**IMPLEMENTASI KOMBINASI CHACHA20 DAN MODIFIKASI AES MELALUI *ADDROUNDKEY* PADA**

**STUDI KASUS KEAMANAN *FILE***

**Proposal Tugas Akhir Mata Kuliah Penulisan Proposal**

## (CAK4EAB2)

Disusun oleh:

## EMILIA RAMONA 1203210017

**PROGRAM STUDI SARJANA INFORMATIKA DIREKTORAT KAMPUS SURABAYA**

**UNIVERSITAS TELKOM SURABAYA TAHUN 2025**

# ABSTRAK

Keamanan data menjadi tantangan besar di era Revolusi Industri 4.0, di mana informasi penting perlu dilindungi dari akses tidak sah. Ancaman serangan siber dan peretasan mengharuskan pengembangan solusi keamanan yang lebih efektif. Penelitian ini bertujuan untuk mengimplementasikan dan mengevaluasi kombinasi algoritma ChaCha20 dan modifikasi *Advanced Encryption* Standard (AES) dengan penambahan langkah *AddRoundKey* sebelum *MixColumns* untuk meningkatkan keamanan *file*. Metode yang digunakan adalah enkripsi ganda, dengan AES modifikasi sebagai *block cipher* yang kuat dan ChaCha20 sebagai *stream cipher* yang cepat. Penelitian ini mencakup implementasi algoritma dengan menggunakan pustaka kriptografi untuk menguji kinerja algoritma melalui parameter seperti waktu enkripsi, waktu *dekripsi*, kecepatan enkripsi, kecepatan *dekripsi*, dan tingkat keacakan data menggunakan analisis entropi. *File* dengan berbagai format dan ukuran, PDF, Word, dan TXT, digunakan sebagai data pengujian untuk mengevaluasi efisiensi algoritma. Selain itu, antarmuka pengguna berbasis web dirancang untuk mempermudah proses enkripsi dan *dekripsi*, melibatkan langkah- langkah seperti pemilihan *file* dan *input* kunci enkripsi. Penelitian ini diharapkan menunjukkan bahwa kombinasi AES modifikasi dan ChaCha20 mampu meningkatkan keamanan data serta memberikan efisiensi yang lebih baik dalam perlindungan *file*.

**Kata Kunci**: Keamanan *File*, *Advanced Encryption Standard* (AES), ChaCha20, Enkripsi Ganda, *AddRoundkey*, *Entropi*.

# ABSTRACT

*Data security has become a major challenge in the era of the Industrial Revolution 4.0, where critical information must be protected from unauthorized access. Cyberattacks and hacking threats demand the development of more effective security solutions. This research aims to implement and evaluate the combination of the ChaCha20 algorithm and a modified Advanced Encryption Standard (AES) by adding an AddRoundKey step before MixColumns to enhance file security. The method used is double encryption, with the modified AES as a robust block cipher and ChaCha20 as a fast stream cipher. This study involves implementing the algorithms using cryptographic libraries to test their performance through parameters such as encryption time, decryption time, encryption speed, decryption speed, and data randomness level using entropy analysis. Files of various formats and sizes, including PDF, Word, and TXT, are utilized as test data to evaluate the efficiency of the algorithms. Additionally, a web-based user interface is designed to simplify the encryption and decryption process, involving steps such as file selection and encryption key input. This research is expected to demonstrate that the combination of modified AES and ChaCha20 can enhance data security and provide better efficiency in file protection.*

***Keywords****: File Security, Advanced Encryption Standard (AES), ChaCha20, Double Encryption, AddRoundkey, Entropy*

# KATA PENGANTAR

Assalamu’alaikum Warrahmatullahi Wabarakatuh, puja puji syukur ke hadirat Allah SWT yang telah memberikan nikmat dan karunia-Nya yang sangat luar biasa kepada hamba – hambanya. Tidak lupa shalawat serta salam kita curahkan atas bimbingan nabi kita nabi Muhammad SAW yang telah membawa cahaya islam ke dalam dunia ini dan menjadi panutan bagi setiap umatnya hingga akhir zaman.

Setelah melalui berbagai proses kerja keras, penyusunan skripsi yang berjudul “Implementasi Kombinasi ChaCha20 dan Modifikasi AES melalui *AddRoundkey* Pada Studi Kasus Keamanan *File*” akhirnya dapat terselesaikan. Selain sebagai salah satu syarat memperoleh gelar S-Kom, penulis berharap juga skripsi ini dapat bermanfaat bagi semua pihak yang berkepentingan. Oleh karena itu, penulis ingin mengungkapkan rasa terima kasih kepada pihak-pihak yang telah memberikan dukungan dan doa sehingga penulis dapat menyelesaikan skripsi ini tepat pada waktunya. Beberapa pihak tersebut antara lain:

1. **Kepada Mama** yang sangat penulis cintai yang tidak pernah lupa selalu memberikan dukungannya dan doa agar diberikan kemudahan dan kelancaran dalam penyusunan skripsi ini sehingga penulis dapat menyelesaikan studi Informatika di Universitas IT Telkom Surabaya.
2. **Kepada Alm.Ibu Suhartatik**, yang penulis cintai dan sayangi, yang telah memberikan motivasi untuk saya.
3. **Kepada Saudara Kembar**, yang selalu mendukung, terima kasih atas bantuan dan waktunya dalam memberikan masukan terkait susunan skripsi ini.
4. **Keluarga Besar penulis** yang telah memberikan motivasi dan kepercayaan besar sehingga penulis bisa mewujudkan impian untuk meraih gelar S-Kom.
5. **Dosen Pembimbing**, Pak Rizky Fenaldo Maulana, S.Kom., M.Kom. dan Tanzila Mustaqim, S.Kom., M.Kom. Yang telah memberikan solusi, waktu, dan bimbingan penuh kesabaran selama proses penyusunan skripsi ini.
6. **Teman-teman dekat**, termasuk Jus’an, Erica Dyah, Qori Emalia, Ardina Firdatana, dan Elma Saskia yang selalu memberikan semangat, mendengarkan keluh kesah, serta menemani dalam berbagai kondisi selama penulisan skripsi.
7. **Semua pihak lainnya**, yang tidak dapat disebutkan satu per satu, atas bantuan dan dukungannya selama proses ini.
8. **Diri penulis sendiri**, atas perjuangan dan kerja keras hingga akhirnya skripsi ini selesai tepat waktu.

Penulis menyadari bahwa skripsi ini masih jauh dari kesempurnaan. Oleh karena itu, saran dan kritik yang membangun sangat diharapkan untuk memperbaiki skripsi ini ke depannya.

Terima kasih.

### Wassalamu’alaikum Warrahmatullahi Wabarakatuh.

Surabaya, 24 Desember 2024 Penulis

Emilia Ramona

# DAFTAR ISI

[LEMBAR PERSETUJUAN ii](#_bookmark0)

[ABSTRAK iii](#_bookmark1)

[ABSTRACT iv](#_bookmark2)

[KATA PENGANTAR v](#_bookmark3)

[DAFTAR ISI vii](#_bookmark4)

[DAFTAR TABEL ix](#_bookmark5)

[DAFTAR GAMBAR x](#_bookmark6)

[DAFTAR RUMUS xi](#_bookmark7)

[BAB I PENDAHULUAN 1](#_bookmark8)

* 1. [Latar Belakang 1](#_bookmark9)
  2. [Rumusan Masalah 4](#_bookmark10)
  3. [Tujuan Penelitian 4](#_bookmark11)
  4. [Batasan dan Asumsi Penelitian 5](#_bookmark12)
  5. [Manfaat Penelitian 5](#_bookmark13)
     1. [Manfaat Akademis 5](#_bookmark14)
     2. [Manfaat Ilmiah 5](#_bookmark15)
  6. [Sistematika Penulisan 6](#_bookmark16)

[BAB II LANDASAN TEORI 7](#_bookmark17)

* 1. [Literatur Terkait Teori 7](#_bookmark18)
  2. [Kajian Teori 10](#_bookmark20)
     1. [Kriptografi 10](#_bookmark21)
     2. [AES (Advanced Encryption Standard) 12](#_bookmark24)
     3. [Chacha20 20](#_bookmark32)
     4. [Entropi 22](#_bookmark35)

[BAB III METODOLOGI PENELITIAN 24](#_bookmark37)

* 1. [Sistematika Penyelesaian Masalah 24](#_bookmark38)
  2. [Alat dan Bahan 24](#_bookmark39)
     1. [Alat 24](#_bookmark40)
     2. [Bahan 25](#_bookmark41)
  3. [Rancangan Penelitian 25](#_bookmark42)
     1. [Studi Literatur 27](#_bookmark45)
     2. [Pengumpulan Data 27](#_bookmark46)
     3. [Implementasi Modifikasi AES 28](#_bookmark47)
     4. [Implementasi Kombinasi AES Modifikasi dan ChaCha20 31](#_bookmark49)
     5. [Implementasi Antarmuka Pengguna (UI) 36](#_bookmark52)
     6. [Pengujian Algoritma 38](#_bookmark54)
     7. [Analisis Hasil 42](#_bookmark59)

[DAFTAR PUSTAKA 44](#_bookmark61)

# DAFTAR TABEL

[Tabel II.1 Literatur Penelitian Terkait Kriptografi 7](#_bookmark19)

[Tabel II.2 Panjang Kunci 13](#_bookmark25)

[Tabel II.3 Tabel Substitusi (S-Box) 16](#_bookmark28)

[Tabel II.4 Transformasi ShiftRows 17](#_bookmark29)

[Tabel III.1 Jenis, Ukuran, dan Isi File untuk Pengujian 40](#_bookmark58)

[Tabel III.2 Evaluasi Hasil 43](#_bookmark60)

# DAFTAR GAMBAR

[Gambar II.1 Alur Proses Enkripsi Dan Dekripsi 11](#_bookmark22)

[Gambar II.2 Proses Kunci Simetris 11](#_bookmark23)

[Gambar II.3.Transformasi ShiftRows 15](#_bookmark27)

[Gambar II.4 Proses AES 19](#_bookmark31)

[Gambar III.1 Diagram Alur Penelitian 26](#_bookmark43)

[Gambar III.2 Diagram Proses Enkripsi AES 28](#_bookmark48)

[Gambar III.3 Diagram Proses Double Encrypt 32](#_bookmark50)

[Gambar III.4 Proses Dekripsi 36](#_bookmark51)

[Gambar III.5 Desain Antarmuka Pengguna Enkripsi dan Dekripsi 37](#_bookmark53)

# DAFTAR RUMUS

[(II-1) Rumus AddRoundKey 14](#_bookmark26)

[(II-2) Operasi Perkalian MixColumns 17](#_bookmark30)

[(II-3) Rumus Keystream 22](#_bookmark33)

[(II-4) Rumus Proses Enkripsi XOR pada Keystream 22](#_bookmark34)

[(II-5) Entropi 22](#_bookmark36)

[(III-1) Pengujian Waktu Enkripsi 39](#_bookmark55)

[(III-2) Pengujian Waktu Dekripsi 39](#_bookmark56)

[(III-3) Pengujian Waktu Kecepatan 39](#_bookmark57)

# BAB I PENDAHULUAN

### Latar Belakang

Semakin meningkatnya teknologi yang semakin canggih membawa sebuah perubahan yang cukup signifikan. Perubahan tersebut dikenal dengan istilah Revolusi Industri 4.0 yang merupakan sebuah kemajuan teknologi yang sering digunakan oleh banyak orang. Revolusi industri 4.0 adalah sebuah data yang digerakkan dengan adanya campur tangan komputer menggunakan metode machine learning atau AI (Alimuddin et al., 2023). Menurut Klaus Schawab revolusi Industri 4.0 memiliki tantangan yang harus dihadapi salah satunya seperti keamanan *cyber* dan privasi data.

Maka dari itu, dengan semakin pesatnya informasi teknologi maka perlunya untuk mengamankan data atau *file*. Keamanan data atau *file* merupakan salah satu langkah yang mudah untuk melindungi sebuah informasi bersifat penting yang berasal dari akses tidak sah. Keamanan *file* ini sangat diperlukan pada era Revolusi Industri 4.0, karena maraknya serangan *cyber* dan peretasan komputer yang bisa mengganggu keberlangsungan suatu industri. Menurut (Mulyana*, 2016*), keamanan *file* merupakan salah satu aksi pencegahan dari sebuah penganggu pengguna *file* atau pengguna yang memakai jaringan yang tidak berkewajiban memiliki akses (Soesanto et al., 2023). Dengan semakin pesatnya dunia teknologi ini membuat keamanan *file* menjadi sangat penting karena untuk melindungi privasi dan informasi dari pihak yang tidak memiliki akses.

Keamanan *file* adalah serangkaian metode atau teknik yang untuk melindungi *file* dari akses yang tidak sah, seperti pencurian atau kerusakan. Tujuannya untuk menjaga informasi dalam *file* tetap rahasia, aman, dan hanya dapat di akses oleh pihak yang berwenang. Oleh karena itu, upaya untuk memastikan agar *file* terjaga dengan menggunakan metode ChaCha20 dan *Advanced Encryption Standard* (AES). ChaCha20 merupakan metode enkripsi yang cepat dan aman. ChaCha20 dipopulerkan oleh Daniel J.

Bernstein sebagai varian dari algoritma. Dalam algoritma ChaCha20 sangat populer dalam pengimplementasi enkripsi dikarenakan percampuran antara kecepatan dan keamanannya yang dapat diunggulkan. Chaca20 mempunyai desain yang hampir sama dengan Salsa20, pada algoritma ChaCha20 menggunakan kunci 256-bit yang mempunyai 8 putaran (Kebande, 2023).

Sedangkan AES (*Advanced Encryption Standard)* adalah sebuah algoritma yang melakukan transformasi terhadap blok data yang menggunakan kunci enkripsi yang sama. AES bekerja pada sistem yang melibatkan operasi matematika dengan level tingkat sulit (Saripa, 2023). Proses kerja enkripsi dasar AES adalah *subbytes* (menggunakan tabel S-Box untuk menggabungkan *byte* pada blok matriks), *shift row* (digunakan untuk menggeser baris matriks), *MixColumns* (melakukan pergeseran dari kanan ke kiri), dan *AddRoundkey* (blok di XOR kan dengan kunci yang diperluas) (Aryanto et al., 2023). Pada proses *dekripsi* AES terdapat tahap-tahapan kebalikan dari proses enkripsi. Dengan memakai kunci yang sama untuk *dekripsi*, blok data yang telah terenkripsi dapat dipulihkan kembali ke bentuk awalnya. algoritma AES adalah jenis *block chipper* yang memiliki panjang kunci 128 bit, 192 bit dan 256 bit (Surbakti & Purwanto, 2024).

Algoritma Chacha20 dan AES merupakan suatu enkripsi yang dapat di digunakan dalam menjaga keamanan *file*. Pada algoritma AES terdapat kelebihan salah satunya adalah sifat karakteristiknya yang di mana selalu ada bidang khusus untuk bilangan prima, memicu kekuatan AES(Surbakti & Purwanto, 2024) . AES juga memiliki sebuah kelemahan salah satunya yaitu algoritma kriptografi yang menggunakan kunci simetris dalam proses pengiriman dan penerimanya data. Namun, penggunaan kunci simetris ini memiliki risiko lebih tinggi terhadap kebocoran, terutama seiring waktu (Hidayatulloh et al., 2023). Pada algoritma ChaCha20 terdapat keunggulan yaitu menggunakan teknik kriptografi ARX (*addition, rotation*, XOR) yang memberikan performa lebih cepat, kompleksitas lebih rendah, dan mampu menghilangkan serangan waktu Klik atau ketuk di sini untuk memasukkan teks.

Meskipun algoritma AES dan Chacha20 telah efektif untuk menjaga peningkatan keamanan *file* dan implementasinya yang cukup rumit sehingga membuat orang yang tidak memiliki akses tidak dapat mengakses *file* tersebut. Hal ini tidak dapat kita pungkiri bahwa semakin pesatnya dunia teknologi membuat banyak orang berbondong-bondong menciptakan pembaruan yang lebih efektif. Metode *double encrypt* ini menawarkan kombinasi yang mampu mengurangi kelemahan dari setiap algoritma jika digunakan secara bersamaan, serta meningkatkan ketahanan terhadap berbagai jenis serangan kriptografi. *Double encrypt* merupakan Teknik kriptografi yang melibatkan proses enkripsi secara dua kali, yang di mana memiliki dua kunci yang berbeda untuk lebih meningkatkan keamanan. *Double encrypt* ini bertujuan untuk meningkatkan keamanan *file* lebih tinggi dibandingkan hanya 1 metode saja (Pangestu et al., 2022).

Pada penelitian terlebih dahulu yang berjudul “Implementasi Sistem Keamanan *File* Menggunakan Algoritma AES untuk Mengamankan *File* Pribadi” yang diteliti oleh Saripa Universitas Negeri Makassar. Dalam penelitian tersebut menjelaskan tentang keamanan *file* dengan menggunakan metode algoritma AES. Di mana hasil penelitian ini menunjukkan bahwa algoritma AES telah berhasil mengamankan *file* yaitu dokumen Word, PDF, Excel, dan Power Poin, *file* yang dienkripsi menjadi tidak terbaca oleh pihak yang tidak memiliki akses sehingga dapat menjaga kerahasiaan data dan AES mampu menjaga keamanan data dengan baik.

Dalam perbedaan penelitian terdahulu dan penelitian saya akan menggunakan double encrypt untuk keamanan *file* dan juga saya akan menggunakan AES dengan menambahkan metode ChaCha20 dan melakukan modifikasi pada bagian AES. Modifikasi menggunakan penambahan *AddRoundkey* sebelum dengan bertujuan agar proses enkripsi akan lebih susah untuk dipecahkan. Dalam algoritma AES, langkah *AddRoundKey* merupakan komponen krusial yang menggabungkan data dengan kunci enkripsi menggunakan operasi XOR. Penelitian (Luong, 2023) oleh menunjukkan bahwa memodifikasi transformasi *AddRoundKey*

dapat meningkatkan keamanan algoritma dengan menciptakan hubungan yang lebih kompleks antara *plainteks* dan *cipherteks*. Penelitian ini menyoroti pentingnya penyesuaian langkah *AddRoundKey* untuk meningkatkan ketahanan algoritma AES terhadap serangan kriptografi, seperti *differential cryptanalysis* dan linear *cryptanalysis*.

Dengan mendasarkan pada penelitian tersebut, modifikasi algoritma AES melalui penambahan langkah *AddRoundKey* sebelum *MixColumns* bertujuan untuk lebih memperkuat difusi data pada tahap awal enkripsi. Penambahan ini diharapkan dapat meningkatkan kerandoman hasil enkripsi dan mengurangi kemungkinan eksploitasi pola oleh pihak yang tidak berwenang, sehingga keamanan algoritma menjadi lebih optimal. Berdasarkan dari latar belakang di atas penelitian ini memiliki tujuan untuk meningkatkan keamanan *file* agar lebih sulit untuk diretas atau diakses pihak yang tidak berwenang. Oleh karena itu penelitian ini dianggap penting karena maraknya kasus peretasan data oleh oknum yang tidak memiliki akses.

### Rumusan Masalah

1. Bagaimana implementasi dari modifikasi AES dengan penambahan *AddRoundkey* sebelum *MixColumns* dalam meningkatkan keamanan *file*?
2. Bagaimana performa kombinasi antara AES modifikasi dan ChaCha20 dalam hal waktu enrkipsi, waktu *dekripsi* , kecepatan enkripsi, dan kecepatan *dekripsi*?

### Tujuan Penelitian

1. Mampu mengidentifikasi *AddRoundkey* yang akan digunakan dalam menjaga keamanan *file* dan mengimplementasi kemampuan Chacha20 dan AES yang telah dimodifikasi dalam proses menjaga keamanan *file*.
2. Mampu mengevaluasi performa algoritma ChaCha20 dan AES yang telah dimodifikasi berdasarkan waktu enkripsi, waktu *dekripsi*, kecepatan enkripsi, dan kecepatan *dekripsi*, untuk menentukan efisiensi

dan efektivitas kombinasi kedua algoritma tersebut dalam menjaga keamanan *file*

### Batasan dan Asumsi Penelitian

1. Penelitian ini berfokus pada jurnal atau artikel ilmiah dengan batasan tahun 2021-2024 yang membahas tentang :
   * Keamanan *file*
   * Algoritma ChaCha20
   * *Advanced Encryption Standard* (AES)
   * Double encryption
2. Penelitian dilakukan dengan pengujian pada *file* dengan ukuran dan format sebagai berikut :
   * Format PDF dan Word dengan variasi ukuran *file* 100 KB - 100MB.
   * Format *file* teks (TXT) dengan ukuran maksimal 300 byte Batasan ukuran *file* digunakan untuk memastikan pengujian dilakukan secara efektif tanpa mengurangi kinerja proses enkripsi dan *dekripsi*.

### Manfaat Penelitian

### Manfaat Akademis

* + - 1. Memberikan inovasi terbaru mengenai pelindungan keamanan

*file* dengan menambahkan *AddRoundkey* pada algoritma AES.

* + - 1. Memudahkan peneliti lainnya dalam kinerja *AddRoundkey* pada algoritma AES yang berfungsi untuk keamanan *file***.**

### Manfaat Ilmiah

* + - 1. Menjadi sumber acuan dan sumber referensi bagi peneliti lainnya yang akan meneliti topik ChaCha20 dan AES modifikasi, terutama dalam konteks keamanan data dan performa enkripsi.
      2. Membuktikan hipotesis apakah AES yang dimodifikasi dengan tambahan *AddRoundkey* dapat meningkatkan performa keamanan dan efisiensi dibandingkan dengan AES standart, dan sekaligus menunjukkan kinerja kombinasi ChaCha20 dan AES dalam pengamanan *file* berbagai format dan ukuran.

### Sistematika Penulisan

## BAB 1 PENDAHULUAN

Menjelaskan latar belakang penelitian, rumusan masalah, tujuan penelitian, batasam penelitian, manfaat penelitian, serta sistematika penulisan secara umum.

## BAB 2 LANDASAN TEORI

Membahas teori-teori yang relavan dengan penelitian, termasuk algoritma ChaCha20, AES, konsep kriptografi, Serta tinjauan terhadap penelitian terdahulu yang terkait.

## BAB 3 METODOLOGI PENELITIAN

Menjelaskan metode yang digunakan dalam penelitian, termasuk algoritma yang diterapkan (modifikasi AES dan ChaCha20 dengan enkripsi ganda). Diuraikan juga alat dan bahan yang digunakan, serta tahapan penelitian yang meliputi implementasi algoritma, pengujian parameter kinerja (waktu dan kecepatan enkripsi, dekripsi, serta analisis entropi), dan evaluasi hasil pengujian pada berbagai jenis file (PDF, Word, TXT).

## BAB 4 IMPLEMENTASI DAN PENGUJIAN

Menyajikan hasil implementasi algoritma yang dikembangkan, pengujian algoritma terhadap berbagai jenis *file*, serta analisis data hasil pengujian**.**

## BAB 5 EVALUASI HASIL PENGUJIAN

Menyajikan hasil implementasi dari aplikasi yang telah dirancang, pengujian fungsionalitas sistem, serta evaluasi kinerja aplikasi terhadap berbagai skenario penggunaan. Analisis data hasil pengujian juga dilakukan untuk menilai keberhasilan aplikasi dalam mencapai tujuan penelitian.

## BAB 6 KESIMPULAN DAN SARAN

Merangkum temuan utama penelitian serta memberikan saran untuk pengembangan lebih lanjut di masa depan.

# BAB II LANDASAN TEORI

### Literatur Terkait Teori

Keamanan informasi menjadi salah satu isu yang paling mendesak di era digital saat ini, di mana data sensitif sering kali menjadi target serangan siber yang beragam. Dengan meningkatnya ancaman seperti serangan *man-in-the- middle*, *eavesdropping*, dan *DDoS*, penting bagi organisasi dan individu untuk menerapkan metode enkripsi yang efektif untuk melindungi data mereka. Berbagai algoritma enkripsi telah dikembangkan untuk memenuhi kebutuhan ini, dengan *Advanced Encryption Standard* (AES) dan ChaCha20 menjadi dua di antara yang paling banyak digunakan. Penelitian-penelitian terkini menunjukkan bahwa kedua algoritma ini tidak hanya menawarkan tingkat keamanan yang tinggi, tetapi juga dapat diimplementasikan dalam berbagai konteks, mulai dari pengamanan *file* hingga enkripsi konten *livestreaming*.

Pada [Tabel II.1 Literatur Penelitian Terkait Kriptografi](#_bookmark19) menyajikan ringkasan dari beberapa penelitian yang mengeksplorasi efektivitas dan penerapan algoritma AES dan ChaCha20 dalam berbagai aplikasi. Setiap penelitian memberikan wawasan tentang tujuan, hasil, dan persamaan dalam pendekatan yang digunakan, serta menyoroti inovasi yang dilakukan untuk meningkatkan keamanan data. Dengan memahami hasil-hasil ini, kita dapat lebih menghargai perkembangan teknologi enkripsi dan aplikasinya dalam menjaga kerahasiaan serta integritas informasi di dunia yang semakin terhubung.

### Kajian Teori

### Kriptografi

Menurut Menezes, Oorschot, dan Vantone (1996), Kriptografi adalah studi teknik matematika yang berkaitan dengan keamanan informasi seperti kerahasiaan, *autentikasi* keaslian data (Belvin Tannadi, 2022). Kriptografi dapat diartikan sebagai sebuah proses untuk melindungi data atau sebuah sistem yang bertujuan untuk melindungi informasi rahasia dengan cara yang tidak dipahami oleh pihak yang tidak berwenang. Dalam kriptografi yang meliputi proses enkripsi dan proses *dekripsi*. Di dalam Kriptografi terdapat tiga fungsi dasar yang meliputi :

* + - 1. Enkripsi merupakan proses data asli (*plainteks*) yang akan diubah menjadi data yang tidak bisa terbaca oleh pihak yang tidak berwenang (*chiperteks*).
      2. *Dekripsi* merupakan proses pengembalian dari data yang tidak bisa terbaca(*chiperteks*) menjadi data terbaca atau data asli (*plainteks*).
      3. Kunci merupakan proses yang dipakai untuk melakukan proses enkripsi dan *dekripsi*, terdapat 2 jenis kunci, yaitu kunci rahasia (*private key*) dan kunci publik (*public key*).

Pada [Gambar II.1 Alur Proses Enkripsi Dan *Dekripsi*](#_bookmark22)

memperlihatkan alur enkripsi dan *dekripsi.* Proses ini dimulai

dengan *plainteks* yang kemudian dienkripsi dengan menggunakan kunci menjadi *chiperteks*. *Chiperteks* tersebut kemudian *didekripsi* kembali menggunakan kunci yang sama untuk mendapatkan kembali *plainteks*.

*Gambar II.1 Alur Proses Enkripsi Dan Dekripsi*

Berikut algoritma Kriptografi yang berdasarkan kunci :

a. Algoritma Simetris

Algoritma simetris adalah algoritma yang merujuk pada metode enkripsi dan *dekripsi* yang di mana algoritma ini memiliki kunci tunggal atau sama. Sehingga dapat disimpulkan algoritma ini mewajibkan pengirim dan penerima harus menyetujui satu kunci yang digunakan dalam proses enkripsi dan *dekripsi* data. Namun, kelemahan dari algoritma ini yaitu tergantung pada rahasia kunci. Apabila kunci tersebut tersebar maka akan dengan mudah pihak yang tidak memiliki akses dapat melakukan proses enkripsi dan *dekripsi* data secara mudah.

*Gambar II.2 Proses Kunci Simetris*

Pada [Gambar II.2 Proses Kunci Simetris](#_bookmark23) menjelaskan proses enkripsi dan *dekripsi* menggunakan kunci simetris. Dimulai dengan pesan yang dikirimkan, kemudian dienkripsi menjadi kode yang tidak bisa dibaca menggunakan kunci simetris. Kode tersebut kemudian dikirim dan di *dekripsi* di sisi penerima menggunakan kunci simetris yang sama, sehingga pesan dapat dibaca kembali. Pada contoh gambar, pesan "Selamat Datang!" dienkripsi menjadi kode "9CDF4837 AB672019" menggunakan kunci simetris yang sama. Kemudian di *dekripsi* kembali ke pesan awal "Selamat Datang!" Kunci simetris merupakan kunci yang sama digunakan untuk enkripsi dan *dekripsi*, sehingga pengirim dan penerima pesan harus memiliki kunci yang sama agar dapat berkomunikasi.

### AES (Advanced Encryption Standard)

Advanced Encryption Standard (AES) adalah salah satu algoritma enkripsi yang paling terkenal dan banyak digunakan, serta terus dikembangkan sebagai pengganti algoritma Data Encryption Standard (DES). AES adalah algoritma enkripsi yang bekerja dengan memproses data dalam blok dengan ukuran minimal 128 bit, dan melalui beberapa tahapan untuk mengenkripsi data. Tujuan utama dari AES adalah untuk mengamankan data rahasia sehingga tidak dapat dibaca oleh pihak yang tidak berwenang (Wahyu Putra, 2023). Pada [Tabel II.2 Panjang Kunci](#_bookmark25) dalam proses enkripsi *Advanced Encryption Standard* (AES) menggunakan ukuran blok dengan panjang sebesar 126 bit, dengan pilihan panjang kunci yang bervariasi, yaitu 128 bit, 192 bit, dan 256 bit (Sekar Putri Ananda et al., 2022). Proses enkripsi dalam algoritma ini dilakukan melalui beberapa tahap yang dikenal sebagai ronde. Jumlah ronde yang digunakan bergantung pada panjang kunci yang akan dipilih. Berikut adalah hubungan panjang kunci dan banyaknya ronde.

Algoritma AES juga menawarkan tingkat keamanan yang sangat tinggi dikarenakan ukuran kunci dan jumlah ronde yang ditentukan. Ukuran kunci menentukan jumlah kemungkinan kombinasi kunci, sehingga membuat serangan *bruteforce* menjadi sangat sulit dilakukan. Di sisi lain, jumlah ronde yang besar dapat menghasilkan keamanan yang kuat, sehingga perubahan kecil pada *plainteks* dapat memengaruhi seluruh bagian *cipherteks*. Selain keunggulan keamanannya, AES juga memiliki efisiensi tinggi dalam hal waktu dan ruang. Algoritma ini dapat diimplementasikan baik pada perangkat keras maupun perangkat lunak, menjadikannya ideal untuk berbagai aplikasi, termasuk perlindungan data (Assyahid, 2024). Pada tahap awal enkripsi, data masukan yang telah disalin ke dalam matriks *state* akan menjalani transformasi *AddRoundkey*. Selanjutnya, matriks *state* akan melalui proses transformasi secara berulang, mencakup *SubBytes*, *ShiftRows*, *MixColumns*, dan *AddRoundkey*, dengan jumlah pengulangan sesuai dengan nilai Nr (Handoyo & Subakti, 2020). *Dekripsi* adalah proses mengembalikan *ciphertext* menjadi plaintext, atau mengubah data terenkripsi kembali ke bentuk aslinya. Pada proses *inverse cipher*, transformasi yang digunakan meliputi *InvShiftRows*, *InvSubBytes*, *InvMixColumns*, dan *AddRoundkey*. Berikut ini adalah cara kerja algoritma AES secara umum:

### Proses Enkripsi AES Modifikasi

* + - * 1. *AddRoundkey*

*AddRoundkey* adalah proses menggabungkan state masukan dengan kunci ronde dengan menggunakan operasi XOR. Dalam proses transformasi enkripsi dan *dekripsi* pada AES, dilakukan operasi penambahan kunci putaran (*round key*)

pada state menggunakan operasi XOR (Djong & Siswanto, 2022). Setiap kunci putaran terdiri dari *Nb word* di mana setiap *word* tersebut akan digabungkan dengan *word* atau kolom yang sesuai pada *state* . Secara matematis, hal ini (Putri et al., 2021).

Keterangan Algoritma dari *AddRoundkey* di mana :

Pada [Gambar II.3.Transformasi ShiftRows](#_bookmark27) di atas di mana S0,0 merupakan elemen di baris ke-0, kolom ke-0, S3,3 merupakan elemen di baris ke-3, kolom ke-3, dan 𝑊3,3 merupakan elemen di baris ke-3, kolom ke-3. Pada label S0,0 dan lainnya merupakan state saat ini, state ini representasi internal data yang sedang dienkripsi ataupun *didekripsi* dan setiap elemen adalah *byte* dari blok data input. Sedangkan pada label 𝑊0,0 merupakan kunci ronde yang dihasilkan dari proses *keyexpansion*, di mana ini bagian dari kunci utama yang dipecah untuk menjadi beberapa ronde dan setiap elemen adalah *byte* dari kunci untuk ronde tertentu. Operasi XOR dilakukan antara setiap *byte* pada *state* dengan *byte* kunci ronde yang sesuai. Hasil di atas dihasilkan dari operasi XOR yang akan digunakan untuk langkah transformasi berikutnya.

* + - * 1. *SubBytes*

Transformasi *transformasi byte* di mana setiap elemen dalam matriks *state* dipetakan menggunakan tabel substitusi yang dikenal sebagai S-Box. Setiap *byte* dalam matriks *state* direpresentasikan sebagai S’[r, c], yang merupakan elemen hasil dari perpotongan antara baris (x) dan kolom (y) (Kurnia Mahesa Naibaho et al., 2021). S-Box berupa matriks

berukuran 16×16 yang berisi nilai-nilai heksadesimal di mana nilai-nilai ini diperoleh melalui perhitungan tertentu (Hena & Kusumaningsih, 2024). Berikut ini tabel S-Box :

Pada [Tabel II.3 Tabel Substitusi (S-Box)](#_bookmark28) sebagai contoh, nilai heksadesimal {62} merujuk ke baris 7 dan kolom 3 dalam S-Box, yang berisi nilai {aa}. Dengan demikian, nilai

{62} dipetakan menjadi nilai {aa}.

* + - * 1. *ShiftRows*

Transformasi *ShiftRows* merupakan proses pergeseran *byte* di mana *byte* paling kiri dipindahkan ke posisi paling kanan (rotasi bit). Jumlah pergeseran dalam melakukan *ShiftRows* ini dilakukan perbedaan setiap baris dalam matriks *state*. Pada baris pertama, tidak ada pergeseran yang dilakukan. Kemudian pada baris kedua barulah mengalami pergeseran satu *byte* ke kiri, lalu baris ketiga digeser dua *byte* ke kiri, dan baris keempat digeser tiga *byte* ke kiri. Tujuan dari proses ini adalah untuk menciptakan *diffusion*(), yang memiliki tujuan untuk membantu menyebarkan efek transformasi non-linear ke seluruh baris dalam matriks *state* untuk proses selanjutnya (Cristy & Riandari, 2021). Proses pergeseran ini bertujuan untuk mencampur elemen-elemen dalam *state* sehingga memperkuat kerahasiaan data.

*MixColumns* adalah proses transformasi di mana setiap elemen dalam blok *cipher* dikalikan dengan matriks . Transformasi ini bertujuan untuk mencampur elemen- elemen dalam setiap kolom *state* berguna meningkatkan difusi data dan memperkuat keamanan enkripsi . Transformasi ini bekerja dengan membagi matriks atau *state* menjadi empat bagian, untuk setiap kolomnya (Widyawan & Imelda, 2021). Setiap kolom dalam matriks *state* diperlakukan sebagai *polinomial* dengan empat bagian dalam *Galois Field* (GF), pada persamaan [(II-2)](#_bookmark30) dilakukan Algoritma dari *AddRoundkey* di mana :

𝑋8 = Elemen dengan derajat ke-8 yang akan muncul sebagai 100000000 (1 diposisi ke 9 dari kanan) di bilangan biner.

X = Menunjukkan elemen dengan derajat ke 1 yang akan muncul sebagai 00000010 ( 1 di posisi ke 2 dari kanan

) dalam bilangan biner

1 = Elemen konstanta yang akan muncul sebagai

00000001 ( 1 diposisi ke 1 dari kanan), dimana elemen 1 ini menyatakan bahwa *polinomial* ini memiliki koefisien konstan.

Proses *MixColumns* juga dapat direpresentasikan sebagai operasi perkalian matriks, di mana setiap kolom matriks dikalikan dengan matriks tertentu dalam *Galois Field* (Cristy & Riandari, 2021).

* + - 1. **Proses *dekripsi* AES :**

Proses *dekripsi* merupakan kebalikan dari proses enkripsi. *Transformasi bytes* yang digunakan dalam *dekripsi* mencakup *AddRoundKey*, *InvShiftRows*, *InvSubBytes,* dan *InvMixColumns*. Pada iterasi pertama *dekripsi*, dilakukan *transformasi AddRoundKey, Inverse ShiftRows, dan Inverse SubBytes. Ciphertex*t kemudian akan mengalami transformasi *AddRoundKey* .

* + - * 1. *InvShiftRows*

Transformasi *byte* yang merupakan kebalikan dari proses *ShiftRows*. Dalam transformasi ini, bit-bit pada *state* digeser ke arah kanan, berbeda dengan *ShiftRows* yang melakukan pergeseran bit ke arah kiri.

* + - * 1. *InvSubBytes*

Transformasi *byte* yang menjadi kebalikan dari *SubBytes*. Dalam proses ini, setiap elemen pada *state* dipetakan menggunakan tabel substitusi *Inverse* S-Box.

* + - * 1. *InvMixColumns*

Setiap kolom dalam *state* dikalikan dengan matriks khusus yang digunakan dalam proses perkalian pada algoritma AES (Syahputra Sianipar et al., 2024)

*Gambar II.4 Proses AES*

Pada [Gambar II.4 Proses AES](#_bookmark31) sebelah kiri menunjukkan tahapan proses enkripsi pada algoritma AES Modifikasi. Proses dimulai dengan memasukkan plainteks, kemudian melalui serangkaian langkah seperti *AddRoundkey*, SubBytes, ShiftRows, MixColumns, dan diakhiri dengan *AddRoundkey* tambahan. Langkah ini diulang sebanyak Nr−1 ronde untuk meningkatkan keamanan data, menghasilkan cipherteks di akhir proses. Flowchart sebelah kanan menjelaskan tahapan proses *dekripsi*, yang merupakan kebalikan dari proses enkripsi. Proses dimulai dengan memasukkan cipherteks, kemudian melalui langkah-langkah seperti *AddRoundkey*, Inverse ShiftRows, Inverse SubBytes, dan Inverse MixColumns. Setelah semua ronde selesai, data dikembalikan ke bentuk plainteks asli.

### Chacha20

Algoritma ChaCha20 adalah salah satu jenis *stream cipher* yang dikembangkan sebagai varian dari algoritma Salsa20. Algoritma ini berbasis desain yang serupa dengan Salsa20, namun ChaCha20 meningkatkan tingkat *diffusion* pada setiap ronde untuk memperkuat keamanan proses enkripsi (Anindita & 18219086, 2023). Chacha20 beroperasi pada unit data 32-bit menggunakan kunci 256-bit. Algoritma ini menghasilkan *output* berupa *stream* kunci sepanjang 512-bit, yang kemudian akan di XOR kan dengan *plainteks* untuk menghasilkan *chiperteks*(Alyas & Abdullah, 2021b). ChaCha20 dimulai dengan membuat *state* 512-bit, yang berupa *array* yang berukuran 16x32-bit. State ini disusun dari empat konstanta 32 bit, kunci rahasia sepanjang 256-bit, penghitung blok 32-bit, serta nonce Ini menunjukkan struktur *state* ChaCha20 yang terdiri dari matriks 4x4 dengan 16 elemen 32-bit. Matriks ini diinisialisasi dengan empat konstanta 32-bit, kunci 256-bit yang dibagi menjadi 8 bagian, penghitung 32 blok 32-bit, dan Nonce 96-bit. Elemen-elemen ini digunakan untuk menghasilkan *keystream* yang kemudian di XOR kan dengan *plainteks* untuk menghasilkan *chiperteks*. Di mana *const* (konstanta) ini berisi empat nilai tetap 32-bit, *key* (kunci) yang terbagi 8 bagian 32-bit dari kunci rahasia 256-bit, *counter* (penghitung), dan Nonce (nilai acak) yang berukuran 96-bit.

Langkah – Langkah Algoritma ChaCha20 :

* + - 1. *Inisialisasi* State

Algoritma ChaCha20 dimulai dengan membangun sebuah state internal berbentuk matriks 4x4 (32-bit) (Barbero et al., 2022) . Matriks ini terdiri dari:

* + - * 1. Baris pertama: Berisi 4 konstanta tetap (128-bit), yang digunakan untuk mengenali algoritma.
        2. Baris kedua dan ketiga: Berisi kunci rahasia 256-bit (8 elemen 32-bit), yang digunakan untuk proses enkripsi.
        3. Baris keempat: Berisi kombinasi 64-bit counter dan 64-bit nonce (masing-masing terdiri dari 2 elemen 32-bit).
      1. *Quarter Round*

Operasi ini memodifikasi 4 elemen dari matriks ( a, b, c, d ) (Almukhlifi & Vora, 2020)

𝑎 += 𝑏; 𝑑 ^ = 𝑎; 𝑑 <<<= 16;

𝑐 += 𝑑; 𝑏 ^ = 𝑐; 𝑏 <<<= 12;

𝑎 += 𝑏; 𝑏 ^ = 𝑎; 𝑑 <<<= 8;

𝑐 += 𝑑; 𝑏 ^ = 𝑐; 𝑏 <<<= 7;

* + - 1. *Double Round*

*Double Round* adalah kombinasi dari dua *Quarter Round* yang dijalankan secara berurutan. Setiap iterasi *Double Round* mencakup:

* + - * 1. Operasi pada kolom matriks *(Column Round):*

melakukan operasi Quarter Round pada setiap kolom matriks 4x4. Tujuannya adalah untuk mencampur data antar elemen dalam kolom, sehingga meningkatkan difusi data secara vertikal.

* + - * 1. Operasi pada diagonal matriks (Diagonal Round):

Setelah operasi pada kolom selesai, langkah berikutnya adalah menerapkan Quarter Round pada elemen-elemen diagonal dalam matriks.

Proses Double Round dilakukan sebanyak 10 iterasi, menghasilkan total 20 ronde Quarter Round untuk setiap blok data yang diproses.

* + - 1. Finalisasi State
      2. *Keystream Generation*

Setelah 20 ronde selesai*, state* akhir ditambahkan kembali ke

### Entropi

Entropi dalam teori informasi digunakan untuk mengukur keacakan data, yang mencerminkan kekuatan algoritma enkripsi. Semakin tinggi nilai entropi, semakin acak data tersebut, yang berarti algoritma enkripsi lebih tahan terhadap serangan (Wahyudi et al., 2024). Entropi dihitung dengan menggunakan rumus pada

Dimana:

* 𝐻𝑚 : Menunjukkan entropi data, dimana tingkat keacakan dalam

# BAB III METODOLOGI PENELITIAN

### Sistematika Penyelesaian Masalah

Penelitian ini bersifat kuantitatif dengan tujuan mengevaluasi performa algoritma AES modifikasi dan kombinasi ChaCha20 dan AES modifikasi. Penilaian performa ini dilakukan berdasarkan waktu enkripsi, waktu *dekripsi*, kecepatan enkripsi, dan kecepatan *dekripsi*, yang akan dianalisis menggunakan data hasil pengujian pada berbagai jenis *file* dengan format PDF, Word dan TXT, yang hasilnya berbentuk tabel untuk mempermudah analisis perbandingan.

### Alat dan Bahan

### Alat

1. Perangkat Keras (Hardware) Laptop dengan spesifikasi :
   1. Prosesor : Intel *Core* I5
   2. RAM : 8 GB
   3. Penyimpanan : 256 GB SSD atau lebih
2. Perangkat Lunak (*Software*)
   1. *Python* :

Menggunakan *library cryptography, pycryptodome*, dan *library* lain yang mendukung algoritma AES dan ChaCha20. Berikut beberapa pustaka yang digunakan :

* + - PyCryptodome : Digunakan Iplementasi algoritma AES dan ChaCha20
    - Cryptography : Implementasi kemanan tambahan (jika nanti dibutuhkan )
    - Numpy : Digunakan untuk analisis data (perhitungan entropi)
    - Time : Mencatat waktu eksekusi algoritma, baik untuk enkripsi maupun dekripsi.
  1. *Text* Editor :

Visual Studio Code dan *Jupyter Notebook*

* 1. Sistem Operasi : Windows

### Bahan

1. Data set *file* :
   1. *File* PDF : Ukuran *file* 100 KB – 100 MB
   2. *File* Word : Ukuran *file* 100 KB – 100 MB
   3. *File* TXT : Ukuran *file* maksimal 300 *byte*
2. Literatur :

Jurnal atau artikel ilmiah tentang AES, ChaCha20 dan *Double Encrypt* (tahun 2020-2024)

1. *Spreadsheet* Analisis Data :

Format CSV (Comma Separated Values)

### Rancangan Penelitian

Pada penelitian ini, dilakukan tahapan yang dirancang secara sistematis untuk mencapai tujuan utama, yaitu Implementasi kombinasi Advanced Encryption Standard (AES) yang dimodifikasi dan ChaCha20. Proses penelitian dimulai dengan studi literatur untuk memahami konsep dasar dan pengembangan terbaru terkait algoritma kriptografi. Setelah itu, dilakukan pengumpulan data yang relevan sebagai dasar implementasi. Tahapan selanjutnya mencakup implementasi algoritma modifikasi AES serta kombinasi algoritma tersebut dengan ChaCha20. Untuk mengukur efektivitas dan efisiensi algoritma yang dikembangkan, pengujian dilakukan pada berbagai jenis *file* seperti PDF, Word, dan TXT. Analisis hasil dari pengujian ini memberikan dasar untuk menarik kesimpulan dan memberikan rekomendasi untuk pengembangan lebih lanjut.

*Gambar III.1 Diagram Alur Penelitian*

[Gambar III.1 Diagram Alur Penelitian](#_bookmark44) ini terdapat beberapa tahapan yaitu:

1. Mulai adalah tahap awal dari penelitian. Pada tahap ini menentukan tujuan dan menyusun langkah-langkah yang akan dilakukan dalam penelitian.
2. Studi literatur dilakukan pengumpulan referensi yang berkaitan dengan algoritma AES, ChaCha20 dan *Double Encrypt. Referensi* berupa jurnal, artikel ilmiah.
3. Pengumpulan data untuk mengumpulkan *file* atau data set yang akan digunakan untuk pengujian algoritma. Format *file* yang digunakan adalah PDF, Word dan TXT.
4. Implementasi modifikasi AES, pada tahap ini algoritma AES dimodifikasi dengan menambahkan langkah *AddRoundkey* sebelum *Mixcolumns*. Tujuannya adalah untuk meningkatkan keamanan dan kecepatan enkripsi dibandingkan dengan AES standar.
5. Implementasi kombinasi AES modifikasi dan ChaCha20, menggabungkan algoritma AES yang sudah dimodifikasi dengan ChaCha20 untuk melakukan *double encypt*. Di mana proses ini bertujuan untuk meningkatkan keamanan *file* lebih baik daripada menggunakan *single encrypt*.
6. Pengujian algoritma, untuk menguji performa algoritma yang sudah diimplementasikan, di mana data set yang dikumpulkan sebelumnya itu digunakan untuk uji coba. Parameter yang diukur pada pengujian ialah

waktu enkripsi dan *dekripsi,* kecepatan enkripsi dan *dekripsi*, dan hasil pengujian ini akan digunakan untuk mengevaluasi kinerja algoritma.

1. Analisis hasil, data pengujian yang sudah terkumpul akan dianalisis. Analisis dilakukan untuk membandingkan performa AES standar, AES yang sudah dimodifikasi, dan kombinasi AES modifikasi dan ChaCha20. Tujuannya sendiri untuk mengetahui algoritma mana yang lebih efisien.
2. Selanjutnya ke kesimpulan dan saran, pada tahap ini merangkum hasil penelitian termasuk kelebihan dan kekurangan algoritma yang sudah diuji. Saran juga untuk pengembangan penelitian lebih lanjut
3. Selesai

### Studi Literatur

Studi literatur dilakukan untuk memahami konsep dasar dan perkembangan terbaru terkait algoritma kriptografi yang digunakan dalam penelitian ini, yaitu Advanced Encryption Standard (AES) dan ChaCha20. Proses ini melibatkan pengumpulan referensi dari jurnal, paper, dan buku yang membahas metode enkripsi, keamanan data, serta teknik evaluasi seperti entropi dan analisis performa. Literatur yang dikaji juga mencakup penelitian sebelumnya tentang algoritma kriptografi untuk keamanan *file*, sehingga memberikan dasar teori yang kuat untuk modifikasi algoritma dan penerapan metode double encryption dalam penelitian ini.

### Pengumpulan Data

Data yang digunakan dalam penelitian ini dibuat sendiri dengan mengikuti ketentuan yang telah ditentukan. Format data meliputi *file* PDF, Word, dan TXT dengan ukuran bervariasi, mulai dari 100 KB hingga 100 MB untuk *file* PDF dan Word, serta hingga 300 *byte* untuk *file* TXT. Data ini dirancang sedemikian rupa agar sesuai dengan kebutuhan pengujian algoritma enkripsi, seperti mengukur waktu enkripsi, waktu *dekripsi*, kecepatan, dan tingkat keacakan menggunakan analisis entropi. Variasi format dan ukuran data ini

bertujuan untuk memastikan bahwa pengujian mencakup berbagai jenis isi *file*.

### Implementasi Modifikasi AES

Pengembangan algoritma *Advanced Encryption Standard* (AES) dilakukan dengan memodifikasi. Modifikasi dilakukan dengan menambahkan *AddRoundkey* sebelum *Mixcolumns*. Penambahan ini bertujuan untuk memperkuat keamanan dan meningkatkan kompleksitas pada proses enkripsi, sehingga lebih sulit untuk diretas dengan pihak tidak berwenang. Berikut adalah proses alur dari modifikasi algoritma AES.

