan dU digunakan sebagai pembanding untuk menentukan hasil dari tes menggunakan metode Durbin Watson.

5.1.2.1 Barisan Kunci Awal

Pengujian menggunakan program SPSS didapatkan hasil yang ditunjukkan pada Tabel 5. 2 berikut:

Tabel 5. 2 Hasil pengujian pada barisan 1 sampai 200

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate	Durbin-Watson
1	.010[a]	.000	-.005	72.59388	2.058

Nilai Durbin Watson yang didapatkan dari program SPSS pada barisan 1 sampai 200 adalah 2.058, sehingga dapat diambil kesimpulan sebagai berikut:

Berdasarkan Uji Positif

d > dU = tidak terdapat autokorelasi positif

Berdasarkan Uji Negatif

(4-d) > dU = tidak terdapat autokorelasi negatif

Berdasarkan dari hasil pengujian pada Tabel 5. 2 yang kemudian dibandingkan dengan ketentuan Durbin Watson. Perbandingan mendapatkan hasil bahwa barisan nilai yang didapatkan tidak terdapat autokrelasi positif maupun negatif, sehingga bilangan yang dihasilkan tidak memiliki tidak memiliki korelasi satu dengan lainnya.

5.1.2.2 Barisan Kunci Tengah

Barisan kunci pada pengujian kedua adalah pada urutan 50.001 sampai 50.200, sehingga didapatkan hasil sebagai berikut:

Tabel 5. 3 Hasil pengujian pada barisan 50.001 sampai 50.200

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate	Durbin-Watson
1	.020[a]	.000	-.005	72.73444	2.119

Nillai Durbin Watson yang didapatkan dari program SPSS pada barisan 50.001 sampai 50.200 adalah 2.119, sehingga dapat diambil kesimpulan sebagai berikut:

Berdasarkan Uji Positif

d > dU = tidak terdapat autokorelasi positif

Berdasarkan Uji Negatif

(4-d) > dU = tidak terdapat autokorelasi negatif

Berdasarkan dari hasil pengujian pada Tabel 5. 3 yang kemudian dibandingkan dengan ketentuan Durbin Watson. Perbandingan mendapatkan hasil bahwa barisan nilai yang didapatkan tidak terdapat autokrelasi positif maupun negatif, sehingga bilangan yang dihasilkan tidak memiliki tidak memiliki korelasi satu dengan lainnya.

5.1.2.3 Barisan Kunci Akhir

Barisan kunci pada pengujian terakhir adalah pada urutan 119.801 sampai 120.000, sehingga didapatkan hasil sebagai berikut:

Tabel 5. 4 Hasil pengujian pada barisan 119.801 sampai 120.000

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate	Durbin-Watson
1	.118[a]	.014	.009	69.18313	1.984

Nillai Durbin Watson yang didapatkan dari program SPSS pada barisan 500.000 sampai 500.200 adalah 2.119, sehingga dapat diambil kesimpulan sebagai berikut:

Berdasarkan Uji Positif

d > dU = tidak terdapat autokorelasi positif

Berdasarkan Uji Negatif

(4-d) > dU = tidak terdapat autokorelasi negatif

Berdasarkan dari hasil pengujian pada Tabel 5. 4 yang kemudian dibandingkan dengan ketentuan Durbin Watson. Perbandingan mendapatkan hasil bahwa barisan nilai yang didapatkan tidak terdapat autokrelasi positif maupun negatif, sehingga bilangan yang dihasilkan tidak memiliki tidak memiliki korelasi satu dengan lainnya.

5.2 Analisis

Pada subbab analisis dilakukan pembahasan tentang hasil pengujian yang telah dilakukan, yaitu hasil pengujian kriptografi, estimasi waktu, dan keacakan kunci.

5.2.1 Analisis Pengujian Kriptografi

Berdasarkan hasil yang didapatkan dari pengujian enkripsi dan dekripsi pada citra, didapatkan bahwa enkripsi citra menggunakan metode *One-Time Pad* dengan barisan PRNG sebagai kunci menghasilkan citra acak yang tidak dapt dikenali dan dekripsi juga menghasilkan tingkat kecocokan citra sebesar 99,92% dengan aslinya.

Berdasarkan hasil pengujian pada Gambar 5. 1, ternyata ukuran file berpengaruh pada lama waktu enkripsi. Semakin besar ukuran citra yang digunakan, semakin lama pula waktu enkripsi yang digunakan.

5.2.2 Analisis Pengujian Keacakan Kunci

Berdasarkan hasil pengujian pada bab 5.1.2 didapatkan 200 barisan bilangan awal, tengah, dan akhir tidak terdapat korelasi positif maupun negatif yang artinya tidak terdapat hubungan antara bilangan pertama dengan bilangan selanjutnya. Barisan bilangan yang dibangkitkan tidak dapat diprediksi, sehingga dapat dikatakan benar-benar acak.

BAB 6 PENUTUP

6.1 Kesimpulan

Berdasarkan rumusan masalah, hesil perancangan, implementasi, dan pengujian dilakukan, maka diambil kesimpulan sebagai berikut:

1. Tingkat kecocokan antara gambar asli sebelum dilakukan enkripsi dengan hasil dekripsi memiliki persentase sebesar 99,92%.
2. Semakin besar ukuran citra yang digunakan, semakin lama waktu yang dibutuhkan untuk enkripsi dan dekripsi.
3. Berdasarkan uji autokorelasi dengan metode Durbin Watson, PRNG Mersenne Twister memiliki keacakan yang tidak memiliki hubungan antara bilangan satu dengan yang lainnya, sehingga dapat dikatakan benar-benar acak.

6.2 Saran

Saran yang dapat diberikan untuk penelitian selanjutnya adalah:

1. Sistem pada penelitian ini dapat dikembangkan lagi dengan menambahkan kamera sebagai penangkap citra, yang kemudian dienkripsi sebelum disimpan.
2. Untuk meningkatkan kecepatan enkripsi dapat menggunakan komputasi paralel dan penambahan atau perbaikan dari metode yang telah dipakai.

DAFTAR PUSTAKA

Munir, Rinaldi. 2011. *Algoritma Enkripsi Citra dengan Pseudo One-Time Pad yang Menggunakan Sistem Chaos*. Konferensi Nasional Informatika 2011.

Munir, Rinaldi. 2006. *Kriptografi*. Bandung: Informatika.

Krikor Lala, Baba Sami, Arif T, Shaaban Zyad. 2009. *Image Encryption Using DCT and Stream Cipher*. European Journal of Scientific Research : ISSN 1450-216X Vol.32 No.1 (2009), pp.48-58.

Stalling, William. 2003. *Cryptography and Network Security, Principle and Practice 3rd Edition*. Pearson Education, Inc.

Sadikin, Rifki. 2012. *Kriptografi untuk Keamanan Jaringan*. Yogyakarta: Andi.

