JURNAL FASILKOM
Volume 14 No. 3 | Desember 2024: 615-626

P-ISSN : 2089-3353
E-ISSN : 2808-9162

Implementasi Algoritma *ChaCha20* Pada Pengamanan File Citra Bitmap

Fitra Rahim[1], Yusuf Ramadhan Nasution[2], Supiyandi[3]
[1, 2] Ilmu Komputer, Sains dan Teknologi, Universitas Islam Negeri Sumatera Utara Medan
[3] Universitas Pembangunan Panca Budi
[1]fitra.rahim@uinsu.ac.id *, [2]ramadhannst@uinsu.ac.id, [3]supiyandi.mkom@gmail.com

Abstract

The advancement of information technology has facilitated the exchange of digital images, including bitmap formats such as BMP and PNG. However, these formats lack inherent security features, leaving them vulnerable to unauthorized access. To address this issue, the ChaCha20 stream cipher algorithm is explored as a lightweight and efficient solution for securing bitmap images. ChaCha20 generates a keystream from a 512-bit initial state block via quarterround operations, enabling rapid encryption and decryption without sacrificing security. The results of this study show that ChaCha20 effectively secures images, as evidenced by the marked differences between the original and encrypted images, observed through visual inspection and histogram analysis. Of the 20 decrypted images, 18 were identical to the originals, while two showed minor changes in file size and imperceptible pixel color alterations, attributed to differences in OpenCV codec interpretation. File size has a greater impact on execution time than its format, and the performance analysis of encryption-decryption shows minimal execution time differences in most ChaCha20 samples, indicating the stability of this algorithm's performance. Each ciphertext increased by 12 bytes to store the nonce at the end of the file. Furthermore, PNG format images exhibited a file size increase of up to 2,704%, indicates inefficiency in compressing encrypted data. Therefore, further research is recommended to optimize system performance and efficiency while adding support for JPG and GIF formats, in order to provide broader benefits in image file security. This enhancement will increase the applicability of the algorithm across various types of image files commonly used in the industry.

Keywords: chacha20 algorithm, cryptography, data security, image encryption, stream cipher

Abstrak

Kemajuan teknologi informasi telah mempermudah pertukaran citra digital, termasuk format bitmap seperti BMP dan PNG. Namun, format citra ini tidak memiliki fitur keamanan bawaan, sehingga rentan terhadap akses tidak sah. Untuk mengatasi hal ini, algoritma stream cipher ChaCha20 dieksplorasi sebagai solusi alternatif yang ringan dan efisien dalam mengamankan citra bitmap. ChaCha20 bekerja dengan menghasilkan keystream dari blok initial state berukuran 512-bit melalui operasi quarterround, yang memungkinkan enkripsi dan dekripsi data secara cepat tanpa mengorbankan keamanan. Hasil penelitian menunjukkan bahwa ChaCha20 efektif dalam mengamankan citra, terbukti dari perbedaan mencolok antara citra asli dan terenkripsi, yang terdeteksi melalui pengamatan visual dan analisis histogram. Dari 20 citra yang didekripsi, 18 di antaranya identik dengan citra asli, sedangkan dua lainnya mengalami perubahan ukuran berkas dan warna piksel yang tidak kasat mata, yang disebabkan oleh perbedaan interpretasi codec OpenCV. Ukuran berkas lebih memengaruhi waktu pemrosesan dibandingkan formatnya, dan analisis kinerja enkripsi-dekripsi menunjukkan perbedaan waktu yang minimal pada sebagian besar sampel ChaCha20, mengindikasikan stabilitas performa algoritma ini. Setiap ciphertext mengalami peningkatan ukuran 12 byte untuk penyimpanan nonce di akhir berkas. Selain itu, format PNG mengalami peningkatan ukuran file hingga 2.704%, menunjukkan adanya inefisiensi dalam kompresi data terenkripsi. Oleh karena itu, penelitian lanjutan disarankan untuk melakukan pengoptimalan & efisiensi kinerja sistem serta menambah dukungan format JPG dan GIF, agar memberikan manfaat yang lebih luas dalam pengamanan file citra. Penambahan ini akan meningkatkan penerapan algoritma pada berbagai jenis file citra yang umum di industri.

Kata kunci: algoritma chacha20, kriptografi, keamanan data, enkripsi citra, stream cipher

1. Pendahuluan

Dengan majunya perkembangan teknologi informasi, pertukaran informasi dari satu orang ke yang lain jadi semakin dimudahkan. Namun begitu, hal ini dapat menjadi bumerang jika data informasi yang dipertukarkan bersifat sensitif dan rahasia. Salah satu data yang cukup sering dipertukarkan saat ini adalah berkas citra bitmap. Citra bitmap merupakan representasi dari citra grafis yang terdiri dari susunan titik yang tersimpan di memori komputer. Bitmap merupakan salah satu bentuk representasi citra digital di komputer, yang umumnya diwujudkan dalam berkas dengan ekstensi BMP dan PNG. Kedua format berkas ini banyak digunakan karena kompatibilitasnya yang luas dengan berbagai program penampil citra. Berkas bitmap dapat mengandung informasi sensitif dan rahasia yang penting bagi individu, organisasi, perusahaan, bahkan pemerintah. Oleh karena itu, akses terhadap informasi tersebut harus dibatasi hanya kepada pihak yang memiliki otorisasi autentik. Mengingat sifat rahasia dari informasi yang terkandung dalam citra, keamanan data harus dijaga agar tidak bocor ke pihak yang tidak berwenang. Untuk tujuan ini, kriptografi digunakan sebagai teknik

P-ISSN : 2089-3353
E-ISSN : 2808-9162

untuk melindungi informasi dengan mengenkripsi isi citra sehingga isinya tidak dapat dipahami oleh pihak yang tidak berwenang. Terdapat berbagai algoritma kriptografi yang dapat digunakan untuk mengamankan data, salah satunya adalah block cipher AES (Advanced Encryption Standard). AES secara luas digunakan karena standar keamanannya yang tinggi, meskipun proses enkripsinya kompleks dan membutuhkan sumber daya komputasi yang signifikan. Namun, Langley dan Nir [1] berpendapat bahwa bergantung hanya pada AES tidak bijaksana, karena jika algoritma ini berhasil dipecahkan, dibutuhkan alternatif cepat dan aman. Salah satu alternatif yang diusulkan adalah ChaCha20, algoritma enkripsi yang diperkenalkan oleh Daniel Julius Bernstein pada 2008. ChaCha20 adalah stream cipher kunci simetris, di mana enkripsi dilakukan dengan operasi XOR antara bit plaintext dan keystream. Algoritma ini menghasilkan keystream acak menggunakan fungsi pseudorandom yang mengacak empat masukan utama : konstanta, kunci, counter, dan nonce, menggunakan operasi ARX (Addition, Rotation & XOR). Sementara, proses dekripsi dilakukan dengan operasi XOR antara ciphertext dan keystream. Penelitian [2] menunjukkan penggunaan ChaCha20 dalam pengamanan pesan dengan kombinasi steganografi dan kriptografi, di mana pesan dienkripsi dan disisipkan dalam bit acak dengan penanda dinamis. Penelitian lain [3] membahas pengamanan data teks dalam citra digital melalui teknik XOR, sementara penelitian [4] merancang perangkat lunak keamanan citra digital menggunakan algoritma Hill Cipher. Selain itu, penelitian [5] menghasilkan cipherimage yang menciptakan pola dari citra asli dengan penyebaran nilai piksel yang tidak merata, tetapi tetap dapat didekripsi tanpa mengubah nilai piksel aslinya.

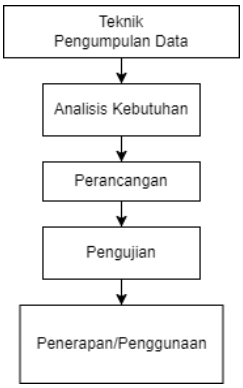
Berangkat dari paparan di atas, peneliti berinisiatif untuk membuat sebuah solusi dari permasalahan keamanan citra digital dengan menggunakan algoritma ChaCha20. Karena itu, dilakukanlah sebuah penelitian yang bertujuan untuk mengamankan citra bitmap berekstensi BMP dan PNG menggunakan algoritma cipher ChaCha20. Penelitian ini juga ditujukan untuk memahami alur implementasi serta cara kerja algoritma ini dalam mengamankan citra bitmap, termasuk merancang aplikasi pengamanan berbasis desktop. Di samping itu, penelitian ini mengevaluasi efektivitas ChaCha20 dalam menghadapi serangan Known-Plaintext Attack, untuk menilai sejauh mana algoritma ini dapat diandalkan dalam menjaga keamanan informasi.