*Gambar III.2 Diagram Proses Enkripsi AES*

Pada [Gambar III.2 Diagram Proses Enkripsi AES](#_bookmark48) ini digunakan untuk mengamankan *file* dengan melakukan tahap enkripsi terhadap data *plainteks* yang menggunakan kunci tertentu. Dalam penelitian ini, modifikasi dilakukan dengan menambahkan langkah *AddRoundkey* untuk meningkatkan tingkat keamanan dan kompleksitas enkripsi. Berikut langkah-langkah proses modifikasi AES .

1. Input *file* dan kunci

Proses enkripsi dimulai dengan memasukkan *file* input (*file* asli yang akan dienkripsi) dan kunci enkripsi. *File* input dibaca dalam mode biner, diubah menjadi *byte array*, lalu dipecah menjadi blok-blok data berukuran 16 *byte*. Jika panjang data tidak habis dibagi 16 *byte, padding* dilakukan dengan metode PKCS#7 untuk melengkapinya.

Contoh :

* 1. *File* input : contoh.pdf

Isi *file* dalam biner : [0x25, 0x50, 0x44, 0x46, ...]

*Byte array* : [37, 80, 68, 70, 10, 0, 45, 78, ...]

* 1. Kunci enkripsi :

Kunci : surabayaitelkom

ASCII : [115, 117, 114, 97, 98, 97, 121, 97, 105, 116, 116,

101, 108, 107, 111, 109]

* 1. Matriks data *file* :

|  |  |  |  |
| --- | --- | --- | --- |
| [37, | 80, | 68, | 70] |
| [10, | 0, | 45, | 78] |
| [255, | 100, | 32, | 12] |
| [80, | 50, | 60, | 72] |

1. *AddRoundkey*

Merupakan tahap awal proses enkripsi, di mana *plainteks* ini digabungkan dengan kunci awal yang menggunakan operasi logika yaitu XOR. Di mana blok data *file* digabungkan dengan kunci awal (W0-W3) menggunakan operasi XOR untuk menambahkan lapisan keamanan awal.

Contoh :

* 1. Proses XOR :

37 XOR 115 = 86

80 XOR 117 = 37

68 XOR 114 = 54

70 XOR 97 = 39

* 1. Matriks setelah *AddRoundkey* :

[86, 37, 54, 39]

[108, 97, 76, 47]

[150, 16, 84, 97]

[28, 89, 83, 37]

1. Proses ronde 1-9
   1. *SubBytes*

Pada proses ini, nantinya setiap *byte* pada matriks *state* ( data yang diperoleh *AddRounKey*) akan diubah menggunakan tabel substitusi yang biasa disebut dengan S-Box. *Tranformasi* ini bertujuan untuk meningkatkan keacakan data agar sulit diprediksi.

Contoh proses :

*Byte* 37 diganti dengan nilai dari tabel S-Box jadi 120

* 1. *ShiftRows*

Setiap baris dalam matriks *state* nantinya digeser ke kiri pada baris pertama tidak ada pergeseran, baris kedua di geser 1 *byte* ke kiri, baris ketiga ini digeser 2 *byte* ke kiri, dan terakhir baris keempat yang nantinya digeser 3 *byte* ke kiri. Contoh proses :

[244, 120, 99, 200] [244, 120, 99, 200]

[60, 107, 55, 175] [107, 55, 175, 60]

[12, 243, 172, 85] [172, 85, 12, 243]

[180, 97, 92, 65] [65, 180, 97, 92]

* 1. *AddRoundkey* (W4-W7)

Setelah *ShiftRows* dilakukan operasi XOR antara matriks *state* dengan kunci ronde berikutnya yaitu (W4-W7) Contoh proses :

*Byte* 244 XOR 115 = 135

*Byte* 120 XOR 117 = 13

Hasil *AddRounKey* : [135, 13, 17, 169]

[38, 98, 85, 12]

[55, 32, 82, 84]

[49, 53, 180, 193]

* 1. *MixColumns*

Setiap kolom pada matriks *state* dimodifikasi dengan operasi matematis berbasis matriks yang menggunakan bidang *Galois* (GF). Di mana proses ini bertujuan untuk mencampur elemen-elemen dalam kolom dan meningkatkan difusi data. [2, 3, 1, 1]

[1, 2, 3, 1]

[1, 1, 2, 3]

[3, 1, 1, 2]

* 1. *AddRoundkey* (W36-W39)

Dilakukan setelah *MixColums* untuk menambahkan lapisan keamanan dengan kunci ronde tambahan (W36-W39).

1. Pada ronde terakhir, hanya dilakukan tiga langkah yaitu

*SubBytes, ShiftRows*, dan *AddRoundkey* tanpa *MixColumns.*

Proses modifikasi algoritma AES ini digunakan untuk mengenkripsi *file* dengan menambahkan langkah *AddRoundkey* tambahan, yang meningkatkan keamanan dengan memberikan lapisan XOR tambahan pada setiap ronde enkripsi.

* + 1. **Implementasi Kombinasi AES Modifikasi dan ChaCha20** Algoritma AES modifikasi dan ChaCha20 dengan menerapkan *double encypt.* Proses ini memakai AES modifikasi dengan memberikan perlindungan berbasis blok *chiper* yang kuat dengan memodifikasi *AddRoundkey* tambahan sebelum *MixColumns,* sedangkan ChaCha20 sebagai *stream cipher* yang cepat dan aman, meningkatkan kompleksitas keseluruhan dengan memanfaatkan proses enkripsi berbasis *keystream.*

Pada metode ini, setiap *file* menggunakan 1 kunci enkripsi. Kunci yang digunakan memiliki panjang awal 128-bit, yang kemudian diperluas menjadi 256-bit pada ChaCha20 dengan menambahkan padding nol (0x00) jika panjangnya kurang dari 256-bit. Proses

dimulai dengan mengenkripsi *file* menggunakan AES Modifikasi untuk menghasilkan cipherteks awal dalam format matriks 4x4. Cipherteks ini kemudian diubah menjadi byte stream untuk kompatibilitas dengan ChaCha20. Selanjutnya, ChaCha20 melakukan proses enkripsi kedua dengan memanfaatkan kunci 256- bit, nonce sepanjang 96-bit, dan counter. Hasil akhir adalah cipherteks final yang lebih aman karena telah melalui dua lapisan enkripsi. Penggunaan double encryption dengan satu kunci untuk setiap *file* memberikan keamanan yang lebih tinggi, di mana kombinasi kedua algoritma menciptakan perlindungan yang kuat terhadap berbagai jenis serangan kriptografi.

*Gambar III.3 Diagram Proses Double Encrypt*

Pada [Gambar III.3 Diagram Proses Double Encrypt](#_bookmark50) menunjukkan langkah-langkah enkripsi ganda. Proses dimulai dengan mengenkripsi *plainteks* menggunakan AES modifikasi untuk menghasilkan *chiperteks* AES. *Chiperteks* ini kemudian dienkripsi ulang menggunakan ChaCha20, yang melibatkan matriks 4x4, proses *quarter-round,* dan operasi XOR dengan *keystream*. Algoritma AES modifikasi dan ChaCha20 menggunakan kunci yang sama, yaitu kunci AES sepanjang 128-bit. berikut adalah tahapan alur di atas :

1. Input *plainteks* dan *key* dan enkripsi AES modifikasi
2. Input *cipherteks* AES ke ChaCha20

*Chiperteks* yang dihasilkan dari proses enkripsi pertama diubah menjadi input untuk algoritma ChaCha20. Ini dilakukan untuk menerapkan enkripsi berlapis yaitu *double encryption,* sehingga keamanan data akan meningkat secara signifikan. *Chiperteks* AES yang biasanya dalam format matriks (4x4) ini diubah ke format *byte stream* untuk kompatibilitas dengan algoritma ChaCha20. Byte stream ini memungkinkan untuk memproses data, sesuai dengan *stream chiper*.

Contoh :

[115, 75, 189, 234, 201, 97, 58, 14, 245, 142, 12, 120, 53, 190,

201, 20] data ini sudah dalam bentuk *byte stream*.

1. Validasi panjang *key* (256-bit *Key*)

Ini untuk memastikan bahwa kunci enkripsi untuk ChaCha20 sesuai dengan spesifikasi algoritma, yaitu panjangnya harus tepat 256 bit (32 *byte*). Kunci ini untuk menghasilkan *keystream* yang beda dalam proses enkripsi. Jika panjang ke kurang dari 256 bit, ditambahkan *padding* berupa *byte* nol (0x00) untuk memenuhi panjang. Jika *key* yang lebih panjang, *key* akan dipotong hingga hanya menyisakan 256 bit. *Key* kemudian dibagi menjadi 8 bagian ( di mana masing-masing 32 bit) untuk digunakan dalam proses pembentukan matriks awal (*state* matriks)

Contoh :

* 1. *Key* input “surabayaittelkom”

[115, 117, 114, 97, 98, 97, 121, 97, 105, 116, 116, 101, 108,

107, 111, 109]

* 1. Panjang *key* adalah 16 *byte* (128 bit), sehingga tidak memenuhi syarat panjang 256 bit. Jadi ditambahkan *padding* nol (0x00) hingga mencapai 32 *byte:*

[115, 117, 114, 97, 98, 97, 121, 97, 105, 116, 116, 101, 108,

107, 111, 109, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

1. Membuat *Initial* State Matriks 4x4

Membentuk *state* matriks awal yang menjadi dasar operasi dalam ChaCha20. Matriks ini berisi kunci, *counter*, dan *nonce.* Matriks 4x4 dalam ChaCha20 terdiri 16 elemen (masing-masing 32 bit) Contoh :

* 1. 4 elemen konstanta tetap “*expand* 32-byte k”4 elemen 32 bit [1634760805, 857760878, 2036477234, 1797285236]
  2. Elemen kunci 256-bit dibagi menjadi 8 elemen 32-bit: [1936028945, 1885434721, 1701147239, 1647539057,

1225269481, 1226844037, 1214606441, 0]

* 1. 1 *Counter*: Nilai awal 0.
  2. 3 Elemen Nonce: Nilai *nonce* yang unik untuk sesi enkripsi: Contoh *nonce* : [12345678, 87654321, 11223344]
  3. Matriks yang dibentuk

[1634760805, 857760878, 2036477234, 1797285236]

[1936028945, 1885434721, 1701147239, 1647539057]

[1225269481, 1226844037, 1214606441, 0]

[0, 12345678, 87654321, 11223344]

Baris ke 1 konstanta, ke 2 *key,* ke 3 *key* dan ke 4 *counter* dan

*nonce*

1. Operasi Quarter Round (20 iterasi)

Melakukan proses pencampuran data di dalam matriks untuk meningkatkan kerandoman dan keamanan *keystream* yang dihasilkan. Quarter Round adalah inti dari algoritma ChaCha20. Quarter Round dilakukan pada 4 elemen tertentu dari matriks. Penjumlahan Modular (mod 2³²): Menambah elemen dalam matriks untuk mencampur data. Operasi XOR: Mencampur data antar elemen melalui operasi logika XOR. Rotasi Bit (ROTL): Menggeser bit elemen untuk mendistribusikan informasi secara acak. Setelah 20 iterasi (10 putaran double round), matriks akan diubah menjadi bentuk yang sangat acak.

1. *Counter* +=1

*Counter* bertambah untuk setiap blok data yang diproses, memastikan *keystream* yang dihasilkan unik untuk setiap blok. Setelah satu blok data selesai diproses, *counter* di matriks bertambah sebesar 1.

Contoh :

*Counter* baru : [1, 12345678, 87654321, 11223344]

1. XOR *keystream* dengan *chiperteks* AES

Menggabungkan *keystream* hasil ChaCha20 dengan *cipherteks AES* melalui operasi XOR untuk menghasilkan *final ciphertext*. XOR dipilih karena sifatnya yang sederhana namun aman dan mudah dibalik. *Keystream* yang dihasilkan dari matriks ChaCha20 di-XOR-kan dengan *cipherteks AES*.

Operasi XOR:

Jika *ciphertext* = *keystream* XOR *plaintext*, maka *dekripsi*

dilakukan dengan cara *plaintext* = *keystream* XOR *ciphertext*.

* + - 1. **Proses *Dekripsi* Double Encryption**

Proses *dekripsi* dilakukan untuk mengembalikan data yang telah terenkripsi ke bentuk aslinya atau *plainteks*. Di mana pada tahap ini, *chiperteks* hasil dari kombinasi algoritma AES modifikasi dan Chacha20 akan *didekrpsi* secara urut. Proses nantinya akan dimulai dengan *mendekripsi chiperteks* dengan ChaCha20 untuk mendapatkan kembali *chiperteks* AES modifikasi, kemudian dilanjutkan dengan *mendekripsi chiprteks* AES modifikasi untuk memperoleh *plainteks* asli.

Pada [Gambar III.4 Proses Dekripsi](#_bookmark51) Diagram alur di atas menjelaskan proses *dekripsi* dari kombinasi algoritma ChaCha20 dan modifikasi AES. Proses dimulai dengan memasukkan cipherteks hasil enkripsi sebagai input. Langkah pertama adalah men*dekripsi* cipherteks menggunakan algoritma ChaCha20 untuk menghasilkan keystream. Keystream ini kemudian dikombinasikan dengan cipherteks menggunakan operasi XOR untuk mendapatkan cipherteks AES.

Selanjutnya, proses ekspansi kunci dilakukan untuk menghasilkan kunci yang akan digunakan pada setiap ronde *dekripsi* AES. Proses *dekripsi* AES terdiri dari tiga tahapan utama: ronde terakhir (Round 10) yang melibatkan *AddRoundKey*, InvShiftRows, dan InvSubBytes; ronde utama (Round 9-1) yang melibatkan *AddRoundKey*, InvMixColumns, *AddRoundKey, InvShiftRows,* dan *InvSubBytes* tambahan; dan ronde awal (Round 0) yang menggunakan *AddRoundKey* (W0-W3). Hasil akhir dari seluruh proses ini adalah plainteks, yaitu data asli yang berhasil di*dekripsi*. Kombinasi kedua algoritma ini memastikan keamanan data yang lebih kuat melalui enkripsi ganda.

*Gambar III.4 Proses Dekripsi*

### Implementasi Antarmuka Pengguna (UI)

Antarmuka pengguna (UI) dirancang untuk mempermudah interaksi, berfungsi sebagai media interaktif yang memungkinkan pengguna untuk memilih file, memasukkan kunci enkripsi, serta memulai proses enkripsi dan dekripsi dengan mudah. Implementasinya menggunakan Flask sebagai framework backend untuk menangani

proses enkripsi dan dekripsi, sedangkan HTML digunakan untuk membuat form input file dan kunci enkripsi. CSS digunakan untuk mempercantik tampilan agar antarmuka lebih menarik, jelas, dan mudah digunakan oleh pengguna dalam proses enkripsi dan dekripsi file menggunakan kombinasi algoritma ChaCha20 dan AES yang telah dimodifikasi.

*Gambar III.5 Desain Antarmuka Pengguna Enkripsi dan Dekripsi*

Pada [Gambar III.5 Desain Antarmuka Pengguna Enkripsi dan](#_bookmark53) [Dekripsi](#_bookmark53) Gambar tersebut menampilkan desain antarmuka pengguna (UI) dari sebuah aplikasi berbasis web yang digunakan untuk enkripsi dan *dekripsi file*. Antarmuka ini memiliki beberapa elemen input pengguna untuk mengunggah *file* dan memilih opsi enkripsi atau *dekripsi*. Terdapat tiga kolom utama pada desain ini: pertama, Pilih *File*, untuk pengguna memilih *file* yang akan diproses. Kedua, Enkripsi atau *Dekripsi*, yang memberikan pilihan kepada pengguna apakah *file* yang dipilih akan dienkripsi atau di *dekripsi*, yang merupakan opsi penting untuk menentukan jenis operasi yang akan dilakukan pada *file* tersebut. Ketiga, Key, di mana pengguna diminta untuk memasukkan kunci yang diperlukan untuk proses enkripsi atau *dekripsi*. Kunci ini memainkan peran penting dalam menjaga keamanan data yang diproses. Di bagian bawah antarmuka, terdapat

tombol Proses yang akan menjalankan operasi sesuai dengan pilihan pengguna, baik itu enkripsi maupun *dekripsi*.

### Pengujian Algoritma

Menjelaskan pengujian yang dilakukan untuk mengevaluasi kinerja dan keamanan algoritma berdasarkan parameter seperti entropi, waktu enkripsi dan *dekripsi*, serta kecapatan.

### Pengujian Entropi

Pengujian kekuatan algoritma dilakukan dengan mengukur tingkat keacakan data hasil enkripsi menggunakan analisis entropi. Keacakan data merupakan indikator penting dalam menentukan seberapa baik algoritma enkripsi menyamarkan pola asli dari data. Semakin tinggi tingkat keacakan, semakin sulit bagi pihak yang tidak berwenang untuk menganalisis pola atau memprediksi isi data terenkripsi. Pengujian ini dilakukan dengan menghitung nilai entropi menggunakan distribusi probabilitas setiap simbol dalam cipherteks. Nilai entropi yang mendekati maksimum menunjukkan bahwa distribusi simbol dalam cipherteks hampir merata, menandakan bahwa algoritma telah berhasil menghasilkan data yang acak. Data dengan keacakan tinggi lebih tahan terhadap serangan berbasis statistik, sehingga pengujian ini memberikan evaluasi kuantitatif terhadap kekuatan keamanan algoritma yang diuji. Hasil pengujian dibandingkan antara AES Standar, AES Modifikasi, Chacha20 dan Double Encryption untuk menunjukkan efektivitas modifikasi dalam meningkatkan keamanan.

### Pengujian Waktu dan Kecepatan

Menguji algoritma AES modifikasi dan *Double encrypt* AES modifikasi dan ChaCha20. Pengujian ini dilakukan untuk menilai kinerja algoritma dalam berbagai parameter. Penelitian ini akan menganalisis parameter pengujian menggunakan matriks dan formula berikut :

* + - * 1. Waktu enkripsi :

mengukur waktu yang dibutuhkan untuk mengenkripsi data dari

*plainteks* menjadi *chiperteks* dijelaskan dalam persamaan [(III-1)](#_bookmark55).

|  |  |
| --- | --- |
| 𝐸𝑡 = 𝐸𝑁𝑡 – 𝑆𝑇𝑡 | (III-1) |

Di mana:

𝐸𝑡 = waktu enkripsi dalam mili detik (ms)

𝐸𝑁𝑡 = waktu selesai enkripsi

𝑆𝑇𝑡 = waktu mulai enkripsi

Waktu ini akan dihitung untuk setiap *file* yang diuji, dan hasilnya nanti akan memberikan gambaran mengenai kecepatan algoritma dalam melakukan proses enkripsi.

* + - * 1. Waktu *dekripsi* :

Mengukur waktu yang dibutuhkan untuk *mendekripsi* data dari

*chiperteks* menjadi *plainteks* dijelaskan pada persamaan [(III-2)](#_bookmark56).

Dimana:

𝐷𝑡 = waktu *dekripsi* (ms)

𝐸𝑁𝑡 = waktu selesai *dekripsi*

𝑆𝑇𝑡 = waktu mulai *dekripsi*

Hasil waktu *dekripsi* ini juga perlu untuk masing-masing algoritma karena ini menganalisis perbandingan kinerja antara algoritma enkripsi.

* + - * 1. Kecepatan :

Menunjukkan jumlah data dalam *kilobyte* yang dapat dienkripsi

Et = waktu enkripsi atau *dekripsi* (s)

### Pengujian pada Berbagai Jenis *File*

*File* PDF yang digunakan dalam pengujian dapat berisi elemen seperti teks, gambar, dan tabel yang semuanya akan dienkripsi secara utuh karena algoritma bekerja pada level data biner. *File* PDF dengan ukuran 100 KB, 1 MB, 10 MB, hingga 100 MB diuji untuk memastikan algoritma dapat bekerja pada dokumen berukuran kecil hingga besar. Hal serupa juga berlaku untuk *file* Word dengan ukuran yang sama, meskipun memiliki struktur internal yang lebih kompleks dibandingkan PDF. *File* TXT dengan ukuran maksimal 300 *byte*, yang biasanya hanya berisi teks sederhana, juga dapat dienkripsi ChaCha20 (*double encrypt*). Di mana pengujian ini bertujuan untuk

mengukur kinerja algoritma. Berikut ini adalah contoh hasil pengujian terhadap *file* PDF :

*Tabel III.2 Evaluasi Hasil*

Pada [Tabel III.2 Evaluasi Hasil](#_bookmark60) menunjukkan hasil evaluasi simulasi kinerja algoritma enkripsi dan *dekripsi*. pada Nilai-nilai tersebut memberikan gambaran awal mengenai performa beberapa metode, termasuk AES Standar, ChaCha20, AES Modifikasi, dan kombinasi AES Modifikasi dengan ChaCha20. Hasil ini digunakan sebagai contoh output yang diharapkan dari model algoritma yang akan dirancang, ini belum merupakan data dari eksperimen nyata.

# DAFTAR PUSTAKA

Alimuddin, A., Juntak, J. N. S., Jusnita, R. A. E., Murniawaty, I., & Wono, H. Y. (2023). Teknologi dalam pendidikan: Membantu siswa beradaptasi dengan revolusi industri 4.0. *Journal on Education*, *5*(4), 11777–11790.

Almukhlifi, R., & Vora, P. L. (2020). Linear cryptanalysis of reduced-round simon using super rounds. *Cryptography*, *4*(1), 1–34. https://doi.org/10.3390/cryptography4010009

Alyas, H. H., & Abdullah, A. A. (2021). Enhancement the ChaCha20 Encryption Algorithm Based on Chaotic Maps. *Lecture Notes in Networks and Systems*, *201*, 91–107. https://doi.org/10.1007/978-981-16-0666-3\_10

Anindita, M., & 18219086, A. A. (2023). *Analisis Perbandingan Algoritma AES dan ChaCha20-Poly1305 dalam Enkripsi Konten Livestreaming*.

Aryanto, M. B., Tahir, M., Devita, S. I., Mustofa, Z. N., Ainiyah, Q., & Sundoro,

S. (2023). Implementasi Enkrip Dan Dekrip File Menggunakan Metode Advance Encryption Standard (AES-128). *Jurnal Ilmiah Sistem Informasi Dan Ilmu Komputer*, *3*(1), 89–104.

Assyahid, M. H. (2024). *Implementasi interpolasi pixel wadah citra dan enkripsi aes pada steganografi least significan bit (LSB)*. https://repository.uinjkt.ac.id/dspace/handle/123456789/76727

Barbero, S., Bazzanella, D., & Bellini, E. (2022). Rotational Cryptanalysis on ChaCha Stream Cipher. *Symmetry*, *14*(6). https://doi.org/10.3390/sym14061087

Belvin Tannadi. (2022). *Ilmu Crypto* (Digital). Jakarta : Elex Media Komputindo.

Cristy, N., & Riandari, F. (2021). *Niolinda Cristy 1 , Fristi Riandari 2 [Implementasi Metode Advanced Encryption Standard (AES 128 Bit) Untuk Mengamankan Data Keuangan*. *4*(2), 75.

Djong, H. S., & Siswanto, S. (2022). Implementasi Kriptografi Dengan Menggunakan Metode RC4 dan AES-256 Untuk Mengamankan File Dokumen pada PT Varnion Technology Semesta. *Prosiding Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI)*, *1*(1), 149–158.

Handoyo, J., & Subakti, Y. M. (2020). Keamanan Dokumen Menggunakan Algoritma Advanced Encryption Standard (AES). *Jurnal SITECH : Sistem Informasi Dan Teknologi*, *3*(2), 143–152. https://jurnal.umk.ac.id/index.php/sitech/article/view/5865

Hena, N., & Kusumaningsih, D. (2024). Pengamanan File Pada Sistem Master Vendor Berbasis WEB Menggunakan Algoritma AES Pada Trinitiland. *Prosiding Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI)*, *3*(2), 30–37.

Hidayatulloh, N. W., Tahir, M., Amalia, H., Basyar, N. A., Prianggara, A. F., & Yasin, M. (2023). Mengenal Advance Encryption Standard (AES) sebagai Algoritma Kriptografi dalam Mengamankan Data. *Digital Transformation Technology*, *3*(1), 1–10.

Kebande, V. R. (2023). Extended-Chacha20 Stream Cipher With Enhanced Quarter Round Function. *IEEE Access*, *11*, 114220–114237. https://doi.org/10.1109/ACCESS.2023.3324612

Kim, B. G., & Wong, D. (2023). Smart Contract-based Secure Verifiable Random Function using ChaCha20 Sequence in Blockchain. *ACM International Conference Proceeding Series*, 41–51. https://doi.org/10.1145/3638025.3638028

Kurnia Mahesa Naibaho, A., Putri, A., & Fikri Husaini, Y. (2021). Analisis Sistem Keamanan Informasi Chat Online dengan Menggunakan Algoritma AES. In *Ilmu Komputer* (Vol. 6, Issue 2).

Luong, T. T. (2023). Strengthening AES Security Through Key-Dependent ShiftRow and AddRoundKey Transformations Utilizing Permutation. In *IJACSA) International Journal of Advanced Computer Science and Applications* (Vol. 14, Issue 11). [www.ijacsa.thesai.org](http://www.ijacsa.thesai.org/)

Muhammed, R. K., Rashid Aziz, R., Hassan, A. A., Aladdin, A. M., Saydah, S. J., Rashid, T. A., & Hassan, A. (2024). *Comparative Analysis of AES, Blowfish, Twofish, Salsa20, and ChaCha20 for Image Encryption*.

Olivia Putri Irine Irawan, B., Tahir, M., Ayu Windrastuti, N., Yurina Cholili, D., Mulaikah, D., Batsul Mushofi Septian wachid, A., & Pendidikan Informatika,

P. (2023). Implementasi Kriptografi Pada Keamanan Data Menggunakan Algoritma Advance Encryption Standard (AES). *Jurnal Simantec*, *11*(2), 167–

174. https://doi.org/10.21107/SIMANTEC.V11I2.20034

Pangestu, G. Y., Hadiana, A. I., & Sabrina, P. N. (2022). Kriptografi Untuk Enkripsi Ganda Pada Gambar Menggunakan Algoritma AES (Advanced Encryption Standard) Dan RC5 (Rivest Code 5). *Vol*, *1*, 25–32.

Putri, A. E., Kartikadewi, A., & Rosyid, L. A. A. (2021). Implementasi Kriptografi dengan Algoritma Advanced Encryption Standard (AES) 128 Bit dan Steganografi menggunakan Metode End of File (EOF) Berbasis Java Desktop

pada Dinas Pendidikan Kabupaten Tangerang. *Appl. Inf. Syst. Manag*, *3*(2), 69–78.

Saripa, S. (2023). Implementation Implementation of a File Security System Using the AES Algorithm to Secure Personal Files: Implementasi Sistem Keamanan File Menggunakan Algoritma AES untuk Mengamankan File Pribadi. *Progressive Information, Security, Computer, and Embedded System*, *1*(2), 138–148.

Sekar Putri Ananda, Saepul Lukman, & Irfan. (2022). *Analisa Metode Kriptogra Modern Advance Encryption Standar (AES) 128 Bit dalam Mengenkripsi dan Mendekripsi File Dokumen Digital*. https://doi.org/10.32409/jikstik.21.3.2973

Soesanto, E., Purba, L. M., Aprilia, B., Putra, D. R., & Putri, S. D. (2023). Implementasi Objek Vital, Pengamanan File dan Pengamanan Cyber di PT Pertamina. *IJM: Indonesian Journal of Multidisciplinary*, *1*(1).

Surbakti, P. S., & Purwanto, P. (2024). Implementasi Kriptografi Dengan Menggunakan Algoritma AES-128 Untuk Enkripsi dan Dekripsi File Dokumen Berbasis Web Pada Law Office Erdi Surbakti, SH & Rekan. *Prosiding Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI)*, *3*(1), 1–8.

Syahputra Sianipar, J., Budi Nuugroho, N., Mariami, I., Informasi, S., & Triguna Dharma, S. (2024). Pengamanan Data Gaji Karyawan Dengan Menggunakan Metode Advanced Encryption Standard (AES). *Jurnal Sistem Informasi Triguna Dharma (JURSI TGD)*, *3*(1), 35–45. https://doi.org/10.53513/JURSI.V3I1.5653

Wahyu Putra, M. R. F. T. H. (2023, July 2). *Implementasi Algoritma Advanced Encryption Standard Untuk Kemanan Dokumen*. https://journal.grahamitra.id/index.php/jurikti/article/view/55/181

Wahyudi, E. N., Ardhianto, E., Handoko, W. T., Murti, H., Supriyanto, E., Lestariningsih, E., & Redjeki, R. S. (2024). Peningkatan Keamanan Data Melalui Teknik Super Enkripsi Menggunakan Algoritma Vigenere dan Caesar. *Jurnal Informatika Polinema*, *10*(3), 315–322. https://doi.org/10.33795/JIP.V10I3.5131

Widyawan, D., & Imelda, I. (2021). Pengamanan File Menggunakan Kriptografi Dengan Metode Aes-128 Berbasis Web Di Komite Nasional Keselamatan Transportasi. *SKANIKA: Sistem Komputer Dan Teknik Informatika*, *4*(1), 15– 22.

**IMPLEMENTASI KOMBINASI CHACHA20 DAN MODIFIKASI AES MELALUI *ADDROUNDKEY* PADA**

**STUDI KASUS KEAMANAN *FILE***

**Proposal Tugas Akhir Mata Kuliah Penulisan Proposal**

## (CAK4EAB2)

Disusun oleh:

## EMILIA RAMONA 1203210017

**PROGRAM STUDI SARJANA INFORMATIKA DIREKTORAT KAMPUS SURABAYA**

**UNIVERSITAS TELKOM SURABAYA TAHUN 2025**

# ABSTRAK

Keamanan data menjadi tantangan besar di era Revolusi Industri 4.0, di mana informasi penting perlu dilindungi dari akses tidak sah. Ancaman serangan siber dan peretasan mengharuskan pengembangan solusi keamanan yang lebih efektif. Penelitian ini bertujuan untuk mengimplementasikan dan mengevaluasi kombinasi algoritma ChaCha20 dan modifikasi *Advanced Encryption* Standard (AES) dengan penambahan langkah *AddRoundKey* sebelum *MixColumns* untuk meningkatkan keamanan *file*. Metode yang digunakan adalah enkripsi ganda, dengan AES modifikasi sebagai *block cipher* yang kuat dan ChaCha20 sebagai *stream cipher* yang cepat. Penelitian ini mencakup implementasi algoritma dengan menggunakan pustaka kriptografi untuk menguji kinerja algoritma melalui parameter seperti waktu enkripsi, waktu *dekripsi*, kecepatan enkripsi, kecepatan *dekripsi*, dan tingkat keacakan data menggunakan analisis entropi. *File* dengan berbagai format dan ukuran, PDF, Word, dan TXT, digunakan sebagai data pengujian untuk mengevaluasi efisiensi algoritma. Selain itu, antarmuka pengguna berbasis web dirancang untuk mempermudah proses enkripsi dan *dekripsi*, melibatkan langkah- langkah seperti pemilihan *file* dan *input* kunci enkripsi. Penelitian ini diharapkan menunjukkan bahwa kombinasi AES modifikasi dan ChaCha20 mampu meningkatkan keamanan data serta memberikan efisiensi yang lebih baik dalam perlindungan *file*.

**Kata Kunci**: Keamanan *File*, *Advanced Encryption Standard* (AES), ChaCha20, Enkripsi Ganda, *AddRoundkey*, *Entropi*.

# ABSTRACT

*Data security has become a major challenge in the era of the Industrial Revolution 4.0, where critical information must be protected from unauthorized access. Cyberattacks and hacking threats demand the development of more effective security solutions. This research aims to implement and evaluate the combination of the ChaCha20 algorithm and a modified Advanced Encryption Standard (AES) by adding an AddRoundKey step before MixColumns to enhance file security. The method used is double encryption, with the modified AES as a robust block cipher and ChaCha20 as a fast stream cipher. This study involves implementing the algorithms using cryptographic libraries to test their performance through parameters such as encryption time, decryption time, encryption speed, decryption speed, and data randomness level using entropy analysis. Files of various formats and sizes, including PDF, Word, and TXT, are utilized as test data to evaluate the efficiency of the algorithms. Additionally, a web-based user interface is designed to simplify the encryption and decryption process, involving steps such as file selection and encryption key input. This research is expected to demonstrate that the combination of modified AES and ChaCha20 can enhance data security and provide better efficiency in file protection.*

***Keywords****: File Security, Advanced Encryption Standard (AES), ChaCha20, Double Encryption, AddRoundkey, Entropy*

# KATA PENGANTAR

Assalamu’alaikum Warrahmatullahi Wabarakatuh, puja puji syukur ke hadirat Allah SWT yang telah memberikan nikmat dan karunia-Nya yang sangat luar biasa kepada hamba – hambanya. Tidak lupa shalawat serta salam kita curahkan atas bimbingan nabi kita nabi Muhammad SAW yang telah membawa cahaya islam ke dalam dunia ini dan menjadi panutan bagi setiap umatnya hingga akhir zaman.

Setelah melalui berbagai proses kerja keras, penyusunan skripsi yang berjudul “Implementasi Kombinasi ChaCha20 dan Modifikasi AES melalui *AddRoundkey* Pada Studi Kasus Keamanan *File*” akhirnya dapat terselesaikan. Selain sebagai salah satu syarat memperoleh gelar S-Kom, penulis berharap juga skripsi ini dapat bermanfaat bagi semua pihak yang berkepentingan. Oleh karena itu, penulis ingin mengungkapkan rasa terima kasih kepada pihak-pihak yang telah memberikan dukungan dan doa sehingga penulis dapat menyelesaikan skripsi ini tepat pada waktunya. Beberapa pihak tersebut antara lain:

1. **Kepada Mama** yang sangat penulis cintai yang tidak pernah lupa selalu memberikan dukungannya dan doa agar diberikan kemudahan dan kelancaran dalam penyusunan skripsi ini sehingga penulis dapat menyelesaikan studi Informatika di Universitas IT Telkom Surabaya.
2. **Kepada Alm.Ibu Suhartatik**, yang penulis cintai dan sayangi, yang telah memberikan motivasi untuk saya.
3. **Kepada Saudara Kembar**, yang selalu mendukung, terima kasih atas bantuan dan waktunya dalam memberikan masukan terkait susunan skripsi ini.
4. **Keluarga Besar penulis** yang telah memberikan motivasi dan kepercayaan besar sehingga penulis bisa mewujudkan impian untuk meraih gelar S-Kom.
5. **Dosen Pembimbing**, Pak Rizky Fenaldo Maulana, S.Kom., M.Kom. dan Tanzila Mustaqim, S.Kom., M.Kom. Yang telah memberikan solusi, waktu, dan bimbingan penuh kesabaran selama proses penyusunan skripsi ini.
6. **Teman-teman dekat**, termasuk Jus’an, Erica Dyah, Qori Emalia, Ardina Firdatana, dan Elma Saskia yang selalu memberikan semangat, mendengarkan keluh kesah, serta menemani dalam berbagai kondisi selama penulisan skripsi.
7. **Semua pihak lainnya**, yang tidak dapat disebutkan satu per satu, atas bantuan dan dukungannya selama proses ini.
8. **Diri penulis sendiri**, atas perjuangan dan kerja keras hingga akhirnya skripsi ini selesai tepat waktu.

Penulis menyadari bahwa skripsi ini masih jauh dari kesempurnaan. Oleh karena itu, saran dan kritik yang membangun sangat diharapkan untuk memperbaiki skripsi ini ke depannya.

Terima kasih.

### Wassalamu’alaikum Warrahmatullahi Wabarakatuh.

Surabaya, 24 Desember 2024 Penulis

Emilia Ramona

# DAFTAR ISI

[LEMBAR PERSETUJUAN ii](#_bookmark0)

[ABSTRAK iii](#_bookmark1)

[ABSTRACT iv](#_bookmark2)

[KATA PENGANTAR v](#_bookmark3)

[DAFTAR ISI vii](#_bookmark4)

[DAFTAR TABEL ix](#_bookmark5)

[DAFTAR GAMBAR x](#_bookmark6)

[DAFTAR RUMUS xi](#_bookmark7)

[BAB I PENDAHULUAN 1](#_bookmark8)

* 1. [Latar Belakang 1](#_bookmark9)
  2. [Rumusan Masalah 4](#_bookmark10)
  3. [Tujuan Penelitian 4](#_bookmark11)
  4. [Batasan dan Asumsi Penelitian 5](#_bookmark12)
  5. [Manfaat Penelitian 5](#_bookmark13)
     1. [Manfaat Akademis 5](#_bookmark14)
     2. [Manfaat Ilmiah 5](#_bookmark15)
  6. [Sistematika Penulisan 6](#_bookmark16)

[BAB II LANDASAN TEORI 7](#_bookmark17)

* 1. [Literatur Terkait Teori 7](#_bookmark18)
  2. [Kajian Teori 10](#_bookmark20)
     1. [Kriptografi 10](#_bookmark21)
     2. [AES (Advanced Encryption Standard) 12](#_bookmark24)
     3. [Chacha20 20](#_bookmark32)
     4. [Entropi 22](#_bookmark35)

[BAB III METODOLOGI PENELITIAN 24](#_bookmark37)

* 1. [Sistematika Penyelesaian Masalah 24](#_bookmark38)
  2. [Alat dan Bahan 24](#_bookmark39)
     1. [Alat 24](#_bookmark40)
     2. [Bahan 25](#_bookmark41)
  3. [Rancangan Penelitian 25](#_bookmark42)
     1. [Studi Literatur 27](#_bookmark45)
     2. [Pengumpulan Data 27](#_bookmark46)
     3. [Implementasi Modifikasi AES 28](#_bookmark47)
     4. [Implementasi Kombinasi AES Modifikasi dan ChaCha20 31](#_bookmark49)
     5. [Implementasi Antarmuka Pengguna (UI) 36](#_bookmark52)
     6. [Pengujian Algoritma 38](#_bookmark54)
     7. [Analisis Hasil 42](#_bookmark59)

[DAFTAR PUSTAKA 44](#_bookmark61)

# DAFTAR TABEL

[Tabel II.1 Literatur Penelitian Terkait Kriptografi 7](#_bookmark19)

[Tabel II.2 Panjang Kunci 13](#_bookmark25)

[Tabel II.3 Tabel Substitusi (S-Box) 16](#_bookmark28)

[Tabel II.4 Transformasi ShiftRows 17](#_bookmark29)

[Tabel III.1 Jenis, Ukuran, dan Isi File untuk Pengujian 40](#_bookmark58)

[Tabel III.2 Evaluasi Hasil 43](#_bookmark60)

# DAFTAR GAMBAR

[Gambar II.1 Alur Proses Enkripsi Dan Dekripsi 11](#_bookmark22)

[Gambar II.2 Proses Kunci Simetris 11](#_bookmark23)

[Gambar II.3.Transformasi ShiftRows 15](#_bookmark27)

[Gambar II.4 Proses AES 19](#_bookmark31)

[Gambar III.1 Diagram Alur Penelitian 26](#_bookmark43)

[Gambar III.2 Diagram Proses Enkripsi AES 28](#_bookmark48)

[Gambar III.3 Diagram Proses Double Encrypt 32](#_bookmark50)

[Gambar III.4 Proses Dekripsi 36](#_bookmark51)

[Gambar III.5 Desain Antarmuka Pengguna Enkripsi dan Dekripsi 37](#_bookmark53)

# DAFTAR RUMUS

[(II-1) Rumus AddRoundKey 14](#_bookmark26)

[(II-2) Operasi Perkalian MixColumns 17](#_bookmark30)

[(II-3) Rumus Keystream 22](#_bookmark33)

[(II-4) Rumus Proses Enkripsi XOR pada Keystream 22](#_bookmark34)

[(II-5) Entropi 22](#_bookmark36)

[(III-1) Pengujian Waktu Enkripsi 39](#_bookmark55)

[(III-2) Pengujian Waktu Dekripsi 39](#_bookmark56)

[(III-3) Pengujian Waktu Kecepatan 39](#_bookmark57)

# BAB I PENDAHULUAN

### Latar Belakang

Semakin meningkatnya teknologi yang semakin canggih membawa sebuah perubahan yang cukup signifikan. Perubahan tersebut dikenal dengan istilah Revolusi Industri 4.0 yang merupakan sebuah kemajuan teknologi yang sering digunakan oleh banyak orang. Revolusi industri 4.0 adalah sebuah data yang digerakkan dengan adanya campur tangan komputer menggunakan metode machine learning atau AI (Alimuddin et al., 2023). Menurut Klaus Schawab revolusi Industri 4.0 memiliki tantangan yang harus dihadapi salah satunya seperti keamanan *cyber* dan privasi data.

Maka dari itu, dengan semakin pesatnya informasi teknologi maka perlunya untuk mengamankan data atau *file*. Keamanan data atau *file* merupakan salah satu langkah yang mudah untuk melindungi sebuah informasi bersifat penting yang berasal dari akses tidak sah. Keamanan *file* ini sangat diperlukan pada era Revolusi Industri 4.0, karena maraknya serangan *cyber* dan peretasan komputer yang bisa mengganggu keberlangsungan suatu industri. Menurut (Mulyana*, 2016*), keamanan *file* merupakan salah satu aksi pencegahan dari sebuah penganggu pengguna *file* atau pengguna yang memakai jaringan yang tidak berkewajiban memiliki akses (Soesanto et al., 2023). Dengan semakin pesatnya dunia teknologi ini membuat keamanan *file* menjadi sangat penting karena untuk melindungi privasi dan informasi dari pihak yang tidak memiliki akses.

Keamanan *file* adalah serangkaian metode atau teknik yang untuk melindungi *file* dari akses yang tidak sah, seperti pencurian atau kerusakan. Tujuannya untuk menjaga informasi dalam *file* tetap rahasia, aman, dan hanya dapat di akses oleh pihak yang berwenang. Oleh karena itu, upaya untuk memastikan agar *file* terjaga dengan menggunakan metode ChaCha20 dan *Advanced Encryption Standard* (AES). ChaCha20 merupakan metode enkripsi yang cepat dan aman. ChaCha20 dipopulerkan oleh Daniel J.

Bernstein sebagai varian dari algoritma. Dalam algoritma ChaCha20 sangat populer dalam pengimplementasi enkripsi dikarenakan percampuran antara kecepatan dan keamanannya yang dapat diunggulkan. Chaca20 mempunyai desain yang hampir sama dengan Salsa20, pada algoritma ChaCha20 menggunakan kunci 256-bit yang mempunyai 8 putaran (Kebande, 2023).

Sedangkan AES (*Advanced Encryption Standard)* adalah sebuah algoritma yang melakukan transformasi terhadap blok data yang menggunakan kunci enkripsi yang sama. AES bekerja pada sistem yang melibatkan operasi matematika dengan level tingkat sulit (Saripa, 2023). Proses kerja enkripsi dasar AES adalah *subbytes* (menggunakan tabel S-Box untuk menggabungkan *byte* pada blok matriks), *shift row* (digunakan untuk menggeser baris matriks), *MixColumns* (melakukan pergeseran dari kanan ke kiri), dan *AddRoundkey* (blok di XOR kan dengan kunci yang diperluas) (Aryanto et al., 2023). Pada proses *dekripsi* AES terdapat tahap-tahapan kebalikan dari proses enkripsi. Dengan memakai kunci yang sama untuk *dekripsi*, blok data yang telah terenkripsi dapat dipulihkan kembali ke bentuk awalnya. algoritma AES adalah jenis *block chipper* yang memiliki panjang kunci 128 bit, 192 bit dan 256 bit (Surbakti & Purwanto, 2024).

Algoritma Chacha20 dan AES merupakan suatu enkripsi yang dapat di digunakan dalam menjaga keamanan *file*. Pada algoritma AES terdapat kelebihan salah satunya adalah sifat karakteristiknya yang di mana selalu ada bidang khusus untuk bilangan prima, memicu kekuatan AES(Surbakti & Purwanto, 2024) . AES juga memiliki sebuah kelemahan salah satunya yaitu algoritma kriptografi yang menggunakan kunci simetris dalam proses pengiriman dan penerimanya data. Namun, penggunaan kunci simetris ini memiliki risiko lebih tinggi terhadap kebocoran, terutama seiring waktu (Hidayatulloh et al., 2023). Pada algoritma ChaCha20 terdapat keunggulan yaitu menggunakan teknik kriptografi ARX (*addition, rotation*, XOR) yang memberikan performa lebih cepat, kompleksitas lebih rendah, dan mampu menghilangkan serangan waktu Klik atau ketuk di sini untuk memasukkan teks.

Meskipun algoritma AES dan Chacha20 telah efektif untuk menjaga peningkatan keamanan *file* dan implementasinya yang cukup rumit sehingga membuat orang yang tidak memiliki akses tidak dapat mengakses *file* tersebut. Hal ini tidak dapat kita pungkiri bahwa semakin pesatnya dunia teknologi membuat banyak orang berbondong-bondong menciptakan pembaruan yang lebih efektif. Metode *double encrypt* ini menawarkan kombinasi yang mampu mengurangi kelemahan dari setiap algoritma jika digunakan secara bersamaan, serta meningkatkan ketahanan terhadap berbagai jenis serangan kriptografi. *Double encrypt* merupakan Teknik kriptografi yang melibatkan proses enkripsi secara dua kali, yang di mana memiliki dua kunci yang berbeda untuk lebih meningkatkan keamanan. *Double encrypt* ini bertujuan untuk meningkatkan keamanan *file* lebih tinggi dibandingkan hanya 1 metode saja (Pangestu et al., 2022).

Pada penelitian terlebih dahulu yang berjudul “Implementasi Sistem Keamanan *File* Menggunakan Algoritma AES untuk Mengamankan *File* Pribadi” yang diteliti oleh Saripa Universitas Negeri Makassar. Dalam penelitian tersebut menjelaskan tentang keamanan *file* dengan menggunakan metode algoritma AES. Di mana hasil penelitian ini menunjukkan bahwa algoritma AES telah berhasil mengamankan *file* yaitu dokumen Word, PDF, Excel, dan Power Poin, *file* yang dienkripsi menjadi tidak terbaca oleh pihak yang tidak memiliki akses sehingga dapat menjaga kerahasiaan data dan AES mampu menjaga keamanan data dengan baik.

Dalam perbedaan penelitian terdahulu dan penelitian saya akan menggunakan double encrypt untuk keamanan *file* dan juga saya akan menggunakan AES dengan menambahkan metode ChaCha20 dan melakukan modifikasi pada bagian AES. Modifikasi menggunakan penambahan *AddRoundkey* sebelum dengan bertujuan agar proses enkripsi akan lebih susah untuk dipecahkan. Dalam algoritma AES, langkah *AddRoundKey* merupakan komponen krusial yang menggabungkan data dengan kunci enkripsi menggunakan operasi XOR. Penelitian (Luong, 2023) oleh menunjukkan bahwa memodifikasi transformasi *AddRoundKey*

dapat meningkatkan keamanan algoritma dengan menciptakan hubungan yang lebih kompleks antara *plainteks* dan *cipherteks*. Penelitian ini menyoroti pentingnya penyesuaian langkah *AddRoundKey* untuk meningkatkan ketahanan algoritma AES terhadap serangan kriptografi, seperti *differential cryptanalysis* dan linear *cryptanalysis*.

Dengan mendasarkan pada penelitian tersebut, modifikasi algoritma AES melalui penambahan langkah *AddRoundKey* sebelum *MixColumns* bertujuan untuk lebih memperkuat difusi data pada tahap awal enkripsi. Penambahan ini diharapkan dapat meningkatkan kerandoman hasil enkripsi dan mengurangi kemungkinan eksploitasi pola oleh pihak yang tidak berwenang, sehingga keamanan algoritma menjadi lebih optimal. Berdasarkan dari latar belakang di atas penelitian ini memiliki tujuan untuk meningkatkan keamanan *file* agar lebih sulit untuk diretas atau diakses pihak yang tidak berwenang. Oleh karena itu penelitian ini dianggap penting karena maraknya kasus peretasan data oleh oknum yang tidak memiliki akses.

### Rumusan Masalah

1. Bagaimana implementasi dari modifikasi AES dengan penambahan *AddRoundkey* sebelum *MixColumns* dalam meningkatkan keamanan *file*?
2. Bagaimana performa kombinasi antara AES modifikasi dan ChaCha20 dalam hal waktu enrkipsi, waktu *dekripsi* , kecepatan enkripsi, dan kecepatan *dekripsi*?

### Tujuan Penelitian

1. Mampu mengidentifikasi *AddRoundkey* yang akan digunakan dalam menjaga keamanan *file* dan mengimplementasi kemampuan Chacha20 dan AES yang telah dimodifikasi dalam proses menjaga keamanan *file*.
2. Mampu mengevaluasi performa algoritma ChaCha20 dan AES yang telah dimodifikasi berdasarkan waktu enkripsi, waktu *dekripsi*, kecepatan enkripsi, dan kecepatan *dekripsi*, untuk menentukan efisiensi

dan efektivitas kombinasi kedua algoritma tersebut dalam menjaga keamanan *file*

### Batasan dan Asumsi Penelitian

1. Penelitian ini berfokus pada jurnal atau artikel ilmiah dengan batasan tahun 2021-2024 yang membahas tentang :
   * Keamanan *file*
   * Algoritma ChaCha20
   * *Advanced Encryption Standard* (AES)
   * Double encryption
2. Penelitian dilakukan dengan pengujian pada *file* dengan ukuran dan format sebagai berikut :
   * Format PDF dan Word dengan variasi ukuran *file* 100 KB - 100MB.
   * Format *file* teks (TXT) dengan ukuran maksimal 300 byte Batasan ukuran *file* digunakan untuk memastikan pengujian dilakukan secara efektif tanpa mengurangi kinerja proses enkripsi dan *dekripsi*.

### Manfaat Penelitian

### Manfaat Akademis

* + - 1. Memberikan inovasi terbaru mengenai pelindungan keamanan

*file* dengan menambahkan *AddRoundkey* pada algoritma AES.