IBM Knowledge Center. 2011. *Random Number Generator*. Tersedia di: <http:// www-01.ibm.com/support/knowledgecenter/SSLVMB_20.0.0/com.ibm. spss.statistics.help/idh_seed.htm> [Diakses 25 November 2015]

Matsumoto, M., Nishimura, T., Hagita, M., Saito, M. 2005. *Cryptographic Mersenne Twister and Fubuki Stream/Block Cipher*. Departement of Mathematics, Hiroshima University.

LAMPIRAN A

A.1 Source Code Enkripsi

A. asd

#/home/enkrip.py

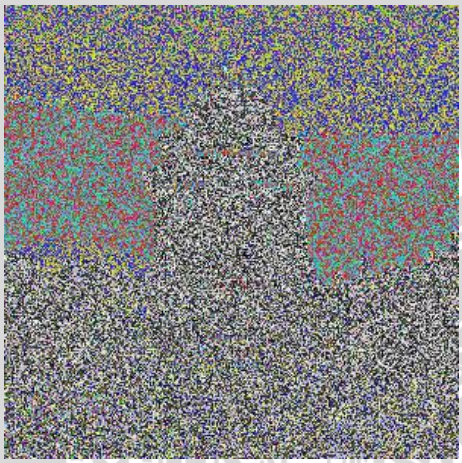
```
from PIL import Image
import random
import getpass
import time

im = Image.open("/home/asli.png")
pixdata = im.load()
rgb_im = im.convert('RGB')
width = im.size[0]
height = im.size[0]
a = getpass.getpass("Masukkan password: ")
rnd = random.Random (a)
start_time = time.time()
for y in range(im.height):
  for x in range(im.width):
    R, G, B = rgb_im.getpixel((x, y))
    sandi = rnd.randint(1, 255)
    enR = R ^ sandi
    enG = G ^ sandi
    enB = B ^ sandi
    pixdata[x, y] = (enR, enG, enB)
end_time = time.time() - start_time
print end_time
print 'done'
im.save("/home/asli.enkrip.png")
```

A.2 Source Code Dekripsi

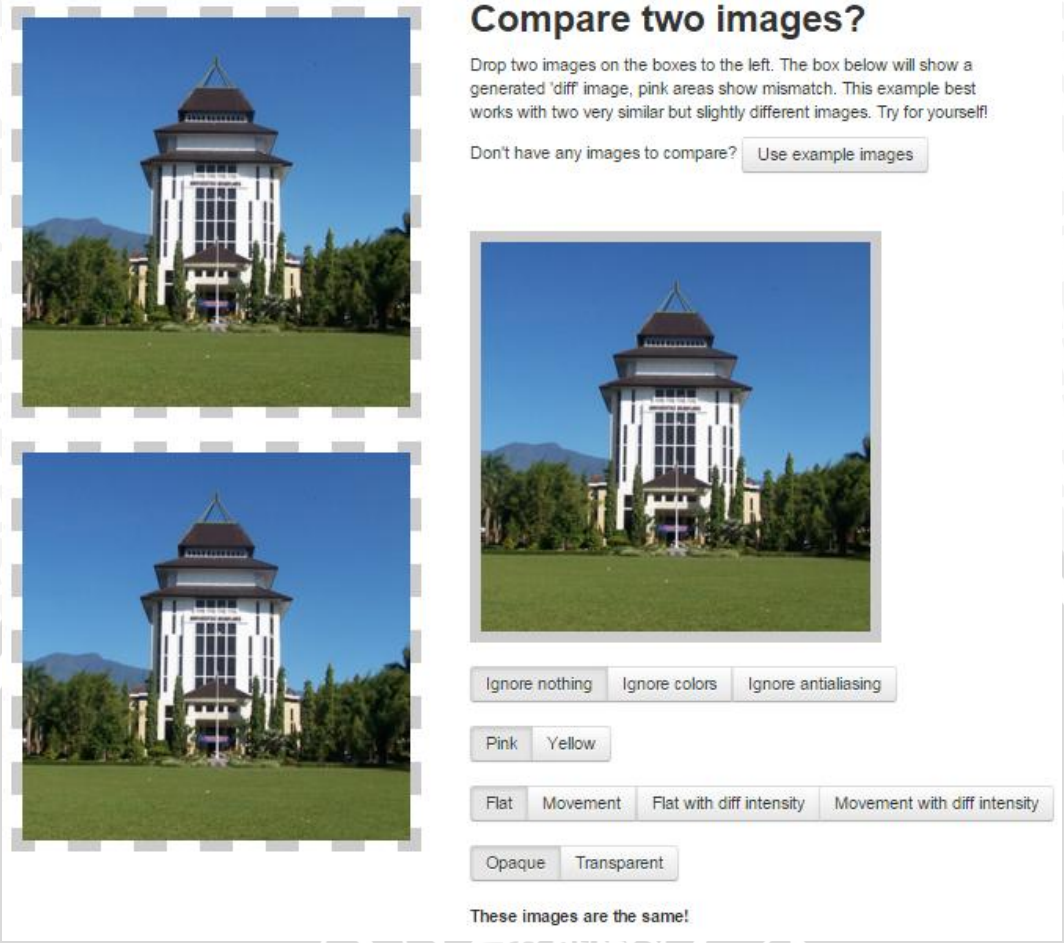
```
#/home/dekrip.py

from PIL import Image
import random
import getpass
import time

im = Image.open("/home/asli.enkrip.png")
pixdata = im.load()
rgb_im = im.convert('RGB')
width = im.size[0]
height = im.size[0]
a = getpass.getpass("Masukkan password: ")
rnd = random.Random (a)
def dekrip():
   output = rnd.randint(1, 255)
   enR = R ^ output
   enG = G ^ output
   enB = B ^ output
   pixdata[x, y] = (enR, enG, enB)
start_time = time.time()
for y in range(im.height):
   for x in range(im.width):
      R, G, B = rgb_im.getpixel((x, y))
       dekrip()
end_time = time.time() - start_time
print end_time
im.save("/home/asli.dekrip.png")
print 'done'
#conn.close()
```