2. Metode Penelitian

Bagian ini menjelaskan secara detail tahapan-tahapan yang digunakan dalam pelaksanaan penelitian. Metode yang digunakan dirancang untuk memberikan gambaran yang jelas tentang prosedur yang diterapkan, sehingga hasil penelitian dapat dinilai dan dipertanggungjawabkan.

2.1. Alur Penelitian

Metode penelitian yang digunakan dalam penelitian terdiri dari tahapan alur penelitian sebagaimana Gambar 1 berikut :



Gambar 1 Tahapan Alur Perencanaan.

Langkah awal dimulai dengan melakukan pengumpulan data, yakni teknik yang digunakan dalam mengumpulkan data penelitian, mulai dari studi literatur, referensi internet serta kamera digital untuk mengumpulkan citra sampel pengujian. Selanjutnya adalah tahap analisis kebutuhan, yang merupakan proses untuk memperoleh informasi dan bayangan dari suatu perangkat lunak yang akan dibangun agar sesuai dengan kebutuhan serta sebagai penentu batasan-batasan sistem yang akan dirancang. Pada tahap perancangan, semua data serta analisis yang telah dikumpulkan maka dibentuk konsep desain perancangan sistem pengamanan citra digital dengan menggunakan algoritma ChaCha20. Setelahnya, dilakukan tahapan pengujian terhadap sistem, proses uji coba bertujuan untuk mengevaluasi kinerja sistem yang telah dikembangkan, termasuk untuk menilai fungsi dari setiap proses, kecepatan, hasil dan termasuk uji keamanan sistem. Terakhir adalah penerapan, tahapan ini dilakukan dengan penggunaan sistem keamanan citra digital baik enkripsi maupun dekripsi.

2.2. Citra

Citra adalah representasi visual yang bisa berupa analog (kontinu) atau digital (diskrit). Citra digital, yang terdiri dari piksel dengan koordinat (x, y) dan intensitas warna f(x, y), termasuk dalam format bitmap [6]. Bitmap adalah citra yang dibentuk dari sejumlah piksel yang diletakkan pada lokasi tertentu dengan warna masing-masing, membentuk gambar. Kualitas gambar bitmap tergantung pada kerapatan piksel; semakin rapat piksel, semakin baik kualitas gambar. Namun, jika diperbesar, citra bitmap dapat terlihat pecah. Format bitmap yang umum meliputi

P-ISSN : 2089-3353
E-ISSN : 2808-9162

.bmp, .jpg, .png, dan .gif [7]. Namun fokus penelitian ini hanya pada bitmap BMP dan PNG. Dalam penelitian ini digunakan 20 sampel citra dari format BMP dan PNG. Untuk menjaga konsistensi, setiap citra sampel ditetapkan dengan standar tinggi resolusi 512 piksel. Dari total sampel yang digunakan, hanya dua sampel yang sudah memenuhi standar tersebut, sedangkan 18 sampel lainnya diselaraskan ke tinggi resolusi 512 piksel menggunakan OpenCV [8].

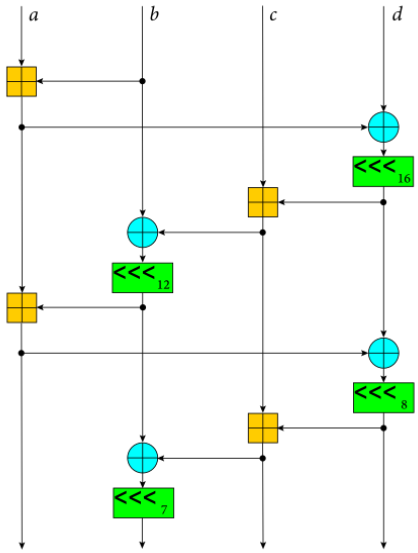
2.3. ChaCha20

ChaCha20 adalah algoritma kriptografi simetris berbasis stream cipher yang dikembangkan oleh Daniel J. Bernstein pada 2008, sebagai pengembangan dari Salsa20. Algoritma ChaCha20 menunjukkan fleksibilitas dan efektivitasnya dalam berbagai aplikasi keamanan, mulai dari sistem tertanam [9], keamanan jaringan sensor nirkabel [10] hingga pengamanan mesin virtual pusat data [11], termasuk penggunaannya dalam protokol SSL/TLS oleh perusahaan seperti Google dan Cloudflare [12]. Algoritma ini meningkatkan difusi per putaran dengan tetap mempertahankan kinerja tinggi. ChaCha20 memanfaatkan fungsi pseudorandom berbasis operasi Add-Rotate-XOR dan menggunakan struktur initial state berupa matriks 4x4 yang terdiri dari 16 word berukuran 32-bit (32-bit unsigned integers), dengan total ukuran 512-bit (64 byte), untuk menghasilkan keystream. Masukan utamanya mencakup kunci 256-bit, konstanta 128-bit, nonce 96-bit, dan counter 32-bit. Algoritma ini mendukung kunci 128-bit atau 256-bit, dengan konstanta "expand 32-byte k" dan nonce sebagai bilangan acak sekali pakai. Dalam ChaCha20, word disimpan dalam format little-endian. Adapun susunan matriks-nya disusun seperti pada Gambar 2 berikut :

Cons	Cons	Cons	Cons
Key	Key	Key	Key
Key	Key	Key	Key
Counter	Nonce	Nonce	Nonce

Gambar 2 Matriks Initial State ChaCha20 [13]

ChaCha20 memiliki sebuah fungsi inti bernama Quarter Round. Ini beroperasi berdasarkan prinsip ARX (Add-Rotate-XOR), yaitu operasi bitwise XOR (⊕), 32-bit penambahan modulus 2^32 ⊞, dan operasi rotasi jarak konstan ke kiri, yang dinyatakan sebagai "<<< n," di mana n merepresentasikan jumlah bit yang diputar ke kiri (menuju bit bernilai tinggi).



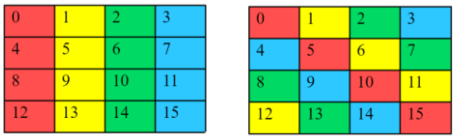
Gambar 3 Fungsi Quarter-Round ChaCha20 [13]

Dari Gambar 3 diatas, fungsi Quarter Round dapat didefinisikan sebagai QR(a, b, c, d), dimana :

$$a \mathrel{+}= b;\ d \mathrel{\hat{}}= a;\ d \lll= 16; \quad (1)$$
$$c \mathrel{+}= d;\ b \mathrel{\hat{}}= c;\ b \lll= 12; \quad (2)$$
$$a \mathrel{+}= b;\ d \mathrel{\hat{}}= a;\ d \lll= 8; \quad (3)$$
$$c \mathrel{+}= d;\ b \mathrel{\hat{}}= c;\ b \lll= 7; \quad (4)$$

dengan a, b, c, d mewakili posisi elemen dalam matriks (word) pada baris dan kolom tertentu. Setiap nilai pada matriks di urutan tertentu per Quarter Round akan dilakukan operasi penambahan (+), XOR (^) dan rotasi jarak konstan (<<<).

ChaCha20 menggunakan fungsi Quarter Round untuk mengacak masukan Initial State menjadi keystream. Prosesnya dimulai dengan mengubah masukan pada initial state ke format little-endian sebelum dikirim ke generator pseudo-random melalui fungsi Quarter Round. Blok matriks initial state 512-bit akan diacak melalui 80 kali aplikasi fungsi Quarter Round. Dalam setiap Quarter Round, 4 word 32-bit (128-bit) diacak, dan proses ini diulang hingga seluruh blok teracak. ChaCha20 menjalankan 20 putaran, yang terdiri dari 10 double round. Seperti terlihat pada Gambar 4, setiap double round mencakup 2 putaran: putaran ganjil yang terdiri dari 4 quarter round kolom dan putaran genap dengan 4 quarter round diagonal. ChaCha20 melakukan 10 double round, setara dengan 20 putaran atau 80 aplikasi quarter round [1].