* + - 1. Memudahkan peneliti lainnya dalam kinerja *AddRoundkey* pada algoritma AES yang berfungsi untuk keamanan *file***.**

### Manfaat Ilmiah

* + - 1. Menjadi sumber acuan dan sumber referensi bagi peneliti lainnya yang akan meneliti topik ChaCha20 dan AES modifikasi, terutama dalam konteks keamanan data dan performa enkripsi.
      2. Membuktikan hipotesis apakah AES yang dimodifikasi dengan tambahan *AddRoundkey* dapat meningkatkan performa keamanan dan efisiensi dibandingkan dengan AES standart, dan sekaligus menunjukkan kinerja kombinasi ChaCha20 dan AES dalam pengamanan *file* berbagai format dan ukuran.

### Sistematika Penulisan

## BAB 1 PENDAHULUAN

Menjelaskan latar belakang penelitian, rumusan masalah, tujuan penelitian, batasam penelitian, manfaat penelitian, serta sistematika penulisan secara umum.

## BAB 2 LANDASAN TEORI

Membahas teori-teori yang relavan dengan penelitian, termasuk algoritma ChaCha20, AES, konsep kriptografi, Serta tinjauan terhadap penelitian terdahulu yang terkait.

## BAB 3 METODOLOGI PENELITIAN

Menjelaskan metode yang digunakan dalam penelitian, termasuk algoritma yang diterapkan (modifikasi AES dan ChaCha20 dengan enkripsi ganda). Diuraikan juga alat dan bahan yang digunakan, serta tahapan penelitian yang meliputi implementasi algoritma, pengujian parameter kinerja (waktu dan kecepatan enkripsi, dekripsi, serta analisis entropi), dan evaluasi hasil pengujian pada berbagai jenis file (PDF, Word, TXT).

## BAB 4 IMPLEMENTASI DAN PENGUJIAN

Menyajikan hasil implementasi algoritma yang dikembangkan, pengujian algoritma terhadap berbagai jenis *file*, serta analisis data hasil pengujian**.**

## BAB 5 EVALUASI HASIL PENGUJIAN

Menyajikan hasil implementasi dari aplikasi yang telah dirancang, pengujian fungsionalitas sistem, serta evaluasi kinerja aplikasi terhadap berbagai skenario penggunaan. Analisis data hasil pengujian juga dilakukan untuk menilai keberhasilan aplikasi dalam mencapai tujuan penelitian.

## BAB 6 KESIMPULAN DAN SARAN

Merangkum temuan utama penelitian serta memberikan saran untuk pengembangan lebih lanjut di masa depan.

# BAB II LANDASAN TEORI

### Literatur Terkait Teori

Keamanan informasi menjadi salah satu isu yang paling mendesak di era digital saat ini, di mana data sensitif sering kali menjadi target serangan siber yang beragam. Dengan meningkatnya ancaman seperti serangan *man-in-the- middle*, *eavesdropping*, dan *DDoS*, penting bagi organisasi dan individu untuk menerapkan metode enkripsi yang efektif untuk melindungi data mereka. Berbagai algoritma enkripsi telah dikembangkan untuk memenuhi kebutuhan ini, dengan *Advanced Encryption Standard* (AES) dan ChaCha20 menjadi dua di antara yang paling banyak digunakan. Penelitian-penelitian terkini menunjukkan bahwa kedua algoritma ini tidak hanya menawarkan tingkat keamanan yang tinggi, tetapi juga dapat diimplementasikan dalam berbagai konteks, mulai dari pengamanan *file* hingga enkripsi konten *livestreaming*.

Pada [Tabel II.1 Literatur Penelitian Terkait Kriptografi](#_bookmark19) menyajikan ringkasan dari beberapa penelitian yang mengeksplorasi efektivitas dan penerapan algoritma AES dan ChaCha20 dalam berbagai aplikasi. Setiap penelitian memberikan wawasan tentang tujuan, hasil, dan persamaan dalam pendekatan yang digunakan, serta menyoroti inovasi yang dilakukan untuk meningkatkan keamanan data. Dengan memahami hasil-hasil ini, kita dapat lebih menghargai perkembangan teknologi enkripsi dan aplikasinya dalam menjaga kerahasiaan serta integritas informasi di dunia yang semakin terhubung.

### Kajian Teori

### Kriptografi

Menurut Menezes, Oorschot, dan Vantone (1996), Kriptografi adalah studi teknik matematika yang berkaitan dengan keamanan informasi seperti kerahasiaan, *autentikasi* keaslian data (Belvin Tannadi, 2022). Kriptografi dapat diartikan sebagai sebuah proses untuk melindungi data atau sebuah sistem yang bertujuan untuk melindungi informasi rahasia dengan cara yang tidak dipahami oleh pihak yang tidak berwenang. Dalam kriptografi yang meliputi proses enkripsi dan proses *dekripsi*. Di dalam Kriptografi terdapat tiga fungsi dasar yang meliputi :

* + - 1. Enkripsi merupakan proses data asli (*plainteks*) yang akan diubah menjadi data yang tidak bisa terbaca oleh pihak yang tidak berwenang (*chiperteks*).
      2. *Dekripsi* merupakan proses pengembalian dari data yang tidak bisa terbaca(*chiperteks*) menjadi data terbaca atau data asli (*plainteks*).
      3. Kunci merupakan proses yang dipakai untuk melakukan proses enkripsi dan *dekripsi*, terdapat 2 jenis kunci, yaitu kunci rahasia (*private key*) dan kunci publik (*public key*).

Pada [Gambar II.1 Alur Proses Enkripsi Dan *Dekripsi*](#_bookmark22)

memperlihatkan alur enkripsi dan *dekripsi.* Proses ini dimulai

dengan *plainteks* yang kemudian dienkripsi dengan menggunakan kunci menjadi *chiperteks*. *Chiperteks* tersebut kemudian *didekripsi* kembali menggunakan kunci yang sama untuk mendapatkan kembali *plainteks*.

*Gambar II.1 Alur Proses Enkripsi Dan Dekripsi*

Berikut algoritma Kriptografi yang berdasarkan kunci :

a. Algoritma Simetris

Algoritma simetris adalah algoritma yang merujuk pada metode enkripsi dan *dekripsi* yang di mana algoritma ini memiliki kunci tunggal atau sama. Sehingga dapat disimpulkan algoritma ini mewajibkan pengirim dan penerima harus menyetujui satu kunci yang digunakan dalam proses enkripsi dan *dekripsi* data. Namun, kelemahan dari algoritma ini yaitu tergantung pada rahasia kunci. Apabila kunci tersebut tersebar maka akan dengan mudah pihak yang tidak memiliki akses dapat melakukan proses enkripsi dan *dekripsi* data secara mudah.

*Gambar II.2 Proses Kunci Simetris*

Pada [Gambar II.2 Proses Kunci Simetris](#_bookmark23) menjelaskan proses enkripsi dan *dekripsi* menggunakan kunci simetris. Dimulai dengan pesan yang dikirimkan, kemudian dienkripsi menjadi kode yang tidak bisa dibaca menggunakan kunci simetris. Kode tersebut kemudian dikirim dan di *dekripsi* di sisi penerima menggunakan kunci simetris yang sama, sehingga pesan dapat dibaca kembali. Pada contoh gambar, pesan "Selamat Datang!" dienkripsi menjadi kode "9CDF4837 AB672019" menggunakan kunci simetris yang sama. Kemudian di *dekripsi* kembali ke pesan awal "Selamat Datang!" Kunci simetris merupakan kunci yang sama digunakan untuk enkripsi dan *dekripsi*, sehingga pengirim dan penerima pesan harus memiliki kunci yang sama agar dapat berkomunikasi.

### AES (Advanced Encryption Standard)

Advanced Encryption Standard (AES) adalah salah satu algoritma enkripsi yang paling terkenal dan banyak digunakan, serta terus dikembangkan sebagai pengganti algoritma Data Encryption Standard (DES). AES adalah algoritma enkripsi yang bekerja dengan memproses data dalam blok dengan ukuran minimal 128 bit, dan melalui beberapa tahapan untuk mengenkripsi data. Tujuan utama dari AES adalah untuk mengamankan data rahasia sehingga tidak dapat dibaca oleh pihak yang tidak berwenang (Wahyu Putra, 2023). Pada [Tabel II.2 Panjang Kunci](#_bookmark25) dalam proses enkripsi *Advanced Encryption Standard* (AES) menggunakan ukuran blok dengan panjang sebesar 126 bit, dengan pilihan panjang kunci yang bervariasi, yaitu 128 bit, 192 bit, dan 256 bit (Sekar Putri Ananda et al., 2022). Proses enkripsi dalam algoritma ini dilakukan melalui beberapa tahap yang dikenal sebagai ronde. Jumlah ronde yang digunakan bergantung pada panjang kunci yang akan dipilih. Berikut adalah hubungan panjang kunci dan banyaknya ronde.

Algoritma AES juga menawarkan tingkat keamanan yang sangat tinggi dikarenakan ukuran kunci dan jumlah ronde yang ditentukan. Ukuran kunci menentukan jumlah kemungkinan kombinasi kunci, sehingga membuat serangan *bruteforce* menjadi sangat sulit dilakukan. Di sisi lain, jumlah ronde yang besar dapat menghasilkan keamanan yang kuat, sehingga perubahan kecil pada *plainteks* dapat memengaruhi seluruh bagian *cipherteks*. Selain keunggulan keamanannya, AES juga memiliki efisiensi tinggi dalam hal waktu dan ruang. Algoritma ini dapat diimplementasikan baik pada perangkat keras maupun perangkat lunak, menjadikannya ideal untuk berbagai aplikasi, termasuk perlindungan data (Assyahid, 2024). Pada tahap awal enkripsi, data masukan yang telah disalin ke dalam matriks *state* akan menjalani transformasi *AddRoundkey*. Selanjutnya, matriks *state* akan melalui proses transformasi secara berulang, mencakup *SubBytes*, *ShiftRows*, *MixColumns*, dan *AddRoundkey*, dengan jumlah pengulangan sesuai dengan nilai Nr (Handoyo & Subakti, 2020). *Dekripsi* adalah proses mengembalikan *ciphertext* menjadi plaintext, atau mengubah data terenkripsi kembali ke bentuk aslinya. Pada proses *inverse cipher*, transformasi yang digunakan meliputi *InvShiftRows*, *InvSubBytes*, *InvMixColumns*, dan *AddRoundkey*. Berikut ini adalah cara kerja algoritma AES secara umum:

### Proses Enkripsi AES Modifikasi

* + - * 1. *AddRoundkey*

*AddRoundkey* adalah proses menggabungkan state masukan dengan kunci ronde dengan menggunakan operasi XOR. Dalam proses transformasi enkripsi dan *dekripsi* pada AES, dilakukan operasi penambahan kunci putaran (*round key*)

pada state menggunakan operasi XOR (Djong & Siswanto, 2022). Setiap kunci putaran terdiri dari *Nb word* di mana setiap *word* tersebut akan digabungkan dengan *word* atau kolom yang sesuai pada *state* . Secara matematis, hal ini (Putri et al., 2021).

Keterangan Algoritma dari *AddRoundkey* di mana :

Pada [Gambar II.3.Transformasi ShiftRows](#_bookmark27) di atas di mana S0,0 merupakan elemen di baris ke-0, kolom ke-0, S3,3 merupakan elemen di baris ke-3, kolom ke-3, dan 𝑊3,3 merupakan elemen di baris ke-3, kolom ke-3. Pada label S0,0 dan lainnya merupakan state saat ini, state ini representasi internal data yang sedang dienkripsi ataupun *didekripsi* dan setiap elemen adalah *byte* dari blok data input. Sedangkan pada label 𝑊0,0 merupakan kunci ronde yang dihasilkan dari proses *keyexpansion*, di mana ini bagian dari kunci utama yang dipecah untuk menjadi beberapa ronde dan setiap elemen adalah *byte* dari kunci untuk ronde tertentu. Operasi XOR dilakukan antara setiap *byte* pada *state* dengan *byte* kunci ronde yang sesuai. Hasil di atas dihasilkan dari operasi XOR yang akan digunakan untuk langkah transformasi berikutnya.

* + - * 1. *SubBytes*

Transformasi *transformasi byte* di mana setiap elemen dalam matriks *state* dipetakan menggunakan tabel substitusi yang dikenal sebagai S-Box. Setiap *byte* dalam matriks *state* direpresentasikan sebagai S’[r, c], yang merupakan elemen hasil dari perpotongan antara baris (x) dan kolom (y) (Kurnia Mahesa Naibaho et al., 2021). S-Box berupa matriks

berukuran 16×16 yang berisi nilai-nilai heksadesimal di mana nilai-nilai ini diperoleh melalui perhitungan tertentu (Hena & Kusumaningsih, 2024). Berikut ini tabel S-Box :

Pada [Tabel II.3 Tabel Substitusi (S-Box)](#_bookmark28) sebagai contoh, nilai heksadesimal {62} merujuk ke baris 7 dan kolom 3 dalam S-Box, yang berisi nilai {aa}. Dengan demikian, nilai

{62} dipetakan menjadi nilai {aa}.

* + - * 1. *ShiftRows*

Transformasi *ShiftRows* merupakan proses pergeseran *byte* di mana *byte* paling kiri dipindahkan ke posisi paling kanan (rotasi bit). Jumlah pergeseran dalam melakukan *ShiftRows* ini dilakukan perbedaan setiap baris dalam matriks *state*. Pada baris pertama, tidak ada pergeseran yang dilakukan. Kemudian pada baris kedua barulah mengalami pergeseran satu *byte* ke kiri, lalu baris ketiga digeser dua *byte* ke kiri, dan baris keempat digeser tiga *byte* ke kiri. Tujuan dari proses ini adalah untuk menciptakan *diffusion*(), yang memiliki tujuan untuk membantu menyebarkan efek transformasi non-linear ke seluruh baris dalam matriks *state* untuk proses selanjutnya (Cristy & Riandari, 2021). Proses pergeseran ini bertujuan untuk mencampur elemen-elemen dalam *state* sehingga memperkuat kerahasiaan data.

*MixColumns* adalah proses transformasi di mana setiap elemen dalam blok *cipher* dikalikan dengan matriks . Transformasi ini bertujuan untuk mencampur elemen- elemen dalam setiap kolom *state* berguna meningkatkan difusi data dan memperkuat keamanan enkripsi . Transformasi ini bekerja dengan membagi matriks atau *state* menjadi empat bagian, untuk setiap kolomnya (Widyawan & Imelda, 2021). Setiap kolom dalam matriks *state* diperlakukan sebagai *polinomial* dengan empat bagian dalam *Galois Field* (GF), pada persamaan [(II-2)](#_bookmark30) dilakukan Algoritma dari *AddRoundkey* di mana :

𝑋8 = Elemen dengan derajat ke-8 yang akan muncul sebagai 100000000 (1 diposisi ke 9 dari kanan) di bilangan biner.

X = Menunjukkan elemen dengan derajat ke 1 yang akan muncul sebagai 00000010 ( 1 di posisi ke 2 dari kanan

) dalam bilangan biner

1 = Elemen konstanta yang akan muncul sebagai

00000001 ( 1 diposisi ke 1 dari kanan), dimana elemen 1 ini menyatakan bahwa *polinomial* ini memiliki koefisien konstan.

Proses *MixColumns* juga dapat direpresentasikan sebagai operasi perkalian matriks, di mana setiap kolom matriks dikalikan dengan matriks tertentu dalam *Galois Field* (Cristy & Riandari, 2021).

* + - 1. **Proses *dekripsi* AES :**

Proses *dekripsi* merupakan kebalikan dari proses enkripsi. *Transformasi bytes* yang digunakan dalam *dekripsi* mencakup *AddRoundKey*, *InvShiftRows*, *InvSubBytes,* dan *InvMixColumns*. Pada iterasi pertama *dekripsi*, dilakukan *transformasi AddRoundKey, Inverse ShiftRows, dan Inverse SubBytes. Ciphertex*t kemudian akan mengalami transformasi *AddRoundKey* .

* + - * 1. *InvShiftRows*

Transformasi *byte* yang merupakan kebalikan dari proses *ShiftRows*. Dalam transformasi ini, bit-bit pada *state* digeser ke arah kanan, berbeda dengan *ShiftRows* yang melakukan pergeseran bit ke arah kiri.

* + - * 1. *InvSubBytes*

Transformasi *byte* yang menjadi kebalikan dari *SubBytes*. Dalam proses ini, setiap elemen pada *state* dipetakan menggunakan tabel substitusi *Inverse* S-Box.

* + - * 1. *InvMixColumns*

Setiap kolom dalam *state* dikalikan dengan matriks khusus yang digunakan dalam proses perkalian pada algoritma AES (Syahputra Sianipar et al., 2024)

*Gambar II.4 Proses AES*

Pada [Gambar II.4 Proses AES](#_bookmark31) sebelah kiri menunjukkan tahapan proses enkripsi pada algoritma AES Modifikasi. Proses dimulai dengan memasukkan plainteks, kemudian melalui serangkaian langkah seperti *AddRoundkey*, SubBytes, ShiftRows, MixColumns, dan diakhiri dengan *AddRoundkey* tambahan. Langkah ini diulang sebanyak Nr−1 ronde untuk meningkatkan keamanan data, menghasilkan cipherteks di akhir proses. Flowchart sebelah kanan menjelaskan tahapan proses *dekripsi*, yang merupakan kebalikan dari proses enkripsi. Proses dimulai dengan memasukkan cipherteks, kemudian melalui langkah-langkah seperti *AddRoundkey*, Inverse ShiftRows, Inverse SubBytes, dan Inverse MixColumns. Setelah semua ronde selesai, data dikembalikan ke bentuk plainteks asli.

### Chacha20

Algoritma ChaCha20 adalah salah satu jenis *stream cipher* yang dikembangkan sebagai varian dari algoritma Salsa20. Algoritma ini berbasis desain yang serupa dengan Salsa20, namun ChaCha20 meningkatkan tingkat *diffusion* pada setiap ronde untuk memperkuat keamanan proses enkripsi (Anindita & 18219086, 2023). Chacha20 beroperasi pada unit data 32-bit menggunakan kunci 256-bit. Algoritma ini menghasilkan *output* berupa *stream* kunci sepanjang 512-bit, yang kemudian akan di XOR kan dengan *plainteks* untuk menghasilkan *chiperteks*(Alyas & Abdullah, 2021b). ChaCha20 dimulai dengan membuat *state* 512-bit, yang berupa *array* yang berukuran 16x32-bit. State ini disusun dari empat konstanta 32 bit, kunci rahasia sepanjang 256-bit, penghitung blok 32-bit, serta nonce Ini menunjukkan struktur *state* ChaCha20 yang terdiri dari matriks 4x4 dengan 16 elemen 32-bit. Matriks ini diinisialisasi dengan empat konstanta 32-bit, kunci 256-bit yang dibagi menjadi 8 bagian, penghitung 32 blok 32-bit, dan Nonce 96-bit. Elemen-elemen ini digunakan untuk menghasilkan *keystream* yang kemudian di XOR kan dengan *plainteks* untuk menghasilkan *chiperteks*. Di mana *const* (konstanta) ini berisi empat nilai tetap 32-bit, *key* (kunci) yang terbagi 8 bagian 32-bit dari kunci rahasia 256-bit, *counter* (penghitung), dan Nonce (nilai acak) yang berukuran 96-bit.

Langkah – Langkah Algoritma ChaCha20 :

* + - 1. *Inisialisasi* State

Algoritma ChaCha20 dimulai dengan membangun sebuah state internal berbentuk matriks 4x4 (32-bit) (Barbero et al., 2022) . Matriks ini terdiri dari:

* + - * 1. Baris pertama: Berisi 4 konstanta tetap (128-bit), yang digunakan untuk mengenali algoritma.
        2. Baris kedua dan ketiga: Berisi kunci rahasia 256-bit (8 elemen 32-bit), yang digunakan untuk proses enkripsi.
        3. Baris keempat: Berisi kombinasi 64-bit counter dan 64-bit nonce (masing-masing terdiri dari 2 elemen 32-bit).
      1. *Quarter Round*

Operasi ini memodifikasi 4 elemen dari matriks ( a, b, c, d ) (Almukhlifi & Vora, 2020)

𝑎 += 𝑏; 𝑑 ^ = 𝑎; 𝑑 <<<= 16;

𝑐 += 𝑑; 𝑏 ^ = 𝑐; 𝑏 <<<= 12;

𝑎 += 𝑏; 𝑏 ^ = 𝑎; 𝑑 <<<= 8;

𝑐 += 𝑑; 𝑏 ^ = 𝑐; 𝑏 <<<= 7;

* + - 1. *Double Round*

*Double Round* adalah kombinasi dari dua *Quarter Round* yang dijalankan secara berurutan. Setiap iterasi *Double Round* mencakup:

* + - * 1. Operasi pada kolom matriks *(Column Round):*

melakukan operasi Quarter Round pada setiap kolom matriks 4x4. Tujuannya adalah untuk mencampur data antar elemen dalam kolom, sehingga meningkatkan difusi data secara vertikal.

* + - * 1. Operasi pada diagonal matriks (Diagonal Round):

Setelah operasi pada kolom selesai, langkah berikutnya adalah menerapkan Quarter Round pada elemen-elemen diagonal dalam matriks.

Proses Double Round dilakukan sebanyak 10 iterasi, menghasilkan total 20 ronde Quarter Round untuk setiap blok data yang diproses.

* + - 1. Finalisasi State
      2. *Keystream Generation*

Setelah 20 ronde selesai*, state* akhir ditambahkan kembali ke

### Entropi

Entropi dalam teori informasi digunakan untuk mengukur keacakan data, yang mencerminkan kekuatan algoritma enkripsi. Semakin tinggi nilai entropi, semakin acak data tersebut, yang berarti algoritma enkripsi lebih tahan terhadap serangan (Wahyudi et al., 2024). Entropi dihitung dengan menggunakan rumus pada

Dimana:

* 𝐻𝑚 : Menunjukkan entropi data, dimana tingkat keacakan dalam

# BAB III METODOLOGI PENELITIAN

### Sistematika Penyelesaian Masalah

Penelitian ini bersifat kuantitatif dengan tujuan mengevaluasi performa algoritma AES modifikasi dan kombinasi ChaCha20 dan AES modifikasi. Penilaian performa ini dilakukan berdasarkan waktu enkripsi, waktu *dekripsi*, kecepatan enkripsi, dan kecepatan *dekripsi*, yang akan dianalisis menggunakan data hasil pengujian pada berbagai jenis *file* dengan format PDF, Word dan TXT, yang hasilnya berbentuk tabel untuk mempermudah analisis perbandingan.

### Alat dan Bahan

### Alat

1. Perangkat Keras (Hardware) Laptop dengan spesifikasi :
   1. Prosesor : Intel *Core* I5
   2. RAM : 8 GB
   3. Penyimpanan : 256 GB SSD atau lebih
2. Perangkat Lunak (*Software*)
   1. *Python* :

Menggunakan *library cryptography, pycryptodome*, dan *library* lain yang mendukung algoritma AES dan ChaCha20. Berikut beberapa pustaka yang digunakan :

* + - PyCryptodome : Digunakan Iplementasi algoritma AES dan ChaCha20
    - Cryptography : Implementasi kemanan tambahan (jika nanti dibutuhkan )
    - Numpy : Digunakan untuk analisis data (perhitungan entropi)
    - Time : Mencatat waktu eksekusi algoritma, baik untuk enkripsi maupun dekripsi.
  1. *Text* Editor :

Visual Studio Code dan *Jupyter Notebook*

* 1. Sistem Operasi : Windows

### Bahan

1. Data set *file* :
   1. *File* PDF : Ukuran *file* 100 KB – 100 MB
   2. *File* Word : Ukuran *file* 100 KB – 100 MB
   3. *File* TXT : Ukuran *file* maksimal 300 *byte*
2. Literatur :

Jurnal atau artikel ilmiah tentang AES, ChaCha20 dan *Double Encrypt* (tahun 2020-2024)

1. *Spreadsheet* Analisis Data :

Format CSV (Comma Separated Values)

### Rancangan Penelitian

Pada penelitian ini, dilakukan tahapan yang dirancang secara sistematis untuk mencapai tujuan utama, yaitu Implementasi kombinasi Advanced Encryption Standard (AES) yang dimodifikasi dan ChaCha20. Proses penelitian dimulai dengan studi literatur untuk memahami konsep dasar dan pengembangan terbaru terkait algoritma kriptografi. Setelah itu, dilakukan pengumpulan data yang relevan sebagai dasar implementasi. Tahapan selanjutnya mencakup implementasi algoritma modifikasi AES serta kombinasi algoritma tersebut dengan ChaCha20. Untuk mengukur efektivitas dan efisiensi algoritma yang dikembangkan, pengujian dilakukan pada berbagai jenis *file* seperti PDF, Word, dan TXT. Analisis hasil dari pengujian ini memberikan dasar untuk menarik kesimpulan dan memberikan rekomendasi untuk pengembangan lebih lanjut.

*Gambar III.1 Diagram Alur Penelitian*

[Gambar III.1 Diagram Alur Penelitian](#_bookmark44) ini terdapat beberapa tahapan yaitu:

1. Mulai adalah tahap awal dari penelitian. Pada tahap ini menentukan tujuan dan menyusun langkah-langkah yang akan dilakukan dalam penelitian.
2. Studi literatur dilakukan pengumpulan referensi yang berkaitan dengan algoritma AES, ChaCha20 dan *Double Encrypt. Referensi* berupa jurnal, artikel ilmiah.
3. Pengumpulan data untuk mengumpulkan *file* atau data set yang akan digunakan untuk pengujian algoritma. Format *file* yang digunakan adalah PDF, Word dan TXT.
4. Implementasi modifikasi AES, pada tahap ini algoritma AES dimodifikasi dengan menambahkan langkah *AddRoundkey* sebelum *Mixcolumns*. Tujuannya adalah untuk meningkatkan keamanan dan kecepatan enkripsi dibandingkan dengan AES standar.
5. Implementasi kombinasi AES modifikasi dan ChaCha20, menggabungkan algoritma AES yang sudah dimodifikasi dengan ChaCha20 untuk melakukan *double encypt*. Di mana proses ini bertujuan untuk meningkatkan keamanan *file* lebih baik daripada menggunakan *single encrypt*.
6. Pengujian algoritma, untuk menguji performa algoritma yang sudah diimplementasikan, di mana data set yang dikumpulkan sebelumnya itu digunakan untuk uji coba. Parameter yang diukur pada pengujian ialah

waktu enkripsi dan *dekripsi,* kecepatan enkripsi dan *dekripsi*, dan hasil pengujian ini akan digunakan untuk mengevaluasi kinerja algoritma.

1. Analisis hasil, data pengujian yang sudah terkumpul akan dianalisis. Analisis dilakukan untuk membandingkan performa AES standar, AES yang sudah dimodifikasi, dan kombinasi AES modifikasi dan ChaCha20. Tujuannya sendiri untuk mengetahui algoritma mana yang lebih efisien.
2. Selanjutnya ke kesimpulan dan saran, pada tahap ini merangkum hasil penelitian termasuk kelebihan dan kekurangan algoritma yang sudah diuji. Saran juga untuk pengembangan penelitian lebih lanjut
3. Selesai

### Studi Literatur

Studi literatur dilakukan untuk memahami konsep dasar dan perkembangan terbaru terkait algoritma kriptografi yang digunakan dalam penelitian ini, yaitu Advanced Encryption Standard (AES) dan ChaCha20. Proses ini melibatkan pengumpulan referensi dari jurnal, paper, dan buku yang membahas metode enkripsi, keamanan data, serta teknik evaluasi seperti entropi dan analisis performa. Literatur yang dikaji juga mencakup penelitian sebelumnya tentang algoritma kriptografi untuk keamanan *file*, sehingga memberikan dasar teori yang kuat untuk modifikasi algoritma dan penerapan metode double encryption dalam penelitian ini.

### Pengumpulan Data

Data yang digunakan dalam penelitian ini dibuat sendiri dengan mengikuti ketentuan yang telah ditentukan. Format data meliputi *file* PDF, Word, dan TXT dengan ukuran bervariasi, mulai dari 100 KB hingga 100 MB untuk *file* PDF dan Word, serta hingga 300 *byte* untuk *file* TXT. Data ini dirancang sedemikian rupa agar sesuai dengan kebutuhan pengujian algoritma enkripsi, seperti mengukur waktu enkripsi, waktu *dekripsi*, kecepatan, dan tingkat keacakan menggunakan analisis entropi. Variasi format dan ukuran data ini

bertujuan untuk memastikan bahwa pengujian mencakup berbagai jenis isi *file*.

### Implementasi Modifikasi AES

Pengembangan algoritma *Advanced Encryption Standard* (AES) dilakukan dengan memodifikasi. Modifikasi dilakukan dengan menambahkan *AddRoundkey* sebelum *Mixcolumns*. Penambahan ini bertujuan untuk memperkuat keamanan dan meningkatkan kompleksitas pada proses enkripsi, sehingga lebih sulit untuk diretas dengan pihak tidak berwenang. Berikut adalah proses alur dari modifikasi algoritma AES.

*Gambar III.2 Diagram Proses Enkripsi AES*

Pada [Gambar III.2 Diagram Proses Enkripsi AES](#_bookmark48) ini digunakan untuk mengamankan *file* dengan melakukan tahap enkripsi terhadap data *plainteks* yang menggunakan kunci tertentu. Dalam penelitian ini, modifikasi dilakukan dengan menambahkan langkah *AddRoundkey* untuk meningkatkan tingkat keamanan dan kompleksitas enkripsi. Berikut langkah-langkah proses modifikasi AES .

1. Input *file* dan kunci

Proses enkripsi dimulai dengan memasukkan *file* input (*file* asli yang akan dienkripsi) dan kunci enkripsi. *File* input dibaca dalam mode biner, diubah menjadi *byte array*, lalu dipecah menjadi blok-blok data berukuran 16 *byte*. Jika panjang data tidak habis dibagi 16 *byte, padding* dilakukan dengan metode PKCS#7 untuk melengkapinya.

Contoh :

* 1. *File* input : contoh.pdf

Isi *file* dalam biner : [0x25, 0x50, 0x44, 0x46, ...]

*Byte array* : [37, 80, 68, 70, 10, 0, 45, 78, ...]

* 1. Kunci enkripsi :

Kunci : surabayaitelkom

ASCII : [115, 117, 114, 97, 98, 97, 121, 97, 105, 116, 116,

101, 108, 107, 111, 109]

* 1. Matriks data *file* :

|  |  |  |  |
| --- | --- | --- | --- |
| [37, | 80, | 68, | 70] |
| [10, | 0, | 45, | 78] |
| [255, | 100, | 32, | 12] |
| [80, | 50, | 60, | 72] |

1. *AddRoundkey*

Merupakan tahap awal proses enkripsi, di mana *plainteks* ini digabungkan dengan kunci awal yang menggunakan operasi logika yaitu XOR. Di mana blok data *file* digabungkan dengan kunci awal (W0-W3) menggunakan operasi XOR untuk menambahkan lapisan keamanan awal.

Contoh :

* 1. Proses XOR :

37 XOR 115 = 86

80 XOR 117 = 37

68 XOR 114 = 54

70 XOR 97 = 39

* 1. Matriks setelah *AddRoundkey* :

[86, 37, 54, 39]

[108, 97, 76, 47]

[150, 16, 84, 97]

[28, 89, 83, 37]

1. Proses ronde 1-9
   1. *SubBytes*

Pada proses ini, nantinya setiap *byte* pada matriks *state* ( data yang diperoleh *AddRounKey*) akan diubah menggunakan tabel substitusi yang biasa disebut dengan S-Box. *Tranformasi* ini bertujuan untuk meningkatkan keacakan data agar sulit diprediksi.

Contoh proses :

*Byte* 37 diganti dengan nilai dari tabel S-Box jadi 120

* 1. *ShiftRows*

Setiap baris dalam matriks *state* nantinya digeser ke kiri pada baris pertama tidak ada pergeseran, baris kedua di geser 1 *byte* ke kiri, baris ketiga ini digeser 2 *byte* ke kiri, dan terakhir baris keempat yang nantinya digeser 3 *byte* ke kiri. Contoh proses :

[244, 120, 99, 200] [244, 120, 99, 200]

[60, 107, 55, 175] [107, 55, 175, 60]

[12, 243, 172, 85] [172, 85, 12, 243]

[180, 97, 92, 65] [65, 180, 97, 92]

* 1. *AddRoundkey* (W4-W7)

Setelah *ShiftRows* dilakukan operasi XOR antara matriks *state* dengan kunci ronde berikutnya yaitu (W4-W7) Contoh proses :

*Byte* 244 XOR 115 = 135

*Byte* 120 XOR 117 = 13

Hasil *AddRounKey* : [135, 13, 17, 169]

[38, 98, 85, 12]

[55, 32, 82, 84]

[49, 53, 180, 193]

* 1. *MixColumns*

Setiap kolom pada matriks *state* dimodifikasi dengan operasi matematis berbasis matriks yang menggunakan bidang *Galois* (GF). Di mana proses ini bertujuan untuk mencampur elemen-elemen dalam kolom dan meningkatkan difusi data. [2, 3, 1, 1]

[1, 2, 3, 1]

[1, 1, 2, 3]

[3, 1, 1, 2]

* 1. *AddRoundkey* (W36-W39)

Dilakukan setelah *MixColums* untuk menambahkan lapisan keamanan dengan kunci ronde tambahan (W36-W39).

1. Pada ronde terakhir, hanya dilakukan tiga langkah yaitu

*SubBytes, ShiftRows*, dan *AddRoundkey* tanpa *MixColumns.*

Proses modifikasi algoritma AES ini digunakan untuk mengenkripsi *file* dengan menambahkan langkah *AddRoundkey* tambahan, yang meningkatkan keamanan dengan memberikan lapisan XOR tambahan pada setiap ronde enkripsi.

* + 1. **Implementasi Kombinasi AES Modifikasi dan ChaCha20** Algoritma AES modifikasi dan ChaCha20 dengan menerapkan *double encypt.* Proses ini memakai AES modifikasi dengan memberikan perlindungan berbasis blok *chiper* yang kuat dengan memodifikasi *AddRoundkey* tambahan sebelum *MixColumns,* sedangkan ChaCha20 sebagai *stream cipher* yang cepat dan aman, meningkatkan kompleksitas keseluruhan dengan memanfaatkan proses enkripsi berbasis *keystream.*

Pada metode ini, setiap *file* menggunakan 1 kunci enkripsi. Kunci yang digunakan memiliki panjang awal 128-bit, yang kemudian diperluas menjadi 256-bit pada ChaCha20 dengan menambahkan padding nol (0x00) jika panjangnya kurang dari 256-bit. Proses

dimulai dengan mengenkripsi *file* menggunakan AES Modifikasi untuk menghasilkan cipherteks awal dalam format matriks 4x4. Cipherteks ini kemudian diubah menjadi byte stream untuk kompatibilitas dengan ChaCha20. Selanjutnya, ChaCha20 melakukan proses enkripsi kedua dengan memanfaatkan kunci 256- bit, nonce sepanjang 96-bit, dan counter. Hasil akhir adalah cipherteks final yang lebih aman karena telah melalui dua lapisan enkripsi. Penggunaan double encryption dengan satu kunci untuk setiap *file* memberikan keamanan yang lebih tinggi, di mana kombinasi kedua algoritma menciptakan perlindungan yang kuat terhadap berbagai jenis serangan kriptografi.

*Gambar III.3 Diagram Proses Double Encrypt*

Pada [Gambar III.3 Diagram Proses Double Encrypt](#_bookmark50) menunjukkan langkah-langkah enkripsi ganda. Proses dimulai dengan mengenkripsi *plainteks* menggunakan AES modifikasi untuk menghasilkan *chiperteks* AES. *Chiperteks* ini kemudian dienkripsi ulang menggunakan ChaCha20, yang melibatkan matriks 4x4, proses *quarter-round,* dan operasi XOR dengan *keystream*. Algoritma AES modifikasi dan ChaCha20 menggunakan kunci yang sama, yaitu kunci AES sepanjang 128-bit. berikut adalah tahapan alur di atas :

1. Input *plainteks* dan *key* dan enkripsi AES modifikasi
2. Input *cipherteks* AES ke ChaCha20

*Chiperteks* yang dihasilkan dari proses enkripsi pertama diubah menjadi input untuk algoritma ChaCha20. Ini dilakukan untuk menerapkan enkripsi berlapis yaitu *double encryption,* sehingga keamanan data akan meningkat secara signifikan. *Chiperteks* AES yang biasanya dalam format matriks (4x4) ini diubah ke format *byte stream* untuk kompatibilitas dengan algoritma ChaCha20. Byte stream ini memungkinkan untuk memproses data, sesuai dengan *stream chiper*.

Contoh :

[115, 75, 189, 234, 201, 97, 58, 14, 245, 142, 12, 120, 53, 190,

201, 20] data ini sudah dalam bentuk *byte stream*.

1. Validasi panjang *key* (256-bit *Key*)

Ini untuk memastikan bahwa kunci enkripsi untuk ChaCha20 sesuai dengan spesifikasi algoritma, yaitu panjangnya harus tepat 256 bit (32 *byte*). Kunci ini untuk menghasilkan *keystream* yang beda dalam proses enkripsi. Jika panjang ke kurang dari 256 bit, ditambahkan *padding* berupa *byte* nol (0x00) untuk memenuhi panjang. Jika *key* yang lebih panjang, *key* akan dipotong hingga hanya menyisakan 256 bit. *Key* kemudian dibagi menjadi 8 bagian ( di mana masing-masing 32 bit) untuk digunakan dalam proses pembentukan matriks awal (*state* matriks)

Contoh :

* 1. *Key* input “surabayaittelkom”

[115, 117, 114, 97, 98, 97, 121, 97, 105, 116, 116, 101, 108,

107, 111, 109]

* 1. Panjang *key* adalah 16 *byte* (128 bit), sehingga tidak memenuhi syarat panjang 256 bit. Jadi ditambahkan *padding* nol (0x00) hingga mencapai 32 *byte:*

[115, 117, 114, 97, 98, 97, 121, 97, 105, 116, 116, 101, 108,

107, 111, 109, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

1. Membuat *Initial* State Matriks 4x4

Membentuk *state* matriks awal yang menjadi dasar operasi dalam ChaCha20. Matriks ini berisi kunci, *counter*, dan *nonce.* Matriks 4x4 dalam ChaCha20 terdiri 16 elemen (masing-masing 32 bit) Contoh :

* 1. 4 elemen konstanta tetap “*expand* 32-byte k”4 elemen 32 bit [1634760805, 857760878, 2036477234, 1797285236]
  2. Elemen kunci 256-bit dibagi menjadi 8 elemen 32-bit: [1936028945, 1885434721, 1701147239, 1647539057,

1225269481, 1226844037, 1214606441, 0]

* 1. 1 *Counter*: Nilai awal 0.
  2. 3 Elemen Nonce: Nilai *nonce* yang unik untuk sesi enkripsi: Contoh *nonce* : [12345678, 87654321, 11223344]
  3. Matriks yang dibentuk

[1634760805, 857760878, 2036477234, 1797285236]

[1936028945, 1885434721, 1701147239, 1647539057]

[1225269481, 1226844037, 1214606441, 0]

[0, 12345678, 87654321, 11223344]

Baris ke 1 konstanta, ke 2 *key,* ke 3 *key* dan ke 4 *counter* dan

*nonce*

1. Operasi Quarter Round (20 iterasi)

Melakukan proses pencampuran data di dalam matriks untuk meningkatkan kerandoman dan keamanan *keystream* yang dihasilkan. Quarter Round adalah inti dari algoritma ChaCha20. Quarter Round dilakukan pada 4 elemen tertentu dari matriks. Penjumlahan Modular (mod 2³²): Menambah elemen dalam matriks untuk mencampur data. Operasi XOR: Mencampur data antar elemen melalui operasi logika XOR. Rotasi Bit (ROTL): Menggeser bit elemen untuk mendistribusikan informasi secara acak. Setelah 20 iterasi (10 putaran double round), matriks akan diubah menjadi bentuk yang sangat acak.

1. *Counter* +=1

*Counter* bertambah untuk setiap blok data yang diproses, memastikan *keystream* yang dihasilkan unik untuk setiap blok. Setelah satu blok data selesai diproses, *counter* di matriks bertambah sebesar 1.

Contoh :

*Counter* baru : [1, 12345678, 87654321, 11223344]

1. XOR *keystream* dengan *chiperteks* AES

Menggabungkan *keystream* hasil ChaCha20 dengan *cipherteks AES* melalui operasi XOR untuk menghasilkan *final ciphertext*. XOR dipilih karena sifatnya yang sederhana namun aman dan mudah dibalik. *Keystream* yang dihasilkan dari matriks ChaCha20 di-XOR-kan dengan *cipherteks AES*.

Operasi XOR:

Jika *ciphertext* = *keystream* XOR *plaintext*, maka *dekripsi*

dilakukan dengan cara *plaintext* = *keystream* XOR *ciphertext*.

* + - 1. **Proses *Dekripsi* Double Encryption**

Proses *dekripsi* dilakukan untuk mengembalikan data yang telah terenkripsi ke bentuk aslinya atau *plainteks*. Di mana pada tahap ini, *chiperteks* hasil dari kombinasi algoritma AES modifikasi dan Chacha20 akan *didekrpsi* secara urut. Proses nantinya akan dimulai dengan *mendekripsi chiperteks* dengan ChaCha20 untuk mendapatkan kembali *chiperteks* AES modifikasi, kemudian dilanjutkan dengan *mendekripsi chiprteks* AES modifikasi untuk memperoleh *plainteks* asli.

Pada [Gambar III.4 Proses Dekripsi](#_bookmark51) Diagram alur di atas menjelaskan proses *dekripsi* dari kombinasi algoritma ChaCha20 dan modifikasi AES. Proses dimulai dengan memasukkan cipherteks hasil enkripsi sebagai input. Langkah pertama adalah men*dekripsi* cipherteks menggunakan algoritma ChaCha20 untuk menghasilkan keystream. Keystream ini kemudian dikombinasikan dengan cipherteks menggunakan operasi XOR untuk mendapatkan cipherteks AES.

Selanjutnya, proses ekspansi kunci dilakukan untuk menghasilkan kunci yang akan digunakan pada setiap ronde *dekripsi* AES. Proses *dekripsi* AES terdiri dari tiga tahapan utama: ronde terakhir (Round 10) yang melibatkan *AddRoundKey*, InvShiftRows, dan InvSubBytes; ronde utama (Round 9-1) yang melibatkan *AddRoundKey*, InvMixColumns, *AddRoundKey, InvShiftRows,* dan *InvSubBytes* tambahan; dan ronde awal (Round 0) yang menggunakan *AddRoundKey* (W0-W3). Hasil akhir dari seluruh proses ini adalah plainteks, yaitu data asli yang berhasil di*dekripsi*. Kombinasi kedua algoritma ini memastikan keamanan data yang lebih kuat melalui enkripsi ganda.

*Gambar III.4 Proses Dekripsi*

### Implementasi Antarmuka Pengguna (UI)

Antarmuka pengguna (UI) dirancang untuk mempermudah interaksi, berfungsi sebagai media interaktif yang memungkinkan pengguna untuk memilih file, memasukkan kunci enkripsi, serta memulai proses enkripsi dan dekripsi dengan mudah. Implementasinya menggunakan Flask sebagai framework backend untuk menangani

proses enkripsi dan dekripsi, sedangkan HTML digunakan untuk membuat form input file dan kunci enkripsi. CSS digunakan untuk mempercantik tampilan agar antarmuka lebih menarik, jelas, dan mudah digunakan oleh pengguna dalam proses enkripsi dan dekripsi file menggunakan kombinasi algoritma ChaCha20 dan AES yang telah dimodifikasi.

*Gambar III.5 Desain Antarmuka Pengguna Enkripsi dan Dekripsi*

Pada [Gambar III.5 Desain Antarmuka Pengguna Enkripsi dan](#_bookmark53) [Dekripsi](#_bookmark53) Gambar tersebut menampilkan desain antarmuka pengguna (UI) dari sebuah aplikasi berbasis web yang digunakan untuk enkripsi dan *dekripsi file*. Antarmuka ini memiliki beberapa elemen input pengguna untuk mengunggah *file* dan memilih opsi enkripsi atau *dekripsi*. Terdapat tiga kolom utama pada desain ini: pertama, Pilih *File*, untuk pengguna memilih *file* yang akan diproses. Kedua, Enkripsi atau *Dekripsi*, yang memberikan pilihan kepada pengguna apakah *file* yang dipilih akan dienkripsi atau di *dekripsi*, yang merupakan opsi penting untuk menentukan jenis operasi yang akan dilakukan pada *file* tersebut. Ketiga, Key, di mana pengguna diminta untuk memasukkan kunci yang diperlukan untuk proses enkripsi atau *dekripsi*. Kunci ini memainkan peran penting dalam menjaga keamanan data yang diproses. Di bagian bawah antarmuka, terdapat

tombol Proses yang akan menjalankan operasi sesuai dengan pilihan pengguna, baik itu enkripsi maupun *dekripsi*.

### Pengujian Algoritma

Menjelaskan pengujian yang dilakukan untuk mengevaluasi kinerja dan keamanan algoritma berdasarkan parameter seperti entropi, waktu enkripsi dan *dekripsi*, serta kecapatan.

### Pengujian Entropi

Pengujian kekuatan algoritma dilakukan dengan mengukur tingkat keacakan data hasil enkripsi menggunakan analisis entropi. Keacakan data merupakan indikator penting dalam menentukan seberapa baik algoritma enkripsi menyamarkan pola asli dari data. Semakin tinggi tingkat keacakan, semakin sulit bagi pihak yang tidak berwenang untuk menganalisis pola atau memprediksi isi data terenkripsi. Pengujian ini dilakukan dengan menghitung nilai entropi menggunakan distribusi probabilitas setiap simbol dalam cipherteks. Nilai entropi yang mendekati maksimum menunjukkan bahwa distribusi simbol dalam cipherteks hampir merata, menandakan bahwa algoritma telah berhasil menghasilkan data yang acak. Data dengan keacakan tinggi lebih tahan terhadap serangan berbasis statistik, sehingga pengujian ini memberikan evaluasi kuantitatif terhadap kekuatan keamanan algoritma yang diuji. Hasil pengujian dibandingkan antara AES Standar, AES Modifikasi, Chacha20 dan Double Encryption untuk menunjukkan efektivitas modifikasi dalam meningkatkan keamanan.

### Pengujian Waktu dan Kecepatan

Menguji algoritma AES modifikasi dan *Double encrypt* AES modifikasi dan ChaCha20. Pengujian ini dilakukan untuk menilai kinerja algoritma dalam berbagai parameter. Penelitian ini akan menganalisis parameter pengujian menggunakan matriks dan formula berikut :

* + - * 1. Waktu enkripsi :

mengukur waktu yang dibutuhkan untuk mengenkripsi data dari

*plainteks* menjadi *chiperteks* dijelaskan dalam persamaan [(III-1)](#_bookmark55).

|  |  |
| --- | --- |
| 𝐸𝑡 = 𝐸𝑁𝑡 – 𝑆𝑇𝑡 | (III-1) |

Di mana:

𝐸𝑡 = waktu enkripsi dalam mili detik (ms)

𝐸𝑁𝑡 = waktu selesai enkripsi

𝑆𝑇𝑡 = waktu mulai enkripsi

Waktu ini akan dihitung untuk setiap *file* yang diuji, dan hasilnya nanti akan memberikan gambaran mengenai kecepatan algoritma dalam melakukan proses enkripsi.

* + - * 1. Waktu *dekripsi* :

Mengukur waktu yang dibutuhkan untuk *mendekripsi* data dari

*chiperteks* menjadi *plainteks* dijelaskan pada persamaan [(III-2)](#_bookmark56).

Dimana:

𝐷𝑡 = waktu *dekripsi* (ms)

𝐸𝑁𝑡 = waktu selesai *dekripsi*

𝑆𝑇𝑡 = waktu mulai *dekripsi*

Hasil waktu *dekripsi* ini juga perlu untuk masing-masing algoritma karena ini menganalisis perbandingan kinerja antara algoritma enkripsi.

* + - * 1. Kecepatan :

Menunjukkan jumlah data dalam *kilobyte* yang dapat dienkripsi

Et = waktu enkripsi atau *dekripsi* (s)

### Pengujian pada Berbagai Jenis *File*

*File* PDF yang digunakan dalam pengujian dapat berisi elemen seperti teks, gambar, dan tabel yang semuanya akan dienkripsi secara utuh karena algoritma bekerja pada level data biner. *File* PDF dengan ukuran 100 KB, 1 MB, 10 MB, hingga 100 MB diuji untuk memastikan algoritma dapat bekerja pada dokumen berukuran kecil hingga besar. Hal serupa juga berlaku untuk *file* Word dengan ukuran yang sama, meskipun memiliki struktur internal yang lebih kompleks dibandingkan PDF. *File* TXT dengan ukuran maksimal 300 *byte*, yang biasanya hanya berisi teks sederhana, juga dapat dienkripsi ChaCha20 (*double encrypt*). Di mana pengujian ini bertujuan untuk

mengukur kinerja algoritma. Berikut ini adalah contoh hasil pengujian terhadap *file* PDF :

*Tabel III.2 Evaluasi Hasil*

Pada [Tabel III.2 Evaluasi Hasil](#_bookmark60) menunjukkan hasil evaluasi simulasi kinerja algoritma enkripsi dan *dekripsi*. pada Nilai-nilai tersebut memberikan gambaran awal mengenai performa beberapa metode, termasuk AES Standar, ChaCha20, AES Modifikasi, dan kombinasi AES Modifikasi dengan ChaCha20. Hasil ini digunakan sebagai contoh output yang diharapkan dari model algoritma yang akan dirancang, ini belum merupakan data dari eksperimen nyata.

# DAFTAR PUSTAKA

Alimuddin, A., Juntak, J. N. S., Jusnita, R. A. E., Murniawaty, I., & Wono, H. Y. (2023). Teknologi dalam pendidikan: Membantu siswa beradaptasi dengan revolusi industri 4.0. *Journal on Education*, *5*(4), 11777–11790.