LAMPIRAN B

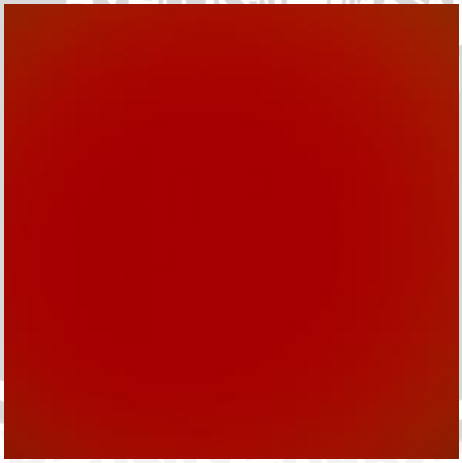
B.1 Pengujian 1 Citra 300 x 300 Piksel

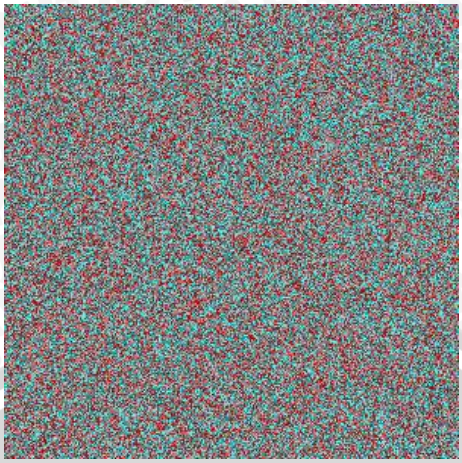


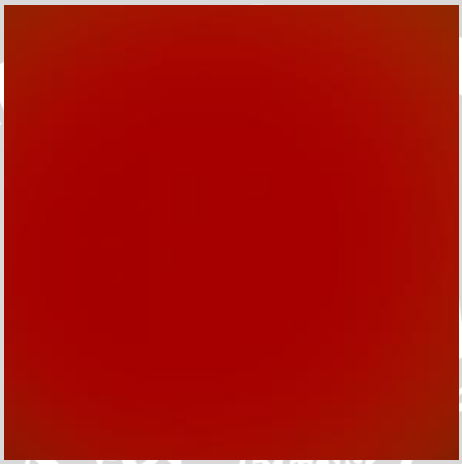
```
192.168.137.199 - PuTTY
root@raspberrypi:/home# python enkrip.py
Masukkan password:
6.40887403488
done
```



B.2 Pengujian 1 Citra 300 x 300 Piksel

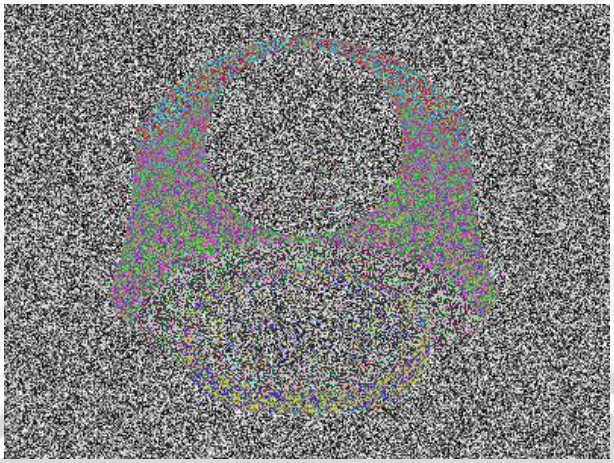




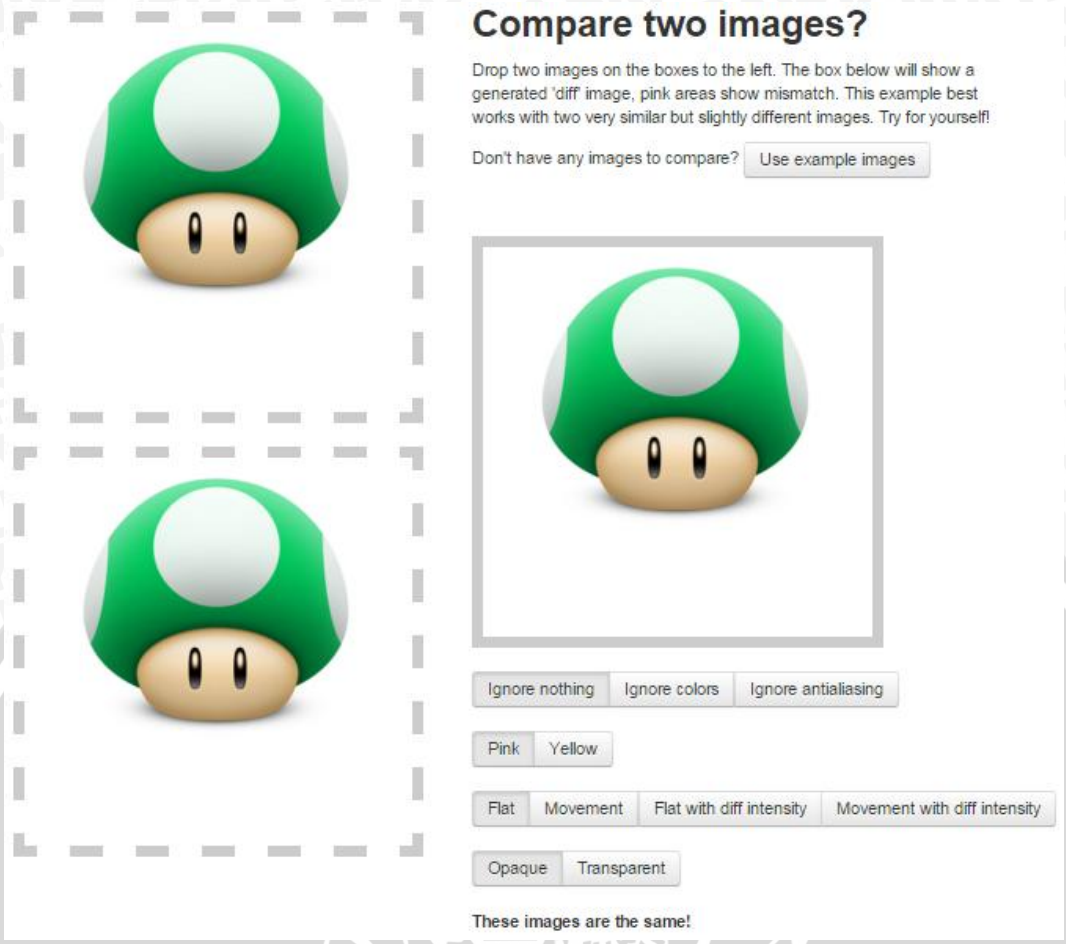


```
root@raspberrypi:/home# python enkrip.py
Masukkan password:
6.41119718552
done
```