(a). Satu round ganjil (kolom) (b). Satu round genap (diagonal)

Gambar 4 Double Round [1]

P-ISSN : 2089-3353
E-ISSN : 2808-9162

Hasil putaran terakhir dijumlahkan dengan initial state asli dalam format little-endian melalui penambahan. Pada tahap akhir, hasil penjumlahan disimpan kembali dalam format little-endian sebagai keystream yang siap digunakan. Mirip Salsa20 [14], setiap blok keystream ChaCha20 bersifat independen, memungkinkan akses acak dan komputasi paralel untuk beberapa blok secara simultan. Jika plaintext panjangnya melebihi 512-bit (64-byte), ChaCha20 menghasilkan keystream tambahan dengan meng-increment nilai block counter untuk setiap blok yang diperlukan. Setiap blok keystream di-XOR dengan bagian plaintext yang sesuai untuk menghasilkan ciphertext. ChaCha20 menggunakan keystream untuk mengenkripsi dan mendekripsi data. Dalam enkripsi, plaintext di-XOR dengan keystream yang dihasilkan dari fungsi pseudorandom berbasis kunci dan nonce, sementara dalam dekripsi, ciphertext di-XOR dengan keystream yang sama untuk memperoleh plaintext. Hubungan ini dijelaskan dengan persamaan 5 dan 6 dibawah ini [1].

$$ciphertext = plaintext \oplus keystream\,(key, nonce) \quad (5)$$

$$plaintext = ciphertext \oplus keystream\,(key, nonce) \quad (6)$$

2.4. Perancangan Kebutuhan Sistem

Penelitian ini memerlukan dua jenis kebutuhan sistem: fungsional dan non-fungsional. Kebutuhan fungsional mencakup proses dan fitur yang harus ada dalam sistem, sedangkan kebutuhan non-fungsional mencakup spesifikasi komponen hardware (laptop) dan perangkat lunak yang digunakan selama penelitian. Kebutuhan fungsional utama untuk aplikasi pengamanan citra dengan algoritma ChaCha20 dijelaskan dalam Tabel 1 berikut.

Tabel 1 Kebutuhan Fungsional

No	ID	Kebutuhan Fungsional
1.		Enkripsi
	KF1	Perangkat lunak harus menyediakan fungsionalitas untuk mengenkripsi file gambar menggunakan algoritma enkripsi ChaCha20.
	KF2	Pengguna harus dapat memilih dan menyimpan file gambar yang telah dienkripsi ke penyimpanan lokal.
2.		Dekripsi
	KF3	Perangkat lunak harus memungkinkan pengguna untuk mendekripsi file gambar yang telah dienkripsi sebelumnya.
	KF4	Pengguna harus dapat memilih dan menyimpan file gambar yang telah didekripsi ke penyimpanan lokal.
3.		Kompabilitas Gambar
	KF5	Perangkat lunak harus mendukung format gambar seperti PNG dan BMP.
	KF6	Harus dipastikan bahwa proses enkripsi dan dekripsi tidak merusak isi citra gambar.

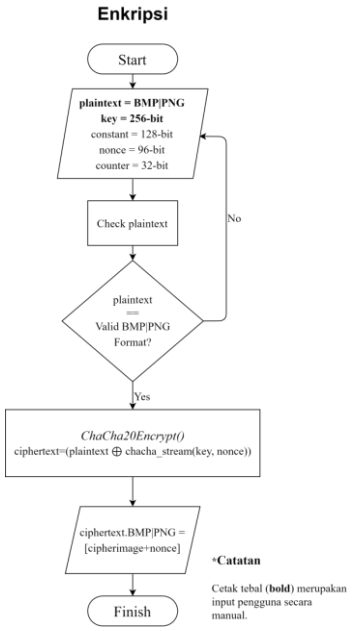
Kebutuhan non-fungsional mencakup spesifikasi terkait kompatibilitas perangkat keras dan perangkat lunak yang digunakan untuk pengujian, sebagaimana yang tampak pada Tabel 2.

Tabel 2 Kebutuhan non-fungsional

No	Hardware & Software	Rincian
1.	Perangkat Keras (Laptop)	
	Prosesor	Intel® Core™ i3-3217U CPU @ 1.80GHz
	Kartu Grafis	Intel HD 4000
	Memori	10GB RAM DDR3L
	Media Penyimpanan	1TB SSD + 500GB HDD
2.	Perangkat Lunak	
	Windows 10 64-bit	Sistem Operasi
	Visual Studio Code	Editor kode
	Python versi 3.9.6	Bahasa pemrograman
	Microsoft Office Word 2007	Pengolah kata
	Hex Editor Neo & HxD	Penampil binari berkas

2.5. Perancangan Sistem Enkripsi dan Dekripsi

Bagian ini menjelaskan perancangan sistem untuk enkripsi dan dekripsi citra digital. Penekanan utamanya adalah pada desain diagram alir yang menggambarkan proses enkripsi dan dekripsi data citra secara akurat dan efisien, sesuai dengan masukan algoritma ChaCha20.

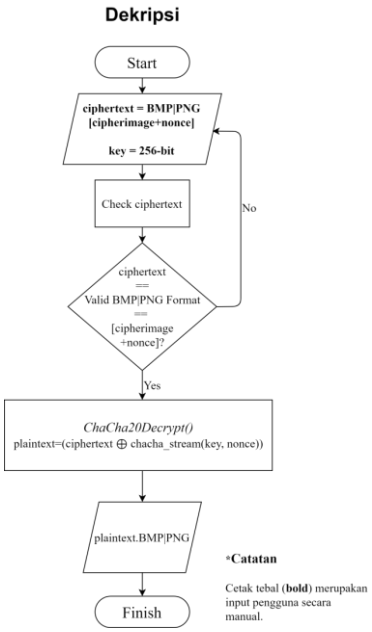


Gambar 5 Diagram Alir Sistem Enkripsi

Diagram alir enkripsi Gambar 5 dapat dijelaskan sebagai berikut: Proses dimulai dengan dimasukkannya plaintext (citra .bmp atau .png) dan key (kata sandi) oleh pengguna, sementara konstanta, nonce, dan counter dimasukkan otomatis oleh sistem. Sistem kemudian memeriksa dan memvalidasi format citra plaintext; jika format tidak valid, proses kembali ke input, tetapi jika valid, sistem melanjutkan ke tahap enkripsi menggunakan algoritma ChaCha20, di mana

P-ISSN : 2089-3353
E-ISSN : 2808-9162

sistem membaca nilai piksel dari citra plaintext untuk kemudian di-XOR dengan chacha_stream yang dihasilkan dari key dan nonce. Hasil enkripsi adalah ciphertext dalam format .bmp atau .png yang mencakup cipherimage dan nonce yang ditambahkan di akhir berkas. Setelah ini, proses enkripsi selesai.

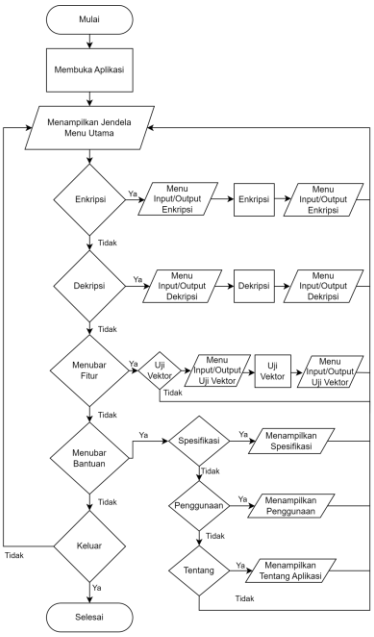


Gambar 6 Diagram Alir Sistem Dekripsi

Diagram alir proses dekripsi pada Gambar 6 dimulai dengan memberikan masukan berupa ciphertext/cipherimage (citra .bmp atau .png) dan key (kata sandi) oleh pengguna. Selanjutnya, masukan ciphertext diperiksa untuk memvalidasi apakah valid dalam format .bmp atau .png serta memastikan bahwa berkas tersebut terdiri dari cipherimage dan nonce. Jika sesuai, proses dekripsi dimulai dengan sistem membaca nilai piksel dari citra terenkripsi (ciphertext), lalu di-XOR menggunakan chacha_stream yang dihasilkan key yang dimasukkan pengguna serta nonce yang diambil dari bagian akhir berkas ciphertext. Dari operasi tersebut, dihasilkan plaintext dengan format .bmp atau .png.