Almukhlifi, R., & Vora, P. L. (2020). Linear cryptanalysis of reduced-round simon using super rounds. *Cryptography*, *4*(1), 1–34. https://doi.org/10.3390/cryptography4010009

Alyas, H. H., & Abdullah, A. A. (2021). Enhancement the ChaCha20 Encryption Algorithm Based on Chaotic Maps. *Lecture Notes in Networks and Systems*, *201*, 91–107. https://doi.org/10.1007/978-981-16-0666-3\_10

Anindita, M., & 18219086, A. A. (2023). *Analisis Perbandingan Algoritma AES dan ChaCha20-Poly1305 dalam Enkripsi Konten Livestreaming*.

Aryanto, M. B., Tahir, M., Devita, S. I., Mustofa, Z. N., Ainiyah, Q., & Sundoro,

S. (2023). Implementasi Enkrip Dan Dekrip File Menggunakan Metode Advance Encryption Standard (AES-128). *Jurnal Ilmiah Sistem Informasi Dan Ilmu Komputer*, *3*(1), 89–104.

Assyahid, M. H. (2024). *Implementasi interpolasi pixel wadah citra dan enkripsi aes pada steganografi least significan bit (LSB)*. https://repository.uinjkt.ac.id/dspace/handle/123456789/76727

Barbero, S., Bazzanella, D., & Bellini, E. (2022). Rotational Cryptanalysis on ChaCha Stream Cipher. *Symmetry*, *14*(6). https://doi.org/10.3390/sym14061087

Belvin Tannadi. (2022). *Ilmu Crypto* (Digital). Jakarta : Elex Media Komputindo.

Cristy, N., & Riandari, F. (2021). *Niolinda Cristy 1 , Fristi Riandari 2 [Implementasi Metode Advanced Encryption Standard (AES 128 Bit) Untuk Mengamankan Data Keuangan*. *4*(2), 75.

Djong, H. S., & Siswanto, S. (2022). Implementasi Kriptografi Dengan Menggunakan Metode RC4 dan AES-256 Untuk Mengamankan File Dokumen pada PT Varnion Technology Semesta. *Prosiding Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI)*, *1*(1), 149–158.

Handoyo, J., & Subakti, Y. M. (2020). Keamanan Dokumen Menggunakan Algoritma Advanced Encryption Standard (AES). *Jurnal SITECH : Sistem Informasi Dan Teknologi*, *3*(2), 143–152. https://jurnal.umk.ac.id/index.php/sitech/article/view/5865

Hena, N., & Kusumaningsih, D. (2024). Pengamanan File Pada Sistem Master Vendor Berbasis WEB Menggunakan Algoritma AES Pada Trinitiland. *Prosiding Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI)*, *3*(2), 30–37.

Hidayatulloh, N. W., Tahir, M., Amalia, H., Basyar, N. A., Prianggara, A. F., & Yasin, M. (2023). Mengenal Advance Encryption Standard (AES) sebagai Algoritma Kriptografi dalam Mengamankan Data. *Digital Transformation Technology*, *3*(1), 1–10.

Kebande, V. R. (2023). Extended-Chacha20 Stream Cipher With Enhanced Quarter Round Function. *IEEE Access*, *11*, 114220–114237. https://doi.org/10.1109/ACCESS.2023.3324612

Kim, B. G., & Wong, D. (2023). Smart Contract-based Secure Verifiable Random Function using ChaCha20 Sequence in Blockchain. *ACM International Conference Proceeding Series*, 41–51. https://doi.org/10.1145/3638025.3638028

Kurnia Mahesa Naibaho, A., Putri, A., & Fikri Husaini, Y. (2021). Analisis Sistem Keamanan Informasi Chat Online dengan Menggunakan Algoritma AES. In *Ilmu Komputer* (Vol. 6, Issue 2).

Luong, T. T. (2023). Strengthening AES Security Through Key-Dependent ShiftRow and AddRoundKey Transformations Utilizing Permutation. In *IJACSA) International Journal of Advanced Computer Science and Applications* (Vol. 14, Issue 11). [www.ijacsa.thesai.org](http://www.ijacsa.thesai.org/)

Muhammed, R. K., Rashid Aziz, R., Hassan, A. A., Aladdin, A. M., Saydah, S. J., Rashid, T. A., & Hassan, A. (2024). *Comparative Analysis of AES, Blowfish, Twofish, Salsa20, and ChaCha20 for Image Encryption*.

Olivia Putri Irine Irawan, B., Tahir, M., Ayu Windrastuti, N., Yurina Cholili, D., Mulaikah, D., Batsul Mushofi Septian wachid, A., & Pendidikan Informatika,

P. (2023). Implementasi Kriptografi Pada Keamanan Data Menggunakan Algoritma Advance Encryption Standard (AES). *Jurnal Simantec*, *11*(2), 167–

174. https://doi.org/10.21107/SIMANTEC.V11I2.20034

Pangestu, G. Y., Hadiana, A. I., & Sabrina, P. N. (2022). Kriptografi Untuk Enkripsi Ganda Pada Gambar Menggunakan Algoritma AES (Advanced Encryption Standard) Dan RC5 (Rivest Code 5). *Vol*, *1*, 25–32.

Putri, A. E., Kartikadewi, A., & Rosyid, L. A. A. (2021). Implementasi Kriptografi dengan Algoritma Advanced Encryption Standard (AES) 128 Bit dan Steganografi menggunakan Metode End of File (EOF) Berbasis Java Desktop

pada Dinas Pendidikan Kabupaten Tangerang. *Appl. Inf. Syst. Manag*, *3*(2), 69–78.

Saripa, S. (2023). Implementation Implementation of a File Security System Using the AES Algorithm to Secure Personal Files: Implementasi Sistem Keamanan File Menggunakan Algoritma AES untuk Mengamankan File Pribadi. *Progressive Information, Security, Computer, and Embedded System*, *1*(2), 138–148.

Sekar Putri Ananda, Saepul Lukman, & Irfan. (2022). *Analisa Metode Kriptogra Modern Advance Encryption Standar (AES) 128 Bit dalam Mengenkripsi dan Mendekripsi File Dokumen Digital*. https://doi.org/10.32409/jikstik.21.3.2973

Soesanto, E., Purba, L. M., Aprilia, B., Putra, D. R., & Putri, S. D. (2023). Implementasi Objek Vital, Pengamanan File dan Pengamanan Cyber di PT Pertamina. *IJM: Indonesian Journal of Multidisciplinary*, *1*(1).

Surbakti, P. S., & Purwanto, P. (2024). Implementasi Kriptografi Dengan Menggunakan Algoritma AES-128 Untuk Enkripsi dan Dekripsi File Dokumen Berbasis Web Pada Law Office Erdi Surbakti, SH & Rekan. *Prosiding Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI)*, *3*(1), 1–8.

Syahputra Sianipar, J., Budi Nuugroho, N., Mariami, I., Informasi, S., & Triguna Dharma, S. (2024). Pengamanan Data Gaji Karyawan Dengan Menggunakan Metode Advanced Encryption Standard (AES). *Jurnal Sistem Informasi Triguna Dharma (JURSI TGD)*, *3*(1), 35–45. https://doi.org/10.53513/JURSI.V3I1.5653

Wahyu Putra, M. R. F. T. H. (2023, July 2). *Implementasi Algoritma Advanced Encryption Standard Untuk Kemanan Dokumen*. https://journal.grahamitra.id/index.php/jurikti/article/view/55/181

Wahyudi, E. N., Ardhianto, E., Handoko, W. T., Murti, H., Supriyanto, E., Lestariningsih, E., & Redjeki, R. S. (2024). Peningkatan Keamanan Data Melalui Teknik Super Enkripsi Menggunakan Algoritma Vigenere dan Caesar. *Jurnal Informatika Polinema*, *10*(3), 315–322. https://doi.org/10.33795/JIP.V10I3.5131

Widyawan, D., & Imelda, I. (2021). Pengamanan File Menggunakan Kriptografi Dengan Metode Aes-128 Berbasis Web Di Komite Nasional Keselamatan Transportasi. *SKANIKA: Sistem Komputer Dan Teknik Informatika*, *4*(1), 15– 22.

**IMPLEMENTASI KOMBINASI CHACHA20 DAN MODIFIKASI AES MELALUI *ADDROUNDKEY* PADA**

**STUDI KASUS KEAMANAN *FILE***

**Proposal Tugas Akhir Mata Kuliah Penulisan Proposal**

## (CAK4EAB2)

Disusun oleh:

## EMILIA RAMONA 1203210017

**PROGRAM STUDI SARJANA INFORMATIKA DIREKTORAT KAMPUS SURABAYA**

**UNIVERSITAS TELKOM SURABAYA TAHUN 2025**

# ABSTRAK

Keamanan data menjadi tantangan besar di era Revolusi Industri 4.0, di mana informasi penting perlu dilindungi dari akses tidak sah. Ancaman serangan siber dan peretasan mengharuskan pengembangan solusi keamanan yang lebih efektif. Penelitian ini bertujuan untuk mengimplementasikan dan mengevaluasi kombinasi algoritma ChaCha20 dan modifikasi *Advanced Encryption* Standard (AES) dengan penambahan langkah *AddRoundKey* sebelum *MixColumns* untuk meningkatkan keamanan *file*. Metode yang digunakan adalah enkripsi ganda, dengan AES modifikasi sebagai *block cipher* yang kuat dan ChaCha20 sebagai *stream cipher* yang cepat. Penelitian ini mencakup implementasi algoritma dengan menggunakan pustaka kriptografi untuk menguji kinerja algoritma melalui parameter seperti waktu enkripsi, waktu *dekripsi*, kecepatan enkripsi, kecepatan *dekripsi*, dan tingkat keacakan data menggunakan analisis entropi. *File* dengan berbagai format dan ukuran, PDF, Word, dan TXT, digunakan sebagai data pengujian untuk mengevaluasi efisiensi algoritma. Selain itu, antarmuka pengguna berbasis web dirancang untuk mempermudah proses enkripsi dan *dekripsi*, melibatkan langkah- langkah seperti pemilihan *file* dan *input* kunci enkripsi. Penelitian ini diharapkan menunjukkan bahwa kombinasi AES modifikasi dan ChaCha20 mampu meningkatkan keamanan data serta memberikan efisiensi yang lebih baik dalam perlindungan *file*.

**Kata Kunci**: Keamanan *File*, *Advanced Encryption Standard* (AES), ChaCha20, Enkripsi Ganda, *AddRoundkey*, *Entropi*.

# ABSTRACT

*Data security has become a major challenge in the era of the Industrial Revolution 4.0, where critical information must be protected from unauthorized access. Cyberattacks and hacking threats demand the development of more effective security solutions. This research aims to implement and evaluate the combination of the ChaCha20 algorithm and a modified Advanced Encryption Standard (AES) by adding an AddRoundKey step before MixColumns to enhance file security. The method used is double encryption, with the modified AES as a robust block cipher and ChaCha20 as a fast stream cipher. This study involves implementing the algorithms using cryptographic libraries to test their performance through parameters such as encryption time, decryption time, encryption speed, decryption speed, and data randomness level using entropy analysis. Files of various formats and sizes, including PDF, Word, and TXT, are utilized as test data to evaluate the efficiency of the algorithms. Additionally, a web-based user interface is designed to simplify the encryption and decryption process, involving steps such as file selection and encryption key input. This research is expected to demonstrate that the combination of modified AES and ChaCha20 can enhance data security and provide better efficiency in file protection.*

***Keywords****: File Security, Advanced Encryption Standard (AES), ChaCha20, Double Encryption, AddRoundkey, Entropy*

# KATA PENGANTAR

Assalamu’alaikum Warrahmatullahi Wabarakatuh, puja puji syukur ke hadirat Allah SWT yang telah memberikan nikmat dan karunia-Nya yang sangat luar biasa kepada hamba – hambanya. Tidak lupa shalawat serta salam kita curahkan atas bimbingan nabi kita nabi Muhammad SAW yang telah membawa cahaya islam ke dalam dunia ini dan menjadi panutan bagi setiap umatnya hingga akhir zaman.

Setelah melalui berbagai proses kerja keras, penyusunan skripsi yang berjudul “Implementasi Kombinasi ChaCha20 dan Modifikasi AES melalui *AddRoundkey* Pada Studi Kasus Keamanan *File*” akhirnya dapat terselesaikan. Selain sebagai salah satu syarat memperoleh gelar S-Kom, penulis berharap juga skripsi ini dapat bermanfaat bagi semua pihak yang berkepentingan. Oleh karena itu, penulis ingin mengungkapkan rasa terima kasih kepada pihak-pihak yang telah memberikan dukungan dan doa sehingga penulis dapat menyelesaikan skripsi ini tepat pada waktunya. Beberapa pihak tersebut antara lain:

1. **Kepada Mama** yang sangat penulis cintai yang tidak pernah lupa selalu memberikan dukungannya dan doa agar diberikan kemudahan dan kelancaran dalam penyusunan skripsi ini sehingga penulis dapat menyelesaikan studi Informatika di Universitas IT Telkom Surabaya.
2. **Kepada Alm.Ibu Suhartatik**, yang penulis cintai dan sayangi, yang telah memberikan motivasi untuk saya.
3. **Kepada Saudara Kembar**, yang selalu mendukung, terima kasih atas bantuan dan waktunya dalam memberikan masukan terkait susunan skripsi ini.
4. **Keluarga Besar penulis** yang telah memberikan motivasi dan kepercayaan besar sehingga penulis bisa mewujudkan impian untuk meraih gelar S-Kom.
5. **Dosen Pembimbing**, Pak Rizky Fenaldo Maulana, S.Kom., M.Kom. dan Tanzila Mustaqim, S.Kom., M.Kom. Yang telah memberikan solusi, waktu, dan bimbingan penuh kesabaran selama proses penyusunan skripsi ini.
6. **Teman-teman dekat**, termasuk Jus’an, Erica Dyah, Qori Emalia, Ardina Firdatana, dan Elma Saskia yang selalu memberikan semangat, mendengarkan keluh kesah, serta menemani dalam berbagai kondisi selama penulisan skripsi.
7. **Semua pihak lainnya**, yang tidak dapat disebutkan satu per satu, atas bantuan dan dukungannya selama proses ini.
8. **Diri penulis sendiri**, atas perjuangan dan kerja keras hingga akhirnya skripsi ini selesai tepat waktu.

Penulis menyadari bahwa skripsi ini masih jauh dari kesempurnaan. Oleh karena itu, saran dan kritik yang membangun sangat diharapkan untuk memperbaiki skripsi ini ke depannya.

Terima kasih.

### Wassalamu’alaikum Warrahmatullahi Wabarakatuh.

Surabaya, 24 Desember 2024 Penulis

Emilia Ramona

# DAFTAR ISI

[LEMBAR PERSETUJUAN ii](#_bookmark0)

[ABSTRAK iii](#_bookmark1)

[ABSTRACT iv](#_bookmark2)

[KATA PENGANTAR v](#_bookmark3)

[DAFTAR ISI vii](#_bookmark4)

[DAFTAR TABEL ix](#_bookmark5)

[DAFTAR GAMBAR x](#_bookmark6)

[DAFTAR RUMUS xi](#_bookmark7)

[BAB I PENDAHULUAN 1](#_bookmark8)

* 1. [Latar Belakang 1](#_bookmark9)
  2. [Rumusan Masalah 4](#_bookmark10)
  3. [Tujuan Penelitian 4](#_bookmark11)
  4. [Batasan dan Asumsi Penelitian 5](#_bookmark12)
  5. [Manfaat Penelitian 5](#_bookmark13)
     1. [Manfaat Akademis 5](#_bookmark14)
     2. [Manfaat Ilmiah 5](#_bookmark15)
  6. [Sistematika Penulisan 6](#_bookmark16)

[BAB II LANDASAN TEORI 7](#_bookmark17)

* 1. [Literatur Terkait Teori 7](#_bookmark18)
  2. [Kajian Teori 10](#_bookmark20)
     1. [Kriptografi 10](#_bookmark21)
     2. [AES (Advanced Encryption Standard) 12](#_bookmark24)
     3. [Chacha20 20](#_bookmark32)
     4. [Entropi 22](#_bookmark35)

[BAB III METODOLOGI PENELITIAN 24](#_bookmark37)

* 1. [Sistematika Penyelesaian Masalah 24](#_bookmark38)
  2. [Alat dan Bahan 24](#_bookmark39)
     1. [Alat 24](#_bookmark40)
     2. [Bahan 25](#_bookmark41)
  3. [Rancangan Penelitian 25](#_bookmark42)
     1. [Studi Literatur 27](#_bookmark45)
     2. [Pengumpulan Data 27](#_bookmark46)
     3. [Implementasi Modifikasi AES 28](#_bookmark47)
     4. [Implementasi Kombinasi AES Modifikasi dan ChaCha20 31](#_bookmark49)
     5. [Implementasi Antarmuka Pengguna (UI) 36](#_bookmark52)
     6. [Pengujian Algoritma 38](#_bookmark54)
     7. [Analisis Hasil 42](#_bookmark59)

[DAFTAR PUSTAKA 44](#_bookmark61)

# DAFTAR TABEL

[Tabel II.1 Literatur Penelitian Terkait Kriptografi 7](#_bookmark19)

[Tabel II.2 Panjang Kunci 13](#_bookmark25)

[Tabel II.3 Tabel Substitusi (S-Box) 16](#_bookmark28)

[Tabel II.4 Transformasi ShiftRows 17](#_bookmark29)

[Tabel III.1 Jenis, Ukuran, dan Isi File untuk Pengujian 40](#_bookmark58)

[Tabel III.2 Evaluasi Hasil 43](#_bookmark60)

# DAFTAR GAMBAR

[Gambar II.1 Alur Proses Enkripsi Dan Dekripsi 11](#_bookmark22)

[Gambar II.2 Proses Kunci Simetris 11](#_bookmark23)

[Gambar II.3.Transformasi ShiftRows 15](#_bookmark27)

[Gambar II.4 Proses AES 19](#_bookmark31)

[Gambar III.1 Diagram Alur Penelitian 26](#_bookmark43)

[Gambar III.2 Diagram Proses Enkripsi AES 28](#_bookmark48)

[Gambar III.3 Diagram Proses Double Encrypt 32](#_bookmark50)

[Gambar III.4 Proses Dekripsi 36](#_bookmark51)

[Gambar III.5 Desain Antarmuka Pengguna Enkripsi dan Dekripsi 37](#_bookmark53)

# DAFTAR RUMUS

[(II-1) Rumus AddRoundKey 14](#_bookmark26)

[(II-2) Operasi Perkalian MixColumns 17](#_bookmark30)

[(II-3) Rumus Keystream 22](#_bookmark33)

[(II-4) Rumus Proses Enkripsi XOR pada Keystream 22](#_bookmark34)

[(II-5) Entropi 22](#_bookmark36)

[(III-1) Pengujian Waktu Enkripsi 39](#_bookmark55)

[(III-2) Pengujian Waktu Dekripsi 39](#_bookmark56)

[(III-3) Pengujian Waktu Kecepatan 39](#_bookmark57)

# BAB I PENDAHULUAN

### Latar Belakang

Semakin meningkatnya teknologi yang semakin canggih membawa sebuah perubahan yang cukup signifikan. Perubahan tersebut dikenal dengan istilah Revolusi Industri 4.0 yang merupakan sebuah kemajuan teknologi yang sering digunakan oleh banyak orang. Revolusi industri 4.0 adalah sebuah data yang digerakkan dengan adanya campur tangan komputer menggunakan metode machine learning atau AI (Alimuddin et al., 2023). Menurut Klaus Schawab revolusi Industri 4.0 memiliki tantangan yang harus dihadapi salah satunya seperti keamanan *cyber* dan privasi data.

Maka dari itu, dengan semakin pesatnya informasi teknologi maka perlunya untuk mengamankan data atau *file*. Keamanan data atau *file* merupakan salah satu langkah yang mudah untuk melindungi sebuah informasi bersifat penting yang berasal dari akses tidak sah. Keamanan *file* ini sangat diperlukan pada era Revolusi Industri 4.0, karena maraknya serangan *cyber* dan peretasan komputer yang bisa mengganggu keberlangsungan suatu industri. Menurut (Mulyana*, 2016*), keamanan *file* merupakan salah satu aksi pencegahan dari sebuah penganggu pengguna *file* atau pengguna yang memakai jaringan yang tidak berkewajiban memiliki akses (Soesanto et al., 2023). Dengan semakin pesatnya dunia teknologi ini membuat keamanan *file* menjadi sangat penting karena untuk melindungi privasi dan informasi dari pihak yang tidak memiliki akses.

Keamanan *file* adalah serangkaian metode atau teknik yang untuk melindungi *file* dari akses yang tidak sah, seperti pencurian atau kerusakan. Tujuannya untuk menjaga informasi dalam *file* tetap rahasia, aman, dan hanya dapat di akses oleh pihak yang berwenang. Oleh karena itu, upaya untuk memastikan agar *file* terjaga dengan menggunakan metode ChaCha20 dan *Advanced Encryption Standard* (AES). ChaCha20 merupakan metode enkripsi yang cepat dan aman. ChaCha20 dipopulerkan oleh Daniel J.

Bernstein sebagai varian dari algoritma. Dalam algoritma ChaCha20 sangat populer dalam pengimplementasi enkripsi dikarenakan percampuran antara kecepatan dan keamanannya yang dapat diunggulkan. Chaca20 mempunyai desain yang hampir sama dengan Salsa20, pada algoritma ChaCha20 menggunakan kunci 256-bit yang mempunyai 8 putaran (Kebande, 2023).

Sedangkan AES (*Advanced Encryption Standard)* adalah sebuah algoritma yang melakukan transformasi terhadap blok data yang menggunakan kunci enkripsi yang sama. AES bekerja pada sistem yang melibatkan operasi matematika dengan level tingkat sulit (Saripa, 2023). Proses kerja enkripsi dasar AES adalah *subbytes* (menggunakan tabel S-Box untuk menggabungkan *byte* pada blok matriks), *shift row* (digunakan untuk menggeser baris matriks), *MixColumns* (melakukan pergeseran dari kanan ke kiri), dan *AddRoundkey* (blok di XOR kan dengan kunci yang diperluas) (Aryanto et al., 2023). Pada proses *dekripsi* AES terdapat tahap-tahapan kebalikan dari proses enkripsi. Dengan memakai kunci yang sama untuk *dekripsi*, blok data yang telah terenkripsi dapat dipulihkan kembali ke bentuk awalnya. algoritma AES adalah jenis *block chipper* yang memiliki panjang kunci 128 bit, 192 bit dan 256 bit (Surbakti & Purwanto, 2024).

Algoritma Chacha20 dan AES merupakan suatu enkripsi yang dapat di digunakan dalam menjaga keamanan *file*. Pada algoritma AES terdapat kelebihan salah satunya adalah sifat karakteristiknya yang di mana selalu ada bidang khusus untuk bilangan prima, memicu kekuatan AES(Surbakti & Purwanto, 2024) . AES juga memiliki sebuah kelemahan salah satunya yaitu algoritma kriptografi yang menggunakan kunci simetris dalam proses pengiriman dan penerimanya data. Namun, penggunaan kunci simetris ini memiliki risiko lebih tinggi terhadap kebocoran, terutama seiring waktu (Hidayatulloh et al., 2023). Pada algoritma ChaCha20 terdapat keunggulan yaitu menggunakan teknik kriptografi ARX (*addition, rotation*, XOR) yang memberikan performa lebih cepat, kompleksitas lebih rendah, dan mampu menghilangkan serangan waktu Klik atau ketuk di sini untuk memasukkan teks.

Meskipun algoritma AES dan Chacha20 telah efektif untuk menjaga peningkatan keamanan *file* dan implementasinya yang cukup rumit sehingga membuat orang yang tidak memiliki akses tidak dapat mengakses *file* tersebut. Hal ini tidak dapat kita pungkiri bahwa semakin pesatnya dunia teknologi membuat banyak orang berbondong-bondong menciptakan pembaruan yang lebih efektif. Metode *double encrypt* ini menawarkan kombinasi yang mampu mengurangi kelemahan dari setiap algoritma jika digunakan secara bersamaan, serta meningkatkan ketahanan terhadap berbagai jenis serangan kriptografi. *Double encrypt* merupakan Teknik kriptografi yang melibatkan proses enkripsi secara dua kali, yang di mana memiliki dua kunci yang berbeda untuk lebih meningkatkan keamanan. *Double encrypt* ini bertujuan untuk meningkatkan keamanan *file* lebih tinggi dibandingkan hanya 1 metode saja (Pangestu et al., 2022).

Pada penelitian terlebih dahulu yang berjudul “Implementasi Sistem Keamanan *File* Menggunakan Algoritma AES untuk Mengamankan *File* Pribadi” yang diteliti oleh Saripa Universitas Negeri Makassar. Dalam penelitian tersebut menjelaskan tentang keamanan *file* dengan menggunakan metode algoritma AES. Di mana hasil penelitian ini menunjukkan bahwa algoritma AES telah berhasil mengamankan *file* yaitu dokumen Word, PDF, Excel, dan Power Poin, *file* yang dienkripsi menjadi tidak terbaca oleh pihak yang tidak memiliki akses sehingga dapat menjaga kerahasiaan data dan AES mampu menjaga keamanan data dengan baik.

Dalam perbedaan penelitian terdahulu dan penelitian saya akan menggunakan double encrypt untuk keamanan *file* dan juga saya akan menggunakan AES dengan menambahkan metode ChaCha20 dan melakukan modifikasi pada bagian AES. Modifikasi menggunakan penambahan *AddRoundkey* sebelum dengan bertujuan agar proses enkripsi akan lebih susah untuk dipecahkan. Dalam algoritma AES, langkah *AddRoundKey* merupakan komponen krusial yang menggabungkan data dengan kunci enkripsi menggunakan operasi XOR. Penelitian (Luong, 2023) oleh menunjukkan bahwa memodifikasi transformasi *AddRoundKey*

dapat meningkatkan keamanan algoritma dengan menciptakan hubungan yang lebih kompleks antara *plainteks* dan *cipherteks*. Penelitian ini menyoroti pentingnya penyesuaian langkah *AddRoundKey* untuk meningkatkan ketahanan algoritma AES terhadap serangan kriptografi, seperti *differential cryptanalysis* dan linear *cryptanalysis*.

Dengan mendasarkan pada penelitian tersebut, modifikasi algoritma AES melalui penambahan langkah *AddRoundKey* sebelum *MixColumns* bertujuan untuk lebih memperkuat difusi data pada tahap awal enkripsi. Penambahan ini diharapkan dapat meningkatkan kerandoman hasil enkripsi dan mengurangi kemungkinan eksploitasi pola oleh pihak yang tidak berwenang, sehingga keamanan algoritma menjadi lebih optimal. Berdasarkan dari latar belakang di atas penelitian ini memiliki tujuan untuk meningkatkan keamanan *file* agar lebih sulit untuk diretas atau diakses pihak yang tidak berwenang. Oleh karena itu penelitian ini dianggap penting karena maraknya kasus peretasan data oleh oknum yang tidak memiliki akses.

### Rumusan Masalah

1. Bagaimana implementasi dari modifikasi AES dengan penambahan *AddRoundkey* sebelum *MixColumns* dalam meningkatkan keamanan *file*?
2. Bagaimana performa kombinasi antara AES modifikasi dan ChaCha20 dalam hal waktu enrkipsi, waktu *dekripsi* , kecepatan enkripsi, dan kecepatan *dekripsi*?

### Tujuan Penelitian

1. Mampu mengidentifikasi *AddRoundkey* yang akan digunakan dalam menjaga keamanan *file* dan mengimplementasi kemampuan Chacha20 dan AES yang telah dimodifikasi dalam proses menjaga keamanan *file*.
2. Mampu mengevaluasi performa algoritma ChaCha20 dan AES yang telah dimodifikasi berdasarkan waktu enkripsi, waktu *dekripsi*, kecepatan enkripsi, dan kecepatan *dekripsi*, untuk menentukan efisiensi

dan efektivitas kombinasi kedua algoritma tersebut dalam menjaga keamanan *file*

### Batasan dan Asumsi Penelitian

1. Penelitian ini berfokus pada jurnal atau artikel ilmiah dengan batasan tahun 2021-2024 yang membahas tentang :
   * Keamanan *file*
   * Algoritma ChaCha20
   * *Advanced Encryption Standard* (AES)
   * Double encryption
2. Penelitian dilakukan dengan pengujian pada *file* dengan ukuran dan format sebagai berikut :
   * Format PDF dan Word dengan variasi ukuran *file* 100 KB - 100MB.
   * Format *file* teks (TXT) dengan ukuran maksimal 300 byte Batasan ukuran *file* digunakan untuk memastikan pengujian dilakukan secara efektif tanpa mengurangi kinerja proses enkripsi dan *dekripsi*.

### Manfaat Penelitian

### Manfaat Akademis

* + - 1. Memberikan inovasi terbaru mengenai pelindungan keamanan

*file* dengan menambahkan *AddRoundkey* pada algoritma AES.

* + - 1. Memudahkan peneliti lainnya dalam kinerja *AddRoundkey* pada algoritma AES yang berfungsi untuk keamanan *file***.**

### Manfaat Ilmiah

* + - 1. Menjadi sumber acuan dan sumber referensi bagi peneliti lainnya yang akan meneliti topik ChaCha20 dan AES modifikasi, terutama dalam konteks keamanan data dan performa enkripsi.
      2. Membuktikan hipotesis apakah AES yang dimodifikasi dengan tambahan *AddRoundkey* dapat meningkatkan performa keamanan dan efisiensi dibandingkan dengan AES standart, dan sekaligus menunjukkan kinerja kombinasi ChaCha20 dan AES dalam pengamanan *file* berbagai format dan ukuran.

### Sistematika Penulisan

## BAB 1 PENDAHULUAN

Menjelaskan latar belakang penelitian, rumusan masalah, tujuan penelitian, batasam penelitian, manfaat penelitian, serta sistematika penulisan secara umum.

## BAB 2 LANDASAN TEORI

Membahas teori-teori yang relavan dengan penelitian, termasuk algoritma ChaCha20, AES, konsep kriptografi, Serta tinjauan terhadap penelitian terdahulu yang terkait.

## BAB 3 METODOLOGI PENELITIAN

Menjelaskan metode yang digunakan dalam penelitian, termasuk algoritma yang diterapkan (modifikasi AES dan ChaCha20 dengan enkripsi ganda). Diuraikan juga alat dan bahan yang digunakan, serta tahapan penelitian yang meliputi implementasi algoritma, pengujian parameter kinerja (waktu dan kecepatan enkripsi, dekripsi, serta analisis entropi), dan evaluasi hasil pengujian pada berbagai jenis file (PDF, Word, TXT).

## BAB 4 IMPLEMENTASI DAN PENGUJIAN

Menyajikan hasil implementasi algoritma yang dikembangkan, pengujian algoritma terhadap berbagai jenis *file*, serta analisis data hasil pengujian**.**

## BAB 5 EVALUASI HASIL PENGUJIAN

Menyajikan hasil implementasi dari aplikasi yang telah dirancang, pengujian fungsionalitas sistem, serta evaluasi kinerja aplikasi terhadap berbagai skenario penggunaan. Analisis data hasil pengujian juga dilakukan untuk menilai keberhasilan aplikasi dalam mencapai tujuan penelitian.

## BAB 6 KESIMPULAN DAN SARAN

Merangkum temuan utama penelitian serta memberikan saran untuk pengembangan lebih lanjut di masa depan.

# BAB II LANDASAN TEORI

### Literatur Terkait Teori

Keamanan informasi menjadi salah satu isu yang paling mendesak di era digital saat ini, di mana data sensitif sering kali menjadi target serangan siber yang beragam. Dengan meningkatnya ancaman seperti serangan *man-in-the- middle*, *eavesdropping*, dan *DDoS*, penting bagi organisasi dan individu untuk menerapkan metode enkripsi yang efektif untuk melindungi data mereka. Berbagai algoritma enkripsi telah dikembangkan untuk memenuhi kebutuhan ini, dengan *Advanced Encryption Standard* (AES) dan ChaCha20 menjadi dua di antara yang paling banyak digunakan. Penelitian-penelitian terkini menunjukkan bahwa kedua algoritma ini tidak hanya menawarkan tingkat keamanan yang tinggi, tetapi juga dapat diimplementasikan dalam berbagai konteks, mulai dari pengamanan *file* hingga enkripsi konten *livestreaming*.

Pada [Tabel II.1 Literatur Penelitian Terkait Kriptografi](#_bookmark19) menyajikan ringkasan dari beberapa penelitian yang mengeksplorasi efektivitas dan penerapan algoritma AES dan ChaCha20 dalam berbagai aplikasi. Setiap penelitian memberikan wawasan tentang tujuan, hasil, dan persamaan dalam pendekatan yang digunakan, serta menyoroti inovasi yang dilakukan untuk meningkatkan keamanan data. Dengan memahami hasil-hasil ini, kita dapat lebih menghargai perkembangan teknologi enkripsi dan aplikasinya dalam menjaga kerahasiaan serta integritas informasi di dunia yang semakin terhubung.

### Kajian Teori

### Kriptografi

Menurut Menezes, Oorschot, dan Vantone (1996), Kriptografi adalah studi teknik matematika yang berkaitan dengan keamanan informasi seperti kerahasiaan, *autentikasi* keaslian data (Belvin Tannadi, 2022). Kriptografi dapat diartikan sebagai sebuah proses untuk melindungi data atau sebuah sistem yang bertujuan untuk melindungi informasi rahasia dengan cara yang tidak dipahami oleh pihak yang tidak berwenang. Dalam kriptografi yang meliputi proses enkripsi dan proses *dekripsi*. Di dalam Kriptografi terdapat tiga fungsi dasar yang meliputi :

* + - 1. Enkripsi merupakan proses data asli (*plainteks*) yang akan diubah menjadi data yang tidak bisa terbaca oleh pihak yang tidak berwenang (*chiperteks*).
      2. *Dekripsi* merupakan proses pengembalian dari data yang tidak bisa terbaca(*chiperteks*) menjadi data terbaca atau data asli (*plainteks*).
      3. Kunci merupakan proses yang dipakai untuk melakukan proses enkripsi dan *dekripsi*, terdapat 2 jenis kunci, yaitu kunci rahasia (*private key*) dan kunci publik (*public key*).

Pada [Gambar II.1 Alur Proses Enkripsi Dan *Dekripsi*](#_bookmark22)

memperlihatkan alur enkripsi dan *dekripsi.* Proses ini dimulai

dengan *plainteks* yang kemudian dienkripsi dengan menggunakan kunci menjadi *chiperteks*. *Chiperteks* tersebut kemudian *didekripsi* kembali menggunakan kunci yang sama untuk mendapatkan kembali *plainteks*.

*Gambar II.1 Alur Proses Enkripsi Dan Dekripsi*

Berikut algoritma Kriptografi yang berdasarkan kunci :

a. Algoritma Simetris

Algoritma simetris adalah algoritma yang merujuk pada metode enkripsi dan *dekripsi* yang di mana algoritma ini memiliki kunci tunggal atau sama. Sehingga dapat disimpulkan algoritma ini mewajibkan pengirim dan penerima harus menyetujui satu kunci yang digunakan dalam proses enkripsi dan *dekripsi* data. Namun, kelemahan dari algoritma ini yaitu tergantung pada rahasia kunci. Apabila kunci tersebut tersebar maka akan dengan mudah pihak yang tidak memiliki akses dapat melakukan proses enkripsi dan *dekripsi* data secara mudah.

*Gambar II.2 Proses Kunci Simetris*

Pada [Gambar II.2 Proses Kunci Simetris](#_bookmark23) menjelaskan proses enkripsi dan *dekripsi* menggunakan kunci simetris. Dimulai dengan pesan yang dikirimkan, kemudian dienkripsi menjadi kode yang tidak bisa dibaca menggunakan kunci simetris. Kode tersebut kemudian dikirim dan di *dekripsi* di sisi penerima menggunakan kunci simetris yang sama, sehingga pesan dapat dibaca kembali. Pada contoh gambar, pesan "Selamat Datang!" dienkripsi menjadi kode "9CDF4837 AB672019" menggunakan kunci simetris yang sama. Kemudian di *dekripsi* kembali ke pesan awal "Selamat Datang!" Kunci simetris merupakan kunci yang sama digunakan untuk enkripsi dan *dekripsi*, sehingga pengirim dan penerima pesan harus memiliki kunci yang sama agar dapat berkomunikasi.

### AES (Advanced Encryption Standard)

Advanced Encryption Standard (AES) adalah salah satu algoritma enkripsi yang paling terkenal dan banyak digunakan, serta terus dikembangkan sebagai pengganti algoritma Data Encryption Standard (DES). AES adalah algoritma enkripsi yang bekerja dengan memproses data dalam blok dengan ukuran minimal 128 bit, dan melalui beberapa tahapan untuk mengenkripsi data. Tujuan utama dari AES adalah untuk mengamankan data rahasia sehingga tidak dapat dibaca oleh pihak yang tidak berwenang (Wahyu Putra, 2023). Pada [Tabel II.2 Panjang Kunci](#_bookmark25) dalam proses enkripsi *Advanced Encryption Standard* (AES) menggunakan ukuran blok dengan panjang sebesar 126 bit, dengan pilihan panjang kunci yang bervariasi, yaitu 128 bit, 192 bit, dan 256 bit (Sekar Putri Ananda et al., 2022). Proses enkripsi dalam algoritma ini dilakukan melalui beberapa tahap yang dikenal sebagai ronde. Jumlah ronde yang digunakan bergantung pada panjang kunci yang akan dipilih. Berikut adalah hubungan panjang kunci dan banyaknya ronde.

Algoritma AES juga menawarkan tingkat keamanan yang sangat tinggi dikarenakan ukuran kunci dan jumlah ronde yang ditentukan. Ukuran kunci menentukan jumlah kemungkinan kombinasi kunci, sehingga membuat serangan *bruteforce* menjadi sangat sulit dilakukan. Di sisi lain, jumlah ronde yang besar dapat menghasilkan keamanan yang kuat, sehingga perubahan kecil pada *plainteks* dapat memengaruhi seluruh bagian *cipherteks*. Selain keunggulan keamanannya, AES juga memiliki efisiensi tinggi dalam hal waktu dan ruang. Algoritma ini dapat diimplementasikan baik pada perangkat keras maupun perangkat lunak, menjadikannya ideal untuk berbagai aplikasi, termasuk perlindungan data (Assyahid, 2024). Pada tahap awal enkripsi, data masukan yang telah disalin ke dalam matriks *state* akan menjalani transformasi *AddRoundkey*. Selanjutnya, matriks *state* akan melalui proses transformasi secara berulang, mencakup *SubBytes*, *ShiftRows*, *MixColumns*, dan *AddRoundkey*, dengan jumlah pengulangan sesuai dengan nilai Nr (Handoyo & Subakti, 2020). *Dekripsi* adalah proses mengembalikan *ciphertext* menjadi plaintext, atau mengubah data terenkripsi kembali ke bentuk aslinya. Pada proses *inverse cipher*, transformasi yang digunakan meliputi *InvShiftRows*, *InvSubBytes*, *InvMixColumns*, dan *AddRoundkey*. Berikut ini adalah cara kerja algoritma AES secara umum:

### Proses Enkripsi AES Modifikasi

* + - * 1. *AddRoundkey*

*AddRoundkey* adalah proses menggabungkan state masukan dengan kunci ronde dengan menggunakan operasi XOR. Dalam proses transformasi enkripsi dan *dekripsi* pada AES, dilakukan operasi penambahan kunci putaran (*round key*)

pada state menggunakan operasi XOR (Djong & Siswanto, 2022). Setiap kunci putaran terdiri dari *Nb word* di mana setiap *word* tersebut akan digabungkan dengan *word* atau kolom yang sesuai pada *state* . Secara matematis, hal ini (Putri et al., 2021).

Keterangan Algoritma dari *AddRoundkey* di mana :

Pada [Gambar II.3.Transformasi ShiftRows](#_bookmark27) di atas di mana S0,0 merupakan elemen di baris ke-0, kolom ke-0, S3,3 merupakan elemen di baris ke-3, kolom ke-3, dan 𝑊3,3 merupakan elemen di baris ke-3, kolom ke-3. Pada label S0,0 dan lainnya merupakan state saat ini, state ini representasi internal data yang sedang dienkripsi ataupun *didekripsi* dan setiap elemen adalah *byte* dari blok data input. Sedangkan pada label 𝑊0,0 merupakan kunci ronde yang dihasilkan dari proses *keyexpansion*, di mana ini bagian dari kunci utama yang dipecah untuk menjadi beberapa ronde dan setiap elemen adalah *byte* dari kunci untuk ronde tertentu. Operasi XOR dilakukan antara setiap *byte* pada *state* dengan *byte* kunci ronde yang sesuai. Hasil di atas dihasilkan dari operasi XOR yang akan digunakan untuk langkah transformasi berikutnya.

* + - * 1. *SubBytes*

Transformasi *transformasi byte* di mana setiap elemen dalam matriks *state* dipetakan menggunakan tabel substitusi yang dikenal sebagai S-Box. Setiap *byte* dalam matriks *state* direpresentasikan sebagai S’[r, c], yang merupakan elemen hasil dari perpotongan antara baris (x) dan kolom (y) (Kurnia Mahesa Naibaho et al., 2021). S-Box berupa matriks

berukuran 16×16 yang berisi nilai-nilai heksadesimal di mana nilai-nilai ini diperoleh melalui perhitungan tertentu (Hena & Kusumaningsih, 2024). Berikut ini tabel S-Box :

Pada [Tabel II.3 Tabel Substitusi (S-Box)](#_bookmark28) sebagai contoh, nilai heksadesimal {62} merujuk ke baris 7 dan kolom 3 dalam S-Box, yang berisi nilai {aa}. Dengan demikian, nilai

{62} dipetakan menjadi nilai {aa}.

* + - * 1. *ShiftRows*

Transformasi *ShiftRows* merupakan proses pergeseran *byte* di mana *byte* paling kiri dipindahkan ke posisi paling kanan (rotasi bit). Jumlah pergeseran dalam melakukan *ShiftRows* ini dilakukan perbedaan setiap baris dalam matriks *state*. Pada baris pertama, tidak ada pergeseran yang dilakukan. Kemudian pada baris kedua barulah mengalami pergeseran satu *byte* ke kiri, lalu baris ketiga digeser dua *byte* ke kiri, dan baris keempat digeser tiga *byte* ke kiri. Tujuan dari proses ini adalah untuk menciptakan *diffusion*(), yang memiliki tujuan untuk membantu menyebarkan efek transformasi non-linear ke seluruh baris dalam matriks *state* untuk proses selanjutnya (Cristy & Riandari, 2021). Proses pergeseran ini bertujuan untuk mencampur elemen-elemen dalam *state* sehingga memperkuat kerahasiaan data.

*MixColumns* adalah proses transformasi di mana setiap elemen dalam blok *cipher* dikalikan dengan matriks . Transformasi ini bertujuan untuk mencampur elemen- elemen dalam setiap kolom *state* berguna meningkatkan difusi data dan memperkuat keamanan enkripsi . Transformasi ini bekerja dengan membagi matriks atau *state* menjadi empat bagian, untuk setiap kolomnya (Widyawan & Imelda, 2021). Setiap kolom dalam matriks *state* diperlakukan sebagai *polinomial* dengan empat bagian dalam *Galois Field* (GF), pada persamaan [(II-2)](#_bookmark30) dilakukan Algoritma dari *AddRoundkey* di mana :

𝑋8 = Elemen dengan derajat ke-8 yang akan muncul sebagai 100000000 (1 diposisi ke 9 dari kanan) di bilangan biner.

X = Menunjukkan elemen dengan derajat ke 1 yang akan muncul sebagai 00000010 ( 1 di posisi ke 2 dari kanan

) dalam bilangan biner

1 = Elemen konstanta yang akan muncul sebagai

00000001 ( 1 diposisi ke 1 dari kanan), dimana elemen 1 ini menyatakan bahwa *polinomial* ini memiliki koefisien konstan.

Proses *MixColumns* juga dapat direpresentasikan sebagai operasi perkalian matriks, di mana setiap kolom matriks dikalikan dengan matriks tertentu dalam *Galois Field* (Cristy & Riandari, 2021).

* + - 1. **Proses *dekripsi* AES :**

Proses *dekripsi* merupakan kebalikan dari proses enkripsi. *Transformasi bytes* yang digunakan dalam *dekripsi* mencakup *AddRoundKey*, *InvShiftRows*, *InvSubBytes,* dan *InvMixColumns*. Pada iterasi pertama *dekripsi*, dilakukan *transformasi AddRoundKey, Inverse ShiftRows, dan Inverse SubBytes. Ciphertex*t kemudian akan mengalami transformasi *AddRoundKey* .

* + - * 1. *InvShiftRows*

Transformasi *byte* yang merupakan kebalikan dari proses *ShiftRows*. Dalam transformasi ini, bit-bit pada *state* digeser ke arah kanan, berbeda dengan *ShiftRows* yang melakukan pergeseran bit ke arah kiri.

* + - * 1. *InvSubBytes*

Transformasi *byte* yang menjadi kebalikan dari *SubBytes*. Dalam proses ini, setiap elemen pada *state* dipetakan menggunakan tabel substitusi *Inverse* S-Box.

* + - * 1. *InvMixColumns*

Setiap kolom dalam *state* dikalikan dengan matriks khusus yang digunakan dalam proses perkalian pada algoritma AES (Syahputra Sianipar et al., 2024)

*Gambar II.4 Proses AES*

Pada [Gambar II.4 Proses AES](#_bookmark31) sebelah kiri menunjukkan tahapan proses enkripsi pada algoritma AES Modifikasi. Proses dimulai dengan memasukkan plainteks, kemudian melalui serangkaian langkah seperti *AddRoundkey*, SubBytes, ShiftRows, MixColumns, dan diakhiri dengan *AddRoundkey* tambahan. Langkah ini diulang sebanyak Nr−1 ronde untuk meningkatkan keamanan data, menghasilkan cipherteks di akhir proses. Flowchart sebelah kanan menjelaskan tahapan proses *dekripsi*, yang merupakan kebalikan dari proses enkripsi. Proses dimulai dengan memasukkan cipherteks, kemudian melalui langkah-langkah seperti *AddRoundkey*, Inverse ShiftRows, Inverse SubBytes, dan Inverse MixColumns. Setelah semua ronde selesai, data dikembalikan ke bentuk plainteks asli.

### Chacha20

Algoritma ChaCha20 adalah salah satu jenis *stream cipher* yang dikembangkan sebagai varian dari algoritma Salsa20. Algoritma ini berbasis desain yang serupa dengan Salsa20, namun ChaCha20 meningkatkan tingkat *diffusion* pada setiap ronde untuk memperkuat keamanan proses enkripsi (Anindita & 18219086, 2023). Chacha20 beroperasi pada unit data 32-bit menggunakan kunci 256-bit. Algoritma ini menghasilkan *output* berupa *stream* kunci sepanjang 512-bit, yang kemudian akan di XOR kan dengan *plainteks* untuk menghasilkan *chiperteks*(Alyas & Abdullah, 2021b). ChaCha20 dimulai dengan membuat *state* 512-bit, yang berupa *array* yang berukuran 16x32-bit. State ini disusun dari empat konstanta 32 bit, kunci rahasia sepanjang 256-bit, penghitung blok 32-bit, serta nonce Ini menunjukkan struktur *state* ChaCha20 yang terdiri dari matriks 4x4 dengan 16 elemen 32-bit. Matriks ini diinisialisasi dengan empat konstanta 32-bit, kunci 256-bit yang dibagi menjadi 8 bagian, penghitung 32 blok 32-bit, dan Nonce 96-bit. Elemen-elemen ini digunakan untuk menghasilkan *keystream* yang kemudian di XOR kan dengan *plainteks* untuk menghasilkan *chiperteks*. Di mana *const* (konstanta) ini berisi empat nilai tetap 32-bit, *key* (kunci) yang terbagi 8 bagian 32-bit dari kunci rahasia 256-bit, *counter* (penghitung), dan Nonce (nilai acak) yang berukuran 96-bit.

Langkah – Langkah Algoritma ChaCha20 :

* + - 1. *Inisialisasi* State

Algoritma ChaCha20 dimulai dengan membangun sebuah state internal berbentuk matriks 4x4 (32-bit) (Barbero et al., 2022) . Matriks ini terdiri dari:

* + - * 1. Baris pertama: Berisi 4 konstanta tetap (128-bit), yang digunakan untuk mengenali algoritma.
        2. Baris kedua dan ketiga: Berisi kunci rahasia 256-bit (8 elemen 32-bit), yang digunakan untuk proses enkripsi.
        3. Baris keempat: Berisi kombinasi 64-bit counter dan 64-bit nonce (masing-masing terdiri dari 2 elemen 32-bit).
      1. *Quarter Round*

Operasi ini memodifikasi 4 elemen dari matriks ( a, b, c, d ) (Almukhlifi & Vora, 2020)

𝑎 += 𝑏; 𝑑 ^ = 𝑎; 𝑑 <<<= 16;

𝑐 += 𝑑; 𝑏 ^ = 𝑐; 𝑏 <<<= 12;

𝑎 += 𝑏; 𝑏 ^ = 𝑎; 𝑑 <<<= 8;

𝑐 += 𝑑; 𝑏 ^ = 𝑐; 𝑏 <<<= 7;

* + - 1. *Double Round*

*Double Round* adalah kombinasi dari dua *Quarter Round* yang dijalankan secara berurutan. Setiap iterasi *Double Round* mencakup:

* + - * 1. Operasi pada kolom matriks *(Column Round):*

melakukan operasi Quarter Round pada setiap kolom matriks 4x4. Tujuannya adalah untuk mencampur data antar elemen dalam kolom, sehingga meningkatkan difusi data secara vertikal.

* + - * 1. Operasi pada diagonal matriks (Diagonal Round):

Setelah operasi pada kolom selesai, langkah berikutnya adalah menerapkan Quarter Round pada elemen-elemen diagonal dalam matriks.

Proses Double Round dilakukan sebanyak 10 iterasi, menghasilkan total 20 ronde Quarter Round untuk setiap blok data yang diproses.