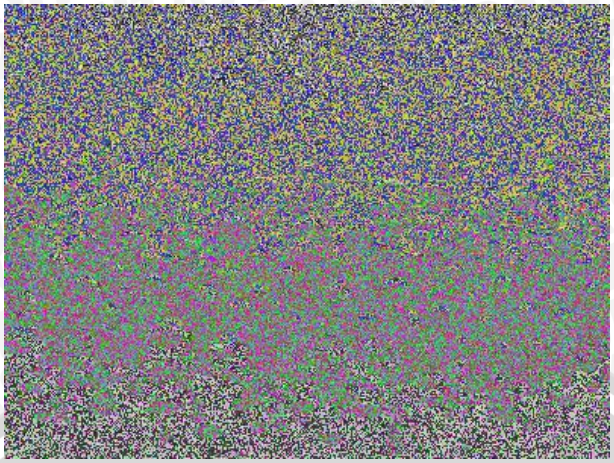
B.3 Pengujian 1 Citra 400 x 300 Piksel



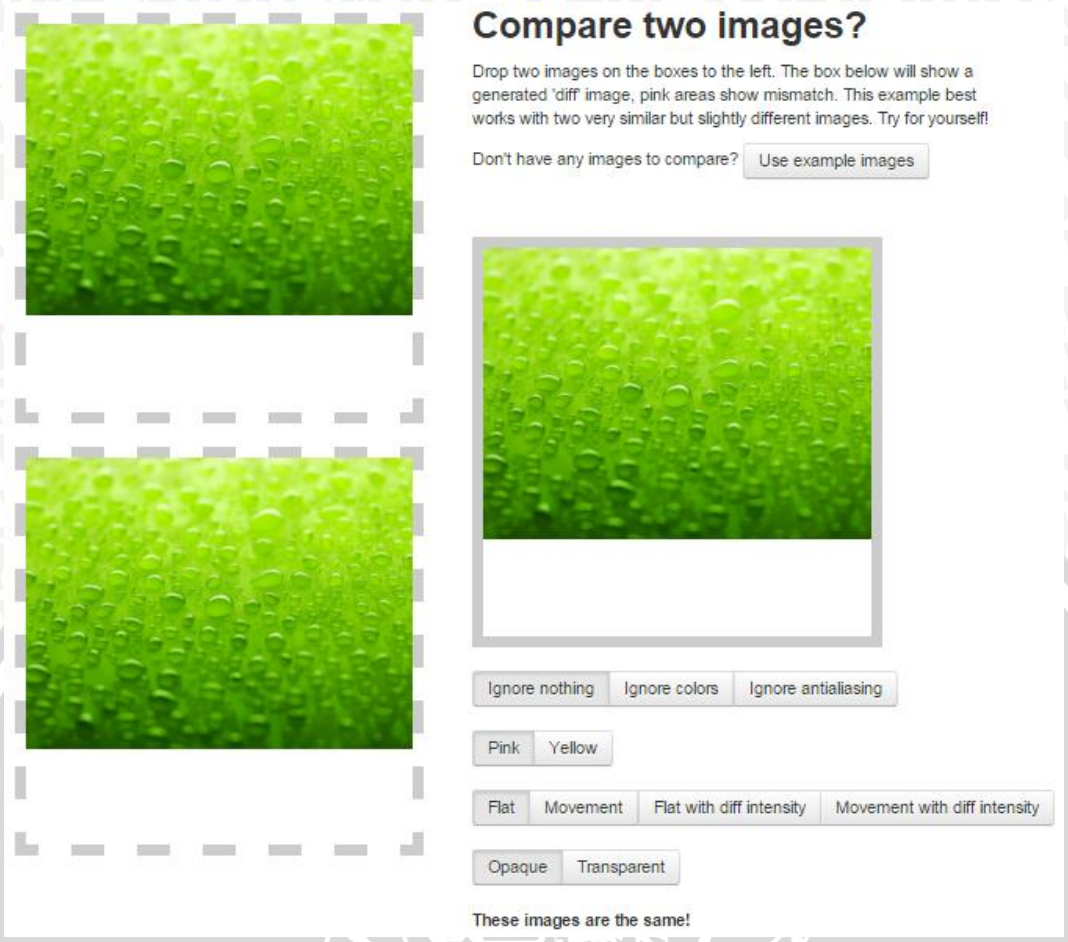
```
root@raspberrypi:/home# python enkrip.py
Masukkan password:
8.51134490967
done
```



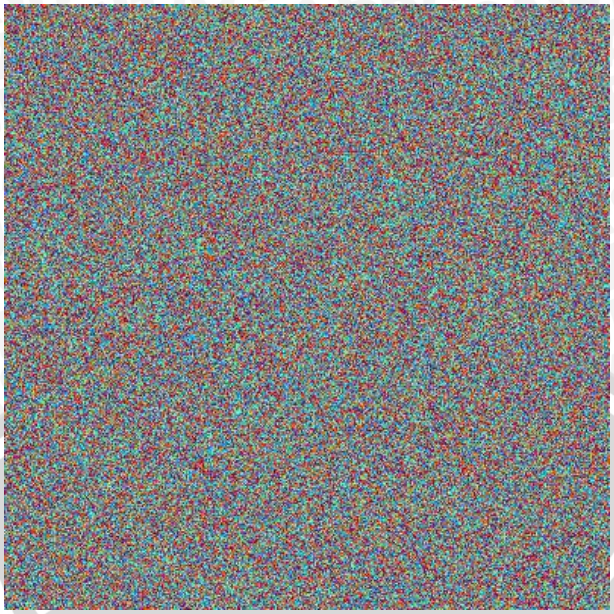
B.4 Pengujian 1 Citra 400 x 300 Piksel



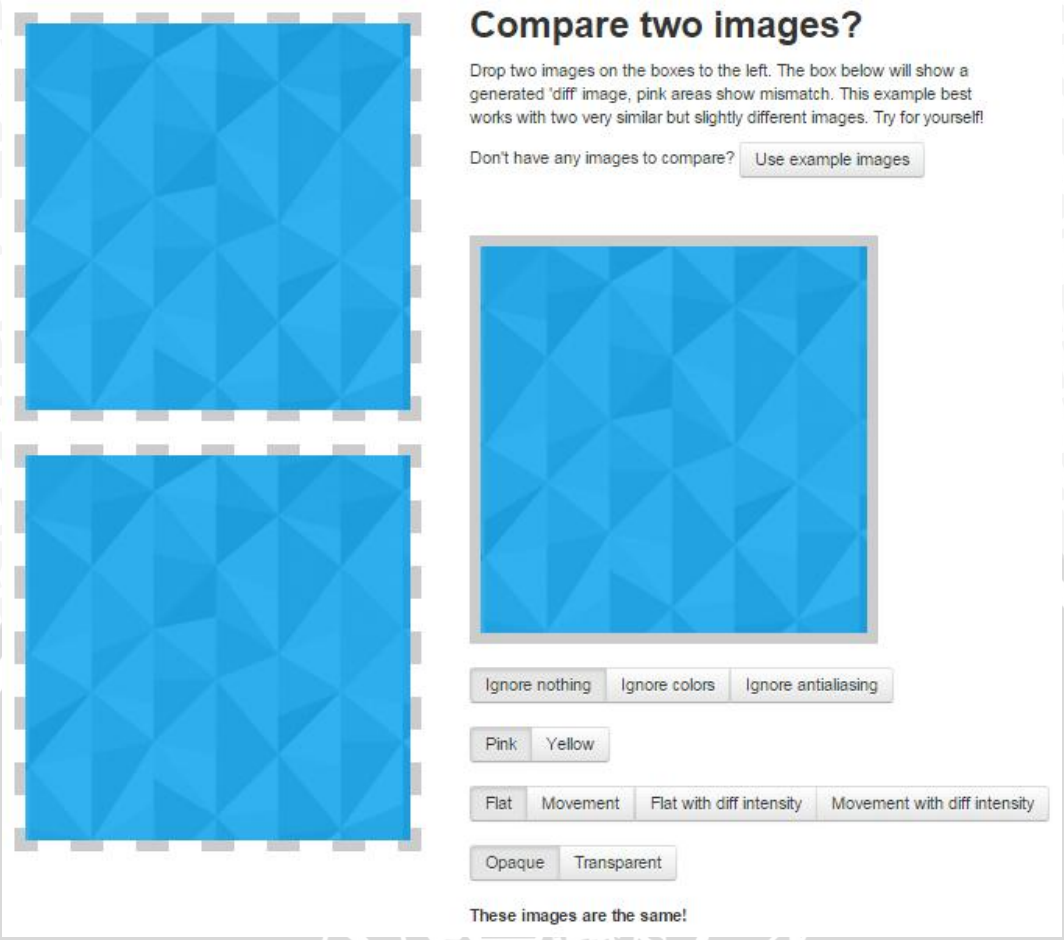
```
root@raspberrypi:/home# python enkrip.py
Masukkan password:
8.64812803268
done
```