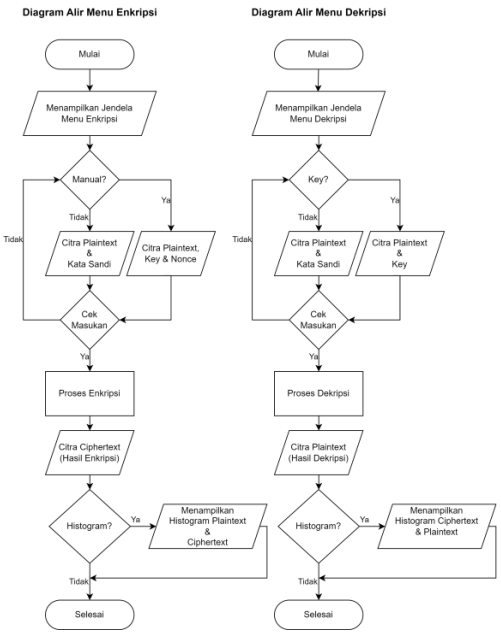
2.6. Perancangan Diagram Alir Antarmuka

Berbeda dengan sebelumnya, perancangan diagram alir antarmuka program fokus pada alur antarmuka program dari sisi perspektif pengguna. Dimulai ketika pengguna menjalankan program utama, yang kemudian menampilkan menu utama dashboard dengan pilihan enkripsi, dekripsi, uji vektor, dan sub-menu bantuan (spesifikasi, penggunaan, dan tentang). Menu enkripsi digunakan untuk mengamankan citra, sementara dekripsi memulihkan citra asli. Uji vektor membantu dalam pengujian algoritma ChaCha20. Sub-menu lainnya menampilkan informasi tentang spesifikasi dan penggunaan aplikasi. Alur lengkapnya digambarkan dalam diagram alir skematis Gambar 7.



Gambar 7 Diagram Alir Dashboard Menu Utama

Proses enkripsi dimulai saat pengguna memilih menu enkripsi, menampilkan jendela dengan opsi manual. Jika dicentang, pengguna memasukkan citra plaintext, key, dan nonce; jika tidak, hanya citra plaintext dan kata sandi. Setelah validasi, sistem mengenkripsi plaintext menjadi ciphertext, atau kembali ke langkah sebelumnya jika tidak valid. Dekripsi dimulai dengan memilih menu dekripsi. Jika key dicentang, masukannya adalah citra ciphertext dan key; jika tidak, citra plaintext dan kata sandi. Setelah validasi, citra plaintext dipulihkan. Kedua alur digambarkan pada diagram alir Gambar 8.



Gambar 8 Diagram Alir Menu Enkripsi dan Dekripsi

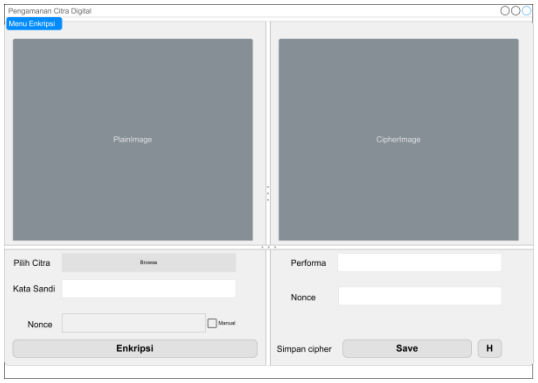
P-ISSN : 2089-3353
E-ISSN : 2808-9162

2.7. Perancangan Antarmuka Sistem

Perancangan antarmuka sistem bertujuan untuk memudahkan pengguna dalam menjalankan proses enkripsi dan dekripsi. Antarmuka sistem dirancang agar intuitif dan sesuai dengan kebutuhan fungsional. Menu utama merupakan jendela pertama yang muncul saat program dijalankan, dengan dua tombol pilihan, yaitu Enkripsi dan Dekripsi. Rancangan antarmukanya tampak pada Gambar 9 berikut.

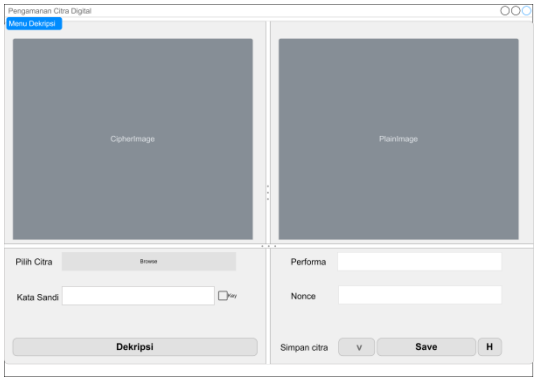
Gambar 9 Rancangan Antarmuka Menu Utama

Saat tombol enkripsi pada menu utama ditekan, jendela antarmuka baru akan muncul untuk memulai proses enkripsi. Desain antarmukanya dapat dilihat pada Gambar 10 berikut.



Gambar 10 Rancangan Antarmuka Enkripsi

Sementara itu, antarmuka dekripsi akan muncul jika tombol dekripsi dipilih dari menu utama, menampilkan jendela yang memungkinkan pengguna menjalankan proses dekripsi. Rancangan antarmuka seperti Gambar 11 berikut.

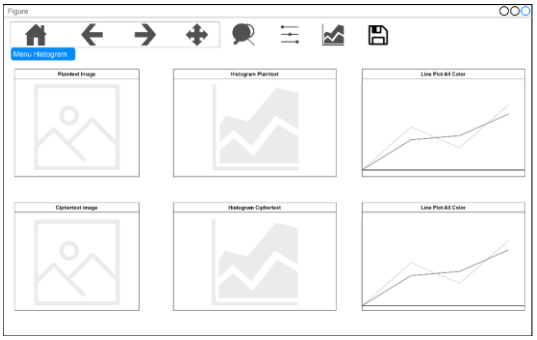


Gambar 11 Rancangan Antarmuka Dekripsi

Menu tes vektor menyediakan antarmuka untuk memasukkan beberapa input guna melakukan pengujian vektor. Menu ini diakses melalui sub-menu fitur pada menu bar. Rancangan antarmukanya sesuai Gambar 12 dibawah ini.

Gambar 12 Rancangan Antarmuka Uji Vektor

Menu histogram adalah jendela terpisah yang menampilkan histogram dari plaintext dan ciphertext. Untuk rancangan antarmukanya, perhatikan Gambar 13 berikut.



Gambar 13 Rancangan Antarmuka Uji Histogram

3. Hasil dan Pembahasan

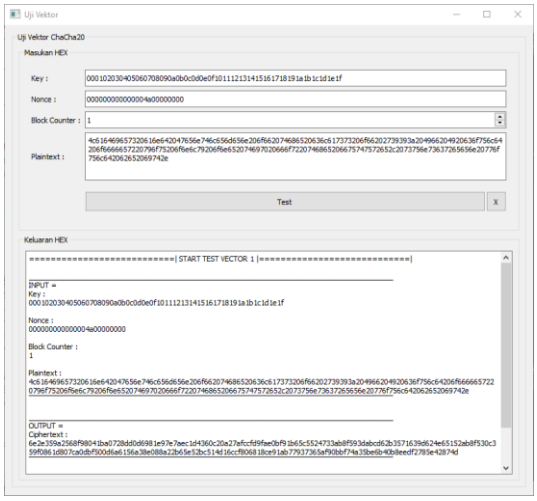
Pada bagian ini disajikan hasil implementasi serta pengujian sistem aplikasi enkripsi dan dekripsi citra digital menggunakan algoritma ChaCha20. Selain itu, dilakukan analisis terhadap kinerja sistem berdasarkan uji vektor dan evaluasi hasil enkripsi serta dekripsi, guna mengukur efektivitas dan ketepatan proses yang diusulkan. Aplikasi atau program yang dihasilkan dari penelitian ini berbasis desktop Windows 10 yang ditulis menggunakan bahasa pemrograman Python menggunakan pustaka antarmuka PyQt5 [15]. Python adalah bahasa pemrograman interpreted, berorientasi objek, dan tingkat tinggi, diciptakan oleh Guido van Rossum pada 1991. Populer karena kemudahannya, Python digunakan dalam ilmu data, pembelajaran mesin, dan AI [16].