* + - 1. Finalisasi State
      2. *Keystream Generation*

Setelah 20 ronde selesai*, state* akhir ditambahkan kembali ke

### Entropi

Entropi dalam teori informasi digunakan untuk mengukur keacakan data, yang mencerminkan kekuatan algoritma enkripsi. Semakin tinggi nilai entropi, semakin acak data tersebut, yang berarti algoritma enkripsi lebih tahan terhadap serangan (Wahyudi et al., 2024). Entropi dihitung dengan menggunakan rumus pada

Dimana:

* 𝐻𝑚 : Menunjukkan entropi data, dimana tingkat keacakan dalam

# BAB III METODOLOGI PENELITIAN

### Sistematika Penyelesaian Masalah

Penelitian ini bersifat kuantitatif dengan tujuan mengevaluasi performa algoritma AES modifikasi dan kombinasi ChaCha20 dan AES modifikasi. Penilaian performa ini dilakukan berdasarkan waktu enkripsi, waktu *dekripsi*, kecepatan enkripsi, dan kecepatan *dekripsi*, yang akan dianalisis menggunakan data hasil pengujian pada berbagai jenis *file* dengan format PDF, Word dan TXT, yang hasilnya berbentuk tabel untuk mempermudah analisis perbandingan.

### Alat dan Bahan

### Alat

1. Perangkat Keras (Hardware) Laptop dengan spesifikasi :
   1. Prosesor : Intel *Core* I5
   2. RAM : 8 GB
   3. Penyimpanan : 256 GB SSD atau lebih
2. Perangkat Lunak (*Software*)
   1. *Python* :

Menggunakan *library cryptography, pycryptodome*, dan *library* lain yang mendukung algoritma AES dan ChaCha20. Berikut beberapa pustaka yang digunakan :

* + - PyCryptodome : Digunakan Iplementasi algoritma AES dan ChaCha20
    - Cryptography : Implementasi kemanan tambahan (jika nanti dibutuhkan )
    - Numpy : Digunakan untuk analisis data (perhitungan entropi)
    - Time : Mencatat waktu eksekusi algoritma, baik untuk enkripsi maupun dekripsi.
  1. *Text* Editor :

Visual Studio Code dan *Jupyter Notebook*

* 1. Sistem Operasi : Windows

### Bahan

1. Data set *file* :
   1. *File* PDF : Ukuran *file* 100 KB – 100 MB
   2. *File* Word : Ukuran *file* 100 KB – 100 MB
   3. *File* TXT : Ukuran *file* maksimal 300 *byte*
2. Literatur :

Jurnal atau artikel ilmiah tentang AES, ChaCha20 dan *Double Encrypt* (tahun 2020-2024)

1. *Spreadsheet* Analisis Data :

Format CSV (Comma Separated Values)

### Rancangan Penelitian

Pada penelitian ini, dilakukan tahapan yang dirancang secara sistematis untuk mencapai tujuan utama, yaitu Implementasi kombinasi Advanced Encryption Standard (AES) yang dimodifikasi dan ChaCha20. Proses penelitian dimulai dengan studi literatur untuk memahami konsep dasar dan pengembangan terbaru terkait algoritma kriptografi. Setelah itu, dilakukan pengumpulan data yang relevan sebagai dasar implementasi. Tahapan selanjutnya mencakup implementasi algoritma modifikasi AES serta kombinasi algoritma tersebut dengan ChaCha20. Untuk mengukur efektivitas dan efisiensi algoritma yang dikembangkan, pengujian dilakukan pada berbagai jenis *file* seperti PDF, Word, dan TXT. Analisis hasil dari pengujian ini memberikan dasar untuk menarik kesimpulan dan memberikan rekomendasi untuk pengembangan lebih lanjut.

*Gambar III.1 Diagram Alur Penelitian*

[Gambar III.1 Diagram Alur Penelitian](#_bookmark44) ini terdapat beberapa tahapan yaitu:

1. Mulai adalah tahap awal dari penelitian. Pada tahap ini menentukan tujuan dan menyusun langkah-langkah yang akan dilakukan dalam penelitian.
2. Studi literatur dilakukan pengumpulan referensi yang berkaitan dengan algoritma AES, ChaCha20 dan *Double Encrypt. Referensi* berupa jurnal, artikel ilmiah.
3. Pengumpulan data untuk mengumpulkan *file* atau data set yang akan digunakan untuk pengujian algoritma. Format *file* yang digunakan adalah PDF, Word dan TXT.
4. Implementasi modifikasi AES, pada tahap ini algoritma AES dimodifikasi dengan menambahkan langkah *AddRoundkey* sebelum *Mixcolumns*. Tujuannya adalah untuk meningkatkan keamanan dan kecepatan enkripsi dibandingkan dengan AES standar.
5. Implementasi kombinasi AES modifikasi dan ChaCha20, menggabungkan algoritma AES yang sudah dimodifikasi dengan ChaCha20 untuk melakukan *double encypt*. Di mana proses ini bertujuan untuk meningkatkan keamanan *file* lebih baik daripada menggunakan *single encrypt*.
6. Pengujian algoritma, untuk menguji performa algoritma yang sudah diimplementasikan, di mana data set yang dikumpulkan sebelumnya itu digunakan untuk uji coba. Parameter yang diukur pada pengujian ialah

waktu enkripsi dan *dekripsi,* kecepatan enkripsi dan *dekripsi*, dan hasil pengujian ini akan digunakan untuk mengevaluasi kinerja algoritma.

1. Analisis hasil, data pengujian yang sudah terkumpul akan dianalisis. Analisis dilakukan untuk membandingkan performa AES standar, AES yang sudah dimodifikasi, dan kombinasi AES modifikasi dan ChaCha20. Tujuannya sendiri untuk mengetahui algoritma mana yang lebih efisien.
2. Selanjutnya ke kesimpulan dan saran, pada tahap ini merangkum hasil penelitian termasuk kelebihan dan kekurangan algoritma yang sudah diuji. Saran juga untuk pengembangan penelitian lebih lanjut
3. Selesai

### Studi Literatur

Studi literatur dilakukan untuk memahami konsep dasar dan perkembangan terbaru terkait algoritma kriptografi yang digunakan dalam penelitian ini, yaitu Advanced Encryption Standard (AES) dan ChaCha20. Proses ini melibatkan pengumpulan referensi dari jurnal, paper, dan buku yang membahas metode enkripsi, keamanan data, serta teknik evaluasi seperti entropi dan analisis performa. Literatur yang dikaji juga mencakup penelitian sebelumnya tentang algoritma kriptografi untuk keamanan *file*, sehingga memberikan dasar teori yang kuat untuk modifikasi algoritma dan penerapan metode double encryption dalam penelitian ini.

### Pengumpulan Data

Data yang digunakan dalam penelitian ini dibuat sendiri dengan mengikuti ketentuan yang telah ditentukan. Format data meliputi *file* PDF, Word, dan TXT dengan ukuran bervariasi, mulai dari 100 KB hingga 100 MB untuk *file* PDF dan Word, serta hingga 300 *byte* untuk *file* TXT. Data ini dirancang sedemikian rupa agar sesuai dengan kebutuhan pengujian algoritma enkripsi, seperti mengukur waktu enkripsi, waktu *dekripsi*, kecepatan, dan tingkat keacakan menggunakan analisis entropi. Variasi format dan ukuran data ini

bertujuan untuk memastikan bahwa pengujian mencakup berbagai jenis isi *file*.

### Implementasi Modifikasi AES

Pengembangan algoritma *Advanced Encryption Standard* (AES) dilakukan dengan memodifikasi. Modifikasi dilakukan dengan menambahkan *AddRoundkey* sebelum *Mixcolumns*. Penambahan ini bertujuan untuk memperkuat keamanan dan meningkatkan kompleksitas pada proses enkripsi, sehingga lebih sulit untuk diretas dengan pihak tidak berwenang. Berikut adalah proses alur dari modifikasi algoritma AES.

*Gambar III.2 Diagram Proses Enkripsi AES*

Pada [Gambar III.2 Diagram Proses Enkripsi AES](#_bookmark48) ini digunakan untuk mengamankan *file* dengan melakukan tahap enkripsi terhadap data *plainteks* yang menggunakan kunci tertentu. Dalam penelitian ini, modifikasi dilakukan dengan menambahkan langkah *AddRoundkey* untuk meningkatkan tingkat keamanan dan kompleksitas enkripsi. Berikut langkah-langkah proses modifikasi AES .

1. Input *file* dan kunci

Proses enkripsi dimulai dengan memasukkan *file* input (*file* asli yang akan dienkripsi) dan kunci enkripsi. *File* input dibaca dalam mode biner, diubah menjadi *byte array*, lalu dipecah menjadi blok-blok data berukuran 16 *byte*. Jika panjang data tidak habis dibagi 16 *byte, padding* dilakukan dengan metode PKCS#7 untuk melengkapinya.

Contoh :

* 1. *File* input : contoh.pdf

Isi *file* dalam biner : [0x25, 0x50, 0x44, 0x46, ...]

*Byte array* : [37, 80, 68, 70, 10, 0, 45, 78, ...]

* 1. Kunci enkripsi :

Kunci : surabayaitelkom

ASCII : [115, 117, 114, 97, 98, 97, 121, 97, 105, 116, 116,

101, 108, 107, 111, 109]

* 1. Matriks data *file* :

|  |  |  |  |
| --- | --- | --- | --- |
| [37, | 80, | 68, | 70] |
| [10, | 0, | 45, | 78] |
| [255, | 100, | 32, | 12] |
| [80, | 50, | 60, | 72] |

1. *AddRoundkey*

Merupakan tahap awal proses enkripsi, di mana *plainteks* ini digabungkan dengan kunci awal yang menggunakan operasi logika yaitu XOR. Di mana blok data *file* digabungkan dengan kunci awal (W0-W3) menggunakan operasi XOR untuk menambahkan lapisan keamanan awal.

Contoh :

* 1. Proses XOR :

37 XOR 115 = 86

80 XOR 117 = 37

68 XOR 114 = 54

70 XOR 97 = 39

* 1. Matriks setelah *AddRoundkey* :

[86, 37, 54, 39]

[108, 97, 76, 47]

[150, 16, 84, 97]

[28, 89, 83, 37]

1. Proses ronde 1-9
   1. *SubBytes*

Pada proses ini, nantinya setiap *byte* pada matriks *state* ( data yang diperoleh *AddRounKey*) akan diubah menggunakan tabel substitusi yang biasa disebut dengan S-Box. *Tranformasi* ini bertujuan untuk meningkatkan keacakan data agar sulit diprediksi.

Contoh proses :

*Byte* 37 diganti dengan nilai dari tabel S-Box jadi 120

* 1. *ShiftRows*

Setiap baris dalam matriks *state* nantinya digeser ke kiri pada baris pertama tidak ada pergeseran, baris kedua di geser 1 *byte* ke kiri, baris ketiga ini digeser 2 *byte* ke kiri, dan terakhir baris keempat yang nantinya digeser 3 *byte* ke kiri. Contoh proses :

[244, 120, 99, 200] [244, 120, 99, 200]

[60, 107, 55, 175] [107, 55, 175, 60]

[12, 243, 172, 85] [172, 85, 12, 243]

[180, 97, 92, 65] [65, 180, 97, 92]

* 1. *AddRoundkey* (W4-W7)

Setelah *ShiftRows* dilakukan operasi XOR antara matriks *state* dengan kunci ronde berikutnya yaitu (W4-W7) Contoh proses :

*Byte* 244 XOR 115 = 135

*Byte* 120 XOR 117 = 13

Hasil *AddRounKey* : [135, 13, 17, 169]

[38, 98, 85, 12]

[55, 32, 82, 84]

[49, 53, 180, 193]

* 1. *MixColumns*

Setiap kolom pada matriks *state* dimodifikasi dengan operasi matematis berbasis matriks yang menggunakan bidang *Galois* (GF). Di mana proses ini bertujuan untuk mencampur elemen-elemen dalam kolom dan meningkatkan difusi data. [2, 3, 1, 1]

[1, 2, 3, 1]

[1, 1, 2, 3]

[3, 1, 1, 2]

* 1. *AddRoundkey* (W36-W39)

Dilakukan setelah *MixColums* untuk menambahkan lapisan keamanan dengan kunci ronde tambahan (W36-W39).

1. Pada ronde terakhir, hanya dilakukan tiga langkah yaitu

*SubBytes, ShiftRows*, dan *AddRoundkey* tanpa *MixColumns.*

Proses modifikasi algoritma AES ini digunakan untuk mengenkripsi *file* dengan menambahkan langkah *AddRoundkey* tambahan, yang meningkatkan keamanan dengan memberikan lapisan XOR tambahan pada setiap ronde enkripsi.

* + 1. **Implementasi Kombinasi AES Modifikasi dan ChaCha20** Algoritma AES modifikasi dan ChaCha20 dengan menerapkan *double encypt.* Proses ini memakai AES modifikasi dengan memberikan perlindungan berbasis blok *chiper* yang kuat dengan memodifikasi *AddRoundkey* tambahan sebelum *MixColumns,* sedangkan ChaCha20 sebagai *stream cipher* yang cepat dan aman, meningkatkan kompleksitas keseluruhan dengan memanfaatkan proses enkripsi berbasis *keystream.*

Pada metode ini, setiap *file* menggunakan 1 kunci enkripsi. Kunci yang digunakan memiliki panjang awal 128-bit, yang kemudian diperluas menjadi 256-bit pada ChaCha20 dengan menambahkan padding nol (0x00) jika panjangnya kurang dari 256-bit. Proses

dimulai dengan mengenkripsi *file* menggunakan AES Modifikasi untuk menghasilkan cipherteks awal dalam format matriks 4x4. Cipherteks ini kemudian diubah menjadi byte stream untuk kompatibilitas dengan ChaCha20. Selanjutnya, ChaCha20 melakukan proses enkripsi kedua dengan memanfaatkan kunci 256- bit, nonce sepanjang 96-bit, dan counter. Hasil akhir adalah cipherteks final yang lebih aman karena telah melalui dua lapisan enkripsi. Penggunaan double encryption dengan satu kunci untuk setiap *file* memberikan keamanan yang lebih tinggi, di mana kombinasi kedua algoritma menciptakan perlindungan yang kuat terhadap berbagai jenis serangan kriptografi.

*Gambar III.3 Diagram Proses Double Encrypt*

Pada [Gambar III.3 Diagram Proses Double Encrypt](#_bookmark50) menunjukkan langkah-langkah enkripsi ganda. Proses dimulai dengan mengenkripsi *plainteks* menggunakan AES modifikasi untuk menghasilkan *chiperteks* AES. *Chiperteks* ini kemudian dienkripsi ulang menggunakan ChaCha20, yang melibatkan matriks 4x4, proses *quarter-round,* dan operasi XOR dengan *keystream*. Algoritma AES modifikasi dan ChaCha20 menggunakan kunci yang sama, yaitu kunci AES sepanjang 128-bit. berikut adalah tahapan alur di atas :

1. Input *plainteks* dan *key* dan enkripsi AES modifikasi
2. Input *cipherteks* AES ke ChaCha20

*Chiperteks* yang dihasilkan dari proses enkripsi pertama diubah menjadi input untuk algoritma ChaCha20. Ini dilakukan untuk menerapkan enkripsi berlapis yaitu *double encryption,* sehingga keamanan data akan meningkat secara signifikan. *Chiperteks* AES yang biasanya dalam format matriks (4x4) ini diubah ke format *byte stream* untuk kompatibilitas dengan algoritma ChaCha20. Byte stream ini memungkinkan untuk memproses data, sesuai dengan *stream chiper*.

Contoh :

[115, 75, 189, 234, 201, 97, 58, 14, 245, 142, 12, 120, 53, 190,

201, 20] data ini sudah dalam bentuk *byte stream*.

1. Validasi panjang *key* (256-bit *Key*)

Ini untuk memastikan bahwa kunci enkripsi untuk ChaCha20 sesuai dengan spesifikasi algoritma, yaitu panjangnya harus tepat 256 bit (32 *byte*). Kunci ini untuk menghasilkan *keystream* yang beda dalam proses enkripsi. Jika panjang ke kurang dari 256 bit, ditambahkan *padding* berupa *byte* nol (0x00) untuk memenuhi panjang. Jika *key* yang lebih panjang, *key* akan dipotong hingga hanya menyisakan 256 bit. *Key* kemudian dibagi menjadi 8 bagian ( di mana masing-masing 32 bit) untuk digunakan dalam proses pembentukan matriks awal (*state* matriks)

Contoh :

* 1. *Key* input “surabayaittelkom”

[115, 117, 114, 97, 98, 97, 121, 97, 105, 116, 116, 101, 108,

107, 111, 109]

* 1. Panjang *key* adalah 16 *byte* (128 bit), sehingga tidak memenuhi syarat panjang 256 bit. Jadi ditambahkan *padding* nol (0x00) hingga mencapai 32 *byte:*

[115, 117, 114, 97, 98, 97, 121, 97, 105, 116, 116, 101, 108,

107, 111, 109, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

1. Membuat *Initial* State Matriks 4x4

Membentuk *state* matriks awal yang menjadi dasar operasi dalam ChaCha20. Matriks ini berisi kunci, *counter*, dan *nonce.* Matriks 4x4 dalam ChaCha20 terdiri 16 elemen (masing-masing 32 bit) Contoh :

* 1. 4 elemen konstanta tetap “*expand* 32-byte k”4 elemen 32 bit [1634760805, 857760878, 2036477234, 1797285236]
  2. Elemen kunci 256-bit dibagi menjadi 8 elemen 32-bit: [1936028945, 1885434721, 1701147239, 1647539057,

1225269481, 1226844037, 1214606441, 0]

* 1. 1 *Counter*: Nilai awal 0.
  2. 3 Elemen Nonce: Nilai *nonce* yang unik untuk sesi enkripsi: Contoh *nonce* : [12345678, 87654321, 11223344]
  3. Matriks yang dibentuk

[1634760805, 857760878, 2036477234, 1797285236]

[1936028945, 1885434721, 1701147239, 1647539057]

[1225269481, 1226844037, 1214606441, 0]

[0, 12345678, 87654321, 11223344]

Baris ke 1 konstanta, ke 2 *key,* ke 3 *key* dan ke 4 *counter* dan

*nonce*

1. Operasi Quarter Round (20 iterasi)

Melakukan proses pencampuran data di dalam matriks untuk meningkatkan kerandoman dan keamanan *keystream* yang dihasilkan. Quarter Round adalah inti dari algoritma ChaCha20. Quarter Round dilakukan pada 4 elemen tertentu dari matriks. Penjumlahan Modular (mod 2³²): Menambah elemen dalam matriks untuk mencampur data. Operasi XOR: Mencampur data antar elemen melalui operasi logika XOR. Rotasi Bit (ROTL): Menggeser bit elemen untuk mendistribusikan informasi secara acak. Setelah 20 iterasi (10 putaran double round), matriks akan diubah menjadi bentuk yang sangat acak.

1. *Counter* +=1

*Counter* bertambah untuk setiap blok data yang diproses, memastikan *keystream* yang dihasilkan unik untuk setiap blok. Setelah satu blok data selesai diproses, *counter* di matriks bertambah sebesar 1.

Contoh :

*Counter* baru : [1, 12345678, 87654321, 11223344]

1. XOR *keystream* dengan *chiperteks* AES

Menggabungkan *keystream* hasil ChaCha20 dengan *cipherteks AES* melalui operasi XOR untuk menghasilkan *final ciphertext*. XOR dipilih karena sifatnya yang sederhana namun aman dan mudah dibalik. *Keystream* yang dihasilkan dari matriks ChaCha20 di-XOR-kan dengan *cipherteks AES*.

Operasi XOR:

Jika *ciphertext* = *keystream* XOR *plaintext*, maka *dekripsi*

dilakukan dengan cara *plaintext* = *keystream* XOR *ciphertext*.

* + - 1. **Proses *Dekripsi* Double Encryption**

Proses *dekripsi* dilakukan untuk mengembalikan data yang telah terenkripsi ke bentuk aslinya atau *plainteks*. Di mana pada tahap ini, *chiperteks* hasil dari kombinasi algoritma AES modifikasi dan Chacha20 akan *didekrpsi* secara urut. Proses nantinya akan dimulai dengan *mendekripsi chiperteks* dengan ChaCha20 untuk mendapatkan kembali *chiperteks* AES modifikasi, kemudian dilanjutkan dengan *mendekripsi chiprteks* AES modifikasi untuk memperoleh *plainteks* asli.

Pada [Gambar III.4 Proses Dekripsi](#_bookmark51) Diagram alur di atas menjelaskan proses *dekripsi* dari kombinasi algoritma ChaCha20 dan modifikasi AES. Proses dimulai dengan memasukkan cipherteks hasil enkripsi sebagai input. Langkah pertama adalah men*dekripsi* cipherteks menggunakan algoritma ChaCha20 untuk menghasilkan keystream. Keystream ini kemudian dikombinasikan dengan cipherteks menggunakan operasi XOR untuk mendapatkan cipherteks AES.

Selanjutnya, proses ekspansi kunci dilakukan untuk menghasilkan kunci yang akan digunakan pada setiap ronde *dekripsi* AES. Proses *dekripsi* AES terdiri dari tiga tahapan utama: ronde terakhir (Round 10) yang melibatkan *AddRoundKey*, InvShiftRows, dan InvSubBytes; ronde utama (Round 9-1) yang melibatkan *AddRoundKey*, InvMixColumns, *AddRoundKey, InvShiftRows,* dan *InvSubBytes* tambahan; dan ronde awal (Round 0) yang menggunakan *AddRoundKey* (W0-W3). Hasil akhir dari seluruh proses ini adalah plainteks, yaitu data asli yang berhasil di*dekripsi*. Kombinasi kedua algoritma ini memastikan keamanan data yang lebih kuat melalui enkripsi ganda.

*Gambar III.4 Proses Dekripsi*

### Implementasi Antarmuka Pengguna (UI)

Antarmuka pengguna (UI) dirancang untuk mempermudah interaksi, berfungsi sebagai media interaktif yang memungkinkan pengguna untuk memilih file, memasukkan kunci enkripsi, serta memulai proses enkripsi dan dekripsi dengan mudah. Implementasinya menggunakan Flask sebagai framework backend untuk menangani

proses enkripsi dan dekripsi, sedangkan HTML digunakan untuk membuat form input file dan kunci enkripsi. CSS digunakan untuk mempercantik tampilan agar antarmuka lebih menarik, jelas, dan mudah digunakan oleh pengguna dalam proses enkripsi dan dekripsi file menggunakan kombinasi algoritma ChaCha20 dan AES yang telah dimodifikasi.

*Gambar III.5 Desain Antarmuka Pengguna Enkripsi dan Dekripsi*

Pada [Gambar III.5 Desain Antarmuka Pengguna Enkripsi dan](#_bookmark53) [Dekripsi](#_bookmark53) Gambar tersebut menampilkan desain antarmuka pengguna (UI) dari sebuah aplikasi berbasis web yang digunakan untuk enkripsi dan *dekripsi file*. Antarmuka ini memiliki beberapa elemen input pengguna untuk mengunggah *file* dan memilih opsi enkripsi atau *dekripsi*. Terdapat tiga kolom utama pada desain ini: pertama, Pilih *File*, untuk pengguna memilih *file* yang akan diproses. Kedua, Enkripsi atau *Dekripsi*, yang memberikan pilihan kepada pengguna apakah *file* yang dipilih akan dienkripsi atau di *dekripsi*, yang merupakan opsi penting untuk menentukan jenis operasi yang akan dilakukan pada *file* tersebut. Ketiga, Key, di mana pengguna diminta untuk memasukkan kunci yang diperlukan untuk proses enkripsi atau *dekripsi*. Kunci ini memainkan peran penting dalam menjaga keamanan data yang diproses. Di bagian bawah antarmuka, terdapat

tombol Proses yang akan menjalankan operasi sesuai dengan pilihan pengguna, baik itu enkripsi maupun *dekripsi*.

### Pengujian Algoritma

Menjelaskan pengujian yang dilakukan untuk mengevaluasi kinerja dan keamanan algoritma berdasarkan parameter seperti entropi, waktu enkripsi dan *dekripsi*, serta kecapatan.

### Pengujian Entropi

Pengujian kekuatan algoritma dilakukan dengan mengukur tingkat keacakan data hasil enkripsi menggunakan analisis entropi. Keacakan data merupakan indikator penting dalam menentukan seberapa baik algoritma enkripsi menyamarkan pola asli dari data. Semakin tinggi tingkat keacakan, semakin sulit bagi pihak yang tidak berwenang untuk menganalisis pola atau memprediksi isi data terenkripsi. Pengujian ini dilakukan dengan menghitung nilai entropi menggunakan distribusi probabilitas setiap simbol dalam cipherteks. Nilai entropi yang mendekati maksimum menunjukkan bahwa distribusi simbol dalam cipherteks hampir merata, menandakan bahwa algoritma telah berhasil menghasilkan data yang acak. Data dengan keacakan tinggi lebih tahan terhadap serangan berbasis statistik, sehingga pengujian ini memberikan evaluasi kuantitatif terhadap kekuatan keamanan algoritma yang diuji. Hasil pengujian dibandingkan antara AES Standar, AES Modifikasi, Chacha20 dan Double Encryption untuk menunjukkan efektivitas modifikasi dalam meningkatkan keamanan.

### Pengujian Waktu dan Kecepatan

Menguji algoritma AES modifikasi dan *Double encrypt* AES modifikasi dan ChaCha20. Pengujian ini dilakukan untuk menilai kinerja algoritma dalam berbagai parameter. Penelitian ini akan menganalisis parameter pengujian menggunakan matriks dan formula berikut :

* + - * 1. Waktu enkripsi :

mengukur waktu yang dibutuhkan untuk mengenkripsi data dari

*plainteks* menjadi *chiperteks* dijelaskan dalam persamaan [(III-1)](#_bookmark55).

|  |  |
| --- | --- |
| 𝐸𝑡 = 𝐸𝑁𝑡 – 𝑆𝑇𝑡 | (III-1) |

Di mana:

𝐸𝑡 = waktu enkripsi dalam mili detik (ms)

𝐸𝑁𝑡 = waktu selesai enkripsi

𝑆𝑇𝑡 = waktu mulai enkripsi

Waktu ini akan dihitung untuk setiap *file* yang diuji, dan hasilnya nanti akan memberikan gambaran mengenai kecepatan algoritma dalam melakukan proses enkripsi.

* + - * 1. Waktu *dekripsi* :

Mengukur waktu yang dibutuhkan untuk *mendekripsi* data dari

*chiperteks* menjadi *plainteks* dijelaskan pada persamaan [(III-2)](#_bookmark56).

Dimana:

𝐷𝑡 = waktu *dekripsi* (ms)

𝐸𝑁𝑡 = waktu selesai *dekripsi*

𝑆𝑇𝑡 = waktu mulai *dekripsi*

Hasil waktu *dekripsi* ini juga perlu untuk masing-masing algoritma karena ini menganalisis perbandingan kinerja antara algoritma enkripsi.

* + - * 1. Kecepatan :

Menunjukkan jumlah data dalam *kilobyte* yang dapat dienkripsi

Et = waktu enkripsi atau *dekripsi* (s)

### Pengujian pada Berbagai Jenis *File*

*File* PDF yang digunakan dalam pengujian dapat berisi elemen seperti teks, gambar, dan tabel yang semuanya akan dienkripsi secara utuh karena algoritma bekerja pada level data biner. *File* PDF dengan ukuran 100 KB, 1 MB, 10 MB, hingga 100 MB diuji untuk memastikan algoritma dapat bekerja pada dokumen berukuran kecil hingga besar. Hal serupa juga berlaku untuk *file* Word dengan ukuran yang sama, meskipun memiliki struktur internal yang lebih kompleks dibandingkan PDF. *File* TXT dengan ukuran maksimal 300 *byte*, yang biasanya hanya berisi teks sederhana, juga dapat dienkripsi ChaCha20 (*double encrypt*). Di mana pengujian ini bertujuan untuk

mengukur kinerja algoritma. Berikut ini adalah contoh hasil pengujian terhadap *file* PDF :

*Tabel III.2 Evaluasi Hasil*

Pada [Tabel III.2 Evaluasi Hasil](#_bookmark60) menunjukkan hasil evaluasi simulasi kinerja algoritma enkripsi dan *dekripsi*. pada Nilai-nilai tersebut memberikan gambaran awal mengenai performa beberapa metode, termasuk AES Standar, ChaCha20, AES Modifikasi, dan kombinasi AES Modifikasi dengan ChaCha20. Hasil ini digunakan sebagai contoh output yang diharapkan dari model algoritma yang akan dirancang, ini belum merupakan data dari eksperimen nyata.

# DAFTAR PUSTAKA

Alimuddin, A., Juntak, J. N. S., Jusnita, R. A. E., Murniawaty, I., & Wono, H. Y. (2023). Teknologi dalam pendidikan: Membantu siswa beradaptasi dengan revolusi industri 4.0. *Journal on Education*, *5*(4), 11777–11790.

Almukhlifi, R., & Vora, P. L. (2020). Linear cryptanalysis of reduced-round simon using super rounds. *Cryptography*, *4*(1), 1–34. https://doi.org/10.3390/cryptography4010009

Alyas, H. H., & Abdullah, A. A. (2021). Enhancement the ChaCha20 Encryption Algorithm Based on Chaotic Maps. *Lecture Notes in Networks and Systems*, *201*, 91–107. https://doi.org/10.1007/978-981-16-0666-3\_10

Anindita, M., & 18219086, A. A. (2023). *Analisis Perbandingan Algoritma AES dan ChaCha20-Poly1305 dalam Enkripsi Konten Livestreaming*.

Aryanto, M. B., Tahir, M., Devita, S. I., Mustofa, Z. N., Ainiyah, Q., & Sundoro,

S. (2023). Implementasi Enkrip Dan Dekrip File Menggunakan Metode Advance Encryption Standard (AES-128). *Jurnal Ilmiah Sistem Informasi Dan Ilmu Komputer*, *3*(1), 89–104.

Assyahid, M. H. (2024). *Implementasi interpolasi pixel wadah citra dan enkripsi aes pada steganografi least significan bit (LSB)*. https://repository.uinjkt.ac.id/dspace/handle/123456789/76727

Barbero, S., Bazzanella, D., & Bellini, E. (2022). Rotational Cryptanalysis on ChaCha Stream Cipher. *Symmetry*, *14*(6). https://doi.org/10.3390/sym14061087

Belvin Tannadi. (2022). *Ilmu Crypto* (Digital). Jakarta : Elex Media Komputindo.

Cristy, N., & Riandari, F. (2021). *Niolinda Cristy 1 , Fristi Riandari 2 [Implementasi Metode Advanced Encryption Standard (AES 128 Bit) Untuk Mengamankan Data Keuangan*. *4*(2), 75.

Djong, H. S., & Siswanto, S. (2022). Implementasi Kriptografi Dengan Menggunakan Metode RC4 dan AES-256 Untuk Mengamankan File Dokumen pada PT Varnion Technology Semesta. *Prosiding Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI)*, *1*(1), 149–158.

Handoyo, J., & Subakti, Y. M. (2020). Keamanan Dokumen Menggunakan Algoritma Advanced Encryption Standard (AES). *Jurnal SITECH : Sistem Informasi Dan Teknologi*, *3*(2), 143–152. https://jurnal.umk.ac.id/index.php/sitech/article/view/5865

Hena, N., & Kusumaningsih, D. (2024). Pengamanan File Pada Sistem Master Vendor Berbasis WEB Menggunakan Algoritma AES Pada Trinitiland. *Prosiding Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI)*, *3*(2), 30–37.

Hidayatulloh, N. W., Tahir, M., Amalia, H., Basyar, N. A., Prianggara, A. F., & Yasin, M. (2023). Mengenal Advance Encryption Standard (AES) sebagai Algoritma Kriptografi dalam Mengamankan Data. *Digital Transformation Technology*, *3*(1), 1–10.

Kebande, V. R. (2023). Extended-Chacha20 Stream Cipher With Enhanced Quarter Round Function. *IEEE Access*, *11*, 114220–114237. https://doi.org/10.1109/ACCESS.2023.3324612

Kim, B. G., & Wong, D. (2023). Smart Contract-based Secure Verifiable Random Function using ChaCha20 Sequence in Blockchain. *ACM International Conference Proceeding Series*, 41–51. https://doi.org/10.1145/3638025.3638028

Kurnia Mahesa Naibaho, A., Putri, A., & Fikri Husaini, Y. (2021). Analisis Sistem Keamanan Informasi Chat Online dengan Menggunakan Algoritma AES. In *Ilmu Komputer* (Vol. 6, Issue 2).

Luong, T. T. (2023). Strengthening AES Security Through Key-Dependent ShiftRow and AddRoundKey Transformations Utilizing Permutation. In *IJACSA) International Journal of Advanced Computer Science and Applications* (Vol. 14, Issue 11). [www.ijacsa.thesai.org](http://www.ijacsa.thesai.org/)

Muhammed, R. K., Rashid Aziz, R., Hassan, A. A., Aladdin, A. M., Saydah, S. J., Rashid, T. A., & Hassan, A. (2024). *Comparative Analysis of AES, Blowfish, Twofish, Salsa20, and ChaCha20 for Image Encryption*.

Olivia Putri Irine Irawan, B., Tahir, M., Ayu Windrastuti, N., Yurina Cholili, D., Mulaikah, D., Batsul Mushofi Septian wachid, A., & Pendidikan Informatika,

P. (2023). Implementasi Kriptografi Pada Keamanan Data Menggunakan Algoritma Advance Encryption Standard (AES). *Jurnal Simantec*, *11*(2), 167–

174. https://doi.org/10.21107/SIMANTEC.V11I2.20034

Pangestu, G. Y., Hadiana, A. I., & Sabrina, P. N. (2022). Kriptografi Untuk Enkripsi Ganda Pada Gambar Menggunakan Algoritma AES (Advanced Encryption Standard) Dan RC5 (Rivest Code 5). *Vol*, *1*, 25–32.

Putri, A. E., Kartikadewi, A., & Rosyid, L. A. A. (2021). Implementasi Kriptografi dengan Algoritma Advanced Encryption Standard (AES) 128 Bit dan Steganografi menggunakan Metode End of File (EOF) Berbasis Java Desktop

pada Dinas Pendidikan Kabupaten Tangerang. *Appl. Inf. Syst. Manag*, *3*(2), 69–78.

Saripa, S. (2023). Implementation Implementation of a File Security System Using the AES Algorithm to Secure Personal Files: Implementasi Sistem Keamanan File Menggunakan Algoritma AES untuk Mengamankan File Pribadi. *Progressive Information, Security, Computer, and Embedded System*, *1*(2), 138–148.

Sekar Putri Ananda, Saepul Lukman, & Irfan. (2022). *Analisa Metode Kriptogra Modern Advance Encryption Standar (AES) 128 Bit dalam Mengenkripsi dan Mendekripsi File Dokumen Digital*. https://doi.org/10.32409/jikstik.21.3.2973

Soesanto, E., Purba, L. M., Aprilia, B., Putra, D. R., & Putri, S. D. (2023). Implementasi Objek Vital, Pengamanan File dan Pengamanan Cyber di PT Pertamina. *IJM: Indonesian Journal of Multidisciplinary*, *1*(1).

Surbakti, P. S., & Purwanto, P. (2024). Implementasi Kriptografi Dengan Menggunakan Algoritma AES-128 Untuk Enkripsi dan Dekripsi File Dokumen Berbasis Web Pada Law Office Erdi Surbakti, SH & Rekan. *Prosiding Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI)*, *3*(1), 1–8.

Syahputra Sianipar, J., Budi Nuugroho, N., Mariami, I., Informasi, S., & Triguna Dharma, S. (2024). Pengamanan Data Gaji Karyawan Dengan Menggunakan Metode Advanced Encryption Standard (AES). *Jurnal Sistem Informasi Triguna Dharma (JURSI TGD)*, *3*(1), 35–45. https://doi.org/10.53513/JURSI.V3I1.5653

Wahyu Putra, M. R. F. T. H. (2023, July 2). *Implementasi Algoritma Advanced Encryption Standard Untuk Kemanan Dokumen*. https://journal.grahamitra.id/index.php/jurikti/article/view/55/181

Wahyudi, E. N., Ardhianto, E., Handoko, W. T., Murti, H., Supriyanto, E., Lestariningsih, E., & Redjeki, R. S. (2024). Peningkatan Keamanan Data Melalui Teknik Super Enkripsi Menggunakan Algoritma Vigenere dan Caesar. *Jurnal Informatika Polinema*, *10*(3), 315–322. https://doi.org/10.33795/JIP.V10I3.5131

Widyawan, D., & Imelda, I. (2021). Pengamanan File Menggunakan Kriptografi Dengan Metode Aes-128 Berbasis Web Di Komite Nasional Keselamatan Transportasi. *SKANIKA: Sistem Komputer Dan Teknik Informatika*, *4*(1), 15– 22.

**IMPLEMENTASI KOMBINASI CHACHA20 DAN MODIFIKASI AES MELALUI *ADDROUNDKEY* PADA**

**STUDI KASUS KEAMANAN *FILE***

**Proposal Tugas Akhir Mata Kuliah Penulisan Proposal**

## (CAK4EAB2)

Disusun oleh:

## EMILIA RAMONA 1203210017

**PROGRAM STUDI SARJANA INFORMATIKA DIREKTORAT KAMPUS SURABAYA**

**UNIVERSITAS TELKOM SURABAYA TAHUN 2025**

# ABSTRAK

Keamanan data menjadi tantangan besar di era Revolusi Industri 4.0, di mana informasi penting perlu dilindungi dari akses tidak sah. Ancaman serangan siber dan peretasan mengharuskan pengembangan solusi keamanan yang lebih efektif. Penelitian ini bertujuan untuk mengimplementasikan dan mengevaluasi kombinasi algoritma ChaCha20 dan modifikasi *Advanced Encryption* Standard (AES) dengan penambahan langkah *AddRoundKey* sebelum *MixColumns* untuk meningkatkan keamanan *file*. Metode yang digunakan adalah enkripsi ganda, dengan AES modifikasi sebagai *block cipher* yang kuat dan ChaCha20 sebagai *stream cipher* yang cepat. Penelitian ini mencakup implementasi algoritma dengan menggunakan pustaka kriptografi untuk menguji kinerja algoritma melalui parameter seperti waktu enkripsi, waktu *dekripsi*, kecepatan enkripsi, kecepatan *dekripsi*, dan tingkat keacakan data menggunakan analisis entropi. *File* dengan berbagai format dan ukuran, PDF, Word, dan TXT, digunakan sebagai data pengujian untuk mengevaluasi efisiensi algoritma. Selain itu, antarmuka pengguna berbasis web dirancang untuk mempermudah proses enkripsi dan *dekripsi*, melibatkan langkah- langkah seperti pemilihan *file* dan *input* kunci enkripsi. Penelitian ini diharapkan menunjukkan bahwa kombinasi AES modifikasi dan ChaCha20 mampu meningkatkan keamanan data serta memberikan efisiensi yang lebih baik dalam perlindungan *file*.

**Kata Kunci**: Keamanan *File*, *Advanced Encryption Standard* (AES), ChaCha20, Enkripsi Ganda, *AddRoundkey*, *Entropi*.

# ABSTRACT

*Data security has become a major challenge in the era of the Industrial Revolution 4.0, where critical information must be protected from unauthorized access. Cyberattacks and hacking threats demand the development of more effective security solutions. This research aims to implement and evaluate the combination of the ChaCha20 algorithm and a modified Advanced Encryption Standard (AES) by adding an AddRoundKey step before MixColumns to enhance file security. The method used is double encryption, with the modified AES as a robust block cipher and ChaCha20 as a fast stream cipher. This study involves implementing the algorithms using cryptographic libraries to test their performance through parameters such as encryption time, decryption time, encryption speed, decryption speed, and data randomness level using entropy analysis. Files of various formats and sizes, including PDF, Word, and TXT, are utilized as test data to evaluate the efficiency of the algorithms. Additionally, a web-based user interface is designed to simplify the encryption and decryption process, involving steps such as file selection and encryption key input. This research is expected to demonstrate that the combination of modified AES and ChaCha20 can enhance data security and provide better efficiency in file protection.*

***Keywords****: File Security, Advanced Encryption Standard (AES), ChaCha20, Double Encryption, AddRoundkey, Entropy*

# KATA PENGANTAR

Assalamu’alaikum Warrahmatullahi Wabarakatuh, puja puji syukur ke hadirat Allah SWT yang telah memberikan nikmat dan karunia-Nya yang sangat luar biasa kepada hamba – hambanya. Tidak lupa shalawat serta salam kita curahkan atas bimbingan nabi kita nabi Muhammad SAW yang telah membawa cahaya islam ke dalam dunia ini dan menjadi panutan bagi setiap umatnya hingga akhir zaman.

Setelah melalui berbagai proses kerja keras, penyusunan skripsi yang berjudul “Implementasi Kombinasi ChaCha20 dan Modifikasi AES melalui *AddRoundkey* Pada Studi Kasus Keamanan *File*” akhirnya dapat terselesaikan. Selain sebagai salah satu syarat memperoleh gelar S-Kom, penulis berharap juga skripsi ini dapat bermanfaat bagi semua pihak yang berkepentingan. Oleh karena itu, penulis ingin mengungkapkan rasa terima kasih kepada pihak-pihak yang telah memberikan dukungan dan doa sehingga penulis dapat menyelesaikan skripsi ini tepat pada waktunya. Beberapa pihak tersebut antara lain:

1. **Kepada Mama** yang sangat penulis cintai yang tidak pernah lupa selalu memberikan dukungannya dan doa agar diberikan kemudahan dan kelancaran dalam penyusunan skripsi ini sehingga penulis dapat menyelesaikan studi Informatika di Universitas IT Telkom Surabaya.
2. **Kepada Alm.Ibu Suhartatik**, yang penulis cintai dan sayangi, yang telah memberikan motivasi untuk saya.
3. **Kepada Saudara Kembar**, yang selalu mendukung, terima kasih atas bantuan dan waktunya dalam memberikan masukan terkait susunan skripsi ini.
4. **Keluarga Besar penulis** yang telah memberikan motivasi dan kepercayaan besar sehingga penulis bisa mewujudkan impian untuk meraih gelar S-Kom.
5. **Dosen Pembimbing**, Pak Rizky Fenaldo Maulana, S.Kom., M.Kom. dan Tanzila Mustaqim, S.Kom., M.Kom. Yang telah memberikan solusi, waktu, dan bimbingan penuh kesabaran selama proses penyusunan skripsi ini.
6. **Teman-teman dekat**, termasuk Jus’an, Erica Dyah, Qori Emalia, Ardina Firdatana, dan Elma Saskia yang selalu memberikan semangat, mendengarkan keluh kesah, serta menemani dalam berbagai kondisi selama penulisan skripsi.
7. **Semua pihak lainnya**, yang tidak dapat disebutkan satu per satu, atas bantuan dan dukungannya selama proses ini.
8. **Diri penulis sendiri**, atas perjuangan dan kerja keras hingga akhirnya skripsi ini selesai tepat waktu.

Penulis menyadari bahwa skripsi ini masih jauh dari kesempurnaan. Oleh karena itu, saran dan kritik yang membangun sangat diharapkan untuk memperbaiki skripsi ini ke depannya.

Terima kasih.

### Wassalamu’alaikum Warrahmatullahi Wabarakatuh.

Surabaya, 24 Desember 2024 Penulis

Emilia Ramona

# DAFTAR ISI

[LEMBAR PERSETUJUAN ii](#_bookmark0)

[ABSTRAK iii](#_bookmark1)

[ABSTRACT iv](#_bookmark2)

[KATA PENGANTAR v](#_bookmark3)

[DAFTAR ISI vii](#_bookmark4)

[DAFTAR TABEL ix](#_bookmark5)

[DAFTAR GAMBAR x](#_bookmark6)

[DAFTAR RUMUS xi](#_bookmark7)

[BAB I PENDAHULUAN 1](#_bookmark8)

* 1. [Latar Belakang 1](#_bookmark9)
  2. [Rumusan Masalah 4](#_bookmark10)
  3. [Tujuan Penelitian 4](#_bookmark11)
  4. [Batasan dan Asumsi Penelitian 5](#_bookmark12)
  5. [Manfaat Penelitian 5](#_bookmark13)
     1. [Manfaat Akademis 5](#_bookmark14)
     2. [Manfaat Ilmiah 5](#_bookmark15)
  6. [Sistematika Penulisan 6](#_bookmark16)

[BAB II LANDASAN TEORI 7](#_bookmark17)

* 1. [Literatur Terkait Teori 7](#_bookmark18)
  2. [Kajian Teori 10](#_bookmark20)
     1. [Kriptografi 10](#_bookmark21)
     2. [AES (Advanced Encryption Standard) 12](#_bookmark24)
     3. [Chacha20 20](#_bookmark32)
     4. [Entropi 22](#_bookmark35)

[BAB III METODOLOGI PENELITIAN 24](#_bookmark37)

* 1. [Sistematika Penyelesaian Masalah 24](#_bookmark38)
  2. [Alat dan Bahan 24](#_bookmark39)
     1. [Alat 24](#_bookmark40)
     2. [Bahan 25](#_bookmark41)
  3. [Rancangan Penelitian 25](#_bookmark42)
     1. [Studi Literatur 27](#_bookmark45)
     2. [Pengumpulan Data 27](#_bookmark46)
     3. [Implementasi Modifikasi AES 28](#_bookmark47)
     4. [Implementasi Kombinasi AES Modifikasi dan ChaCha20 31](#_bookmark49)
     5. [Implementasi Antarmuka Pengguna (UI) 36](#_bookmark52)
     6. [Pengujian Algoritma 38](#_bookmark54)
     7. [Analisis Hasil 42](#_bookmark59)

[DAFTAR PUSTAKA 44](#_bookmark61)

# DAFTAR TABEL

[Tabel II.1 Literatur Penelitian Terkait Kriptografi 7](#_bookmark19)

[Tabel II.2 Panjang Kunci 13](#_bookmark25)

[Tabel II.3 Tabel Substitusi (S-Box) 16](#_bookmark28)

[Tabel II.4 Transformasi ShiftRows 17](#_bookmark29)

[Tabel III.1 Jenis, Ukuran, dan Isi File untuk Pengujian 40](#_bookmark58)

[Tabel III.2 Evaluasi Hasil 43](#_bookmark60)

# DAFTAR GAMBAR

[Gambar II.1 Alur Proses Enkripsi Dan Dekripsi 11](#_bookmark22)

[Gambar II.2 Proses Kunci Simetris 11](#_bookmark23)

[Gambar II.3.Transformasi ShiftRows 15](#_bookmark27)

[Gambar II.4 Proses AES 19](#_bookmark31)

[Gambar III.1 Diagram Alur Penelitian 26](#_bookmark43)

[Gambar III.2 Diagram Proses Enkripsi AES 28](#_bookmark48)

[Gambar III.3 Diagram Proses Double Encrypt 32](#_bookmark50)

[Gambar III.4 Proses Dekripsi 36](#_bookmark51)

[Gambar III.5 Desain Antarmuka Pengguna Enkripsi dan Dekripsi 37](#_bookmark53)

# DAFTAR RUMUS

[(II-1) Rumus AddRoundKey 14](#_bookmark26)

[(II-2) Operasi Perkalian MixColumns 17](#_bookmark30)

[(II-3) Rumus Keystream 22](#_bookmark33)

[(II-4) Rumus Proses Enkripsi XOR pada Keystream 22](#_bookmark34)

[(II-5) Entropi 22](#_bookmark36)

[(III-1) Pengujian Waktu Enkripsi 39](#_bookmark55)

[(III-2) Pengujian Waktu Dekripsi 39](#_bookmark56)

[(III-3) Pengujian Waktu Kecepatan 39](#_bookmark57)

# BAB I PENDAHULUAN

### Latar Belakang

Semakin meningkatnya teknologi yang semakin canggih membawa sebuah perubahan yang cukup signifikan. Perubahan tersebut dikenal dengan istilah Revolusi Industri 4.0 yang merupakan sebuah kemajuan teknologi yang sering digunakan oleh banyak orang. Revolusi industri 4.0 adalah sebuah data yang digerakkan dengan adanya campur tangan komputer menggunakan metode machine learning atau AI (Alimuddin et al., 2023). Menurut Klaus Schawab revolusi Industri 4.0 memiliki tantangan yang harus dihadapi salah satunya seperti keamanan *cyber* dan privasi data.

Maka dari itu, dengan semakin pesatnya informasi teknologi maka perlunya untuk mengamankan data atau *file*. Keamanan data atau *file* merupakan salah satu langkah yang mudah untuk melindungi sebuah informasi bersifat penting yang berasal dari akses tidak sah. Keamanan *file* ini sangat diperlukan pada era Revolusi Industri 4.0, karena maraknya serangan *cyber* dan peretasan komputer yang bisa mengganggu keberlangsungan suatu industri. Menurut (Mulyana*, 2016*), keamanan *file* merupakan salah satu aksi pencegahan dari sebuah penganggu pengguna *file* atau pengguna yang memakai jaringan yang tidak berkewajiban memiliki akses (Soesanto et al., 2023). Dengan semakin pesatnya dunia teknologi ini membuat keamanan *file* menjadi sangat penting karena untuk melindungi privasi dan informasi dari pihak yang tidak memiliki akses.

Keamanan *file* adalah serangkaian metode atau teknik yang untuk melindungi *file* dari akses yang tidak sah, seperti pencurian atau kerusakan. Tujuannya untuk menjaga informasi dalam *file* tetap rahasia, aman, dan hanya dapat di akses oleh pihak yang berwenang. Oleh karena itu, upaya untuk memastikan agar *file* terjaga dengan menggunakan metode ChaCha20 dan *Advanced Encryption Standard* (AES). ChaCha20 merupakan metode enkripsi yang cepat dan aman. ChaCha20 dipopulerkan oleh Daniel J.

Bernstein sebagai varian dari algoritma. Dalam algoritma ChaCha20 sangat populer dalam pengimplementasi enkripsi dikarenakan percampuran antara kecepatan dan keamanannya yang dapat diunggulkan. Chaca20 mempunyai desain yang hampir sama dengan Salsa20, pada algoritma ChaCha20 menggunakan kunci 256-bit yang mempunyai 8 putaran (Kebande, 2023).