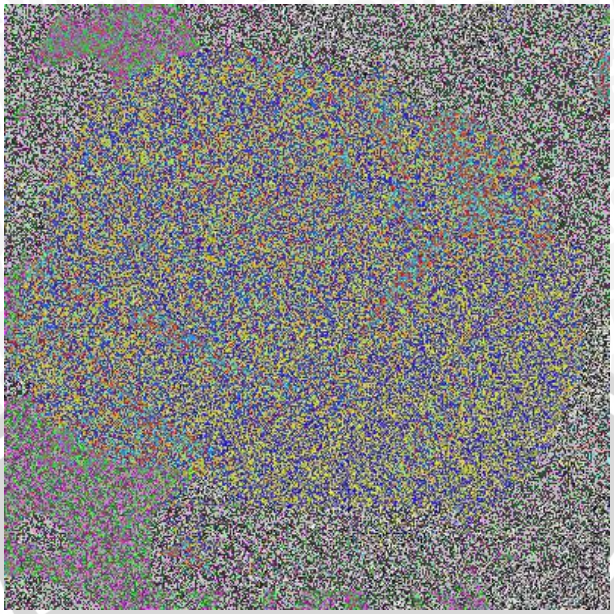
B.5 Pengujian 1 Citra 400 x 400 Piksel



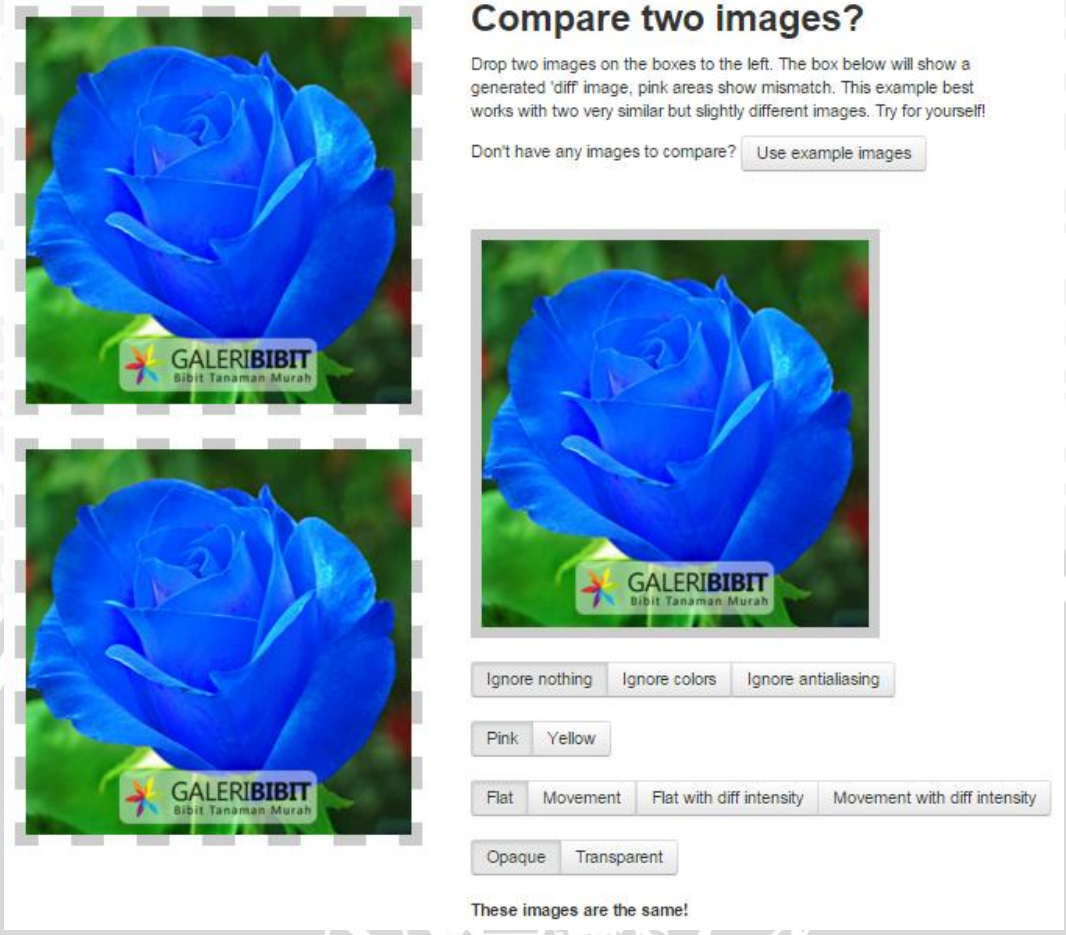
```
root@raspberrypi:/home# python enkrip.py
Masukkan password:
11.1232450008
done
```