3.1. Uji vektor

Uji vektor adalah metode pengujian yang melibatkan sejumlah masukan untuk menguji sistem, dan merupakan bagian dari pengujian perangkat lunak untuk memverifikasi kesesuaian fungsionalitas dengan

P-ISSN : 2089-3353
E-ISSN : 2808-9162

spesifikasi yang ditetapkan. Dalam konteks algoritma ChaCha20, masukan uji vektor berdasarkan RFC8439 [1], seperti yang ditunjukkan pada Gambar 14 yang menggambarkan hasil uji vektor valid.



Gambar 14 Hasil uji vektor

3.2. Uji Black Box

Tujuan dari uji coba sistem berdasarkan fungsionalitas utama adalah untuk memastikan bahwa sistem berfungsi dengan baik dan menghasilkan keluaran yang diharapkan. Uji coba ini dilakukan dengan metode black box testing, yang fokus pada pengujian fungsionalitas tanpa mempertimbangkan struktur internal sistem. Tabel 1 Tabel 3 berikut adalah hasil uji black box yang telah dilakukan.

Tabel 3 Hasil Uji Black Box

ID	Kebutuhan Fungsional	Langkah Pengujian	Hasil
KF1	Perangkat lunak harus mendukung enkripsi file citra dengan ChaCha20.	Buka aplikasi; Pilih menu enkripsi; Pilih file dan masukkan key; Enkripsi file; Verifikasi	Berhasil
KF2	Pengguna harus dapat memilih dan menyimpan file terenkripsi ke penyimpanan.	Buka aplikasi; Pilih menu enkripsi; Verifikasi pemilihan file lokal	Berhasil
KF3	Perangkat lunak harus mendukung dekripsi file citra	Buka aplikasi; Pilih menu dekripsi; Pilih file dan masukkan key; Dekripsi file; Verifikasi.	Berhasil
KF4	Pengguna harus dapat menyimpan file dekripsi ke penyimpanan lokal	Buka aplikasi; Pilih menu dekripsi; Pilih file terdekripsi; Simpan file; Verifikasi.	Berhasil
KF5	Perangkat lunak harus mendukung format PNG dan BMP	Buka aplikasi; Enkripsi dan dekripsi format PNG dan BMP; Verifikasi dukungan format.	Berhasil
KF6	Proses enkripsi dan dekripsi tidak merusak isi citra	Buka aplikasi; Lakukan enkripsi dan dekripsi; Verifikasi tidak ada kerusakan.	Berhasil

3.3. Penerapan

Pada tahap ini, akan dijelaskan implementasi dan perhitungan singkat algoritma ChaCha20 dalam proses enkripsi dan dekripsi. Hasil implementasi akan ditampilkan sesuai dengan rancangan yang telah dibahas pada bab sebelumnya. Penelitian ini hanya melibatkan 20 sampel citra, yang terdiri dari 10 citra berformat BMP dan 10 citra berformat PNG, dengan variasi ukuran berkas dan tipe saluran (channel). Sebagai perwakilan, citra plaintext yang digunakan untuk pengujian adalah Gambar 15, yaitu citra berformat PNG dengan 4 channel, berukuran 512 piksel untuk tinggi dan 910 piksel untuk lebar, yang menampilkan gedung Universitas Islam Negeri Sumatera Utara.

Gambar 15 Citra Plaintext

Untuk keperluan pengujian, dari citra plaintext diatas kemudian dibaca nilai RGBA sepanjang 32 piksel pertama (128 bytes) seperti Tabel 4 berikut.

Tabel 4 Nilai Heksadesimal Plaintext

Plaintext	Nilai	Panjang
gedung_ uinsu.png	896937ff896937ff876735ff886836ff856533ff866735ff866734ff856632ff886834ff886834ff886834ff886834ff886834ff886834ff866632ff886a32ff896a34ff896a34ff896a34ff896a34ff896a34ff896a34ff896a34ff896a34ff896a34ff896a34ff896a34ff896a34ff896a32ff886a36ff8a6839ff	128 byte (1024 bit)

Nilai masukan ChaCha20 yang digunakan untuk melakukan proses pengamanan Gambar 15 dapat dilihat pada Tabel 5.

Tabel 5 Nilai Masukan

Masukan	Masukan Initial State ChaCha20 String	Heksadesimal	Panjang
Constant	expand 32-byte k	657870616e6420333 22d62797465206b	128 bit
Key	256bitadal ahpanjang kuncichacha20	32353662697461646 16c616870616e6a61 6e676b756e6369636 8616368613230	256 bit
B. Counter		0	32 bit
Nonce	96-bit-nonce	39362d6269742d6e6 f6e6365	96 bit

P-ISSN : 2089-3353
E-ISSN : 2808-9162

Masukan Initial state dimasukkan dalam format little-endian, kemudian diproses melalui fungsi QuarterRound sebanyak 80 kali dan dijumlahkan dengan initial state asli untuk menghasilkan keystream, seperti keystream dua blok yang ditunjukkan Tabel 6 ini.

Tabel 6 Dua Blok Keystream

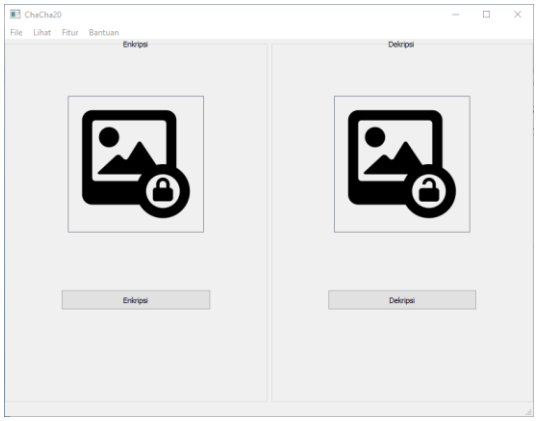
Keystream	Panjang
6ab59bf02dbf23518b8330f950031b2e97 ff2c19d8514c799cbcdf125b2bc243d961 3a53b966caea3986d35ac8ccbb538ec5ca 607b04718702cac3a14c2d111d	64 byte (512 bit)
2f8f9684511181d16c2a51c34ffac9ca5f4 93a75218f7ea250e284455437c3f2659e2 df7196def027f2385ddd73f46667a2bd8da f18ee5a82f9ea8dc64ae37a2	64 byte (512 bit)

Keystream kemudian di XOR dengan plaintext Tabel 4, hasilnya adalah ciphertext seperti Tabel 7.

Tabel 7 Dua Blok Ciphertext

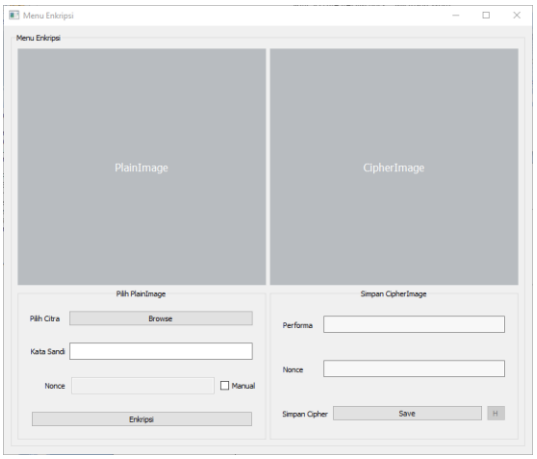
Ciphertext	Panjang
e3dcac0fa4d614ae0ce40506d86b2dd112 9a1fe65e3679861adbebedde4df0bc5109 0eac310efe15b1eee7a540a48fac06adfe9f f36c457884acf15ec44723e2	64 byte (512 bit)
a6e5a27bd87bb52ee540653cc690fd35d6 230e8aa8e54a5dd988b0badd5df70decf4 19089007dbfdf649b1225e557299f341ec 2578e4d757a7f49e23eec60e5d	64 byte (512 bit)

Untuk memulihkan plaintext, cukup dilakukan operasi kebalikan, yaitu dengan melakukan XOR antara ciphertext dan keystream. Selanjutnya, penerapan antarmuka dapat dilihat pada Gambar 16 yang menampilkan dasbor menu utama aplikasi saat pertama kali dijalankan. Dasbor ini menjadi dasar bagi pengguna untuk mengakses fitur enkripsi dan dekripsi.