Sedangkan AES (*Advanced Encryption Standard)* adalah sebuah algoritma yang melakukan transformasi terhadap blok data yang menggunakan kunci enkripsi yang sama. AES bekerja pada sistem yang melibatkan operasi matematika dengan level tingkat sulit (Saripa, 2023). Proses kerja enkripsi dasar AES adalah *subbytes* (menggunakan tabel S-Box untuk menggabungkan *byte* pada blok matriks), *shift row* (digunakan untuk menggeser baris matriks), *MixColumns* (melakukan pergeseran dari kanan ke kiri), dan *AddRoundkey* (blok di XOR kan dengan kunci yang diperluas) (Aryanto et al., 2023). Pada proses *dekripsi* AES terdapat tahap-tahapan kebalikan dari proses enkripsi. Dengan memakai kunci yang sama untuk *dekripsi*, blok data yang telah terenkripsi dapat dipulihkan kembali ke bentuk awalnya. algoritma AES adalah jenis *block chipper* yang memiliki panjang kunci 128 bit, 192 bit dan 256 bit (Surbakti & Purwanto, 2024).

Algoritma Chacha20 dan AES merupakan suatu enkripsi yang dapat di digunakan dalam menjaga keamanan *file*. Pada algoritma AES terdapat kelebihan salah satunya adalah sifat karakteristiknya yang di mana selalu ada bidang khusus untuk bilangan prima, memicu kekuatan AES(Surbakti & Purwanto, 2024) . AES juga memiliki sebuah kelemahan salah satunya yaitu algoritma kriptografi yang menggunakan kunci simetris dalam proses pengiriman dan penerimanya data. Namun, penggunaan kunci simetris ini memiliki risiko lebih tinggi terhadap kebocoran, terutama seiring waktu (Hidayatulloh et al., 2023). Pada algoritma ChaCha20 terdapat keunggulan yaitu menggunakan teknik kriptografi ARX (*addition, rotation*, XOR) yang memberikan performa lebih cepat, kompleksitas lebih rendah, dan mampu menghilangkan serangan waktu Klik atau ketuk di sini untuk memasukkan teks.

Meskipun algoritma AES dan Chacha20 telah efektif untuk menjaga peningkatan keamanan *file* dan implementasinya yang cukup rumit sehingga membuat orang yang tidak memiliki akses tidak dapat mengakses *file* tersebut. Hal ini tidak dapat kita pungkiri bahwa semakin pesatnya dunia teknologi membuat banyak orang berbondong-bondong menciptakan pembaruan yang lebih efektif. Metode *double encrypt* ini menawarkan kombinasi yang mampu mengurangi kelemahan dari setiap algoritma jika digunakan secara bersamaan, serta meningkatkan ketahanan terhadap berbagai jenis serangan kriptografi. *Double encrypt* merupakan Teknik kriptografi yang melibatkan proses enkripsi secara dua kali, yang di mana memiliki dua kunci yang berbeda untuk lebih meningkatkan keamanan. *Double encrypt* ini bertujuan untuk meningkatkan keamanan *file* lebih tinggi dibandingkan hanya 1 metode saja (Pangestu et al., 2022).

Pada penelitian terlebih dahulu yang berjudul “Implementasi Sistem Keamanan *File* Menggunakan Algoritma AES untuk Mengamankan *File* Pribadi” yang diteliti oleh Saripa Universitas Negeri Makassar. Dalam penelitian tersebut menjelaskan tentang keamanan *file* dengan menggunakan metode algoritma AES. Di mana hasil penelitian ini menunjukkan bahwa algoritma AES telah berhasil mengamankan *file* yaitu dokumen Word, PDF, Excel, dan Power Poin, *file* yang dienkripsi menjadi tidak terbaca oleh pihak yang tidak memiliki akses sehingga dapat menjaga kerahasiaan data dan AES mampu menjaga keamanan data dengan baik.

Dalam perbedaan penelitian terdahulu dan penelitian saya akan menggunakan double encrypt untuk keamanan *file* dan juga saya akan menggunakan AES dengan menambahkan metode ChaCha20 dan melakukan modifikasi pada bagian AES. Modifikasi menggunakan penambahan *AddRoundkey* sebelum dengan bertujuan agar proses enkripsi akan lebih susah untuk dipecahkan. Dalam algoritma AES, langkah *AddRoundKey* merupakan komponen krusial yang menggabungkan data dengan kunci enkripsi menggunakan operasi XOR. Penelitian (Luong, 2023) oleh menunjukkan bahwa memodifikasi transformasi *AddRoundKey*

dapat meningkatkan keamanan algoritma dengan menciptakan hubungan yang lebih kompleks antara *plainteks* dan *cipherteks*. Penelitian ini menyoroti pentingnya penyesuaian langkah *AddRoundKey* untuk meningkatkan ketahanan algoritma AES terhadap serangan kriptografi, seperti *differential cryptanalysis* dan linear *cryptanalysis*.

Dengan mendasarkan pada penelitian tersebut, modifikasi algoritma AES melalui penambahan langkah *AddRoundKey* sebelum *MixColumns* bertujuan untuk lebih memperkuat difusi data pada tahap awal enkripsi. Penambahan ini diharapkan dapat meningkatkan kerandoman hasil enkripsi dan mengurangi kemungkinan eksploitasi pola oleh pihak yang tidak berwenang, sehingga keamanan algoritma menjadi lebih optimal. Berdasarkan dari latar belakang di atas penelitian ini memiliki tujuan untuk meningkatkan keamanan *file* agar lebih sulit untuk diretas atau diakses pihak yang tidak berwenang. Oleh karena itu penelitian ini dianggap penting karena maraknya kasus peretasan data oleh oknum yang tidak memiliki akses.

### Rumusan Masalah

1. Bagaimana implementasi dari modifikasi AES dengan penambahan *AddRoundkey* sebelum *MixColumns* dalam meningkatkan keamanan *file*?
2. Bagaimana performa kombinasi antara AES modifikasi dan ChaCha20 dalam hal waktu enrkipsi, waktu *dekripsi* , kecepatan enkripsi, dan kecepatan *dekripsi*?

### Tujuan Penelitian

1. Mampu mengidentifikasi *AddRoundkey* yang akan digunakan dalam menjaga keamanan *file* dan mengimplementasi kemampuan Chacha20 dan AES yang telah dimodifikasi dalam proses menjaga keamanan *file*.
2. Mampu mengevaluasi performa algoritma ChaCha20 dan AES yang telah dimodifikasi berdasarkan waktu enkripsi, waktu *dekripsi*, kecepatan enkripsi, dan kecepatan *dekripsi*, untuk menentukan efisiensi

dan efektivitas kombinasi kedua algoritma tersebut dalam menjaga keamanan *file*

### Batasan dan Asumsi Penelitian

1. Penelitian ini berfokus pada jurnal atau artikel ilmiah dengan batasan tahun 2021-2024 yang membahas tentang :
   * Keamanan *file*
   * Algoritma ChaCha20
   * *Advanced Encryption Standard* (AES)
   * Double encryption
2. Penelitian dilakukan dengan pengujian pada *file* dengan ukuran dan format sebagai berikut :
   * Format PDF dan Word dengan variasi ukuran *file* 100 KB - 100MB.
   * Format *file* teks (TXT) dengan ukuran maksimal 300 byte Batasan ukuran *file* digunakan untuk memastikan pengujian dilakukan secara efektif tanpa mengurangi kinerja proses enkripsi dan *dekripsi*.

### Manfaat Penelitian

### Manfaat Akademis

* + - 1. Memberikan inovasi terbaru mengenai pelindungan keamanan

*file* dengan menambahkan *AddRoundkey* pada algoritma AES.

* + - 1. Memudahkan peneliti lainnya dalam kinerja *AddRoundkey* pada algoritma AES yang berfungsi untuk keamanan *file***.**

### Manfaat Ilmiah

* + - 1. Menjadi sumber acuan dan sumber referensi bagi peneliti lainnya yang akan meneliti topik ChaCha20 dan AES modifikasi, terutama dalam konteks keamanan data dan performa enkripsi.
      2. Membuktikan hipotesis apakah AES yang dimodifikasi dengan tambahan *AddRoundkey* dapat meningkatkan performa keamanan dan efisiensi dibandingkan dengan AES standart, dan sekaligus menunjukkan kinerja kombinasi ChaCha20 dan AES dalam pengamanan *file* berbagai format dan ukuran.

### Sistematika Penulisan

## BAB 1 PENDAHULUAN

Menjelaskan latar belakang penelitian, rumusan masalah, tujuan penelitian, batasam penelitian, manfaat penelitian, serta sistematika penulisan secara umum.

## BAB 2 LANDASAN TEORI

Membahas teori-teori yang relavan dengan penelitian, termasuk algoritma ChaCha20, AES, konsep kriptografi, Serta tinjauan terhadap penelitian terdahulu yang terkait.

## BAB 3 METODOLOGI PENELITIAN

Menjelaskan metode yang digunakan dalam penelitian, termasuk algoritma yang diterapkan (modifikasi AES dan ChaCha20 dengan enkripsi ganda). Diuraikan juga alat dan bahan yang digunakan, serta tahapan penelitian yang meliputi implementasi algoritma, pengujian parameter kinerja (waktu dan kecepatan enkripsi, dekripsi, serta analisis entropi), dan evaluasi hasil pengujian pada berbagai jenis file (PDF, Word, TXT).

## BAB 4 IMPLEMENTASI DAN PENGUJIAN

Menyajikan hasil implementasi algoritma yang dikembangkan, pengujian algoritma terhadap berbagai jenis *file*, serta analisis data hasil pengujian**.**

## BAB 5 EVALUASI HASIL PENGUJIAN

Menyajikan hasil implementasi dari aplikasi yang telah dirancang, pengujian fungsionalitas sistem, serta evaluasi kinerja aplikasi terhadap berbagai skenario penggunaan. Analisis data hasil pengujian juga dilakukan untuk menilai keberhasilan aplikasi dalam mencapai tujuan penelitian.

## BAB 6 KESIMPULAN DAN SARAN

Merangkum temuan utama penelitian serta memberikan saran untuk pengembangan lebih lanjut di masa depan.

# BAB II LANDASAN TEORI

### Literatur Terkait Teori

Keamanan informasi menjadi salah satu isu yang paling mendesak di era digital saat ini, di mana data sensitif sering kali menjadi target serangan siber yang beragam. Dengan meningkatnya ancaman seperti serangan *man-in-the- middle*, *eavesdropping*, dan *DDoS*, penting bagi organisasi dan individu untuk menerapkan metode enkripsi yang efektif untuk melindungi data mereka. Berbagai algoritma enkripsi telah dikembangkan untuk memenuhi kebutuhan ini, dengan *Advanced Encryption Standard* (AES) dan ChaCha20 menjadi dua di antara yang paling banyak digunakan. Penelitian-penelitian terkini menunjukkan bahwa kedua algoritma ini tidak hanya menawarkan tingkat keamanan yang tinggi, tetapi juga dapat diimplementasikan dalam berbagai konteks, mulai dari pengamanan *file* hingga enkripsi konten *livestreaming*.

Pada [Tabel II.1 Literatur Penelitian Terkait Kriptografi](#_bookmark19) menyajikan ringkasan dari beberapa penelitian yang mengeksplorasi efektivitas dan penerapan algoritma AES dan ChaCha20 dalam berbagai aplikasi. Setiap penelitian memberikan wawasan tentang tujuan, hasil, dan persamaan dalam pendekatan yang digunakan, serta menyoroti inovasi yang dilakukan untuk meningkatkan keamanan data. Dengan memahami hasil-hasil ini, kita dapat lebih menghargai perkembangan teknologi enkripsi dan aplikasinya dalam menjaga kerahasiaan serta integritas informasi di dunia yang semakin terhubung.

### Kajian Teori

### Kriptografi

Menurut Menezes, Oorschot, dan Vantone (1996), Kriptografi adalah studi teknik matematika yang berkaitan dengan keamanan informasi seperti kerahasiaan, *autentikasi* keaslian data (Belvin Tannadi, 2022). Kriptografi dapat diartikan sebagai sebuah proses untuk melindungi data atau sebuah sistem yang bertujuan untuk melindungi informasi rahasia dengan cara yang tidak dipahami oleh pihak yang tidak berwenang. Dalam kriptografi yang meliputi proses enkripsi dan proses *dekripsi*. Di dalam Kriptografi terdapat tiga fungsi dasar yang meliputi :

* + - 1. Enkripsi merupakan proses data asli (*plainteks*) yang akan diubah menjadi data yang tidak bisa terbaca oleh pihak yang tidak berwenang (*chiperteks*).
      2. *Dekripsi* merupakan proses pengembalian dari data yang tidak bisa terbaca(*chiperteks*) menjadi data terbaca atau data asli (*plainteks*).
      3. Kunci merupakan proses yang dipakai untuk melakukan proses enkripsi dan *dekripsi*, terdapat 2 jenis kunci, yaitu kunci rahasia (*private key*) dan kunci publik (*public key*).

Pada [Gambar II.1 Alur Proses Enkripsi Dan *Dekripsi*](#_bookmark22)

memperlihatkan alur enkripsi dan *dekripsi.* Proses ini dimulai

dengan *plainteks* yang kemudian dienkripsi dengan menggunakan kunci menjadi *chiperteks*. *Chiperteks* tersebut kemudian *didekripsi* kembali menggunakan kunci yang sama untuk mendapatkan kembali *plainteks*.

*Gambar II.1 Alur Proses Enkripsi Dan Dekripsi*

Berikut algoritma Kriptografi yang berdasarkan kunci :

a. Algoritma Simetris

Algoritma simetris adalah algoritma yang merujuk pada metode enkripsi dan *dekripsi* yang di mana algoritma ini memiliki kunci tunggal atau sama. Sehingga dapat disimpulkan algoritma ini mewajibkan pengirim dan penerima harus menyetujui satu kunci yang digunakan dalam proses enkripsi dan *dekripsi* data. Namun, kelemahan dari algoritma ini yaitu tergantung pada rahasia kunci. Apabila kunci tersebut tersebar maka akan dengan mudah pihak yang tidak memiliki akses dapat melakukan proses enkripsi dan *dekripsi* data secara mudah.

*Gambar II.2 Proses Kunci Simetris*

Pada [Gambar II.2 Proses Kunci Simetris](#_bookmark23) menjelaskan proses enkripsi dan *dekripsi* menggunakan kunci simetris. Dimulai dengan pesan yang dikirimkan, kemudian dienkripsi menjadi kode yang tidak bisa dibaca menggunakan kunci simetris. Kode tersebut kemudian dikirim dan di *dekripsi* di sisi penerima menggunakan kunci simetris yang sama, sehingga pesan dapat dibaca kembali. Pada contoh gambar, pesan "Selamat Datang!" dienkripsi menjadi kode "9CDF4837 AB672019" menggunakan kunci simetris yang sama. Kemudian di *dekripsi* kembali ke pesan awal "Selamat Datang!" Kunci simetris merupakan kunci yang sama digunakan untuk enkripsi dan *dekripsi*, sehingga pengirim dan penerima pesan harus memiliki kunci yang sama agar dapat berkomunikasi.

### AES (Advanced Encryption Standard)

Advanced Encryption Standard (AES) adalah salah satu algoritma enkripsi yang paling terkenal dan banyak digunakan, serta terus dikembangkan sebagai pengganti algoritma Data Encryption Standard (DES). AES adalah algoritma enkripsi yang bekerja dengan memproses data dalam blok dengan ukuran minimal 128 bit, dan melalui beberapa tahapan untuk mengenkripsi data. Tujuan utama dari AES adalah untuk mengamankan data rahasia sehingga tidak dapat dibaca oleh pihak yang tidak berwenang (Wahyu Putra, 2023). Pada [Tabel II.2 Panjang Kunci](#_bookmark25) dalam proses enkripsi *Advanced Encryption Standard* (AES) menggunakan ukuran blok dengan panjang sebesar 126 bit, dengan pilihan panjang kunci yang bervariasi, yaitu 128 bit, 192 bit, dan 256 bit (Sekar Putri Ananda et al., 2022). Proses enkripsi dalam algoritma ini dilakukan melalui beberapa tahap yang dikenal sebagai ronde. Jumlah ronde yang digunakan bergantung pada panjang kunci yang akan dipilih. Berikut adalah hubungan panjang kunci dan banyaknya ronde.

Algoritma AES juga menawarkan tingkat keamanan yang sangat tinggi dikarenakan ukuran kunci dan jumlah ronde yang ditentukan. Ukuran kunci menentukan jumlah kemungkinan kombinasi kunci, sehingga membuat serangan *bruteforce* menjadi sangat sulit dilakukan. Di sisi lain, jumlah ronde yang besar dapat menghasilkan keamanan yang kuat, sehingga perubahan kecil pada *plainteks* dapat memengaruhi seluruh bagian *cipherteks*. Selain keunggulan keamanannya, AES juga memiliki efisiensi tinggi dalam hal waktu dan ruang. Algoritma ini dapat diimplementasikan baik pada perangkat keras maupun perangkat lunak, menjadikannya ideal untuk berbagai aplikasi, termasuk perlindungan data (Assyahid, 2024). Pada tahap awal enkripsi, data masukan yang telah disalin ke dalam matriks *state* akan menjalani transformasi *AddRoundkey*. Selanjutnya, matriks *state* akan melalui proses transformasi secara berulang, mencakup *SubBytes*, *ShiftRows*, *MixColumns*, dan *AddRoundkey*, dengan jumlah pengulangan sesuai dengan nilai Nr (Handoyo & Subakti, 2020). *Dekripsi* adalah proses mengembalikan *ciphertext* menjadi plaintext, atau mengubah data terenkripsi kembali ke bentuk aslinya. Pada proses *inverse cipher*, transformasi yang digunakan meliputi *InvShiftRows*, *InvSubBytes*, *InvMixColumns*, dan *AddRoundkey*. Berikut ini adalah cara kerja algoritma AES secara umum:

### Proses Enkripsi AES Modifikasi

* + - * 1. *AddRoundkey*

*AddRoundkey* adalah proses menggabungkan state masukan dengan kunci ronde dengan menggunakan operasi XOR. Dalam proses transformasi enkripsi dan *dekripsi* pada AES, dilakukan operasi penambahan kunci putaran (*round key*)

pada state menggunakan operasi XOR (Djong & Siswanto, 2022). Setiap kunci putaran terdiri dari *Nb word* di mana setiap *word* tersebut akan digabungkan dengan *word* atau kolom yang sesuai pada *state* . Secara matematis, hal ini (Putri et al., 2021).

Keterangan Algoritma dari *AddRoundkey* di mana :

Pada [Gambar II.3.Transformasi ShiftRows](#_bookmark27) di atas di mana S0,0 merupakan elemen di baris ke-0, kolom ke-0, S3,3 merupakan elemen di baris ke-3, kolom ke-3, dan 𝑊3,3 merupakan elemen di baris ke-3, kolom ke-3. Pada label S0,0 dan lainnya merupakan state saat ini, state ini representasi internal data yang sedang dienkripsi ataupun *didekripsi* dan setiap elemen adalah *byte* dari blok data input. Sedangkan pada label 𝑊0,0 merupakan kunci ronde yang dihasilkan dari proses *keyexpansion*, di mana ini bagian dari kunci utama yang dipecah untuk menjadi beberapa ronde dan setiap elemen adalah *byte* dari kunci untuk ronde tertentu. Operasi XOR dilakukan antara setiap *byte* pada *state* dengan *byte* kunci ronde yang sesuai. Hasil di atas dihasilkan dari operasi XOR yang akan digunakan untuk langkah transformasi berikutnya.

* + - * 1. *SubBytes*

Transformasi *transformasi byte* di mana setiap elemen dalam matriks *state* dipetakan menggunakan tabel substitusi yang dikenal sebagai S-Box. Setiap *byte* dalam matriks *state* direpresentasikan sebagai S’[r, c], yang merupakan elemen hasil dari perpotongan antara baris (x) dan kolom (y) (Kurnia Mahesa Naibaho et al., 2021). S-Box berupa matriks

berukuran 16×16 yang berisi nilai-nilai heksadesimal di mana nilai-nilai ini diperoleh melalui perhitungan tertentu (Hena & Kusumaningsih, 2024). Berikut ini tabel S-Box :

Pada [Tabel II.3 Tabel Substitusi (S-Box)](#_bookmark28) sebagai contoh, nilai heksadesimal {62} merujuk ke baris 7 dan kolom 3 dalam S-Box, yang berisi nilai {aa}. Dengan demikian, nilai

{62} dipetakan menjadi nilai {aa}.

* + - * 1. *ShiftRows*

Transformasi *ShiftRows* merupakan proses pergeseran *byte* di mana *byte* paling kiri dipindahkan ke posisi paling kanan (rotasi bit). Jumlah pergeseran dalam melakukan *ShiftRows* ini dilakukan perbedaan setiap baris dalam matriks *state*. Pada baris pertama, tidak ada pergeseran yang dilakukan. Kemudian pada baris kedua barulah mengalami pergeseran satu *byte* ke kiri, lalu baris ketiga digeser dua *byte* ke kiri, dan baris keempat digeser tiga *byte* ke kiri. Tujuan dari proses ini adalah untuk menciptakan *diffusion*(), yang memiliki tujuan untuk membantu menyebarkan efek transformasi non-linear ke seluruh baris dalam matriks *state* untuk proses selanjutnya (Cristy & Riandari, 2021). Proses pergeseran ini bertujuan untuk mencampur elemen-elemen dalam *state* sehingga memperkuat kerahasiaan data.

*MixColumns* adalah proses transformasi di mana setiap elemen dalam blok *cipher* dikalikan dengan matriks . Transformasi ini bertujuan untuk mencampur elemen- elemen dalam setiap kolom *state* berguna meningkatkan difusi data dan memperkuat keamanan enkripsi . Transformasi ini bekerja dengan membagi matriks atau *state* menjadi empat bagian, untuk setiap kolomnya (Widyawan & Imelda, 2021). Setiap kolom dalam matriks *state* diperlakukan sebagai *polinomial* dengan empat bagian dalam *Galois Field* (GF), pada persamaan [(II-2)](#_bookmark30) dilakukan Algoritma dari *AddRoundkey* di mana :

𝑋8 = Elemen dengan derajat ke-8 yang akan muncul sebagai 100000000 (1 diposisi ke 9 dari kanan) di bilangan biner.

X = Menunjukkan elemen dengan derajat ke 1 yang akan muncul sebagai 00000010 ( 1 di posisi ke 2 dari kanan

) dalam bilangan biner

1 = Elemen konstanta yang akan muncul sebagai

00000001 ( 1 diposisi ke 1 dari kanan), dimana elemen 1 ini menyatakan bahwa *polinomial* ini memiliki koefisien konstan.

Proses *MixColumns* juga dapat direpresentasikan sebagai operasi perkalian matriks, di mana setiap kolom matriks dikalikan dengan matriks tertentu dalam *Galois Field* (Cristy & Riandari, 2021).

* + - 1. **Proses *dekripsi* AES :**

Proses *dekripsi* merupakan kebalikan dari proses enkripsi. *Transformasi bytes* yang digunakan dalam *dekripsi* mencakup *AddRoundKey*, *InvShiftRows*, *InvSubBytes,* dan *InvMixColumns*. Pada iterasi pertama *dekripsi*, dilakukan *transformasi AddRoundKey, Inverse ShiftRows, dan Inverse SubBytes. Ciphertex*t kemudian akan mengalami transformasi *AddRoundKey* .

* + - * 1. *InvShiftRows*

Transformasi *byte* yang merupakan kebalikan dari proses *ShiftRows*. Dalam transformasi ini, bit-bit pada *state* digeser ke arah kanan, berbeda dengan *ShiftRows* yang melakukan pergeseran bit ke arah kiri.

* + - * 1. *InvSubBytes*

Transformasi *byte* yang menjadi kebalikan dari *SubBytes*. Dalam proses ini, setiap elemen pada *state* dipetakan menggunakan tabel substitusi *Inverse* S-Box.

* + - * 1. *InvMixColumns*

Setiap kolom dalam *state* dikalikan dengan matriks khusus yang digunakan dalam proses perkalian pada algoritma AES (Syahputra Sianipar et al., 2024)

*Gambar II.4 Proses AES*

Pada [Gambar II.4 Proses AES](#_bookmark31) sebelah kiri menunjukkan tahapan proses enkripsi pada algoritma AES Modifikasi. Proses dimulai dengan memasukkan plainteks, kemudian melalui serangkaian langkah seperti *AddRoundkey*, SubBytes, ShiftRows, MixColumns, dan diakhiri dengan *AddRoundkey* tambahan. Langkah ini diulang sebanyak Nr−1 ronde untuk meningkatkan keamanan data, menghasilkan cipherteks di akhir proses. Flowchart sebelah kanan menjelaskan tahapan proses *dekripsi*, yang merupakan kebalikan dari proses enkripsi. Proses dimulai dengan memasukkan cipherteks, kemudian melalui langkah-langkah seperti *AddRoundkey*, Inverse ShiftRows, Inverse SubBytes, dan Inverse MixColumns. Setelah semua ronde selesai, data dikembalikan ke bentuk plainteks asli.

### Chacha20

Algoritma ChaCha20 adalah salah satu jenis *stream cipher* yang dikembangkan sebagai varian dari algoritma Salsa20. Algoritma ini berbasis desain yang serupa dengan Salsa20, namun ChaCha20 meningkatkan tingkat *diffusion* pada setiap ronde untuk memperkuat keamanan proses enkripsi (Anindita & 18219086, 2023). Chacha20 beroperasi pada unit data 32-bit menggunakan kunci 256-bit. Algoritma ini menghasilkan *output* berupa *stream* kunci sepanjang 512-bit, yang kemudian akan di XOR kan dengan *plainteks* untuk menghasilkan *chiperteks*(Alyas & Abdullah, 2021b). ChaCha20 dimulai dengan membuat *state* 512-bit, yang berupa *array* yang berukuran 16x32-bit. State ini disusun dari empat konstanta 32 bit, kunci rahasia sepanjang 256-bit, penghitung blok 32-bit, serta nonce Ini menunjukkan struktur *state* ChaCha20 yang terdiri dari matriks 4x4 dengan 16 elemen 32-bit. Matriks ini diinisialisasi dengan empat konstanta 32-bit, kunci 256-bit yang dibagi menjadi 8 bagian, penghitung 32 blok 32-bit, dan Nonce 96-bit. Elemen-elemen ini digunakan untuk menghasilkan *keystream* yang kemudian di XOR kan dengan *plainteks* untuk menghasilkan *chiperteks*. Di mana *const* (konstanta) ini berisi empat nilai tetap 32-bit, *key* (kunci) yang terbagi 8 bagian 32-bit dari kunci rahasia 256-bit, *counter* (penghitung), dan Nonce (nilai acak) yang berukuran 96-bit.

Langkah – Langkah Algoritma ChaCha20 :

* + - 1. *Inisialisasi* State

Algoritma ChaCha20 dimulai dengan membangun sebuah state internal berbentuk matriks 4x4 (32-bit) (Barbero et al., 2022) . Matriks ini terdiri dari:

* + - * 1. Baris pertama: Berisi 4 konstanta tetap (128-bit), yang digunakan untuk mengenali algoritma.
        2. Baris kedua dan ketiga: Berisi kunci rahasia 256-bit (8 elemen 32-bit), yang digunakan untuk proses enkripsi.
        3. Baris keempat: Berisi kombinasi 64-bit counter dan 64-bit nonce (masing-masing terdiri dari 2 elemen 32-bit).
      1. *Quarter Round*

Operasi ini memodifikasi 4 elemen dari matriks ( a, b, c, d ) (Almukhlifi & Vora, 2020)

𝑎 += 𝑏; 𝑑 ^ = 𝑎; 𝑑 <<<= 16;

𝑐 += 𝑑; 𝑏 ^ = 𝑐; 𝑏 <<<= 12;

𝑎 += 𝑏; 𝑏 ^ = 𝑎; 𝑑 <<<= 8;

𝑐 += 𝑑; 𝑏 ^ = 𝑐; 𝑏 <<<= 7;

* + - 1. *Double Round*

*Double Round* adalah kombinasi dari dua *Quarter Round* yang dijalankan secara berurutan. Setiap iterasi *Double Round* mencakup:

* + - * 1. Operasi pada kolom matriks *(Column Round):*

melakukan operasi Quarter Round pada setiap kolom matriks 4x4. Tujuannya adalah untuk mencampur data antar elemen dalam kolom, sehingga meningkatkan difusi data secara vertikal.

* + - * 1. Operasi pada diagonal matriks (Diagonal Round):

Setelah operasi pada kolom selesai, langkah berikutnya adalah menerapkan Quarter Round pada elemen-elemen diagonal dalam matriks.

Proses Double Round dilakukan sebanyak 10 iterasi, menghasilkan total 20 ronde Quarter Round untuk setiap blok data yang diproses.

* + - 1. Finalisasi State
      2. *Keystream Generation*

Setelah 20 ronde selesai*, state* akhir ditambahkan kembali ke

### Entropi

Entropi dalam teori informasi digunakan untuk mengukur keacakan data, yang mencerminkan kekuatan algoritma enkripsi. Semakin tinggi nilai entropi, semakin acak data tersebut, yang berarti algoritma enkripsi lebih tahan terhadap serangan (Wahyudi et al., 2024). Entropi dihitung dengan menggunakan rumus pada

Dimana:

* 𝐻𝑚 : Menunjukkan entropi data, dimana tingkat keacakan dalam

# BAB III METODOLOGI PENELITIAN

### Sistematika Penyelesaian Masalah

Penelitian ini bersifat kuantitatif dengan tujuan mengevaluasi performa algoritma AES modifikasi dan kombinasi ChaCha20 dan AES modifikasi. Penilaian performa ini dilakukan berdasarkan waktu enkripsi, waktu *dekripsi*, kecepatan enkripsi, dan kecepatan *dekripsi*, yang akan dianalisis menggunakan data hasil pengujian pada berbagai jenis *file* dengan format PDF, Word dan TXT, yang hasilnya berbentuk tabel untuk mempermudah analisis perbandingan.

### Alat dan Bahan

### Alat

1. Perangkat Keras (Hardware) Laptop dengan spesifikasi :
   1. Prosesor : Intel *Core* I5
   2. RAM : 8 GB
   3. Penyimpanan : 256 GB SSD atau lebih
2. Perangkat Lunak (*Software*)
   1. *Python* :

Menggunakan *library cryptography, pycryptodome*, dan *library* lain yang mendukung algoritma AES dan ChaCha20. Berikut beberapa pustaka yang digunakan :

* + - PyCryptodome : Digunakan Iplementasi algoritma AES dan ChaCha20
    - Cryptography : Implementasi kemanan tambahan (jika nanti dibutuhkan )
    - Numpy : Digunakan untuk analisis data (perhitungan entropi)
    - Time : Mencatat waktu eksekusi algoritma, baik untuk enkripsi maupun dekripsi.
  1. *Text* Editor :

Visual Studio Code dan *Jupyter Notebook*

* 1. Sistem Operasi : Windows

### Bahan

1. Data set *file* :
   1. *File* PDF : Ukuran *file* 100 KB – 100 MB
   2. *File* Word : Ukuran *file* 100 KB – 100 MB
   3. *File* TXT : Ukuran *file* maksimal 300 *byte*
2. Literatur :

Jurnal atau artikel ilmiah tentang AES, ChaCha20 dan *Double Encrypt* (tahun 2020-2024)

1. *Spreadsheet* Analisis Data :

Format CSV (Comma Separated Values)

### Rancangan Penelitian

Pada penelitian ini, dilakukan tahapan yang dirancang secara sistematis untuk mencapai tujuan utama, yaitu Implementasi kombinasi Advanced Encryption Standard (AES) yang dimodifikasi dan ChaCha20. Proses penelitian dimulai dengan studi literatur untuk memahami konsep dasar dan pengembangan terbaru terkait algoritma kriptografi. Setelah itu, dilakukan pengumpulan data yang relevan sebagai dasar implementasi. Tahapan selanjutnya mencakup implementasi algoritma modifikasi AES serta kombinasi algoritma tersebut dengan ChaCha20. Untuk mengukur efektivitas dan efisiensi algoritma yang dikembangkan, pengujian dilakukan pada berbagai jenis *file* seperti PDF, Word, dan TXT. Analisis hasil dari pengujian ini memberikan dasar untuk menarik kesimpulan dan memberikan rekomendasi untuk pengembangan lebih lanjut.

*Gambar III.1 Diagram Alur Penelitian*

[Gambar III.1 Diagram Alur Penelitian](#_bookmark44) ini terdapat beberapa tahapan yaitu:

1. Mulai adalah tahap awal dari penelitian. Pada tahap ini menentukan tujuan dan menyusun langkah-langkah yang akan dilakukan dalam penelitian.
2. Studi literatur dilakukan pengumpulan referensi yang berkaitan dengan algoritma AES, ChaCha20 dan *Double Encrypt. Referensi* berupa jurnal, artikel ilmiah.
3. Pengumpulan data untuk mengumpulkan *file* atau data set yang akan digunakan untuk pengujian algoritma. Format *file* yang digunakan adalah PDF, Word dan TXT.
4. Implementasi modifikasi AES, pada tahap ini algoritma AES dimodifikasi dengan menambahkan langkah *AddRoundkey* sebelum *Mixcolumns*. Tujuannya adalah untuk meningkatkan keamanan dan kecepatan enkripsi dibandingkan dengan AES standar.
5. Implementasi kombinasi AES modifikasi dan ChaCha20, menggabungkan algoritma AES yang sudah dimodifikasi dengan ChaCha20 untuk melakukan *double encypt*. Di mana proses ini bertujuan untuk meningkatkan keamanan *file* lebih baik daripada menggunakan *single encrypt*.
6. Pengujian algoritma, untuk menguji performa algoritma yang sudah diimplementasikan, di mana data set yang dikumpulkan sebelumnya itu digunakan untuk uji coba. Parameter yang diukur pada pengujian ialah

waktu enkripsi dan *dekripsi,* kecepatan enkripsi dan *dekripsi*, dan hasil pengujian ini akan digunakan untuk mengevaluasi kinerja algoritma.

1. Analisis hasil, data pengujian yang sudah terkumpul akan dianalisis. Analisis dilakukan untuk membandingkan performa AES standar, AES yang sudah dimodifikasi, dan kombinasi AES modifikasi dan ChaCha20. Tujuannya sendiri untuk mengetahui algoritma mana yang lebih efisien.
2. Selanjutnya ke kesimpulan dan saran, pada tahap ini merangkum hasil penelitian termasuk kelebihan dan kekurangan algoritma yang sudah diuji. Saran juga untuk pengembangan penelitian lebih lanjut
3. Selesai

### Studi Literatur

Studi literatur dilakukan untuk memahami konsep dasar dan perkembangan terbaru terkait algoritma kriptografi yang digunakan dalam penelitian ini, yaitu Advanced Encryption Standard (AES) dan ChaCha20. Proses ini melibatkan pengumpulan referensi dari jurnal, paper, dan buku yang membahas metode enkripsi, keamanan data, serta teknik evaluasi seperti entropi dan analisis performa. Literatur yang dikaji juga mencakup penelitian sebelumnya tentang algoritma kriptografi untuk keamanan *file*, sehingga memberikan dasar teori yang kuat untuk modifikasi algoritma dan penerapan metode double encryption dalam penelitian ini.

### Pengumpulan Data

Data yang digunakan dalam penelitian ini dibuat sendiri dengan mengikuti ketentuan yang telah ditentukan. Format data meliputi *file* PDF, Word, dan TXT dengan ukuran bervariasi, mulai dari 100 KB hingga 100 MB untuk *file* PDF dan Word, serta hingga 300 *byte* untuk *file* TXT. Data ini dirancang sedemikian rupa agar sesuai dengan kebutuhan pengujian algoritma enkripsi, seperti mengukur waktu enkripsi, waktu *dekripsi*, kecepatan, dan tingkat keacakan menggunakan analisis entropi. Variasi format dan ukuran data ini

bertujuan untuk memastikan bahwa pengujian mencakup berbagai jenis isi *file*.

### Implementasi Modifikasi AES

Pengembangan algoritma *Advanced Encryption Standard* (AES) dilakukan dengan memodifikasi. Modifikasi dilakukan dengan menambahkan *AddRoundkey* sebelum *Mixcolumns*. Penambahan ini bertujuan untuk memperkuat keamanan dan meningkatkan kompleksitas pada proses enkripsi, sehingga lebih sulit untuk diretas dengan pihak tidak berwenang. Berikut adalah proses alur dari modifikasi algoritma AES.

*Gambar III.2 Diagram Proses Enkripsi AES*

Pada [Gambar III.2 Diagram Proses Enkripsi AES](#_bookmark48) ini digunakan untuk mengamankan *file* dengan melakukan tahap enkripsi terhadap data *plainteks* yang menggunakan kunci tertentu. Dalam penelitian ini, modifikasi dilakukan dengan menambahkan langkah *AddRoundkey* untuk meningkatkan tingkat keamanan dan kompleksitas enkripsi. Berikut langkah-langkah proses modifikasi AES .

1. Input *file* dan kunci

Proses enkripsi dimulai dengan memasukkan *file* input (*file* asli yang akan dienkripsi) dan kunci enkripsi. *File* input dibaca dalam mode biner, diubah menjadi *byte array*, lalu dipecah menjadi blok-blok data berukuran 16 *byte*. Jika panjang data tidak habis dibagi 16 *byte, padding* dilakukan dengan metode PKCS#7 untuk melengkapinya.

Contoh :

* 1. *File* input : contoh.pdf

Isi *file* dalam biner : [0x25, 0x50, 0x44, 0x46, ...]

*Byte array* : [37, 80, 68, 70, 10, 0, 45, 78, ...]

* 1. Kunci enkripsi :

Kunci : surabayaitelkom

ASCII : [115, 117, 114, 97, 98, 97, 121, 97, 105, 116, 116,

101, 108, 107, 111, 109]

* 1. Matriks data *file* :

|  |  |  |  |
| --- | --- | --- | --- |
| [37, | 80, | 68, | 70] |
| [10, | 0, | 45, | 78] |
| [255, | 100, | 32, | 12] |
| [80, | 50, | 60, | 72] |

1. *AddRoundkey*

Merupakan tahap awal proses enkripsi, di mana *plainteks* ini digabungkan dengan kunci awal yang menggunakan operasi logika yaitu XOR. Di mana blok data *file* digabungkan dengan kunci awal (W0-W3) menggunakan operasi XOR untuk menambahkan lapisan keamanan awal.

Contoh :

* 1. Proses XOR :

37 XOR 115 = 86

80 XOR 117 = 37

68 XOR 114 = 54

70 XOR 97 = 39

* 1. Matriks setelah *AddRoundkey* :

[86, 37, 54, 39]

[108, 97, 76, 47]

[150, 16, 84, 97]

[28, 89, 83, 37]

1. Proses ronde 1-9
   1. *SubBytes*

Pada proses ini, nantinya setiap *byte* pada matriks *state* ( data yang diperoleh *AddRounKey*) akan diubah menggunakan tabel substitusi yang biasa disebut dengan S-Box. *Tranformasi* ini bertujuan untuk meningkatkan keacakan data agar sulit diprediksi.

Contoh proses :

*Byte* 37 diganti dengan nilai dari tabel S-Box jadi 120

* 1. *ShiftRows*

Setiap baris dalam matriks *state* nantinya digeser ke kiri pada baris pertama tidak ada pergeseran, baris kedua di geser 1 *byte* ke kiri, baris ketiga ini digeser 2 *byte* ke kiri, dan terakhir baris keempat yang nantinya digeser 3 *byte* ke kiri. Contoh proses :

[244, 120, 99, 200] [244, 120, 99, 200]

[60, 107, 55, 175] [107, 55, 175, 60]

[12, 243, 172, 85] [172, 85, 12, 243]

[180, 97, 92, 65] [65, 180, 97, 92]

* 1. *AddRoundkey* (W4-W7)

Setelah *ShiftRows* dilakukan operasi XOR antara matriks *state* dengan kunci ronde berikutnya yaitu (W4-W7) Contoh proses :

*Byte* 244 XOR 115 = 135

*Byte* 120 XOR 117 = 13

Hasil *AddRounKey* : [135, 13, 17, 169]

[38, 98, 85, 12]

[55, 32, 82, 84]

[49, 53, 180, 193]

* 1. *MixColumns*

Setiap kolom pada matriks *state* dimodifikasi dengan operasi matematis berbasis matriks yang menggunakan bidang *Galois* (GF). Di mana proses ini bertujuan untuk mencampur elemen-elemen dalam kolom dan meningkatkan difusi data. [2, 3, 1, 1]

[1, 2, 3, 1]

[1, 1, 2, 3]

[3, 1, 1, 2]

* 1. *AddRoundkey* (W36-W39)

Dilakukan setelah *MixColums* untuk menambahkan lapisan keamanan dengan kunci ronde tambahan (W36-W39).

1. Pada ronde terakhir, hanya dilakukan tiga langkah yaitu

*SubBytes, ShiftRows*, dan *AddRoundkey* tanpa *MixColumns.*

Proses modifikasi algoritma AES ini digunakan untuk mengenkripsi *file* dengan menambahkan langkah *AddRoundkey* tambahan, yang meningkatkan keamanan dengan memberikan lapisan XOR tambahan pada setiap ronde enkripsi.

* + 1. **Implementasi Kombinasi AES Modifikasi dan ChaCha20** Algoritma AES modifikasi dan ChaCha20 dengan menerapkan *double encypt.* Proses ini memakai AES modifikasi dengan memberikan perlindungan berbasis blok *chiper* yang kuat dengan memodifikasi *AddRoundkey* tambahan sebelum *MixColumns,* sedangkan ChaCha20 sebagai *stream cipher* yang cepat dan aman, meningkatkan kompleksitas keseluruhan dengan memanfaatkan proses enkripsi berbasis *keystream.*

Pada metode ini, setiap *file* menggunakan 1 kunci enkripsi. Kunci yang digunakan memiliki panjang awal 128-bit, yang kemudian diperluas menjadi 256-bit pada ChaCha20 dengan menambahkan padding nol (0x00) jika panjangnya kurang dari 256-bit. Proses

dimulai dengan mengenkripsi *file* menggunakan AES Modifikasi untuk menghasilkan cipherteks awal dalam format matriks 4x4. Cipherteks ini kemudian diubah menjadi byte stream untuk kompatibilitas dengan ChaCha20. Selanjutnya, ChaCha20 melakukan proses enkripsi kedua dengan memanfaatkan kunci 256- bit, nonce sepanjang 96-bit, dan counter. Hasil akhir adalah cipherteks final yang lebih aman karena telah melalui dua lapisan enkripsi. Penggunaan double encryption dengan satu kunci untuk setiap *file* memberikan keamanan yang lebih tinggi, di mana kombinasi kedua algoritma menciptakan perlindungan yang kuat terhadap berbagai jenis serangan kriptografi.

*Gambar III.3 Diagram Proses Double Encrypt*

Pada [Gambar III.3 Diagram Proses Double Encrypt](#_bookmark50) menunjukkan langkah-langkah enkripsi ganda. Proses dimulai dengan mengenkripsi *plainteks* menggunakan AES modifikasi untuk menghasilkan *chiperteks* AES. *Chiperteks* ini kemudian dienkripsi ulang menggunakan ChaCha20, yang melibatkan matriks 4x4, proses *quarter-round,* dan operasi XOR dengan *keystream*. Algoritma AES modifikasi dan ChaCha20 menggunakan kunci yang sama, yaitu kunci AES sepanjang 128-bit. berikut adalah tahapan alur di atas :

1. Input *plainteks* dan *key* dan enkripsi AES modifikasi
2. Input *cipherteks* AES ke ChaCha20

*Chiperteks* yang dihasilkan dari proses enkripsi pertama diubah menjadi input untuk algoritma ChaCha20. Ini dilakukan untuk menerapkan enkripsi berlapis yaitu *double encryption,* sehingga keamanan data akan meningkat secara signifikan. *Chiperteks* AES yang biasanya dalam format matriks (4x4) ini diubah ke format *byte stream* untuk kompatibilitas dengan algoritma ChaCha20. Byte stream ini memungkinkan untuk memproses data, sesuai dengan *stream chiper*.

Contoh :

[115, 75, 189, 234, 201, 97, 58, 14, 245, 142, 12, 120, 53, 190,

201, 20] data ini sudah dalam bentuk *byte stream*.

1. Validasi panjang *key* (256-bit *Key*)

Ini untuk memastikan bahwa kunci enkripsi untuk ChaCha20 sesuai dengan spesifikasi algoritma, yaitu panjangnya harus tepat 256 bit (32 *byte*). Kunci ini untuk menghasilkan *keystream* yang beda dalam proses enkripsi. Jika panjang ke kurang dari 256 bit, ditambahkan *padding* berupa *byte* nol (0x00) untuk memenuhi panjang. Jika *key* yang lebih panjang, *key* akan dipotong hingga hanya menyisakan 256 bit. *Key* kemudian dibagi menjadi 8 bagian ( di mana masing-masing 32 bit) untuk digunakan dalam proses pembentukan matriks awal (*state* matriks)

Contoh :

* 1. *Key* input “surabayaittelkom”

[115, 117, 114, 97, 98, 97, 121, 97, 105, 116, 116, 101, 108,

107, 111, 109]

* 1. Panjang *key* adalah 16 *byte* (128 bit), sehingga tidak memenuhi syarat panjang 256 bit. Jadi ditambahkan *padding* nol (0x00) hingga mencapai 32 *byte:*

[115, 117, 114, 97, 98, 97, 121, 97, 105, 116, 116, 101, 108,

107, 111, 109, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

1. Membuat *Initial* State Matriks 4x4

Membentuk *state* matriks awal yang menjadi dasar operasi dalam ChaCha20. Matriks ini berisi kunci, *counter*, dan *nonce.* Matriks 4x4 dalam ChaCha20 terdiri 16 elemen (masing-masing 32 bit) Contoh :

* 1. 4 elemen konstanta tetap “*expand* 32-byte k”4 elemen 32 bit [1634760805, 857760878, 2036477234, 1797285236]
  2. Elemen kunci 256-bit dibagi menjadi 8 elemen 32-bit: [1936028945, 1885434721, 1701147239, 1647539057,

1225269481, 1226844037, 1214606441, 0]

* 1. 1 *Counter*: Nilai awal 0.
  2. 3 Elemen Nonce: Nilai *nonce* yang unik untuk sesi enkripsi: Contoh *nonce* : [12345678, 87654321, 11223344]
  3. Matriks yang dibentuk

[1634760805, 857760878, 2036477234, 1797285236]

[1936028945, 1885434721, 1701147239, 1647539057]

[1225269481, 1226844037, 1214606441, 0]

[0, 12345678, 87654321, 11223344]

Baris ke 1 konstanta, ke 2 *key,* ke 3 *key* dan ke 4 *counter* dan

*nonce*

1. Operasi Quarter Round (20 iterasi)

Melakukan proses pencampuran data di dalam matriks untuk meningkatkan kerandoman dan keamanan *keystream* yang dihasilkan. Quarter Round adalah inti dari algoritma ChaCha20. Quarter Round dilakukan pada 4 elemen tertentu dari matriks. Penjumlahan Modular (mod 2³²): Menambah elemen dalam matriks untuk mencampur data. Operasi XOR: Mencampur data antar elemen melalui operasi logika XOR. Rotasi Bit (ROTL): Menggeser bit elemen untuk mendistribusikan informasi secara acak. Setelah 20 iterasi (10 putaran double round), matriks akan diubah menjadi bentuk yang sangat acak.

1. *Counter* +=1

*Counter* bertambah untuk setiap blok data yang diproses, memastikan *keystream* yang dihasilkan unik untuk setiap blok. Setelah satu blok data selesai diproses, *counter* di matriks bertambah sebesar 1.

Contoh :

*Counter* baru : [1, 12345678, 87654321, 11223344]

1. XOR *keystream* dengan *chiperteks* AES

Menggabungkan *keystream* hasil ChaCha20 dengan *cipherteks AES* melalui operasi XOR untuk menghasilkan *final ciphertext*. XOR dipilih karena sifatnya yang sederhana namun aman dan mudah dibalik. *Keystream* yang dihasilkan dari matriks ChaCha20 di-XOR-kan dengan *cipherteks AES*.

Operasi XOR:

Jika *ciphertext* = *keystream* XOR *plaintext*, maka *dekripsi*

dilakukan dengan cara *plaintext* = *keystream* XOR *ciphertext*.

* + - 1. **Proses *Dekripsi* Double Encryption**

Proses *dekripsi* dilakukan untuk mengembalikan data yang telah terenkripsi ke bentuk aslinya atau *plainteks*. Di mana pada tahap ini, *chiperteks* hasil dari kombinasi algoritma AES modifikasi dan Chacha20 akan *didekrpsi* secara urut. Proses nantinya akan dimulai dengan *mendekripsi chiperteks* dengan ChaCha20 untuk mendapatkan kembali *chiperteks* AES modifikasi, kemudian dilanjutkan dengan *mendekripsi chiprteks* AES modifikasi untuk memperoleh *plainteks* asli.

Pada [Gambar III.4 Proses Dekripsi](#_bookmark51) Diagram alur di atas menjelaskan proses *dekripsi* dari kombinasi algoritma ChaCha20 dan modifikasi AES. Proses dimulai dengan memasukkan cipherteks hasil enkripsi sebagai input. Langkah pertama adalah men*dekripsi* cipherteks menggunakan algoritma ChaCha20 untuk menghasilkan keystream. Keystream ini kemudian dikombinasikan dengan cipherteks menggunakan operasi XOR untuk mendapatkan cipherteks AES.

Selanjutnya, proses ekspansi kunci dilakukan untuk menghasilkan kunci yang akan digunakan pada setiap ronde *dekripsi* AES. Proses *dekripsi* AES terdiri dari tiga tahapan utama: ronde terakhir (Round 10) yang melibatkan *AddRoundKey*, InvShiftRows, dan InvSubBytes; ronde utama (Round 9-1) yang melibatkan *AddRoundKey*, InvMixColumns, *AddRoundKey, InvShiftRows,* dan *InvSubBytes* tambahan; dan ronde awal (Round 0) yang menggunakan *AddRoundKey* (W0-W3). Hasil akhir dari seluruh proses ini adalah plainteks, yaitu data asli yang berhasil di*dekripsi*. Kombinasi kedua algoritma ini memastikan keamanan data yang lebih kuat melalui enkripsi ganda.