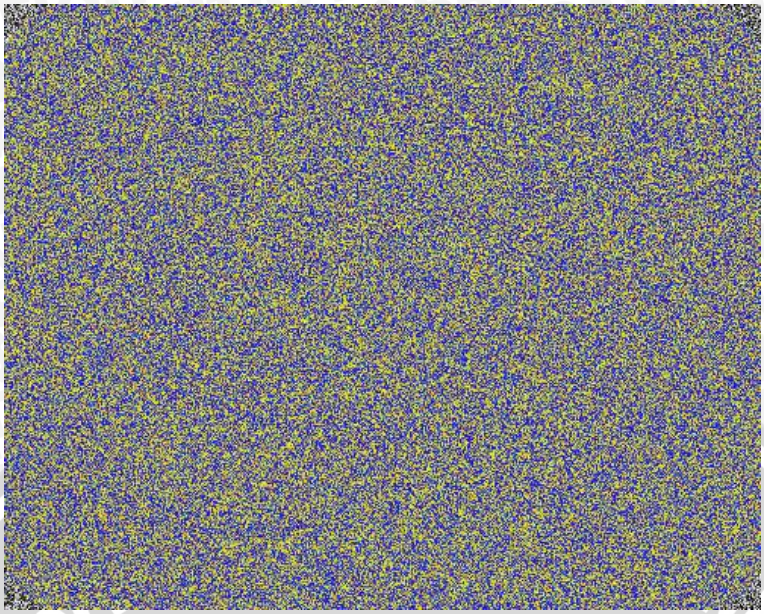
B.6 Pengujian 1 Citra 400 x 400 Piksel




```
root@raspberrypi:/home# python enkrip.py
Masukkan password:
11.3557889462
done
```

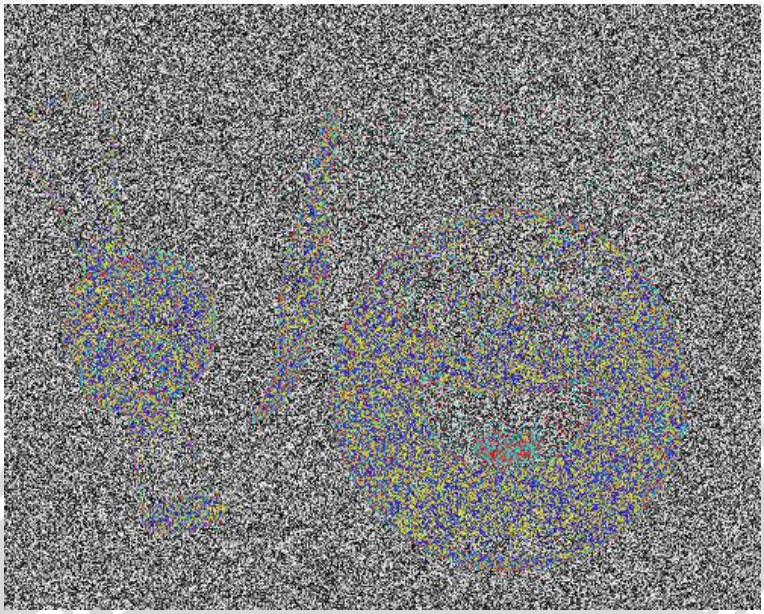


B.7 Pengujian 1 Citra 500 x 400 Piksel

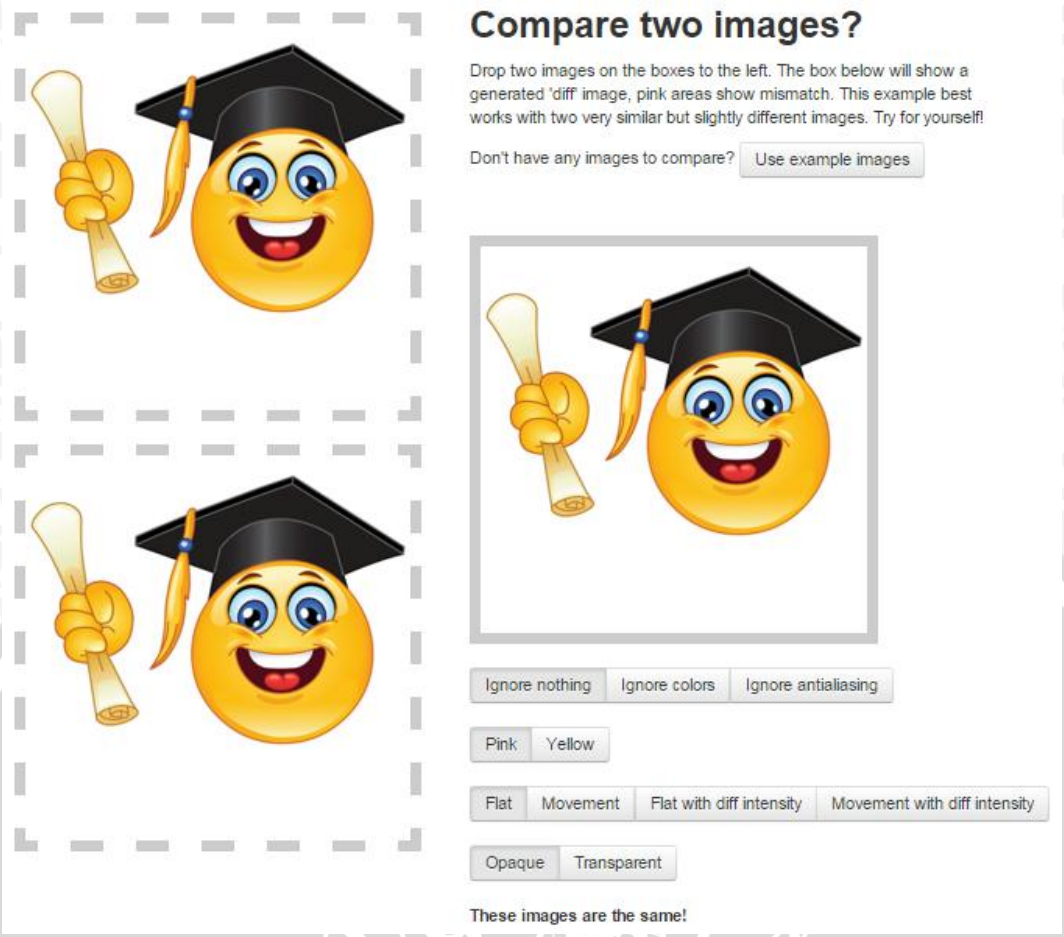



```
root@raspberrypi:/home# python enkrip.py
Masukkan password:
14.0732350349
done
```