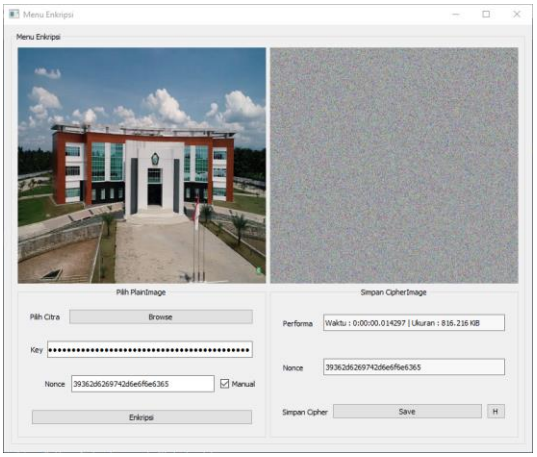
Gambar 16 Antarmuka Dasbor Menu Utama

Gambar 17 menunjukkan implementasi antarmuka pada menu enkripsi, yang berfungsi untuk mengamankan citra melalui proses enkripsi.



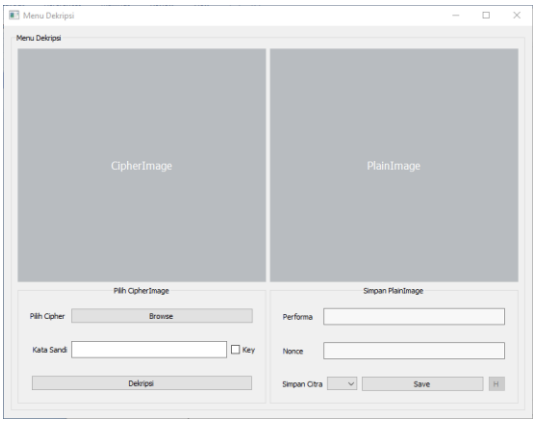
Gambar 17 Antarmuka Menu Enkripsi

Untuk melakukan proses enkripsi, maka dimasukkan citra plaintext dan nilai masukkan Tabel 5 secara manual (key & nonce), maka hasilnya adalah seperti Gambar 18.



Gambar 18 Antarmuka Hasil Enkripsi

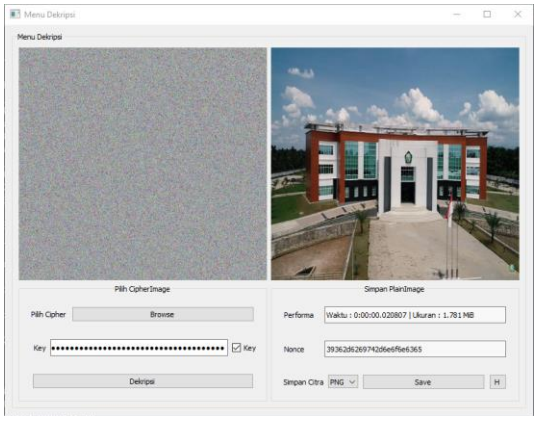
Selanjutnya, hasil implementasi antarmuka menu dekripsi dapat dilihat pada Gambar 19, yang berperan dalam memulihkan citra asli setelah melalui proses enkripsi.



Gambar 19 Antarmuka Menu Dekripsi

P-ISSN : 2089-3353
E-ISSN : 2808-9162

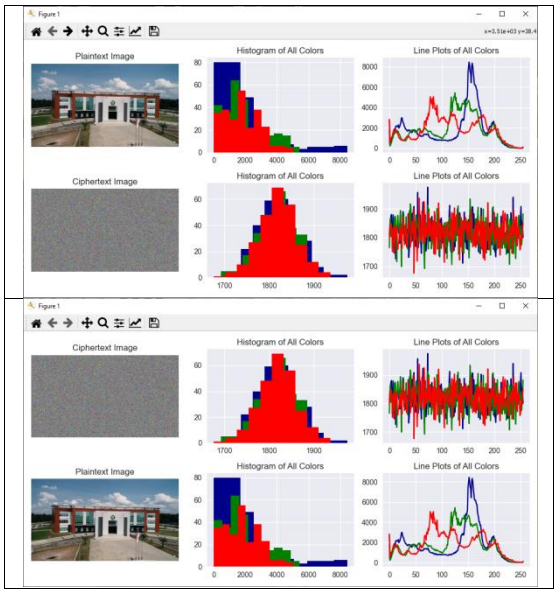
Proses dekripsi dilakukan dengan memasukkan citra ciphertext dan nilai key yang sesuai. Hasilnya seperti Gambar 20 dibawah ini.



Gambar 20 Antarmuka Hasil Dekripsi

3.4. Perbandingan Hasil

Pada bagian ini, dilakukan perbandingan antara citra asli, citra hasil enkripsi, dan citra hasil dekripsi yang telah diproses menggunakan algoritma ChaCha20. Perbandingan ini meliputi ukuran file, visual citra, histogram, waktu proses serta kesesuaian hasil dekripsi dengan citra asli. Tujuan dari perbandingan ini adalah untuk mengukur efektivitas algoritma dalam menjaga integritas data citra selama proses enkripsi dan dekripsi. Evaluasi dilakukan dengan menghitung selisih ukuran file antara citra asli dan ciphertext, serta mengidentifikasi apakah terdapat distorsi atau perubahan pada citra hasil dekripsi dibandingkan dengan citra asli. Perbandingan visual antara citra plaintext, ciphertext dan hasil dekripsi dapat diperhatikan pada Gambar 21.



Gambar 21 Antarmuka Hasil Uji Histogram Enkripsi dan Dekripsi

Dari hasil di atas, terlihat adanya perbedaan visual yang signifikan antara plaintext dan ciphertext, serta antara ciphertext dan plaintext hasil dekripsi. Uji histogram menunjukkan perbedaan signifikan; ciphertext memiliki distribusi intensitas warna yang berbeda secara visual dari plaintext, dengan pengacakan warna dan perubahan intensitas yang tidak berpola. Hal ini mengindikasikan keberhasilan enkripsi dalam menyamarkan informasi visual dan distribusi warna citra asli.

Selain perbedaan visual dan distribusi warna, perubahan ukuran berkas juga menjadi salah satu indikator penting dalam evaluasi performa enkripsi. Hasil uji perbandingan ukuran berkas citra sebelum dan sesudah proses enkripsi dan dekripsi ditunjukkan pada Tabel 8, di mana kolom p = plaintext, c = ciphertext, Δpc = selisih p ke c, d = citra hasil dekripsi, dan Δpd = selisih persentase dari p ke d.

Tabel 8 Hasil Perbandingan Ukuran Berkas Citra sebelum dan setelah Enkripsi dan Dekripsi

No	Tipe	p (KB)	c (KB)	Δpc (%)	d (KB)	Δpd (%)
1	PNG	817	1867	129	817	0
2	PNG	44,4	1245	2.704	44,4	0
3	PNG	393	769	96	393	0
4	PNG	198	346	75	198	0
5	PNG	173	385	123	173	0
6	PNG	670	1576	135	712	6
7	PNG	563	1182	110	680	21
8	PNG	754	1182	57	754	0
9	PNG	326	1401	330	326	0
10	PNG	287	1955	581	287	0
11	BMP	768	768	0	768	0
12	BMP	768	768	0	768	0
13	BMP	768	768	0	768	0
14	BMP	257	257	0	257	0
15	BMP	457	457	0	457	0
16	BMP	768	768	0	768	0
17	BMP	1179	1179	0	1179	0
18	BMP	768	768	0	768	0
19	BMP	768	768	0	768	0
20	BMP	1167	1167	0	1167	0

Tabel perbandingan menunjukkan bahwa ukuran berkas ciphertext PNG meningkat sebesar 57 hingga 2.704 persen dibandingkan plaintext, sementara format BMP tidak mengalami perubahan ukuran. Setiap berkas ciphertext sebenarnya mengalami peningkatan sebesar 12 bytes untuk menampung nonce, meskipun peningkatan ini kurang terlihat pada tabel karena satuan yang digunakan adalah kilobytes. Peningkatan signifikan pada sampel ciphertext PNG disebabkan oleh proses enkripsi yang menghasilkan data piksel acak dan tidak berulang, sehingga kompresi lossless yang digunakan pada format PNG menjadi tidak efektif. Selain itu, peneliti mencatat bahwa dari 20 citra hasil dekripsi, 18 di antaranya tidak mengalami perubahan ukuran maupun pergeseran warna piksel dibandingkan dengan citra plaintext asli. Namun, terdapat dua citra hasil dekripsi yang mengalami peningkatan ukuran dan perbedaan warna piksel. Perubahan ini disebabkan karena kedua

P-ISSN : 2089-3353
E-ISSN : 2808-9162

sampel tersebut yang tidak melalui proses standarisasi resolusi menggunakan OpenCV, sehingga terdapat perbedaan interpretasi nilai piksel yang dihasilkan oleh codec lain saat dibaca oleh program OpenCV. Meskipun demikian, perbedaan warna ini tidak begitu terlihat oleh mata manusia.