*Gambar III.4 Proses Dekripsi*

### Implementasi Antarmuka Pengguna (UI)

Antarmuka pengguna (UI) dirancang untuk mempermudah interaksi, berfungsi sebagai media interaktif yang memungkinkan pengguna untuk memilih file, memasukkan kunci enkripsi, serta memulai proses enkripsi dan dekripsi dengan mudah. Implementasinya menggunakan Flask sebagai framework backend untuk menangani

proses enkripsi dan dekripsi, sedangkan HTML digunakan untuk membuat form input file dan kunci enkripsi. CSS digunakan untuk mempercantik tampilan agar antarmuka lebih menarik, jelas, dan mudah digunakan oleh pengguna dalam proses enkripsi dan dekripsi file menggunakan kombinasi algoritma ChaCha20 dan AES yang telah dimodifikasi.

*Gambar III.5 Desain Antarmuka Pengguna Enkripsi dan Dekripsi*

Pada [Gambar III.5 Desain Antarmuka Pengguna Enkripsi dan](#_bookmark53) [Dekripsi](#_bookmark53) Gambar tersebut menampilkan desain antarmuka pengguna (UI) dari sebuah aplikasi berbasis web yang digunakan untuk enkripsi dan *dekripsi file*. Antarmuka ini memiliki beberapa elemen input pengguna untuk mengunggah *file* dan memilih opsi enkripsi atau *dekripsi*. Terdapat tiga kolom utama pada desain ini: pertama, Pilih *File*, untuk pengguna memilih *file* yang akan diproses. Kedua, Enkripsi atau *Dekripsi*, yang memberikan pilihan kepada pengguna apakah *file* yang dipilih akan dienkripsi atau di *dekripsi*, yang merupakan opsi penting untuk menentukan jenis operasi yang akan dilakukan pada *file* tersebut. Ketiga, Key, di mana pengguna diminta untuk memasukkan kunci yang diperlukan untuk proses enkripsi atau *dekripsi*. Kunci ini memainkan peran penting dalam menjaga keamanan data yang diproses. Di bagian bawah antarmuka, terdapat

tombol Proses yang akan menjalankan operasi sesuai dengan pilihan pengguna, baik itu enkripsi maupun *dekripsi*.

### Pengujian Algoritma

Menjelaskan pengujian yang dilakukan untuk mengevaluasi kinerja dan keamanan algoritma berdasarkan parameter seperti entropi, waktu enkripsi dan *dekripsi*, serta kecapatan.

### Pengujian Entropi

Pengujian kekuatan algoritma dilakukan dengan mengukur tingkat keacakan data hasil enkripsi menggunakan analisis entropi. Keacakan data merupakan indikator penting dalam menentukan seberapa baik algoritma enkripsi menyamarkan pola asli dari data. Semakin tinggi tingkat keacakan, semakin sulit bagi pihak yang tidak berwenang untuk menganalisis pola atau memprediksi isi data terenkripsi. Pengujian ini dilakukan dengan menghitung nilai entropi menggunakan distribusi probabilitas setiap simbol dalam cipherteks. Nilai entropi yang mendekati maksimum menunjukkan bahwa distribusi simbol dalam cipherteks hampir merata, menandakan bahwa algoritma telah berhasil menghasilkan data yang acak. Data dengan keacakan tinggi lebih tahan terhadap serangan berbasis statistik, sehingga pengujian ini memberikan evaluasi kuantitatif terhadap kekuatan keamanan algoritma yang diuji. Hasil pengujian dibandingkan antara AES Standar, AES Modifikasi, Chacha20 dan Double Encryption untuk menunjukkan efektivitas modifikasi dalam meningkatkan keamanan.

### Pengujian Waktu dan Kecepatan

Menguji algoritma AES modifikasi dan *Double encrypt* AES modifikasi dan ChaCha20. Pengujian ini dilakukan untuk menilai kinerja algoritma dalam berbagai parameter. Penelitian ini akan menganalisis parameter pengujian menggunakan matriks dan formula berikut :

* + - * 1. Waktu enkripsi :

mengukur waktu yang dibutuhkan untuk mengenkripsi data dari

*plainteks* menjadi *chiperteks* dijelaskan dalam persamaan [(III-1)](#_bookmark55).

|  |  |
| --- | --- |
| 𝐸𝑡 = 𝐸𝑁𝑡 – 𝑆𝑇𝑡 | (III-1) |

Di mana:

𝐸𝑡 = waktu enkripsi dalam mili detik (ms)

𝐸𝑁𝑡 = waktu selesai enkripsi

𝑆𝑇𝑡 = waktu mulai enkripsi

Waktu ini akan dihitung untuk setiap *file* yang diuji, dan hasilnya nanti akan memberikan gambaran mengenai kecepatan algoritma dalam melakukan proses enkripsi.

* + - * 1. Waktu *dekripsi* :

Mengukur waktu yang dibutuhkan untuk *mendekripsi* data dari

*chiperteks* menjadi *plainteks* dijelaskan pada persamaan [(III-2)](#_bookmark56).

Dimana:

𝐷𝑡 = waktu *dekripsi* (ms)

𝐸𝑁𝑡 = waktu selesai *dekripsi*

𝑆𝑇𝑡 = waktu mulai *dekripsi*

Hasil waktu *dekripsi* ini juga perlu untuk masing-masing algoritma karena ini menganalisis perbandingan kinerja antara algoritma enkripsi.

* + - * 1. Kecepatan :

Menunjukkan jumlah data dalam *kilobyte* yang dapat dienkripsi

Et = waktu enkripsi atau *dekripsi* (s)

### Pengujian pada Berbagai Jenis *File*

*File* PDF yang digunakan dalam pengujian dapat berisi elemen seperti teks, gambar, dan tabel yang semuanya akan dienkripsi secara utuh karena algoritma bekerja pada level data biner. *File* PDF dengan ukuran 100 KB, 1 MB, 10 MB, hingga 100 MB diuji untuk memastikan algoritma dapat bekerja pada dokumen berukuran kecil hingga besar. Hal serupa juga berlaku untuk *file* Word dengan ukuran yang sama, meskipun memiliki struktur internal yang lebih kompleks dibandingkan PDF. *File* TXT dengan ukuran maksimal 300 *byte*, yang biasanya hanya berisi teks sederhana, juga dapat dienkripsi ChaCha20 (*double encrypt*). Di mana pengujian ini bertujuan untuk

mengukur kinerja algoritma. Berikut ini adalah contoh hasil pengujian terhadap *file* PDF :

*Tabel III.2 Evaluasi Hasil*

Pada [Tabel III.2 Evaluasi Hasil](#_bookmark60) menunjukkan hasil evaluasi simulasi kinerja algoritma enkripsi dan *dekripsi*. pada Nilai-nilai tersebut memberikan gambaran awal mengenai performa beberapa metode, termasuk AES Standar, ChaCha20, AES Modifikasi, dan kombinasi AES Modifikasi dengan ChaCha20. Hasil ini digunakan sebagai contoh output yang diharapkan dari model algoritma yang akan dirancang, ini belum merupakan data dari eksperimen nyata.

# DAFTAR PUSTAKA

Alimuddin, A., Juntak, J. N. S., Jusnita, R. A. E., Murniawaty, I., & Wono, H. Y. (2023). Teknologi dalam pendidikan: Membantu siswa beradaptasi dengan revolusi industri 4.0. *Journal on Education*, *5*(4), 11777–11790.

Almukhlifi, R., & Vora, P. L. (2020). Linear cryptanalysis of reduced-round simon using super rounds. *Cryptography*, *4*(1), 1–34. https://doi.org/10.3390/cryptography4010009

Alyas, H. H., & Abdullah, A. A. (2021). Enhancement the ChaCha20 Encryption Algorithm Based on Chaotic Maps. *Lecture Notes in Networks and Systems*, *201*, 91–107. https://doi.org/10.1007/978-981-16-0666-3\_10

Anindita, M., & 18219086, A. A. (2023). *Analisis Perbandingan Algoritma AES dan ChaCha20-Poly1305 dalam Enkripsi Konten Livestreaming*.

Aryanto, M. B., Tahir, M., Devita, S. I., Mustofa, Z. N., Ainiyah, Q., & Sundoro,

S. (2023). Implementasi Enkrip Dan Dekrip File Menggunakan Metode Advance Encryption Standard (AES-128). *Jurnal Ilmiah Sistem Informasi Dan Ilmu Komputer*, *3*(1), 89–104.

Assyahid, M. H. (2024). *Implementasi interpolasi pixel wadah citra dan enkripsi aes pada steganografi least significan bit (LSB)*. https://repository.uinjkt.ac.id/dspace/handle/123456789/76727

Barbero, S., Bazzanella, D., & Bellini, E. (2022). Rotational Cryptanalysis on ChaCha Stream Cipher. *Symmetry*, *14*(6). https://doi.org/10.3390/sym14061087

Belvin Tannadi. (2022). *Ilmu Crypto* (Digital). Jakarta : Elex Media Komputindo.

Cristy, N., & Riandari, F. (2021). *Niolinda Cristy 1 , Fristi Riandari 2 [Implementasi Metode Advanced Encryption Standard (AES 128 Bit) Untuk Mengamankan Data Keuangan*. *4*(2), 75.

Djong, H. S., & Siswanto, S. (2022). Implementasi Kriptografi Dengan Menggunakan Metode RC4 dan AES-256 Untuk Mengamankan File Dokumen pada PT Varnion Technology Semesta. *Prosiding Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI)*, *1*(1), 149–158.

Handoyo, J., & Subakti, Y. M. (2020). Keamanan Dokumen Menggunakan Algoritma Advanced Encryption Standard (AES). *Jurnal SITECH : Sistem Informasi Dan Teknologi*, *3*(2), 143–152. https://jurnal.umk.ac.id/index.php/sitech/article/view/5865

Hena, N., & Kusumaningsih, D. (2024). Pengamanan File Pada Sistem Master Vendor Berbasis WEB Menggunakan Algoritma AES Pada Trinitiland. *Prosiding Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI)*, *3*(2), 30–37.

Hidayatulloh, N. W., Tahir, M., Amalia, H., Basyar, N. A., Prianggara, A. F., & Yasin, M. (2023). Mengenal Advance Encryption Standard (AES) sebagai Algoritma Kriptografi dalam Mengamankan Data. *Digital Transformation Technology*, *3*(1), 1–10.

Kebande, V. R. (2023). Extended-Chacha20 Stream Cipher With Enhanced Quarter Round Function. *IEEE Access*, *11*, 114220–114237. https://doi.org/10.1109/ACCESS.2023.3324612

Kim, B. G., & Wong, D. (2023). Smart Contract-based Secure Verifiable Random Function using ChaCha20 Sequence in Blockchain. *ACM International Conference Proceeding Series*, 41–51. https://doi.org/10.1145/3638025.3638028

Kurnia Mahesa Naibaho, A., Putri, A., & Fikri Husaini, Y. (2021). Analisis Sistem Keamanan Informasi Chat Online dengan Menggunakan Algoritma AES. In *Ilmu Komputer* (Vol. 6, Issue 2).

Luong, T. T. (2023). Strengthening AES Security Through Key-Dependent ShiftRow and AddRoundKey Transformations Utilizing Permutation. In *IJACSA) International Journal of Advanced Computer Science and Applications* (Vol. 14, Issue 11). [www.ijacsa.thesai.org](http://www.ijacsa.thesai.org/)

Muhammed, R. K., Rashid Aziz, R., Hassan, A. A., Aladdin, A. M., Saydah, S. J., Rashid, T. A., & Hassan, A. (2024). *Comparative Analysis of AES, Blowfish, Twofish, Salsa20, and ChaCha20 for Image Encryption*.

Olivia Putri Irine Irawan, B., Tahir, M., Ayu Windrastuti, N., Yurina Cholili, D., Mulaikah, D., Batsul Mushofi Septian wachid, A., & Pendidikan Informatika,

P. (2023). Implementasi Kriptografi Pada Keamanan Data Menggunakan Algoritma Advance Encryption Standard (AES). *Jurnal Simantec*, *11*(2), 167–

174. https://doi.org/10.21107/SIMANTEC.V11I2.20034

Pangestu, G. Y., Hadiana, A. I., & Sabrina, P. N. (2022). Kriptografi Untuk Enkripsi Ganda Pada Gambar Menggunakan Algoritma AES (Advanced Encryption Standard) Dan RC5 (Rivest Code 5). *Vol*, *1*, 25–32.

Putri, A. E., Kartikadewi, A., & Rosyid, L. A. A. (2021). Implementasi Kriptografi dengan Algoritma Advanced Encryption Standard (AES) 128 Bit dan Steganografi menggunakan Metode End of File (EOF) Berbasis Java Desktop

pada Dinas Pendidikan Kabupaten Tangerang. *Appl. Inf. Syst. Manag*, *3*(2), 69–78.

Saripa, S. (2023). Implementation Implementation of a File Security System Using the AES Algorithm to Secure Personal Files: Implementasi Sistem Keamanan File Menggunakan Algoritma AES untuk Mengamankan File Pribadi. *Progressive Information, Security, Computer, and Embedded System*, *1*(2), 138–148.

Sekar Putri Ananda, Saepul Lukman, & Irfan. (2022). *Analisa Metode Kriptogra Modern Advance Encryption Standar (AES) 128 Bit dalam Mengenkripsi dan Mendekripsi File Dokumen Digital*. https://doi.org/10.32409/jikstik.21.3.2973

Soesanto, E., Purba, L. M., Aprilia, B., Putra, D. R., & Putri, S. D. (2023). Implementasi Objek Vital, Pengamanan File dan Pengamanan Cyber di PT Pertamina. *IJM: Indonesian Journal of Multidisciplinary*, *1*(1).

Surbakti, P. S., & Purwanto, P. (2024). Implementasi Kriptografi Dengan Menggunakan Algoritma AES-128 Untuk Enkripsi dan Dekripsi File Dokumen Berbasis Web Pada Law Office Erdi Surbakti, SH & Rekan. *Prosiding Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI)*, *3*(1), 1–8.

Syahputra Sianipar, J., Budi Nuugroho, N., Mariami, I., Informasi, S., & Triguna Dharma, S. (2024). Pengamanan Data Gaji Karyawan Dengan Menggunakan Metode Advanced Encryption Standard (AES). *Jurnal Sistem Informasi Triguna Dharma (JURSI TGD)*, *3*(1), 35–45. https://doi.org/10.53513/JURSI.V3I1.5653

Wahyu Putra, M. R. F. T. H. (2023, July 2). *Implementasi Algoritma Advanced Encryption Standard Untuk Kemanan Dokumen*. https://journal.grahamitra.id/index.php/jurikti/article/view/55/181

Wahyudi, E. N., Ardhianto, E., Handoko, W. T., Murti, H., Supriyanto, E., Lestariningsih, E., & Redjeki, R. S. (2024). Peningkatan Keamanan Data Melalui Teknik Super Enkripsi Menggunakan Algoritma Vigenere dan Caesar. *Jurnal Informatika Polinema*, *10*(3), 315–322. https://doi.org/10.33795/JIP.V10I3.5131

Widyawan, D., & Imelda, I. (2021). Pengamanan File Menggunakan Kriptografi Dengan Metode Aes-128 Berbasis Web Di Komite Nasional Keselamatan Transportasi. *SKANIKA: Sistem Komputer Dan Teknik Informatika*, *4*(1), 15– 22.

**IMPLEMENTASI KOMBINASI CHACHA20 DAN MODIFIKASI AES MELALUI *ADDROUNDKEY* PADA**

**STUDI KASUS KEAMANAN *FILE***

**Proposal Tugas Akhir Mata Kuliah Penulisan Proposal**

## (CAK4EAB2)

Disusun oleh:

## EMILIA RAMONA 1203210017

**PROGRAM STUDI SARJANA INFORMATIKA DIREKTORAT KAMPUS SURABAYA**

**UNIVERSITAS TELKOM SURABAYA TAHUN 2025**

# ABSTRAK

Keamanan data menjadi tantangan besar di era Revolusi Industri 4.0, di mana informasi penting perlu dilindungi dari akses tidak sah. Ancaman serangan siber dan peretasan mengharuskan pengembangan solusi keamanan yang lebih efektif. Penelitian ini bertujuan untuk mengimplementasikan dan mengevaluasi kombinasi algoritma ChaCha20 dan modifikasi *Advanced Encryption* Standard (AES) dengan penambahan langkah *AddRoundKey* sebelum *MixColumns* untuk meningkatkan keamanan *file*. Metode yang digunakan adalah enkripsi ganda, dengan AES modifikasi sebagai *block cipher* yang kuat dan ChaCha20 sebagai *stream cipher* yang cepat. Penelitian ini mencakup implementasi algoritma dengan menggunakan pustaka kriptografi untuk menguji kinerja algoritma melalui parameter seperti waktu enkripsi, waktu *dekripsi*, kecepatan enkripsi, kecepatan *dekripsi*, dan tingkat keacakan data menggunakan analisis entropi. *File* dengan berbagai format dan ukuran, PDF, Word, dan TXT, digunakan sebagai data pengujian untuk mengevaluasi efisiensi algoritma. Selain itu, antarmuka pengguna berbasis web dirancang untuk mempermudah proses enkripsi dan *dekripsi*, melibatkan langkah- langkah seperti pemilihan *file* dan *input* kunci enkripsi. Penelitian ini diharapkan menunjukkan bahwa kombinasi AES modifikasi dan ChaCha20 mampu meningkatkan keamanan data serta memberikan efisiensi yang lebih baik dalam perlindungan *file*.

**Kata Kunci**: Keamanan *File*, *Advanced Encryption Standard* (AES), ChaCha20, Enkripsi Ganda, *AddRoundkey*, *Entropi*.

# ABSTRACT

*Data security has become a major challenge in the era of the Industrial Revolution 4.0, where critical information must be protected from unauthorized access. Cyberattacks and hacking threats demand the development of more effective security solutions. This research aims to implement and evaluate the combination of the ChaCha20 algorithm and a modified Advanced Encryption Standard (AES) by adding an AddRoundKey step before MixColumns to enhance file security. The method used is double encryption, with the modified AES as a robust block cipher and ChaCha20 as a fast stream cipher. This study involves implementing the algorithms using cryptographic libraries to test their performance through parameters such as encryption time, decryption time, encryption speed, decryption speed, and data randomness level using entropy analysis. Files of various formats and sizes, including PDF, Word, and TXT, are utilized as test data to evaluate the efficiency of the algorithms. Additionally, a web-based user interface is designed to simplify the encryption and decryption process, involving steps such as file selection and encryption key input. This research is expected to demonstrate that the combination of modified AES and ChaCha20 can enhance data security and provide better efficiency in file protection.*

***Keywords****: File Security, Advanced Encryption Standard (AES), ChaCha20, Double Encryption, AddRoundkey, Entropy*

# KATA PENGANTAR

Assalamu’alaikum Warrahmatullahi Wabarakatuh, puja puji syukur ke hadirat Allah SWT yang telah memberikan nikmat dan karunia-Nya yang sangat luar biasa kepada hamba – hambanya. Tidak lupa shalawat serta salam kita curahkan atas bimbingan nabi kita nabi Muhammad SAW yang telah membawa cahaya islam ke dalam dunia ini dan menjadi panutan bagi setiap umatnya hingga akhir zaman.

Setelah melalui berbagai proses kerja keras, penyusunan skripsi yang berjudul “Implementasi Kombinasi ChaCha20 dan Modifikasi AES melalui *AddRoundkey* Pada Studi Kasus Keamanan *File*” akhirnya dapat terselesaikan. Selain sebagai salah satu syarat memperoleh gelar S-Kom, penulis berharap juga skripsi ini dapat bermanfaat bagi semua pihak yang berkepentingan. Oleh karena itu, penulis ingin mengungkapkan rasa terima kasih kepada pihak-pihak yang telah memberikan dukungan dan doa sehingga penulis dapat menyelesaikan skripsi ini tepat pada waktunya. Beberapa pihak tersebut antara lain:

1. **Kepada Mama** yang sangat penulis cintai yang tidak pernah lupa selalu memberikan dukungannya dan doa agar diberikan kemudahan dan kelancaran dalam penyusunan skripsi ini sehingga penulis dapat menyelesaikan studi Informatika di Universitas IT Telkom Surabaya.
2. **Kepada Alm.Ibu Suhartatik**, yang penulis cintai dan sayangi, yang telah memberikan motivasi untuk saya.
3. **Kepada Saudara Kembar**, yang selalu mendukung, terima kasih atas bantuan dan waktunya dalam memberikan masukan terkait susunan skripsi ini.
4. **Keluarga Besar penulis** yang telah memberikan motivasi dan kepercayaan besar sehingga penulis bisa mewujudkan impian untuk meraih gelar S-Kom.
5. **Dosen Pembimbing**, Pak Rizky Fenaldo Maulana, S.Kom., M.Kom. dan Tanzila Mustaqim, S.Kom., M.Kom. Yang telah memberikan solusi, waktu, dan bimbingan penuh kesabaran selama proses penyusunan skripsi ini.
6. **Teman-teman dekat**, termasuk Jus’an, Erica Dyah, Qori Emalia, Ardina Firdatana, dan Elma Saskia yang selalu memberikan semangat, mendengarkan keluh kesah, serta menemani dalam berbagai kondisi selama penulisan skripsi.
7. **Semua pihak lainnya**, yang tidak dapat disebutkan satu per satu, atas bantuan dan dukungannya selama proses ini.
8. **Diri penulis sendiri**, atas perjuangan dan kerja keras hingga akhirnya skripsi ini selesai tepat waktu.

Penulis menyadari bahwa skripsi ini masih jauh dari kesempurnaan. Oleh karena itu, saran dan kritik yang membangun sangat diharapkan untuk memperbaiki skripsi ini ke depannya.

Terima kasih.

### Wassalamu’alaikum Warrahmatullahi Wabarakatuh.

Surabaya, 24 Desember 2024 Penulis

Emilia Ramona

# DAFTAR ISI

[LEMBAR PERSETUJUAN ii](#_bookmark0)

[ABSTRAK iii](#_bookmark1)

[ABSTRACT iv](#_bookmark2)

[KATA PENGANTAR v](#_bookmark3)

[DAFTAR ISI vii](#_bookmark4)

[DAFTAR TABEL ix](#_bookmark5)

[DAFTAR GAMBAR x](#_bookmark6)

[DAFTAR RUMUS xi](#_bookmark7)

[BAB I PENDAHULUAN 1](#_bookmark8)

* 1. [Latar Belakang 1](#_bookmark9)
  2. [Rumusan Masalah 4](#_bookmark10)
  3. [Tujuan Penelitian 4](#_bookmark11)
  4. [Batasan dan Asumsi Penelitian 5](#_bookmark12)
  5. [Manfaat Penelitian 5](#_bookmark13)
     1. [Manfaat Akademis 5](#_bookmark14)
     2. [Manfaat Ilmiah 5](#_bookmark15)
  6. [Sistematika Penulisan 6](#_bookmark16)

[BAB II LANDASAN TEORI 7](#_bookmark17)

* 1. [Literatur Terkait Teori 7](#_bookmark18)
  2. [Kajian Teori 10](#_bookmark20)
     1. [Kriptografi 10](#_bookmark21)
     2. [AES (Advanced Encryption Standard) 12](#_bookmark24)
     3. [Chacha20 20](#_bookmark32)
     4. [Entropi 22](#_bookmark35)

[BAB III METODOLOGI PENELITIAN 24](#_bookmark37)

* 1. [Sistematika Penyelesaian Masalah 24](#_bookmark38)
  2. [Alat dan Bahan 24](#_bookmark39)
     1. [Alat 24](#_bookmark40)
     2. [Bahan 25](#_bookmark41)
  3. [Rancangan Penelitian 25](#_bookmark42)
     1. [Studi Literatur 27](#_bookmark45)
     2. [Pengumpulan Data 27](#_bookmark46)
     3. [Implementasi Modifikasi AES 28](#_bookmark47)
     4. [Implementasi Kombinasi AES Modifikasi dan ChaCha20 31](#_bookmark49)
     5. [Implementasi Antarmuka Pengguna (UI) 36](#_bookmark52)
     6. [Pengujian Algoritma 38](#_bookmark54)
     7. [Analisis Hasil 42](#_bookmark59)

[DAFTAR PUSTAKA 44](#_bookmark61)

# DAFTAR TABEL

[Tabel II.1 Literatur Penelitian Terkait Kriptografi 7](#_bookmark19)

[Tabel II.2 Panjang Kunci 13](#_bookmark25)

[Tabel II.3 Tabel Substitusi (S-Box) 16](#_bookmark28)

[Tabel II.4 Transformasi ShiftRows 17](#_bookmark29)

[Tabel III.1 Jenis, Ukuran, dan Isi File untuk Pengujian 40](#_bookmark58)

[Tabel III.2 Evaluasi Hasil 43](#_bookmark60)

# DAFTAR GAMBAR

[Gambar II.1 Alur Proses Enkripsi Dan Dekripsi 11](#_bookmark22)

[Gambar II.2 Proses Kunci Simetris 11](#_bookmark23)

[Gambar II.3.Transformasi ShiftRows 15](#_bookmark27)

[Gambar II.4 Proses AES 19](#_bookmark31)

[Gambar III.1 Diagram Alur Penelitian 26](#_bookmark43)

[Gambar III.2 Diagram Proses Enkripsi AES 28](#_bookmark48)

[Gambar III.3 Diagram Proses Double Encrypt 32](#_bookmark50)

[Gambar III.4 Proses Dekripsi 36](#_bookmark51)

[Gambar III.5 Desain Antarmuka Pengguna Enkripsi dan Dekripsi 37](#_bookmark53)

# DAFTAR RUMUS

[(II-1) Rumus AddRoundKey 14](#_bookmark26)

[(II-2) Operasi Perkalian MixColumns 17](#_bookmark30)

[(II-3) Rumus Keystream 22](#_bookmark33)

[(II-4) Rumus Proses Enkripsi XOR pada Keystream 22](#_bookmark34)

[(II-5) Entropi 22](#_bookmark36)

[(III-1) Pengujian Waktu Enkripsi 39](#_bookmark55)

[(III-2) Pengujian Waktu Dekripsi 39](#_bookmark56)

[(III-3) Pengujian Waktu Kecepatan 39](#_bookmark57)

# BAB I PENDAHULUAN

### Latar Belakang

Semakin meningkatnya teknologi yang semakin canggih membawa sebuah perubahan yang cukup signifikan. Perubahan tersebut dikenal dengan istilah Revolusi Industri 4.0 yang merupakan sebuah kemajuan teknologi yang sering digunakan oleh banyak orang. Revolusi industri 4.0 adalah sebuah data yang digerakkan dengan adanya campur tangan komputer menggunakan metode machine learning atau AI (Alimuddin et al., 2023). Menurut Klaus Schawab revolusi Industri 4.0 memiliki tantangan yang harus dihadapi salah satunya seperti keamanan *cyber* dan privasi data.

Maka dari itu, dengan semakin pesatnya informasi teknologi maka perlunya untuk mengamankan data atau *file*. Keamanan data atau *file* merupakan salah satu langkah yang mudah untuk melindungi sebuah informasi bersifat penting yang berasal dari akses tidak sah. Keamanan *file* ini sangat diperlukan pada era Revolusi Industri 4.0, karena maraknya serangan *cyber* dan peretasan komputer yang bisa mengganggu keberlangsungan suatu industri. Menurut (Mulyana*, 2016*), keamanan *file* merupakan salah satu aksi pencegahan dari sebuah penganggu pengguna *file* atau pengguna yang memakai jaringan yang tidak berkewajiban memiliki akses (Soesanto et al., 2023). Dengan semakin pesatnya dunia teknologi ini membuat keamanan *file* menjadi sangat penting karena untuk melindungi privasi dan informasi dari pihak yang tidak memiliki akses.

Keamanan *file* adalah serangkaian metode atau teknik yang untuk melindungi *file* dari akses yang tidak sah, seperti pencurian atau kerusakan. Tujuannya untuk menjaga informasi dalam *file* tetap rahasia, aman, dan hanya dapat di akses oleh pihak yang berwenang. Oleh karena itu, upaya untuk memastikan agar *file* terjaga dengan menggunakan metode ChaCha20 dan *Advanced Encryption Standard* (AES). ChaCha20 merupakan metode enkripsi yang cepat dan aman. ChaCha20 dipopulerkan oleh Daniel J.

Bernstein sebagai varian dari algoritma. Dalam algoritma ChaCha20 sangat populer dalam pengimplementasi enkripsi dikarenakan percampuran antara kecepatan dan keamanannya yang dapat diunggulkan. Chaca20 mempunyai desain yang hampir sama dengan Salsa20, pada algoritma ChaCha20 menggunakan kunci 256-bit yang mempunyai 8 putaran (Kebande, 2023).

Sedangkan AES (*Advanced Encryption Standard)* adalah sebuah algoritma yang melakukan transformasi terhadap blok data yang menggunakan kunci enkripsi yang sama. AES bekerja pada sistem yang melibatkan operasi matematika dengan level tingkat sulit (Saripa, 2023). Proses kerja enkripsi dasar AES adalah *subbytes* (menggunakan tabel S-Box untuk menggabungkan *byte* pada blok matriks), *shift row* (digunakan untuk menggeser baris matriks), *MixColumns* (melakukan pergeseran dari kanan ke kiri), dan *AddRoundkey* (blok di XOR kan dengan kunci yang diperluas) (Aryanto et al., 2023). Pada proses *dekripsi* AES terdapat tahap-tahapan kebalikan dari proses enkripsi. Dengan memakai kunci yang sama untuk *dekripsi*, blok data yang telah terenkripsi dapat dipulihkan kembali ke bentuk awalnya. algoritma AES adalah jenis *block chipper* yang memiliki panjang kunci 128 bit, 192 bit dan 256 bit (Surbakti & Purwanto, 2024).

Algoritma Chacha20 dan AES merupakan suatu enkripsi yang dapat di digunakan dalam menjaga keamanan *file*. Pada algoritma AES terdapat kelebihan salah satunya adalah sifat karakteristiknya yang di mana selalu ada bidang khusus untuk bilangan prima, memicu kekuatan AES(Surbakti & Purwanto, 2024) . AES juga memiliki sebuah kelemahan salah satunya yaitu algoritma kriptografi yang menggunakan kunci simetris dalam proses pengiriman dan penerimanya data. Namun, penggunaan kunci simetris ini memiliki risiko lebih tinggi terhadap kebocoran, terutama seiring waktu (Hidayatulloh et al., 2023). Pada algoritma ChaCha20 terdapat keunggulan yaitu menggunakan teknik kriptografi ARX (*addition, rotation*, XOR) yang memberikan performa lebih cepat, kompleksitas lebih rendah, dan mampu menghilangkan serangan waktu Klik atau ketuk di sini untuk memasukkan teks.

Meskipun algoritma AES dan Chacha20 telah efektif untuk menjaga peningkatan keamanan *file* dan implementasinya yang cukup rumit sehingga membuat orang yang tidak memiliki akses tidak dapat mengakses *file* tersebut. Hal ini tidak dapat kita pungkiri bahwa semakin pesatnya dunia teknologi membuat banyak orang berbondong-bondong menciptakan pembaruan yang lebih efektif. Metode *double encrypt* ini menawarkan kombinasi yang mampu mengurangi kelemahan dari setiap algoritma jika digunakan secara bersamaan, serta meningkatkan ketahanan terhadap berbagai jenis serangan kriptografi. *Double encrypt* merupakan Teknik kriptografi yang melibatkan proses enkripsi secara dua kali, yang di mana memiliki dua kunci yang berbeda untuk lebih meningkatkan keamanan. *Double encrypt* ini bertujuan untuk meningkatkan keamanan *file* lebih tinggi dibandingkan hanya 1 metode saja (Pangestu et al., 2022).

Pada penelitian terlebih dahulu yang berjudul “Implementasi Sistem Keamanan *File* Menggunakan Algoritma AES untuk Mengamankan *File* Pribadi” yang diteliti oleh Saripa Universitas Negeri Makassar. Dalam penelitian tersebut menjelaskan tentang keamanan *file* dengan menggunakan metode algoritma AES. Di mana hasil penelitian ini menunjukkan bahwa algoritma AES telah berhasil mengamankan *file* yaitu dokumen Word, PDF, Excel, dan Power Poin, *file* yang dienkripsi menjadi tidak terbaca oleh pihak yang tidak memiliki akses sehingga dapat menjaga kerahasiaan data dan AES mampu menjaga keamanan data dengan baik.

Dalam perbedaan penelitian terdahulu dan penelitian saya akan menggunakan double encrypt untuk keamanan *file* dan juga saya akan menggunakan AES dengan menambahkan metode ChaCha20 dan melakukan modifikasi pada bagian AES. Modifikasi menggunakan penambahan *AddRoundkey* sebelum dengan bertujuan agar proses enkripsi akan lebih susah untuk dipecahkan. Dalam algoritma AES, langkah *AddRoundKey* merupakan komponen krusial yang menggabungkan data dengan kunci enkripsi menggunakan operasi XOR. Penelitian (Luong, 2023) oleh menunjukkan bahwa memodifikasi transformasi *AddRoundKey*

dapat meningkatkan keamanan algoritma dengan menciptakan hubungan yang lebih kompleks antara *plainteks* dan *cipherteks*. Penelitian ini menyoroti pentingnya penyesuaian langkah *AddRoundKey* untuk meningkatkan ketahanan algoritma AES terhadap serangan kriptografi, seperti *differential cryptanalysis* dan linear *cryptanalysis*.

Dengan mendasarkan pada penelitian tersebut, modifikasi algoritma AES melalui penambahan langkah *AddRoundKey* sebelum *MixColumns* bertujuan untuk lebih memperkuat difusi data pada tahap awal enkripsi. Penambahan ini diharapkan dapat meningkatkan kerandoman hasil enkripsi dan mengurangi kemungkinan eksploitasi pola oleh pihak yang tidak berwenang, sehingga keamanan algoritma menjadi lebih optimal. Berdasarkan dari latar belakang di atas penelitian ini memiliki tujuan untuk meningkatkan keamanan *file* agar lebih sulit untuk diretas atau diakses pihak yang tidak berwenang. Oleh karena itu penelitian ini dianggap penting karena maraknya kasus peretasan data oleh oknum yang tidak memiliki akses.

### Rumusan Masalah

1. Bagaimana implementasi dari modifikasi AES dengan penambahan *AddRoundkey* sebelum *MixColumns* dalam meningkatkan keamanan *file*?
2. Bagaimana performa kombinasi antara AES modifikasi dan ChaCha20 dalam hal waktu enrkipsi, waktu *dekripsi* , kecepatan enkripsi, dan kecepatan *dekripsi*?

### Tujuan Penelitian

1. Mampu mengidentifikasi *AddRoundkey* yang akan digunakan dalam menjaga keamanan *file* dan mengimplementasi kemampuan Chacha20 dan AES yang telah dimodifikasi dalam proses menjaga keamanan *file*.
2. Mampu mengevaluasi performa algoritma ChaCha20 dan AES yang telah dimodifikasi berdasarkan waktu enkripsi, waktu *dekripsi*, kecepatan enkripsi, dan kecepatan *dekripsi*, untuk menentukan efisiensi

dan efektivitas kombinasi kedua algoritma tersebut dalam menjaga keamanan *file*

### Batasan dan Asumsi Penelitian

1. Penelitian ini berfokus pada jurnal atau artikel ilmiah dengan batasan tahun 2021-2024 yang membahas tentang :
   * Keamanan *file*
   * Algoritma ChaCha20
   * *Advanced Encryption Standard* (AES)
   * Double encryption
2. Penelitian dilakukan dengan pengujian pada *file* dengan ukuran dan format sebagai berikut :
   * Format PDF dan Word dengan variasi ukuran *file* 100 KB - 100MB.
   * Format *file* teks (TXT) dengan ukuran maksimal 300 byte Batasan ukuran *file* digunakan untuk memastikan pengujian dilakukan secara efektif tanpa mengurangi kinerja proses enkripsi dan *dekripsi*.

### Manfaat Penelitian

### Manfaat Akademis

* + - 1. Memberikan inovasi terbaru mengenai pelindungan keamanan

*file* dengan menambahkan *AddRoundkey* pada algoritma AES.

* + - 1. Memudahkan peneliti lainnya dalam kinerja *AddRoundkey* pada algoritma AES yang berfungsi untuk keamanan *file***.**

### Manfaat Ilmiah

* + - 1. Menjadi sumber acuan dan sumber referensi bagi peneliti lainnya yang akan meneliti topik ChaCha20 dan AES modifikasi, terutama dalam konteks keamanan data dan performa enkripsi.
      2. Membuktikan hipotesis apakah AES yang dimodifikasi dengan tambahan *AddRoundkey* dapat meningkatkan performa keamanan dan efisiensi dibandingkan dengan AES standart, dan sekaligus menunjukkan kinerja kombinasi ChaCha20 dan AES dalam pengamanan *file* berbagai format dan ukuran.

### Sistematika Penulisan

## BAB 1 PENDAHULUAN

Menjelaskan latar belakang penelitian, rumusan masalah, tujuan penelitian, batasam penelitian, manfaat penelitian, serta sistematika penulisan secara umum.

## BAB 2 LANDASAN TEORI

Membahas teori-teori yang relavan dengan penelitian, termasuk algoritma ChaCha20, AES, konsep kriptografi, Serta tinjauan terhadap penelitian terdahulu yang terkait.

## BAB 3 METODOLOGI PENELITIAN

Menjelaskan metode yang digunakan dalam penelitian, termasuk algoritma yang diterapkan (modifikasi AES dan ChaCha20 dengan enkripsi ganda). Diuraikan juga alat dan bahan yang digunakan, serta tahapan penelitian yang meliputi implementasi algoritma, pengujian parameter kinerja (waktu dan kecepatan enkripsi, dekripsi, serta analisis entropi), dan evaluasi hasil pengujian pada berbagai jenis file (PDF, Word, TXT).

## BAB 4 IMPLEMENTASI DAN PENGUJIAN

Menyajikan hasil implementasi algoritma yang dikembangkan, pengujian algoritma terhadap berbagai jenis *file*, serta analisis data hasil pengujian**.**

## BAB 5 EVALUASI HASIL PENGUJIAN

Menyajikan hasil implementasi dari aplikasi yang telah dirancang, pengujian fungsionalitas sistem, serta evaluasi kinerja aplikasi terhadap berbagai skenario penggunaan. Analisis data hasil pengujian juga dilakukan untuk menilai keberhasilan aplikasi dalam mencapai tujuan penelitian.

## BAB 6 KESIMPULAN DAN SARAN

Merangkum temuan utama penelitian serta memberikan saran untuk pengembangan lebih lanjut di masa depan.

# BAB II LANDASAN TEORI

### Literatur Terkait Teori

Keamanan informasi menjadi salah satu isu yang paling mendesak di era digital saat ini, di mana data sensitif sering kali menjadi target serangan siber yang beragam. Dengan meningkatnya ancaman seperti serangan *man-in-the- middle*, *eavesdropping*, dan *DDoS*, penting bagi organisasi dan individu untuk menerapkan metode enkripsi yang efektif untuk melindungi data mereka. Berbagai algoritma enkripsi telah dikembangkan untuk memenuhi kebutuhan ini, dengan *Advanced Encryption Standard* (AES) dan ChaCha20 menjadi dua di antara yang paling banyak digunakan. Penelitian-penelitian terkini menunjukkan bahwa kedua algoritma ini tidak hanya menawarkan tingkat keamanan yang tinggi, tetapi juga dapat diimplementasikan dalam berbagai konteks, mulai dari pengamanan *file* hingga enkripsi konten *livestreaming*.

Pada [Tabel II.1 Literatur Penelitian Terkait Kriptografi](#_bookmark19) menyajikan ringkasan dari beberapa penelitian yang mengeksplorasi efektivitas dan penerapan algoritma AES dan ChaCha20 dalam berbagai aplikasi. Setiap penelitian memberikan wawasan tentang tujuan, hasil, dan persamaan dalam pendekatan yang digunakan, serta menyoroti inovasi yang dilakukan untuk meningkatkan keamanan data. Dengan memahami hasil-hasil ini, kita dapat lebih menghargai perkembangan teknologi enkripsi dan aplikasinya dalam menjaga kerahasiaan serta integritas informasi di dunia yang semakin terhubung.

### Kajian Teori

### Kriptografi

Menurut Menezes, Oorschot, dan Vantone (1996), Kriptografi adalah studi teknik matematika yang berkaitan dengan keamanan informasi seperti kerahasiaan, *autentikasi* keaslian data (Belvin Tannadi, 2022). Kriptografi dapat diartikan sebagai sebuah proses untuk melindungi data atau sebuah sistem yang bertujuan untuk melindungi informasi rahasia dengan cara yang tidak dipahami oleh pihak yang tidak berwenang. Dalam kriptografi yang meliputi proses enkripsi dan proses *dekripsi*. Di dalam Kriptografi terdapat tiga fungsi dasar yang meliputi :

* + - 1. Enkripsi merupakan proses data asli (*plainteks*) yang akan diubah menjadi data yang tidak bisa terbaca oleh pihak yang tidak berwenang (*chiperteks*).
      2. *Dekripsi* merupakan proses pengembalian dari data yang tidak bisa terbaca(*chiperteks*) menjadi data terbaca atau data asli (*plainteks*).
      3. Kunci merupakan proses yang dipakai untuk melakukan proses enkripsi dan *dekripsi*, terdapat 2 jenis kunci, yaitu kunci rahasia (*private key*) dan kunci publik (*public key*).

Pada [Gambar II.1 Alur Proses Enkripsi Dan *Dekripsi*](#_bookmark22)

memperlihatkan alur enkripsi dan *dekripsi.* Proses ini dimulai

dengan *plainteks* yang kemudian dienkripsi dengan menggunakan kunci menjadi *chiperteks*. *Chiperteks* tersebut kemudian *didekripsi* kembali menggunakan kunci yang sama untuk mendapatkan kembali *plainteks*.

*Gambar II.1 Alur Proses Enkripsi Dan Dekripsi*

Berikut algoritma Kriptografi yang berdasarkan kunci :

a. Algoritma Simetris

Algoritma simetris adalah algoritma yang merujuk pada metode enkripsi dan *dekripsi* yang di mana algoritma ini memiliki kunci tunggal atau sama. Sehingga dapat disimpulkan algoritma ini mewajibkan pengirim dan penerima harus menyetujui satu kunci yang digunakan dalam proses enkripsi dan *dekripsi* data. Namun, kelemahan dari algoritma ini yaitu tergantung pada rahasia kunci. Apabila kunci tersebut tersebar maka akan dengan mudah pihak yang tidak memiliki akses dapat melakukan proses enkripsi dan *dekripsi* data secara mudah.

*Gambar II.2 Proses Kunci Simetris*

Pada [Gambar II.2 Proses Kunci Simetris](#_bookmark23) menjelaskan proses enkripsi dan *dekripsi* menggunakan kunci simetris. Dimulai dengan pesan yang dikirimkan, kemudian dienkripsi menjadi kode yang tidak bisa dibaca menggunakan kunci simetris. Kode tersebut kemudian dikirim dan di *dekripsi* di sisi penerima menggunakan kunci simetris yang sama, sehingga pesan dapat dibaca kembali. Pada contoh gambar, pesan "Selamat Datang!" dienkripsi menjadi kode "9CDF4837 AB672019" menggunakan kunci simetris yang sama. Kemudian di *dekripsi* kembali ke pesan awal "Selamat Datang!" Kunci simetris merupakan kunci yang sama digunakan untuk enkripsi dan *dekripsi*, sehingga pengirim dan penerima pesan harus memiliki kunci yang sama agar dapat berkomunikasi.

### AES (Advanced Encryption Standard)

Advanced Encryption Standard (AES) adalah salah satu algoritma enkripsi yang paling terkenal dan banyak digunakan, serta terus dikembangkan sebagai pengganti algoritma Data Encryption Standard (DES). AES adalah algoritma enkripsi yang bekerja dengan memproses data dalam blok dengan ukuran minimal 128 bit, dan melalui beberapa tahapan untuk mengenkripsi data. Tujuan utama dari AES adalah untuk mengamankan data rahasia sehingga tidak dapat dibaca oleh pihak yang tidak berwenang (Wahyu Putra, 2023). Pada [Tabel II.2 Panjang Kunci](#_bookmark25) dalam proses enkripsi *Advanced Encryption Standard* (AES) menggunakan ukuran blok dengan panjang sebesar 126 bit, dengan pilihan panjang kunci yang bervariasi, yaitu 128 bit, 192 bit, dan 256 bit (Sekar Putri Ananda et al., 2022). Proses enkripsi dalam algoritma ini dilakukan melalui beberapa tahap yang dikenal sebagai ronde. Jumlah ronde yang digunakan bergantung pada panjang kunci yang akan dipilih. Berikut adalah hubungan panjang kunci dan banyaknya ronde.

Algoritma AES juga menawarkan tingkat keamanan yang sangat tinggi dikarenakan ukuran kunci dan jumlah ronde yang ditentukan. Ukuran kunci menentukan jumlah kemungkinan kombinasi kunci, sehingga membuat serangan *bruteforce* menjadi sangat sulit dilakukan. Di sisi lain, jumlah ronde yang besar dapat menghasilkan keamanan yang kuat, sehingga perubahan kecil pada *plainteks* dapat memengaruhi seluruh bagian *cipherteks*. Selain keunggulan keamanannya, AES juga memiliki efisiensi tinggi dalam hal waktu dan ruang. Algoritma ini dapat diimplementasikan baik pada perangkat keras maupun perangkat lunak, menjadikannya ideal untuk berbagai aplikasi, termasuk perlindungan data (Assyahid, 2024). Pada tahap awal enkripsi, data masukan yang telah disalin ke dalam matriks *state* akan menjalani transformasi *AddRoundkey*. Selanjutnya, matriks *state* akan melalui proses transformasi secara berulang, mencakup *SubBytes*, *ShiftRows*, *MixColumns*, dan *AddRoundkey*, dengan jumlah pengulangan sesuai dengan nilai Nr (Handoyo & Subakti, 2020). *Dekripsi* adalah proses mengembalikan *ciphertext* menjadi plaintext, atau mengubah data terenkripsi kembali ke bentuk aslinya. Pada proses *inverse cipher*, transformasi yang digunakan meliputi *InvShiftRows*, *InvSubBytes*, *InvMixColumns*, dan *AddRoundkey*. Berikut ini adalah cara kerja algoritma AES secara umum:

### Proses Enkripsi AES Modifikasi

* + - * 1. *AddRoundkey*

*AddRoundkey* adalah proses menggabungkan state masukan dengan kunci ronde dengan menggunakan operasi XOR. Dalam proses transformasi enkripsi dan *dekripsi* pada AES, dilakukan operasi penambahan kunci putaran (*round key*)

pada state menggunakan operasi XOR (Djong & Siswanto, 2022). Setiap kunci putaran terdiri dari *Nb word* di mana setiap *word* tersebut akan digabungkan dengan *word* atau kolom yang sesuai pada *state* . Secara matematis, hal ini (Putri et al., 2021).

Keterangan Algoritma dari *AddRoundkey* di mana :

Pada [Gambar II.3.Transformasi ShiftRows](#_bookmark27) di atas di mana S0,0 merupakan elemen di baris ke-0, kolom ke-0, S3,3 merupakan elemen di baris ke-3, kolom ke-3, dan 𝑊3,3 merupakan elemen di baris ke-3, kolom ke-3. Pada label S0,0 dan lainnya merupakan state saat ini, state ini representasi internal data yang sedang dienkripsi ataupun *didekripsi* dan setiap elemen adalah *byte* dari blok data input. Sedangkan pada label 𝑊0,0 merupakan kunci ronde yang dihasilkan dari proses *keyexpansion*, di mana ini bagian dari kunci utama yang dipecah untuk menjadi beberapa ronde dan setiap elemen adalah *byte* dari kunci untuk ronde tertentu. Operasi XOR dilakukan antara setiap *byte* pada *state* dengan *byte* kunci ronde yang sesuai. Hasil di atas dihasilkan dari operasi XOR yang akan digunakan untuk langkah transformasi berikutnya.

* + - * 1. *SubBytes*

Transformasi *transformasi byte* di mana setiap elemen dalam matriks *state* dipetakan menggunakan tabel substitusi yang dikenal sebagai S-Box. Setiap *byte* dalam matriks *state* direpresentasikan sebagai S’[r, c], yang merupakan elemen hasil dari perpotongan antara baris (x) dan kolom (y) (Kurnia Mahesa Naibaho et al., 2021). S-Box berupa matriks

berukuran 16×16 yang berisi nilai-nilai heksadesimal di mana nilai-nilai ini diperoleh melalui perhitungan tertentu (Hena & Kusumaningsih, 2024). Berikut ini tabel S-Box :

Pada [Tabel II.3 Tabel Substitusi (S-Box)](#_bookmark28) sebagai contoh, nilai heksadesimal {62} merujuk ke baris 7 dan kolom 3 dalam S-Box, yang berisi nilai {aa}. Dengan demikian, nilai

{62} dipetakan menjadi nilai {aa}.

* + - * 1. *ShiftRows*

Transformasi *ShiftRows* merupakan proses pergeseran *byte* di mana *byte* paling kiri dipindahkan ke posisi paling kanan (rotasi bit). Jumlah pergeseran dalam melakukan *ShiftRows* ini dilakukan perbedaan setiap baris dalam matriks *state*. Pada baris pertama, tidak ada pergeseran yang dilakukan. Kemudian pada baris kedua barulah mengalami pergeseran satu *byte* ke kiri, lalu baris ketiga digeser dua *byte* ke kiri, dan baris keempat digeser tiga *byte* ke kiri. Tujuan dari proses ini adalah untuk menciptakan *diffusion*(), yang memiliki tujuan untuk membantu menyebarkan efek transformasi non-linear ke seluruh baris dalam matriks *state* untuk proses selanjutnya (Cristy & Riandari, 2021). Proses pergeseran ini bertujuan untuk mencampur elemen-elemen dalam *state* sehingga memperkuat kerahasiaan data.