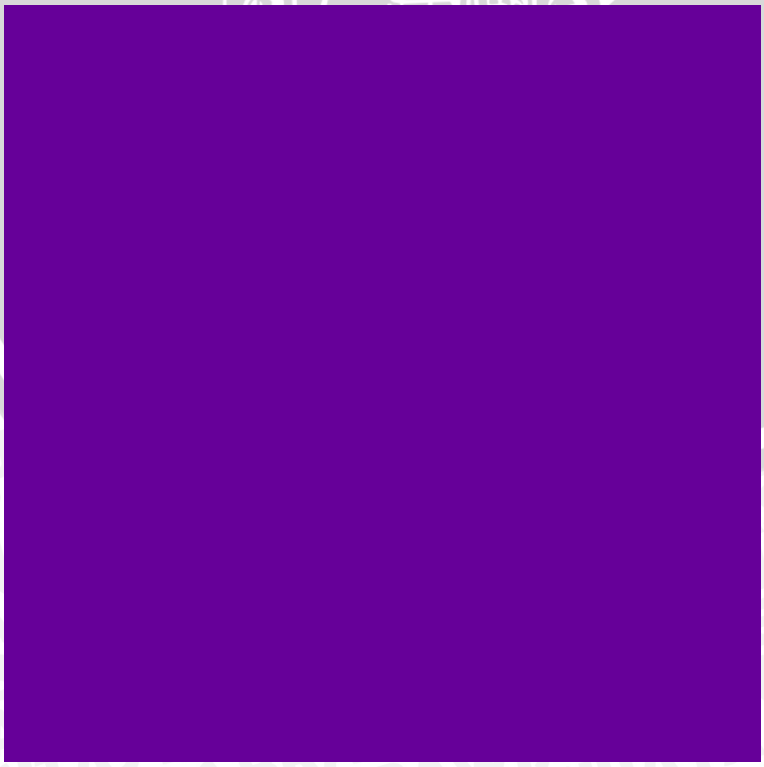
B.8 Pengujian 1 Citra 500 x 400 Piksel

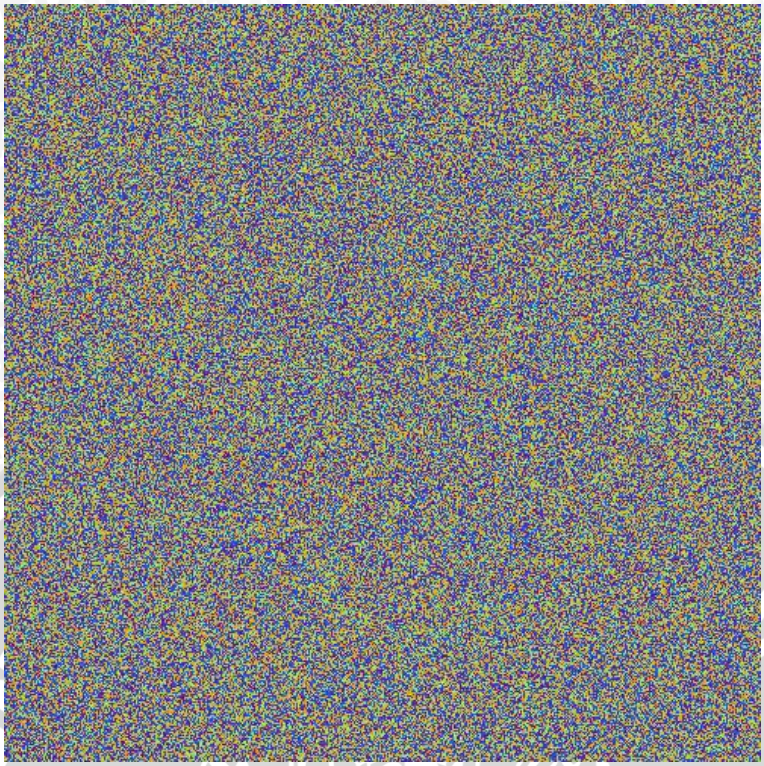


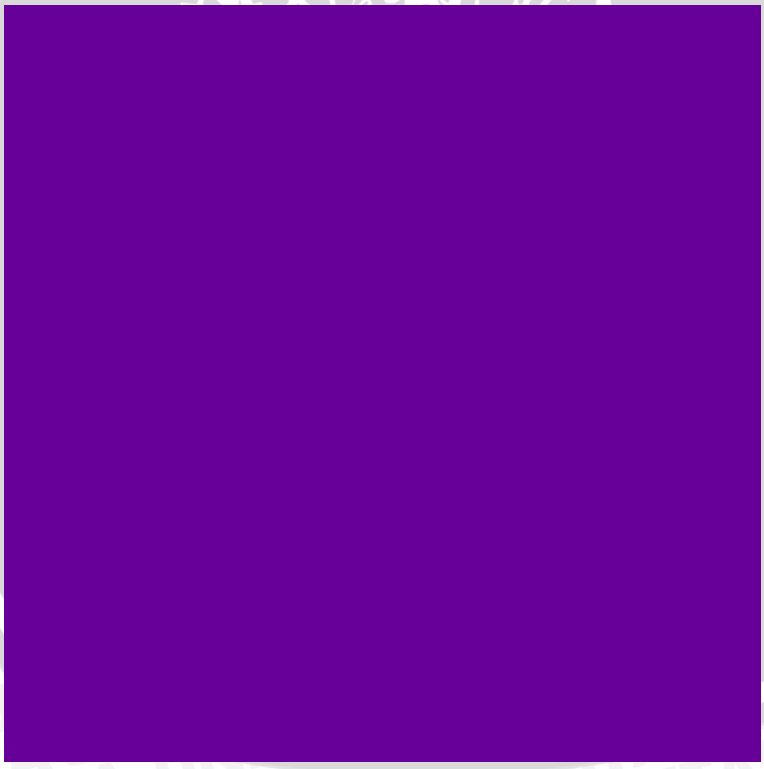
```
root@raspberrypi:/home# python enkrip.py
Masukkan password:
13.8826379776
done
```