Untuk menilai kinerja sistem lebih lanjut, dilakukan pengujian kecepatan enkripsi dan dekripsi, di mana setiap proses diuji sepuluh kali per sampel untuk memperoleh rata-rata performa. Tabel 9 menunjukkan hasil, e = enkripsi, d = dekripsi, dan Δ = selisih waktu antara waktu enkripsi (e) dan waktu dekripsi (d) di mana $\Delta = d - e$. Jika Δ bernilai positif, waktu dekripsi lebih lama daripada waktu enkripsi, sedangkan jika Δ bernilai negatif, waktu dekripsi lebih cepat dibandingkan waktu enkripsi.

Tabel 9 Hasil Uji Rata-Rata Waktu Enkripsi Dan Dekripsi Citra

No	ChaCha20 Waktu ($\bar{x}$)			AES-CBC Waktu ($\bar{x}$)		
	e (ms)	d (ms)	Δ (%)	e (ms)	d (ms)	Δ (%)
1	13.39	13.49	0.76	27.22	27.07	-0.56
2	9.14	8.95	-2.12	18.53	18.02	-2.76
3	6.28	6.06	-3.46	12.24	12.25	0.09
4	2.57	2.53	-1.71	5.24	5.02	-4.30
5	3.22	3.27	1.52	5.89	6.22	5.67
6	11.39	11.52	1.09	23.23	22.75	-2.06
7	8.75	8.79	0.50	17.62	17.25	-2.10
8	8.62	8.53	-1.06	17.40	17.36	-0.23
9	10.23	10.10	-1.24	20.58	20.33	-1.21
10	14.09	14.17	0.60	28.42	28.35	-0.26
11	6.32	6.26	-0.93	12.30	12.05	-2.08
12	6.34	6.45	1.71	11.93	12.13	1.66
13	6.22	6.40	2.85	12.13	11.89	-1.99
14	2.02	1.95	-3.10	4.11	3.94	-3.99
15	3.88	3.81	-1.75	7.33	7.13	-2.66
16	6.28	6.11	-2.84	12.08	11.91	-1.41
17	8.54	8.66	1.40	17.64	17.26	-2.11
18	6.23	6.28	0.70	12.24	11.97	-2.27
19	6.44	6.29	-2.30	12.16	11.80	-2.99
20	8.71	8.41	-3.47	17.90	17.28	-3.44

Tabel di atas menyajikan data mengenai waktu eksekusi enkripsi dan dekripsi yang dilakukan oleh algoritma ChaCha20 dan AES-CBC. Dari total 20 sampel yang diuji, algoritma ChaCha20 menunjukkan kecepatan enkripsi yang lebih baik pada 9 sampel, dengan rentang peningkatan waktu eksekusi antara 0,50 persen hingga 2,85 persen. Sebaliknya, 11 sampel menunjukkan hasil dekripsi yang lebih cepat, dengan penurunan waktu berkisar antara -0,93 persen hingga -3,47 persen. Pada algoritma AES-CBC, terdapat 3 sampel yang menunjukkan kecepatan enkripsi yang lebih baik, dengan rentang waktu eksekusi antara 0,09 persen hingga 5,67 persen, sementara 17 sampel lainnya memiliki kecepatan dekripsi yang lebih cepat, ditunjukkan oleh nilai negatif antara -0,23 persen hingga -4,3 persen. Fluktuasi waktu antara enkripsi dan dekripsi pada ChaCha20 relatif kecil, mengindikasikan stabilitas kinerja. Secara keseluruhan, ChaCha20 terbukti sekitar dua kali lebih cepat dibandingkan AES-CBC. Selain itu, ukuran berkas memiliki pengaruh yang lebih signifikan terhadap waktu pemrosesan dibandingkan format berkas citranya.

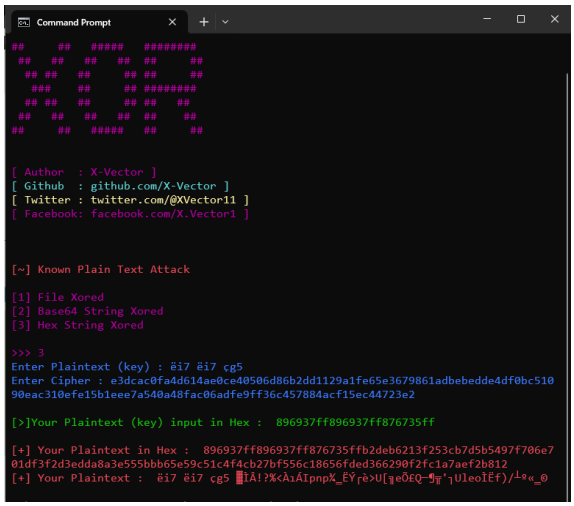
3.5. Uji Keamanan

Uji keamanan bertujuan untuk memastikan bahwa sistem yang dibangun aman dari potensi ancaman. Metode pengujian yang digunakan adalah model penyerangan Known-Plaintext Attack [17]. Dalam metode ini, bagian plaintext dan ciphertext yang sudah diketahui dimasukkan untuk mencoba mengungkap informasi rahasia, seperti kunci (keystream) yang digunakan saat mengamankan ciphertext. Pengujian ini menggunakan sampel sebagian plaintext dan keseluruhan ciphertext, seperti yang ditunjukkan pada Tabel 10 berikut:

Tabel 10 Skenario Serangan

Jenis	No	Nilai	Panjang (bytes)	Total (bytes)
Plain text	1	896937ff896937ff876735ff	12	64
	2	886836ff856533ff866735ff 866734ff856632ff886834ff 886834ff886834ff886834ff	36	
	3	886834ff886834ff866632ff 886a32ff	16	
Ciper text	1	e3dcac0fa4d614ae0ce4050 6d86b2dd1129a1fe65e367	64	64
	2	9861adbebedde4df0bc5109 0eac310efe15b1eee7a540a		
	3	48fac06adfe9ff36c457884a cf15ec44723e2		

Dari skenario diatas, maka dilakukan pengujian dengan hasil seperti Gambar 22 berikut.

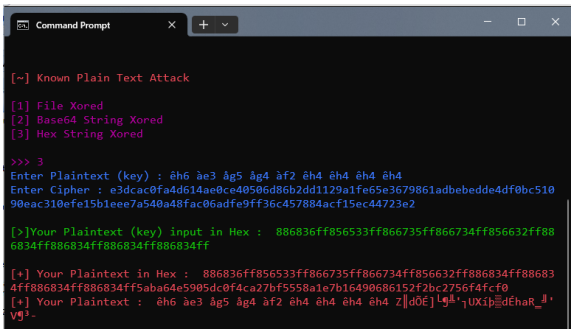


Gambar 22 Skenario Known-Plaintext Attack Pertama

Gambar pengujian diatas memperlihatkan serangan Known Plaintext Attack dengan memasukkan sebagian plaintext (karakter) maka dihasilkan plaintext hasil yang tidak sesuai dengan plaintext aslinya. Hal ini dapat terlihat dengan jelas pada bagian

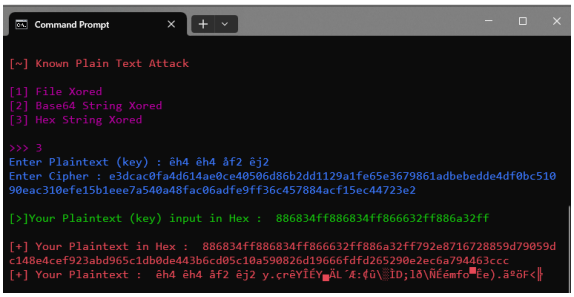
P-ISSN : 2089-3353
E-ISSN : 2808-9162

"Your Plaintext in Hex", yang mengeluarkan nilai Heksadesimal yang tidak sesuai dengan plaintext asli sebagaimana pada tabel skenario sebelumnya.