B.9 Pengujian 1 Citra 500 x 500 Piksel

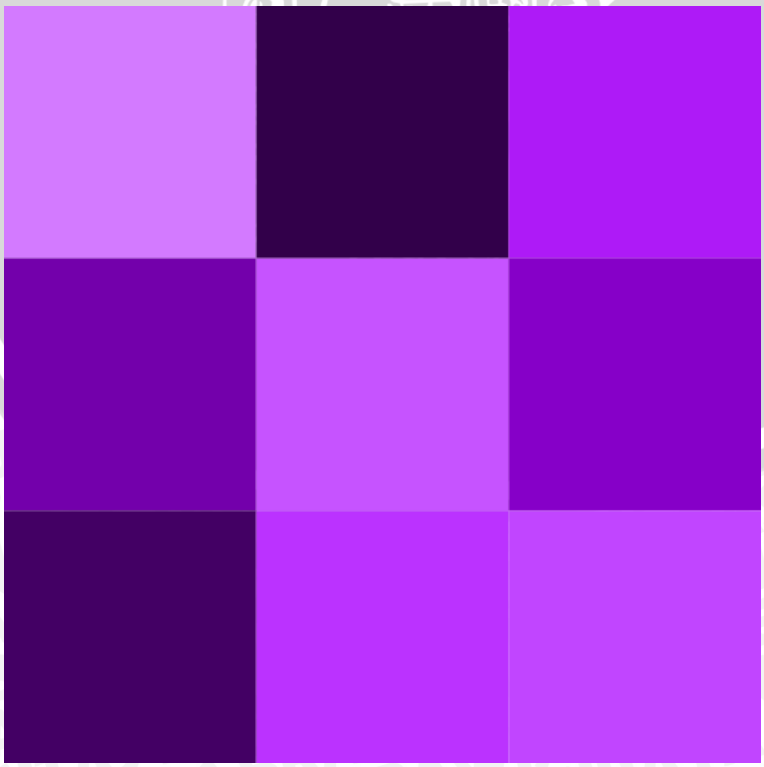


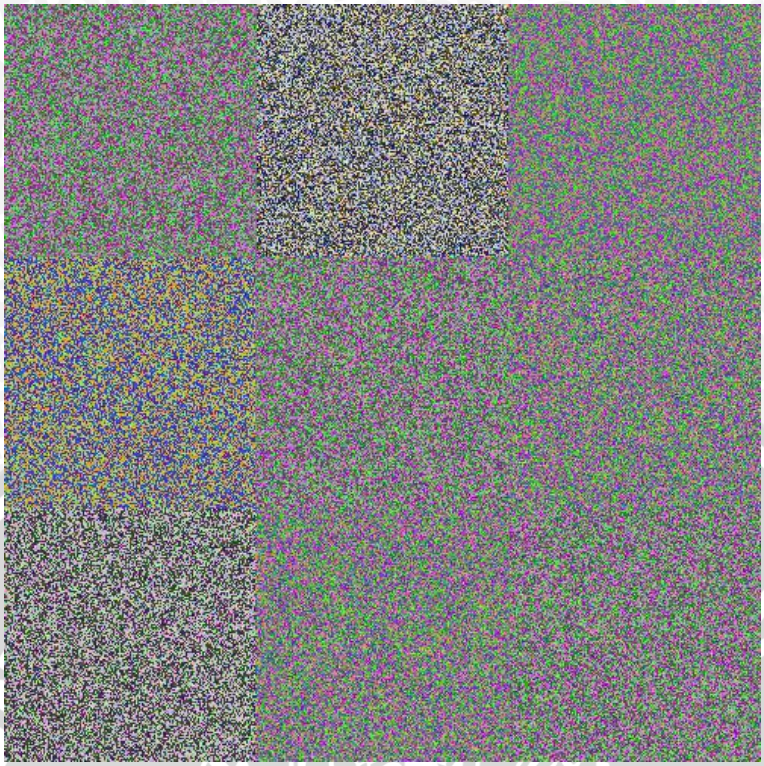


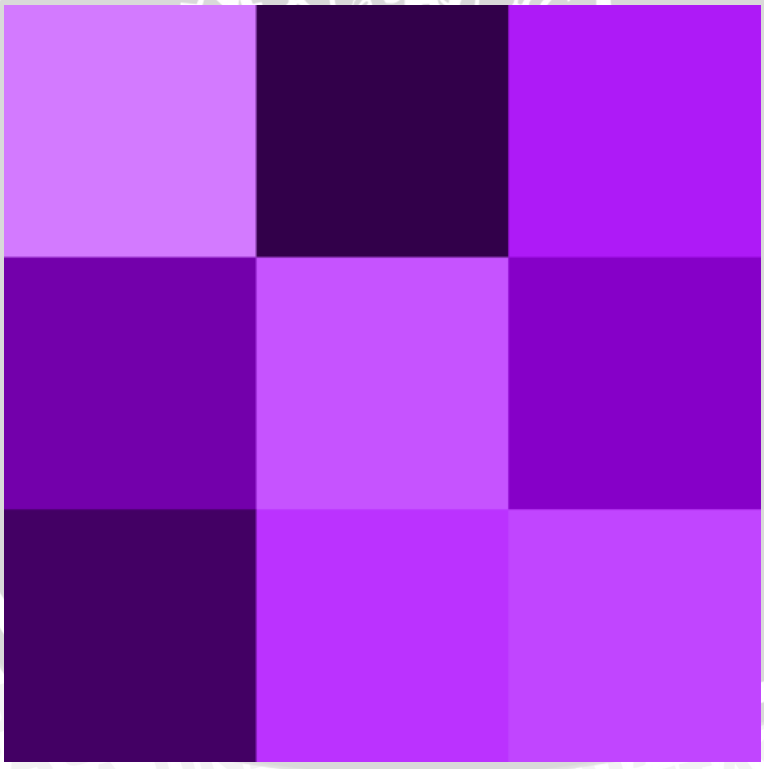


```
root@raspberrypi:/home# python enkrip.py
Masukkan password:
17.7638669014
done
```

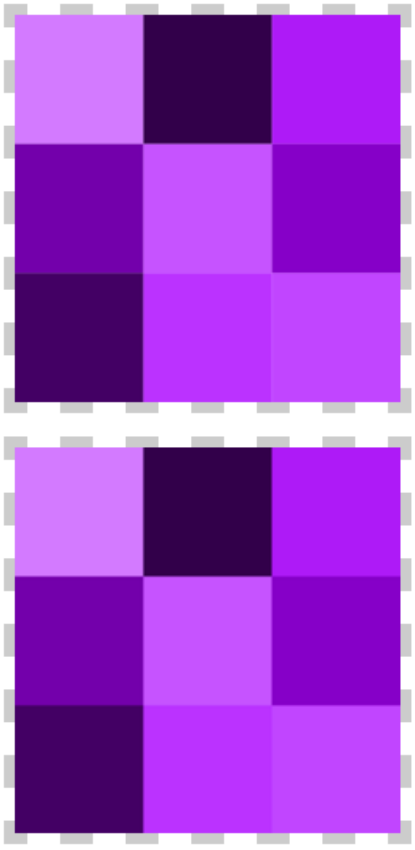
B.10 Pengujian 1 Citra 500 x 500 Piksel







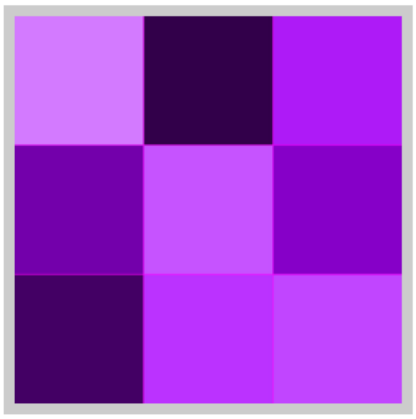
```
root@raspberrypi:/home# python enkrip.py
Masukkan password:
17.9266250134
done
```



Compare two images?

Drop two images on the boxes to the left. The box below will show a generated 'diff' image, pink areas show mismatch. This example best works with two very similar but slightly different images. Try for yourself!

Don't have any images to compare? Use example images



Ignore nothing | Ignore colors | Ignore antialiasing

Pink | Yellow

Flat | Movement | Flat with diff intensity | Movement with diff intensity

Opaque | Transparent

The second image is 0.80% different compared to the first.

Use the buttons above to change the comparison algorithm. Perhaps you don't care about color? Annoying antialiasing causing too much noise? Resemble.js offers multiple comparison options.