Gambar 23 Skenario Known-Plaintext Attack Kedua

Pada Gambar 23, meskipun panjang plaintext yang diketahui telah bertambah, namun plaintext yang dihasilkan sama sekali tidak sesuai dengan plaintext aslinya.



Gambar 24 Skenario Known-Plaintext Attack Ketiga

Pengujian pada skenario terakhir pada Gambar 24 menunjukkan bahwa hasil akhir plaintext tidak sama dengan plaintext aslinya. Berdasarkan ketiga skenario penyerangan Known-Plaintext Attack, dapat disimpulkan bahwa sistem pengamanan data citra ini berhasil melindungi data dari metode serangan tersebut.

4. Kesimpulan

Berdasarkan hasil penelitian, disimpulkan bahwa algoritma ChaCha20 berhasil diimplementasikan secara efektif untuk mengamankan citra digital dalam format BMP dan PNG melalui aplikasi desktop yang sederhana dan fungsional. Uji visual, termasuk perbandingan antara plaintext dan ciphertext serta analisis histogram, menunjukkan bahwa sistem menghasilkan ciphertext yang secara signifikan berbeda dari plaintext, baik dalam distribusi warna maupun intensitas piksel, yang menandakan keberhasilan algoritma dalam menyamarkan informasi visual. Selain itu, sistem ini terbukti tahan terhadap serangan Known-Plaintext Attack (KPA), yang diuji dalam tiga skenario berbeda. Setiap ciphertext mengalami penambahan ukuran sebesar 12 byte untuk menampung nilai nonce pada akhir berkas, dengan format PNG menunjukkan peningkatan ukuran antara 57 hingga 2.704 persen dibandingkan plaintext, sedangkan ukuran berkas BMP tetap stabil. Hasil dekripsi identik dengan plaintext asli pada 18 citra sampel yang distandarisasi pada resolusi 512 piksel menggunakan codec OpenCV. Dua sampel lainnya, dengan resolusi serupa, menunjukkan perubahan ukuran dan warna piksel yang tidak kasat mata akibat perbedaan codec. Variabilitas ini perlu diperhatikan sebagai faktor yang memengaruhi hasil akhir, meskipun untuk mayoritas sampel yang diproses dengan metode standar, kesimpulan penelitian tetap valid. Analisis kinerja enkripsi dan dekripsi menunjukkan bahwa ChaCha20 terbukti lebih cepat hingga dua kali dibandingkan AES-CBC. Perbandingan waktu antara hasil uji enkripsi dan dekripsi pada ChaCha20 menunjukkan perbedaan yang relatif kecil antar sampel, mengindikasikan performa yang stabil untuk kedua proses tersebut. Ukuran file lebih memengaruhi waktu pemrosesan dibandingkan format file. Penelitian ini juga memberikan wawasan mendalam mengenai cara kerja algoritma ChaCha20, terutama dalam pengacakan data masukan melalui quarter-round sebanyak 80 kali sebelum menghasilkan keystream yang di-XOR dengan bit plaintext.

Penelitian ini memiliki keterbatasan, termasuk penggunaan CPU Intel® Core™ i3-3217U @ 1.80GHz yang tidak mendukung AES-NI, sehingga perbandingan antara algoritma ChaCha20 dan AES tetap relevan. Sistem yang digunakan terdiri dari 1TB SSD, 500GB HDD, dan RAM 10GB DDR3L, dengan program pengujian yang ditulis dalam Python 3.9.6 dan dijalankan pada Windows 10 64-bit. Keterbatasan ini perlu dicatat karena berpotensi memengaruhi hasil, khususnya terkait analisis performa dan kecepatan pemrosesan algoritma enkripsi dan dekripsi. Untuk penelitian selanjutnya, peneliti menyarankan penerapan algoritma ChaCha20 pada format file citra lain yang banyak digunakan, seperti JPG dan GIF, untuk mengeksplorasi kinerja dan keamanan sistem enkripsi dan dekripsi. Penerapan ini diharapkan dapat memperluas cakupan penelitian dan memberikan pemahaman yang lebih mendalam tentang efektivitas algoritma ChaCha20. Analisis yang lebih mendetail terhadap efisiensi dan kecepatan pemrosesan pada format-format tersebut dapat memberikan wawasan berharga bagi pengembangan sistem enkripsi yang lebih optimal di masa depan.

Daftar Rujukan

[1] Langley, A., Nir, Y. 2018. *ChaCha20 and Poly1305 for IETF Protocols*. [Online] (Updated 1 June 2018) Tersedia di : https://datatracker.ietf.org/doc/html/rfc8439 [Accessed 1 October 2024]
[2] Rahman, M.I.J. 2017. *Stegokripto Dengan Penanda Dinamis*. Institut Teknologi Bandung
[3] Nasution, Y., Furqan, M., Sinaga, M. 2020. Implementasi Steganografi Menggunakan Metode Spread Spectrum Dalam Pengamanan Data Teks Pada Citra Digital. *J-SAKTI (Jurnal Sains Komputer dan Informatika)*, 4 (2), pp.351–358
[4] Hasugian, A.H. 2017. Perancangan Perangkat Lunak Pengenkripsian Citra BMP, GIF dan JPG dengan Menggunakan

Metode HIL L. *Jurnal Ilmu Komputer dan Informatika*, 01 (November), pp.1–11

[5] Santoso, A.R., Riski, A., Kamsyakawuni, A. 2018. Implementasi Algoritma Reversed Vigenere Encryption pada Pengamanan Citra. *Berkala Sainstek*, 6 (2), pp.61

[6] Salamah, U.G., Ekawati, R. 2021. *Pengolahan Citra Digital*. Bandung: Media Sains Indonesia

[7] Hidayatullah, M. 2020. *Digital imaging menggunakan Adobe Photoshop CS6*. Makassar: Yayasan Barcode

[8] Mordvintsev, A., Rahman, A. 2021. *Introduction to OpenCV-Python Tutorials*. [Online] (Updated 9 October 2021) Tersedia di : https://docs.opencv.org/4.5.4/d0/de3/tutorial_py_intro.html [Accessed 1 October 2024]

[9] Semenov, I. 2020. *An Implementation Of ChaCha20 Stream Cypher in All-Programmable SoCs*. The University of Alabama

[10] Muhammad, F., Ahendyarti, C., Masjudin. 2019. Chacha stream cipher implementation for network security in wireless sensor network. *IOP Conference Series: Materials Science and Engineering*, 673 (1), pp.012064

[11] Ramesh, D., Mishra, R., Nayak, B.S. 2016. Cha-Cha 20: Stream cipher based encryption for cloud data centre. In: *ACM International Conference Proceeding Series*. Udaipur, India, 04 March 2016. Association for Computing Machinery : New York, United States.

[12] Sullivan, N. 2015. *Do the ChaCha: better mobile performance with cryptography*. [Online] (Updated 23 February 2015) Tersedia di : https://blog.cloudflare.com/do-the-chacha-better-mobile-performance-with-cryptography/ [Accessed 27 June 2022]

[13] Bernstein, D.J. 2008. ChaCha, a variant of Salsa20. *Workshop Record of SASC*, 8, pp.1–6

[14] Bernstein, D.J. 2008. The Salsa20 Family of Stream Ciphers. *New Stream Cipher Designs*. Berlin, Heidelberg: Springer Berlin Heidelberg

[15] Riverbank Computing Limited. 2022. *PyQt5*. [Online] (Updated 29 Sept 2024) Tersedia di : https://pypi.org/project/PyQt5/ [Accessed 7 August 2022]

[16] Python Software Foundation. 2024. *General Python FAQ*. [Online] (Updated 9 Sept 2024) Tersedia di : https://docs.python.org/3.9/faq/general.htmlfaq/general.html [Accessed 1 October 2024]

[17] Beyri, M.T.P., Kusyanti, A., Bakhtiar, F.A. 2020. Implementasi Algoritme Salsa20 untuk Pengamanan Search Keyword Dokumen Terenkripsi. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 4 (10), pp.3531–3541