*MixColumns* adalah proses transformasi di mana setiap elemen dalam blok *cipher* dikalikan dengan matriks . Transformasi ini bertujuan untuk mencampur elemen- elemen dalam setiap kolom *state* berguna meningkatkan difusi data dan memperkuat keamanan enkripsi . Transformasi ini bekerja dengan membagi matriks atau *state* menjadi empat bagian, untuk setiap kolomnya (Widyawan & Imelda, 2021). Setiap kolom dalam matriks *state* diperlakukan sebagai *polinomial* dengan empat bagian dalam *Galois Field* (GF), pada persamaan [(II-2)](#_bookmark30) dilakukan Algoritma dari *AddRoundkey* di mana :

𝑋8 = Elemen dengan derajat ke-8 yang akan muncul sebagai 100000000 (1 diposisi ke 9 dari kanan) di bilangan biner.

X = Menunjukkan elemen dengan derajat ke 1 yang akan muncul sebagai 00000010 ( 1 di posisi ke 2 dari kanan

) dalam bilangan biner

1 = Elemen konstanta yang akan muncul sebagai

00000001 ( 1 diposisi ke 1 dari kanan), dimana elemen 1 ini menyatakan bahwa *polinomial* ini memiliki koefisien konstan.

Proses *MixColumns* juga dapat direpresentasikan sebagai operasi perkalian matriks, di mana setiap kolom matriks dikalikan dengan matriks tertentu dalam *Galois Field* (Cristy & Riandari, 2021).

* + - 1. **Proses *dekripsi* AES :**

Proses *dekripsi* merupakan kebalikan dari proses enkripsi. *Transformasi bytes* yang digunakan dalam *dekripsi* mencakup *AddRoundKey*, *InvShiftRows*, *InvSubBytes,* dan *InvMixColumns*. Pada iterasi pertama *dekripsi*, dilakukan *transformasi AddRoundKey, Inverse ShiftRows, dan Inverse SubBytes. Ciphertex*t kemudian akan mengalami transformasi *AddRoundKey* .

* + - * 1. *InvShiftRows*

Transformasi *byte* yang merupakan kebalikan dari proses *ShiftRows*. Dalam transformasi ini, bit-bit pada *state* digeser ke arah kanan, berbeda dengan *ShiftRows* yang melakukan pergeseran bit ke arah kiri.

* + - * 1. *InvSubBytes*

Transformasi *byte* yang menjadi kebalikan dari *SubBytes*. Dalam proses ini, setiap elemen pada *state* dipetakan menggunakan tabel substitusi *Inverse* S-Box.

* + - * 1. *InvMixColumns*

Setiap kolom dalam *state* dikalikan dengan matriks khusus yang digunakan dalam proses perkalian pada algoritma AES (Syahputra Sianipar et al., 2024)

*Gambar II.4 Proses AES*

Pada [Gambar II.4 Proses AES](#_bookmark31) sebelah kiri menunjukkan tahapan proses enkripsi pada algoritma AES Modifikasi. Proses dimulai dengan memasukkan plainteks, kemudian melalui serangkaian langkah seperti *AddRoundkey*, SubBytes, ShiftRows, MixColumns, dan diakhiri dengan *AddRoundkey* tambahan. Langkah ini diulang sebanyak Nr−1 ronde untuk meningkatkan keamanan data, menghasilkan cipherteks di akhir proses. Flowchart sebelah kanan menjelaskan tahapan proses *dekripsi*, yang merupakan kebalikan dari proses enkripsi. Proses dimulai dengan memasukkan cipherteks, kemudian melalui langkah-langkah seperti *AddRoundkey*, Inverse ShiftRows, Inverse SubBytes, dan Inverse MixColumns. Setelah semua ronde selesai, data dikembalikan ke bentuk plainteks asli.

### Chacha20

Algoritma ChaCha20 adalah salah satu jenis *stream cipher* yang dikembangkan sebagai varian dari algoritma Salsa20. Algoritma ini berbasis desain yang serupa dengan Salsa20, namun ChaCha20 meningkatkan tingkat *diffusion* pada setiap ronde untuk memperkuat keamanan proses enkripsi (Anindita & 18219086, 2023). Chacha20 beroperasi pada unit data 32-bit menggunakan kunci 256-bit. Algoritma ini menghasilkan *output* berupa *stream* kunci sepanjang 512-bit, yang kemudian akan di XOR kan dengan *plainteks* untuk menghasilkan *chiperteks*(Alyas & Abdullah, 2021b). ChaCha20 dimulai dengan membuat *state* 512-bit, yang berupa *array* yang berukuran 16x32-bit. State ini disusun dari empat konstanta 32 bit, kunci rahasia sepanjang 256-bit, penghitung blok 32-bit, serta nonce Ini menunjukkan struktur *state* ChaCha20 yang terdiri dari matriks 4x4 dengan 16 elemen 32-bit. Matriks ini diinisialisasi dengan empat konstanta 32-bit, kunci 256-bit yang dibagi menjadi 8 bagian, penghitung 32 blok 32-bit, dan Nonce 96-bit. Elemen-elemen ini digunakan untuk menghasilkan *keystream* yang kemudian di XOR kan dengan *plainteks* untuk menghasilkan *chiperteks*. Di mana *const* (konstanta) ini berisi empat nilai tetap 32-bit, *key* (kunci) yang terbagi 8 bagian 32-bit dari kunci rahasia 256-bit, *counter* (penghitung), dan Nonce (nilai acak) yang berukuran 96-bit.

Langkah – Langkah Algoritma ChaCha20 :

* + - 1. *Inisialisasi* State

Algoritma ChaCha20 dimulai dengan membangun sebuah state internal berbentuk matriks 4x4 (32-bit) (Barbero et al., 2022) . Matriks ini terdiri dari:

* + - * 1. Baris pertama: Berisi 4 konstanta tetap (128-bit), yang digunakan untuk mengenali algoritma.
        2. Baris kedua dan ketiga: Berisi kunci rahasia 256-bit (8 elemen 32-bit), yang digunakan untuk proses enkripsi.
        3. Baris keempat: Berisi kombinasi 64-bit counter dan 64-bit nonce (masing-masing terdiri dari 2 elemen 32-bit).
      1. *Quarter Round*

Operasi ini memodifikasi 4 elemen dari matriks ( a, b, c, d ) (Almukhlifi & Vora, 2020)

𝑎 += 𝑏; 𝑑 ^ = 𝑎; 𝑑 <<<= 16;

𝑐 += 𝑑; 𝑏 ^ = 𝑐; 𝑏 <<<= 12;

𝑎 += 𝑏; 𝑏 ^ = 𝑎; 𝑑 <<<= 8;

𝑐 += 𝑑; 𝑏 ^ = 𝑐; 𝑏 <<<= 7;

* + - 1. *Double Round*

*Double Round* adalah kombinasi dari dua *Quarter Round* yang dijalankan secara berurutan. Setiap iterasi *Double Round* mencakup:

* + - * 1. Operasi pada kolom matriks *(Column Round):*

melakukan operasi Quarter Round pada setiap kolom matriks 4x4. Tujuannya adalah untuk mencampur data antar elemen dalam kolom, sehingga meningkatkan difusi data secara vertikal.

* + - * 1. Operasi pada diagonal matriks (Diagonal Round):

Setelah operasi pada kolom selesai, langkah berikutnya adalah menerapkan Quarter Round pada elemen-elemen diagonal dalam matriks.

Proses Double Round dilakukan sebanyak 10 iterasi, menghasilkan total 20 ronde Quarter Round untuk setiap blok data yang diproses.

* + - 1. Finalisasi State
      2. *Keystream Generation*

Setelah 20 ronde selesai*, state* akhir ditambahkan kembali ke

### Entropi

Entropi dalam teori informasi digunakan untuk mengukur keacakan data, yang mencerminkan kekuatan algoritma enkripsi. Semakin tinggi nilai entropi, semakin acak data tersebut, yang berarti algoritma enkripsi lebih tahan terhadap serangan (Wahyudi et al., 2024). Entropi dihitung dengan menggunakan rumus pada

Dimana:

* 𝐻𝑚 : Menunjukkan entropi data, dimana tingkat keacakan dalam

# BAB III METODOLOGI PENELITIAN

### Sistematika Penyelesaian Masalah

Penelitian ini bersifat kuantitatif dengan tujuan mengevaluasi performa algoritma AES modifikasi dan kombinasi ChaCha20 dan AES modifikasi. Penilaian performa ini dilakukan berdasarkan waktu enkripsi, waktu *dekripsi*, kecepatan enkripsi, dan kecepatan *dekripsi*, yang akan dianalisis menggunakan data hasil pengujian pada berbagai jenis *file* dengan format PDF, Word dan TXT, yang hasilnya berbentuk tabel untuk mempermudah analisis perbandingan.

### Alat dan Bahan

### Alat

1. Perangkat Keras (Hardware) Laptop dengan spesifikasi :
   1. Prosesor : Intel *Core* I5
   2. RAM : 8 GB
   3. Penyimpanan : 256 GB SSD atau lebih
2. Perangkat Lunak (*Software*)
   1. *Python* :

Menggunakan *library cryptography, pycryptodome*, dan *library* lain yang mendukung algoritma AES dan ChaCha20. Berikut beberapa pustaka yang digunakan :

* + - PyCryptodome : Digunakan Iplementasi algoritma AES dan ChaCha20
    - Cryptography : Implementasi kemanan tambahan (jika nanti dibutuhkan )
    - Numpy : Digunakan untuk analisis data (perhitungan entropi)
    - Time : Mencatat waktu eksekusi algoritma, baik untuk enkripsi maupun dekripsi.
  1. *Text* Editor :

Visual Studio Code dan *Jupyter Notebook*

* 1. Sistem Operasi : Windows

### Bahan

1. Data set *file* :
   1. *File* PDF : Ukuran *file* 100 KB – 100 MB
   2. *File* Word : Ukuran *file* 100 KB – 100 MB
   3. *File* TXT : Ukuran *file* maksimal 300 *byte*
2. Literatur :

Jurnal atau artikel ilmiah tentang AES, ChaCha20 dan *Double Encrypt* (tahun 2020-2024)

1. *Spreadsheet* Analisis Data :

Format CSV (Comma Separated Values)

### Rancangan Penelitian

Pada penelitian ini, dilakukan tahapan yang dirancang secara sistematis untuk mencapai tujuan utama, yaitu Implementasi kombinasi Advanced Encryption Standard (AES) yang dimodifikasi dan ChaCha20. Proses penelitian dimulai dengan studi literatur untuk memahami konsep dasar dan pengembangan terbaru terkait algoritma kriptografi. Setelah itu, dilakukan pengumpulan data yang relevan sebagai dasar implementasi. Tahapan selanjutnya mencakup implementasi algoritma modifikasi AES serta kombinasi algoritma tersebut dengan ChaCha20. Untuk mengukur efektivitas dan efisiensi algoritma yang dikembangkan, pengujian dilakukan pada berbagai jenis *file* seperti PDF, Word, dan TXT. Analisis hasil dari pengujian ini memberikan dasar untuk menarik kesimpulan dan memberikan rekomendasi untuk pengembangan lebih lanjut.

*Gambar III.1 Diagram Alur Penelitian*

[Gambar III.1 Diagram Alur Penelitian](#_bookmark44) ini terdapat beberapa tahapan yaitu:

1. Mulai adalah tahap awal dari penelitian. Pada tahap ini menentukan tujuan dan menyusun langkah-langkah yang akan dilakukan dalam penelitian.
2. Studi literatur dilakukan pengumpulan referensi yang berkaitan dengan algoritma AES, ChaCha20 dan *Double Encrypt. Referensi* berupa jurnal, artikel ilmiah.
3. Pengumpulan data untuk mengumpulkan *file* atau data set yang akan digunakan untuk pengujian algoritma. Format *file* yang digunakan adalah PDF, Word dan TXT.
4. Implementasi modifikasi AES, pada tahap ini algoritma AES dimodifikasi dengan menambahkan langkah *AddRoundkey* sebelum *Mixcolumns*. Tujuannya adalah untuk meningkatkan keamanan dan kecepatan enkripsi dibandingkan dengan AES standar.
5. Implementasi kombinasi AES modifikasi dan ChaCha20, menggabungkan algoritma AES yang sudah dimodifikasi dengan ChaCha20 untuk melakukan *double encypt*. Di mana proses ini bertujuan untuk meningkatkan keamanan *file* lebih baik daripada menggunakan *single encrypt*.
6. Pengujian algoritma, untuk menguji performa algoritma yang sudah diimplementasikan, di mana data set yang dikumpulkan sebelumnya itu digunakan untuk uji coba. Parameter yang diukur pada pengujian ialah

waktu enkripsi dan *dekripsi,* kecepatan enkripsi dan *dekripsi*, dan hasil pengujian ini akan digunakan untuk mengevaluasi kinerja algoritma.

1. Analisis hasil, data pengujian yang sudah terkumpul akan dianalisis. Analisis dilakukan untuk membandingkan performa AES standar, AES yang sudah dimodifikasi, dan kombinasi AES modifikasi dan ChaCha20. Tujuannya sendiri untuk mengetahui algoritma mana yang lebih efisien.
2. Selanjutnya ke kesimpulan dan saran, pada tahap ini merangkum hasil penelitian termasuk kelebihan dan kekurangan algoritma yang sudah diuji. Saran juga untuk pengembangan penelitian lebih lanjut
3. Selesai

### Studi Literatur

Studi literatur dilakukan untuk memahami konsep dasar dan perkembangan terbaru terkait algoritma kriptografi yang digunakan dalam penelitian ini, yaitu Advanced Encryption Standard (AES) dan ChaCha20. Proses ini melibatkan pengumpulan referensi dari jurnal, paper, dan buku yang membahas metode enkripsi, keamanan data, serta teknik evaluasi seperti entropi dan analisis performa. Literatur yang dikaji juga mencakup penelitian sebelumnya tentang algoritma kriptografi untuk keamanan *file*, sehingga memberikan dasar teori yang kuat untuk modifikasi algoritma dan penerapan metode double encryption dalam penelitian ini.

### Pengumpulan Data

Data yang digunakan dalam penelitian ini dibuat sendiri dengan mengikuti ketentuan yang telah ditentukan. Format data meliputi *file* PDF, Word, dan TXT dengan ukuran bervariasi, mulai dari 100 KB hingga 100 MB untuk *file* PDF dan Word, serta hingga 300 *byte* untuk *file* TXT. Data ini dirancang sedemikian rupa agar sesuai dengan kebutuhan pengujian algoritma enkripsi, seperti mengukur waktu enkripsi, waktu *dekripsi*, kecepatan, dan tingkat keacakan menggunakan analisis entropi. Variasi format dan ukuran data ini

bertujuan untuk memastikan bahwa pengujian mencakup berbagai jenis isi *file*.

### Implementasi Modifikasi AES

Pengembangan algoritma *Advanced Encryption Standard* (AES) dilakukan dengan memodifikasi. Modifikasi dilakukan dengan menambahkan *AddRoundkey* sebelum *Mixcolumns*. Penambahan ini bertujuan untuk memperkuat keamanan dan meningkatkan kompleksitas pada proses enkripsi, sehingga lebih sulit untuk diretas dengan pihak tidak berwenang. Berikut adalah proses alur dari modifikasi algoritma AES.

*Gambar III.2 Diagram Proses Enkripsi AES*

Pada [Gambar III.2 Diagram Proses Enkripsi AES](#_bookmark48) ini digunakan untuk mengamankan *file* dengan melakukan tahap enkripsi terhadap data *plainteks* yang menggunakan kunci tertentu. Dalam penelitian ini, modifikasi dilakukan dengan menambahkan langkah *AddRoundkey* untuk meningkatkan tingkat keamanan dan kompleksitas enkripsi. Berikut langkah-langkah proses modifikasi AES .

1. Input *file* dan kunci

Proses enkripsi dimulai dengan memasukkan *file* input (*file* asli yang akan dienkripsi) dan kunci enkripsi. *File* input dibaca dalam mode biner, diubah menjadi *byte array*, lalu dipecah menjadi blok-blok data berukuran 16 *byte*. Jika panjang data tidak habis dibagi 16 *byte, padding* dilakukan dengan metode PKCS#7 untuk melengkapinya.

Contoh :

* 1. *File* input : contoh.pdf

Isi *file* dalam biner : [0x25, 0x50, 0x44, 0x46, ...]

*Byte array* : [37, 80, 68, 70, 10, 0, 45, 78, ...]

* 1. Kunci enkripsi :

Kunci : surabayaitelkom

ASCII : [115, 117, 114, 97, 98, 97, 121, 97, 105, 116, 116,

101, 108, 107, 111, 109]

* 1. Matriks data *file* :

|  |  |  |  |
| --- | --- | --- | --- |
| [37, | 80, | 68, | 70] |
| [10, | 0, | 45, | 78] |
| [255, | 100, | 32, | 12] |
| [80, | 50, | 60, | 72] |

1. *AddRoundkey*

Merupakan tahap awal proses enkripsi, di mana *plainteks* ini digabungkan dengan kunci awal yang menggunakan operasi logika yaitu XOR. Di mana blok data *file* digabungkan dengan kunci awal (W0-W3) menggunakan operasi XOR untuk menambahkan lapisan keamanan awal.

Contoh :

* 1. Proses XOR :

37 XOR 115 = 86

80 XOR 117 = 37

68 XOR 114 = 54

70 XOR 97 = 39

* 1. Matriks setelah *AddRoundkey* :

[86, 37, 54, 39]

[108, 97, 76, 47]

[150, 16, 84, 97]

[28, 89, 83, 37]

1. Proses ronde 1-9
   1. *SubBytes*

Pada proses ini, nantinya setiap *byte* pada matriks *state* ( data yang diperoleh *AddRounKey*) akan diubah menggunakan tabel substitusi yang biasa disebut dengan S-Box. *Tranformasi* ini bertujuan untuk meningkatkan keacakan data agar sulit diprediksi.

Contoh proses :

*Byte* 37 diganti dengan nilai dari tabel S-Box jadi 120

* 1. *ShiftRows*

Setiap baris dalam matriks *state* nantinya digeser ke kiri pada baris pertama tidak ada pergeseran, baris kedua di geser 1 *byte* ke kiri, baris ketiga ini digeser 2 *byte* ke kiri, dan terakhir baris keempat yang nantinya digeser 3 *byte* ke kiri. Contoh proses :

[244, 120, 99, 200] [244, 120, 99, 200]

[60, 107, 55, 175] [107, 55, 175, 60]

[12, 243, 172, 85] [172, 85, 12, 243]

[180, 97, 92, 65] [65, 180, 97, 92]

* 1. *AddRoundkey* (W4-W7)

Setelah *ShiftRows* dilakukan operasi XOR antara matriks *state* dengan kunci ronde berikutnya yaitu (W4-W7) Contoh proses :

*Byte* 244 XOR 115 = 135

*Byte* 120 XOR 117 = 13

Hasil *AddRounKey* : [135, 13, 17, 169]

[38, 98, 85, 12]

[55, 32, 82, 84]

[49, 53, 180, 193]

* 1. *MixColumns*

Setiap kolom pada matriks *state* dimodifikasi dengan operasi matematis berbasis matriks yang menggunakan bidang *Galois* (GF). Di mana proses ini bertujuan untuk mencampur elemen-elemen dalam kolom dan meningkatkan difusi data. [2, 3, 1, 1]

[1, 2, 3, 1]

[1, 1, 2, 3]

[3, 1, 1, 2]

* 1. *AddRoundkey* (W36-W39)

Dilakukan setelah *MixColums* untuk menambahkan lapisan keamanan dengan kunci ronde tambahan (W36-W39).

1. Pada ronde terakhir, hanya dilakukan tiga langkah yaitu

*SubBytes, ShiftRows*, dan *AddRoundkey* tanpa *MixColumns.*

Proses modifikasi algoritma AES ini digunakan untuk mengenkripsi *file* dengan menambahkan langkah *AddRoundkey* tambahan, yang meningkatkan keamanan dengan memberikan lapisan XOR tambahan pada setiap ronde enkripsi.

* + 1. **Implementasi Kombinasi AES Modifikasi dan ChaCha20** Algoritma AES modifikasi dan ChaCha20 dengan menerapkan *double encypt.* Proses ini memakai AES modifikasi dengan memberikan perlindungan berbasis blok *chiper* yang kuat dengan memodifikasi *AddRoundkey* tambahan sebelum *MixColumns,* sedangkan ChaCha20 sebagai *stream cipher* yang cepat dan aman, meningkatkan kompleksitas keseluruhan dengan memanfaatkan proses enkripsi berbasis *keystream.*

Pada metode ini, setiap *file* menggunakan 1 kunci enkripsi. Kunci yang digunakan memiliki panjang awal 128-bit, yang kemudian diperluas menjadi 256-bit pada ChaCha20 dengan menambahkan padding nol (0x00) jika panjangnya kurang dari 256-bit. Proses

dimulai dengan mengenkripsi *file* menggunakan AES Modifikasi untuk menghasilkan cipherteks awal dalam format matriks 4x4. Cipherteks ini kemudian diubah menjadi byte stream untuk kompatibilitas dengan ChaCha20. Selanjutnya, ChaCha20 melakukan proses enkripsi kedua dengan memanfaatkan kunci 256- bit, nonce sepanjang 96-bit, dan counter. Hasil akhir adalah cipherteks final yang lebih aman karena telah melalui dua lapisan enkripsi. Penggunaan double encryption dengan satu kunci untuk setiap *file* memberikan keamanan yang lebih tinggi, di mana kombinasi kedua algoritma menciptakan perlindungan yang kuat terhadap berbagai jenis serangan kriptografi.

*Gambar III.3 Diagram Proses Double Encrypt*

Pada [Gambar III.3 Diagram Proses Double Encrypt](#_bookmark50) menunjukkan langkah-langkah enkripsi ganda. Proses dimulai dengan mengenkripsi *plainteks* menggunakan AES modifikasi untuk menghasilkan *chiperteks* AES. *Chiperteks* ini kemudian dienkripsi ulang menggunakan ChaCha20, yang melibatkan matriks 4x4, proses *quarter-round,* dan operasi XOR dengan *keystream*. Algoritma AES modifikasi dan ChaCha20 menggunakan kunci yang sama, yaitu kunci AES sepanjang 128-bit. berikut adalah tahapan alur di atas :

1. Input *plainteks* dan *key* dan enkripsi AES modifikasi
2. Input *cipherteks* AES ke ChaCha20

*Chiperteks* yang dihasilkan dari proses enkripsi pertama diubah menjadi input untuk algoritma ChaCha20. Ini dilakukan untuk menerapkan enkripsi berlapis yaitu *double encryption,* sehingga keamanan data akan meningkat secara signifikan. *Chiperteks* AES yang biasanya dalam format matriks (4x4) ini diubah ke format *byte stream* untuk kompatibilitas dengan algoritma ChaCha20. Byte stream ini memungkinkan untuk memproses data, sesuai dengan *stream chiper*.

Contoh :

[115, 75, 189, 234, 201, 97, 58, 14, 245, 142, 12, 120, 53, 190,

201, 20] data ini sudah dalam bentuk *byte stream*.

1. Validasi panjang *key* (256-bit *Key*)

Ini untuk memastikan bahwa kunci enkripsi untuk ChaCha20 sesuai dengan spesifikasi algoritma, yaitu panjangnya harus tepat 256 bit (32 *byte*). Kunci ini untuk menghasilkan *keystream* yang beda dalam proses enkripsi. Jika panjang ke kurang dari 256 bit, ditambahkan *padding* berupa *byte* nol (0x00) untuk memenuhi panjang. Jika *key* yang lebih panjang, *key* akan dipotong hingga hanya menyisakan 256 bit. *Key* kemudian dibagi menjadi 8 bagian ( di mana masing-masing 32 bit) untuk digunakan dalam proses pembentukan matriks awal (*state* matriks)

Contoh :

* 1. *Key* input “surabayaittelkom”

[115, 117, 114, 97, 98, 97, 121, 97, 105, 116, 116, 101, 108,

107, 111, 109]

* 1. Panjang *key* adalah 16 *byte* (128 bit), sehingga tidak memenuhi syarat panjang 256 bit. Jadi ditambahkan *padding* nol (0x00) hingga mencapai 32 *byte:*

[115, 117, 114, 97, 98, 97, 121, 97, 105, 116, 116, 101, 108,

107, 111, 109, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

1. Membuat *Initial* State Matriks 4x4

Membentuk *state* matriks awal yang menjadi dasar operasi dalam ChaCha20. Matriks ini berisi kunci, *counter*, dan *nonce.* Matriks 4x4 dalam ChaCha20 terdiri 16 elemen (masing-masing 32 bit) Contoh :

* 1. 4 elemen konstanta tetap “*expand* 32-byte k”4 elemen 32 bit [1634760805, 857760878, 2036477234, 1797285236]
  2. Elemen kunci 256-bit dibagi menjadi 8 elemen 32-bit: [1936028945, 1885434721, 1701147239, 1647539057,

1225269481, 1226844037, 1214606441, 0]

* 1. 1 *Counter*: Nilai awal 0.
  2. 3 Elemen Nonce: Nilai *nonce* yang unik untuk sesi enkripsi: Contoh *nonce* : [12345678, 87654321, 11223344]
  3. Matriks yang dibentuk

[1634760805, 857760878, 2036477234, 1797285236]

[1936028945, 1885434721, 1701147239, 1647539057]

[1225269481, 1226844037, 1214606441, 0]

[0, 12345678, 87654321, 11223344]

Baris ke 1 konstanta, ke 2 *key,* ke 3 *key* dan ke 4 *counter* dan

*nonce*

1. Operasi Quarter Round (20 iterasi)

Melakukan proses pencampuran data di dalam matriks untuk meningkatkan kerandoman dan keamanan *keystream* yang dihasilkan. Quarter Round adalah inti dari algoritma ChaCha20. Quarter Round dilakukan pada 4 elemen tertentu dari matriks. Penjumlahan Modular (mod 2³²): Menambah elemen dalam matriks untuk mencampur data. Operasi XOR: Mencampur data antar elemen melalui operasi logika XOR. Rotasi Bit (ROTL): Menggeser bit elemen untuk mendistribusikan informasi secara acak. Setelah 20 iterasi (10 putaran double round), matriks akan diubah menjadi bentuk yang sangat acak.

1. *Counter* +=1

*Counter* bertambah untuk setiap blok data yang diproses, memastikan *keystream* yang dihasilkan unik untuk setiap blok. Setelah satu blok data selesai diproses, *counter* di matriks bertambah sebesar 1.

Contoh :

*Counter* baru : [1, 12345678, 87654321, 11223344]

1. XOR *keystream* dengan *chiperteks* AES

Menggabungkan *keystream* hasil ChaCha20 dengan *cipherteks AES* melalui operasi XOR untuk menghasilkan *final ciphertext*. XOR dipilih karena sifatnya yang sederhana namun aman dan mudah dibalik. *Keystream* yang dihasilkan dari matriks ChaCha20 di-XOR-kan dengan *cipherteks AES*.

Operasi XOR:

Jika *ciphertext* = *keystream* XOR *plaintext*, maka *dekripsi*

dilakukan dengan cara *plaintext* = *keystream* XOR *ciphertext*.

* + - 1. **Proses *Dekripsi* Double Encryption**

Proses *dekripsi* dilakukan untuk mengembalikan data yang telah terenkripsi ke bentuk aslinya atau *plainteks*. Di mana pada tahap ini, *chiperteks* hasil dari kombinasi algoritma AES modifikasi dan Chacha20 akan *didekrpsi* secara urut. Proses nantinya akan dimulai dengan *mendekripsi chiperteks* dengan ChaCha20 untuk mendapatkan kembali *chiperteks* AES modifikasi, kemudian dilanjutkan dengan *mendekripsi chiprteks* AES modifikasi untuk memperoleh *plainteks* asli.

Pada [Gambar III.4 Proses Dekripsi](#_bookmark51) Diagram alur di atas menjelaskan proses *dekripsi* dari kombinasi algoritma ChaCha20 dan modifikasi AES. Proses dimulai dengan memasukkan cipherteks hasil enkripsi sebagai input. Langkah pertama adalah men*dekripsi* cipherteks menggunakan algoritma ChaCha20 untuk menghasilkan keystream. Keystream ini kemudian dikombinasikan dengan cipherteks menggunakan operasi XOR untuk mendapatkan cipherteks AES.

Selanjutnya, proses ekspansi kunci dilakukan untuk menghasilkan kunci yang akan digunakan pada setiap ronde *dekripsi* AES. Proses *dekripsi* AES terdiri dari tiga tahapan utama: ronde terakhir (Round 10) yang melibatkan *AddRoundKey*, InvShiftRows, dan InvSubBytes; ronde utama (Round 9-1) yang melibatkan *AddRoundKey*, InvMixColumns, *AddRoundKey, InvShiftRows,* dan *InvSubBytes* tambahan; dan ronde awal (Round 0) yang menggunakan *AddRoundKey* (W0-W3). Hasil akhir dari seluruh proses ini adalah plainteks, yaitu data asli yang berhasil di*dekripsi*. Kombinasi kedua algoritma ini memastikan keamanan data yang lebih kuat melalui enkripsi ganda.

*Gambar III.4 Proses Dekripsi*

### Implementasi Antarmuka Pengguna (UI)

Antarmuka pengguna (UI) dirancang untuk mempermudah interaksi, berfungsi sebagai media interaktif yang memungkinkan pengguna untuk memilih file, memasukkan kunci enkripsi, serta memulai proses enkripsi dan dekripsi dengan mudah. Implementasinya menggunakan Flask sebagai framework backend untuk menangani

proses enkripsi dan dekripsi, sedangkan HTML digunakan untuk membuat form input file dan kunci enkripsi. CSS digunakan untuk mempercantik tampilan agar antarmuka lebih menarik, jelas, dan mudah digunakan oleh pengguna dalam proses enkripsi dan dekripsi file menggunakan kombinasi algoritma ChaCha20 dan AES yang telah dimodifikasi.

*Gambar III.5 Desain Antarmuka Pengguna Enkripsi dan Dekripsi*

Pada [Gambar III.5 Desain Antarmuka Pengguna Enkripsi dan](#_bookmark53) [Dekripsi](#_bookmark53) Gambar tersebut menampilkan desain antarmuka pengguna (UI) dari sebuah aplikasi berbasis web yang digunakan untuk enkripsi dan *dekripsi file*. Antarmuka ini memiliki beberapa elemen input pengguna untuk mengunggah *file* dan memilih opsi enkripsi atau *dekripsi*. Terdapat tiga kolom utama pada desain ini: pertama, Pilih *File*, untuk pengguna memilih *file* yang akan diproses. Kedua, Enkripsi atau *Dekripsi*, yang memberikan pilihan kepada pengguna apakah *file* yang dipilih akan dienkripsi atau di *dekripsi*, yang merupakan opsi penting untuk menentukan jenis operasi yang akan dilakukan pada *file* tersebut. Ketiga, Key, di mana pengguna diminta untuk memasukkan kunci yang diperlukan untuk proses enkripsi atau *dekripsi*. Kunci ini memainkan peran penting dalam menjaga keamanan data yang diproses. Di bagian bawah antarmuka, terdapat

tombol Proses yang akan menjalankan operasi sesuai dengan pilihan pengguna, baik itu enkripsi maupun *dekripsi*.

### Pengujian Algoritma

Menjelaskan pengujian yang dilakukan untuk mengevaluasi kinerja dan keamanan algoritma berdasarkan parameter seperti entropi, waktu enkripsi dan *dekripsi*, serta kecapatan.

### Pengujian Entropi

Pengujian kekuatan algoritma dilakukan dengan mengukur tingkat keacakan data hasil enkripsi menggunakan analisis entropi. Keacakan data merupakan indikator penting dalam menentukan seberapa baik algoritma enkripsi menyamarkan pola asli dari data. Semakin tinggi tingkat keacakan, semakin sulit bagi pihak yang tidak berwenang untuk menganalisis pola atau memprediksi isi data terenkripsi. Pengujian ini dilakukan dengan menghitung nilai entropi menggunakan distribusi probabilitas setiap simbol dalam cipherteks. Nilai entropi yang mendekati maksimum menunjukkan bahwa distribusi simbol dalam cipherteks hampir merata, menandakan bahwa algoritma telah berhasil menghasilkan data yang acak. Data dengan keacakan tinggi lebih tahan terhadap serangan berbasis statistik, sehingga pengujian ini memberikan evaluasi kuantitatif terhadap kekuatan keamanan algoritma yang diuji. Hasil pengujian dibandingkan antara AES Standar, AES Modifikasi, Chacha20 dan Double Encryption untuk menunjukkan efektivitas modifikasi dalam meningkatkan keamanan.

### Pengujian Waktu dan Kecepatan

Menguji algoritma AES modifikasi dan *Double encrypt* AES modifikasi dan ChaCha20. Pengujian ini dilakukan untuk menilai kinerja algoritma dalam berbagai parameter. Penelitian ini akan menganalisis parameter pengujian menggunakan matriks dan formula berikut :

* + - * 1. Waktu enkripsi :

mengukur waktu yang dibutuhkan untuk mengenkripsi data dari

*plainteks* menjadi *chiperteks* dijelaskan dalam persamaan [(III-1)](#_bookmark55).

|  |  |
| --- | --- |
| 𝐸𝑡 = 𝐸𝑁𝑡 – 𝑆𝑇𝑡 | (III-1) |

Di mana:

𝐸𝑡 = waktu enkripsi dalam mili detik (ms)

𝐸𝑁𝑡 = waktu selesai enkripsi

𝑆𝑇𝑡 = waktu mulai enkripsi

Waktu ini akan dihitung untuk setiap *file* yang diuji, dan hasilnya nanti akan memberikan gambaran mengenai kecepatan algoritma dalam melakukan proses enkripsi.

* + - * 1. Waktu *dekripsi* :

Mengukur waktu yang dibutuhkan untuk *mendekripsi* data dari

*chiperteks* menjadi *plainteks* dijelaskan pada persamaan [(III-2)](#_bookmark56).

Dimana:

𝐷𝑡 = waktu *dekripsi* (ms)

𝐸𝑁𝑡 = waktu selesai *dekripsi*

𝑆𝑇𝑡 = waktu mulai *dekripsi*

Hasil waktu *dekripsi* ini juga perlu untuk masing-masing algoritma karena ini menganalisis perbandingan kinerja antara algoritma enkripsi.

* + - * 1. Kecepatan :

Menunjukkan jumlah data dalam *kilobyte* yang dapat dienkripsi

Et = waktu enkripsi atau *dekripsi* (s)

### Pengujian pada Berbagai Jenis *File*

*File* PDF yang digunakan dalam pengujian dapat berisi elemen seperti teks, gambar, dan tabel yang semuanya akan dienkripsi secara utuh karena algoritma bekerja pada level data biner. *File* PDF dengan ukuran 100 KB, 1 MB, 10 MB, hingga 100 MB diuji untuk memastikan algoritma dapat bekerja pada dokumen berukuran kecil hingga besar. Hal serupa juga berlaku untuk *file* Word dengan ukuran yang sama, meskipun memiliki struktur internal yang lebih kompleks dibandingkan PDF. *File* TXT dengan ukuran maksimal 300 *byte*, yang biasanya hanya berisi teks sederhana, juga dapat dienkripsi ChaCha20 (*double encrypt*). Di mana pengujian ini bertujuan untuk

mengukur kinerja algoritma. Berikut ini adalah contoh hasil pengujian terhadap *file* PDF :

*Tabel III.2 Evaluasi Hasil*

Pada [Tabel III.2 Evaluasi Hasil](#_bookmark60) menunjukkan hasil evaluasi simulasi kinerja algoritma enkripsi dan *dekripsi*. pada Nilai-nilai tersebut memberikan gambaran awal mengenai performa beberapa metode, termasuk AES Standar, ChaCha20, AES Modifikasi, dan kombinasi AES Modifikasi dengan ChaCha20. Hasil ini digunakan sebagai contoh output yang diharapkan dari model algoritma yang akan dirancang, ini belum merupakan data dari eksperimen nyata.

# DAFTAR PUSTAKA

Alimuddin, A., Juntak, J. N. S., Jusnita, R. A. E., Murniawaty, I., & Wono, H. Y. (2023). Teknologi dalam pendidikan: Membantu siswa beradaptasi dengan revolusi industri 4.0. *Journal on Education*, *5*(4), 11777–11790.

Almukhlifi, R., & Vora, P. L. (2020). Linear cryptanalysis of reduced-round simon using super rounds. *Cryptography*, *4*(1), 1–34. https://doi.org/10.3390/cryptography4010009

Alyas, H. H., & Abdullah, A. A. (2021). Enhancement the ChaCha20 Encryption Algorithm Based on Chaotic Maps. *Lecture Notes in Networks and Systems*, *201*, 91–107. https://doi.org/10.1007/978-981-16-0666-3\_10

Anindita, M., & 18219086, A. A. (2023). *Analisis Perbandingan Algoritma AES dan ChaCha20-Poly1305 dalam Enkripsi Konten Livestreaming*.

Aryanto, M. B., Tahir, M., Devita, S. I., Mustofa, Z. N., Ainiyah, Q., & Sundoro,

S. (2023). Implementasi Enkrip Dan Dekrip File Menggunakan Metode Advance Encryption Standard (AES-128). *Jurnal Ilmiah Sistem Informasi Dan Ilmu Komputer*, *3*(1), 89–104.

Assyahid, M. H. (2024). *Implementasi interpolasi pixel wadah citra dan enkripsi aes pada steganografi least significan bit (LSB)*. https://repository.uinjkt.ac.id/dspace/handle/123456789/76727

Barbero, S., Bazzanella, D., & Bellini, E. (2022). Rotational Cryptanalysis on ChaCha Stream Cipher. *Symmetry*, *14*(6). https://doi.org/10.3390/sym14061087

Belvin Tannadi. (2022). *Ilmu Crypto* (Digital). Jakarta : Elex Media Komputindo.

Cristy, N., & Riandari, F. (2021). *Niolinda Cristy 1 , Fristi Riandari 2 [Implementasi Metode Advanced Encryption Standard (AES 128 Bit) Untuk Mengamankan Data Keuangan*. *4*(2), 75.

Djong, H. S., & Siswanto, S. (2022). Implementasi Kriptografi Dengan Menggunakan Metode RC4 dan AES-256 Untuk Mengamankan File Dokumen pada PT Varnion Technology Semesta. *Prosiding Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI)*, *1*(1), 149–158.

Handoyo, J., & Subakti, Y. M. (2020). Keamanan Dokumen Menggunakan Algoritma Advanced Encryption Standard (AES). *Jurnal SITECH : Sistem Informasi Dan Teknologi*, *3*(2), 143–152. https://jurnal.umk.ac.id/index.php/sitech/article/view/5865

Hena, N., & Kusumaningsih, D. (2024). Pengamanan File Pada Sistem Master Vendor Berbasis WEB Menggunakan Algoritma AES Pada Trinitiland. *Prosiding Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI)*, *3*(2), 30–37.

Hidayatulloh, N. W., Tahir, M., Amalia, H., Basyar, N. A., Prianggara, A. F., & Yasin, M. (2023). Mengenal Advance Encryption Standard (AES) sebagai Algoritma Kriptografi dalam Mengamankan Data. *Digital Transformation Technology*, *3*(1), 1–10.

Kebande, V. R. (2023). Extended-Chacha20 Stream Cipher With Enhanced Quarter Round Function. *IEEE Access*, *11*, 114220–114237. https://doi.org/10.1109/ACCESS.2023.3324612

Kim, B. G., & Wong, D. (2023). Smart Contract-based Secure Verifiable Random Function using ChaCha20 Sequence in Blockchain. *ACM International Conference Proceeding Series*, 41–51. https://doi.org/10.1145/3638025.3638028

Kurnia Mahesa Naibaho, A., Putri, A., & Fikri Husaini, Y. (2021). Analisis Sistem Keamanan Informasi Chat Online dengan Menggunakan Algoritma AES. In *Ilmu Komputer* (Vol. 6, Issue 2).

Luong, T. T. (2023). Strengthening AES Security Through Key-Dependent ShiftRow and AddRoundKey Transformations Utilizing Permutation. In *IJACSA) International Journal of Advanced Computer Science and Applications* (Vol. 14, Issue 11). [www.ijacsa.thesai.org](http://www.ijacsa.thesai.org/)

Muhammed, R. K., Rashid Aziz, R., Hassan, A. A., Aladdin, A. M., Saydah, S. J., Rashid, T. A., & Hassan, A. (2024). *Comparative Analysis of AES, Blowfish, Twofish, Salsa20, and ChaCha20 for Image Encryption*.

Olivia Putri Irine Irawan, B., Tahir, M., Ayu Windrastuti, N., Yurina Cholili, D., Mulaikah, D., Batsul Mushofi Septian wachid, A., & Pendidikan Informatika,

P. (2023). Implementasi Kriptografi Pada Keamanan Data Menggunakan Algoritma Advance Encryption Standard (AES). *Jurnal Simantec*, *11*(2), 167–

174. https://doi.org/10.21107/SIMANTEC.V11I2.20034

Pangestu, G. Y., Hadiana, A. I., & Sabrina, P. N. (2022). Kriptografi Untuk Enkripsi Ganda Pada Gambar Menggunakan Algoritma AES (Advanced Encryption Standard) Dan RC5 (Rivest Code 5). *Vol*, *1*, 25–32.

Putri, A. E., Kartikadewi, A., & Rosyid, L. A. A. (2021). Implementasi Kriptografi dengan Algoritma Advanced Encryption Standard (AES) 128 Bit dan Steganografi menggunakan Metode End of File (EOF) Berbasis Java Desktop

pada Dinas Pendidikan Kabupaten Tangerang. *Appl. Inf. Syst. Manag*, *3*(2), 69–78.

Saripa, S. (2023). Implementation Implementation of a File Security System Using the AES Algorithm to Secure Personal Files: Implementasi Sistem Keamanan File Menggunakan Algoritma AES untuk Mengamankan File Pribadi. *Progressive Information, Security, Computer, and Embedded System*, *1*(2), 138–148.

Sekar Putri Ananda, Saepul Lukman, & Irfan. (2022). *Analisa Metode Kriptogra Modern Advance Encryption Standar (AES) 128 Bit dalam Mengenkripsi dan Mendekripsi File Dokumen Digital*. https://doi.org/10.32409/jikstik.21.3.2973

Soesanto, E., Purba, L. M., Aprilia, B., Putra, D. R., & Putri, S. D. (2023). Implementasi Objek Vital, Pengamanan File dan Pengamanan Cyber di PT Pertamina. *IJM: Indonesian Journal of Multidisciplinary*, *1*(1).

Surbakti, P. S., & Purwanto, P. (2024). Implementasi Kriptografi Dengan Menggunakan Algoritma AES-128 Untuk Enkripsi dan Dekripsi File Dokumen Berbasis Web Pada Law Office Erdi Surbakti, SH & Rekan. *Prosiding Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI)*, *3*(1), 1–8.

Syahputra Sianipar, J., Budi Nuugroho, N., Mariami, I., Informasi, S., & Triguna Dharma, S. (2024). Pengamanan Data Gaji Karyawan Dengan Menggunakan Metode Advanced Encryption Standard (AES). *Jurnal Sistem Informasi Triguna Dharma (JURSI TGD)*, *3*(1), 35–45. https://doi.org/10.53513/JURSI.V3I1.5653

Wahyu Putra, M. R. F. T. H. (2023, July 2). *Implementasi Algoritma Advanced Encryption Standard Untuk Kemanan Dokumen*. https://journal.grahamitra.id/index.php/jurikti/article/view/55/181

Wahyudi, E. N., Ardhianto, E., Handoko, W. T., Murti, H., Supriyanto, E., Lestariningsih, E., & Redjeki, R. S. (2024). Peningkatan Keamanan Data Melalui Teknik Super Enkripsi Menggunakan Algoritma Vigenere dan Caesar. *Jurnal Informatika Polinema*, *10*(3), 315–322. https://doi.org/10.33795/JIP.V10I3.5131

Widyawan, D., & Imelda, I. (2021). Pengamanan File Menggunakan Kriptografi Dengan Metode Aes-128 Berbasis Web Di Komite Nasional Keselamatan Transportasi. *SKANIKA: Sistem Komputer Dan Teknik Informatika*, *4*(1), 15– 22.

**IMPLEMENTASI KOMBINASI CHACHA20 DAN MODIFIKASI AES MELALUI *ADDROUNDKEY* PADA**

**STUDI KASUS KEAMANAN *FILE***

**Proposal Tugas Akhir Mata Kuliah Penulisan Proposal**

## (CAK4EAB2)

Disusun oleh:

## EMILIA RAMONA 1203210017

**PROGRAM STUDI SARJANA INFORMATIKA DIREKTORAT KAMPUS SURABAYA**

**UNIVERSITAS TELKOM SURABAYA TAHUN 2025**

# ABSTRAK

Keamanan data menjadi tantangan besar di era Revolusi Industri 4.0, di mana informasi penting perlu dilindungi dari akses tidak sah. Ancaman serangan siber dan peretasan mengharuskan pengembangan solusi keamanan yang lebih efektif. Penelitian ini bertujuan untuk mengimplementasikan dan mengevaluasi kombinasi algoritma ChaCha20 dan modifikasi *Advanced Encryption* Standard (AES) dengan penambahan langkah *AddRoundKey* sebelum *MixColumns* untuk meningkatkan keamanan *file*. Metode yang digunakan adalah enkripsi ganda, dengan AES modifikasi sebagai *block cipher* yang kuat dan ChaCha20 sebagai *stream cipher* yang cepat. Penelitian ini mencakup implementasi algoritma dengan menggunakan pustaka kriptografi untuk menguji kinerja algoritma melalui parameter seperti waktu enkripsi, waktu *dekripsi*, kecepatan enkripsi, kecepatan *dekripsi*, dan tingkat keacakan data menggunakan analisis entropi. *File* dengan berbagai format dan ukuran, PDF, Word, dan TXT, digunakan sebagai data pengujian untuk mengevaluasi efisiensi algoritma. Selain itu, antarmuka pengguna berbasis web dirancang untuk mempermudah proses enkripsi dan *dekripsi*, melibatkan langkah- langkah seperti pemilihan *file* dan *input* kunci enkripsi. Penelitian ini diharapkan menunjukkan bahwa kombinasi AES modifikasi dan ChaCha20 mampu meningkatkan keamanan data serta memberikan efisiensi yang lebih baik dalam perlindungan *file*.

**Kata Kunci**: Keamanan *File*, *Advanced Encryption Standard* (AES), ChaCha20, Enkripsi Ganda, *AddRoundkey*, *Entropi*.

# ABSTRACT

*Data security has become a major challenge in the era of the Industrial Revolution 4.0, where critical information must be protected from unauthorized access. Cyberattacks and hacking threats demand the development of more effective security solutions. This research aims to implement and evaluate the combination of the ChaCha20 algorithm and a modified Advanced Encryption Standard (AES) by adding an AddRoundKey step before MixColumns to enhance file security. The method used is double encryption, with the modified AES as a robust block cipher and ChaCha20 as a fast stream cipher. This study involves implementing the algorithms using cryptographic libraries to test their performance through parameters such as encryption time, decryption time, encryption speed, decryption speed, and data randomness level using entropy analysis. Files of various formats and sizes, including PDF, Word, and TXT, are utilized as test data to evaluate the efficiency of the algorithms. Additionally, a web-based user interface is designed to simplify the encryption and decryption process, involving steps such as file selection and encryption key input. This research is expected to demonstrate that the combination of modified AES and ChaCha20 can enhance data security and provide better efficiency in file protection.*

***Keywords****: File Security, Advanced Encryption Standard (AES), ChaCha20, Double Encryption, AddRoundkey, Entropy*

# KATA PENGANTAR

Assalamu’alaikum Warrahmatullahi Wabarakatuh, puja puji syukur ke hadirat Allah SWT yang telah memberikan nikmat dan karunia-Nya yang sangat luar biasa kepada hamba – hambanya. Tidak lupa shalawat serta salam kita curahkan atas bimbingan nabi kita nabi Muhammad SAW yang telah membawa cahaya islam ke dalam dunia ini dan menjadi panutan bagi setiap umatnya hingga akhir zaman.

Setelah melalui berbagai proses kerja keras, penyusunan skripsi yang berjudul “Implementasi Kombinasi ChaCha20 dan Modifikasi AES melalui *AddRoundkey* Pada Studi Kasus Keamanan *File*” akhirnya dapat terselesaikan. Selain sebagai salah satu syarat memperoleh gelar S-Kom, penulis berharap juga skripsi ini dapat bermanfaat bagi semua pihak yang berkepentingan. Oleh karena itu, penulis ingin mengungkapkan rasa terima kasih kepada pihak-pihak yang telah memberikan dukungan dan doa sehingga penulis dapat menyelesaikan skripsi ini tepat pada waktunya. Beberapa pihak tersebut antara lain:

1. **Kepada Mama** yang sangat penulis cintai yang tidak pernah lupa selalu memberikan dukungannya dan doa agar diberikan kemudahan dan kelancaran dalam penyusunan skripsi ini sehingga penulis dapat menyelesaikan studi Informatika di Universitas IT Telkom Surabaya.
2. **Kepada Alm.Ibu Suhartatik**, yang penulis cintai dan sayangi, yang telah memberikan motivasi untuk saya.
3. **Kepada Saudara Kembar**, yang selalu mendukung, terima kasih atas bantuan dan waktunya dalam memberikan masukan terkait susunan skripsi ini.
4. **Keluarga Besar penulis** yang telah memberikan motivasi dan kepercayaan besar sehingga penulis bisa mewujudkan impian untuk meraih gelar S-Kom.
5. **Dosen Pembimbing**, Pak Rizky Fenaldo Maulana, S.Kom., M.Kom. dan Tanzila Mustaqim, S.Kom., M.Kom. Yang telah memberikan solusi, waktu, dan bimbingan penuh kesabaran selama proses penyusunan skripsi ini.
6. **Teman-teman dekat**, termasuk Jus’an, Erica Dyah, Qori Emalia, Ardina Firdatana, dan Elma Saskia yang selalu memberikan semangat, mendengarkan keluh kesah, serta menemani dalam berbagai kondisi selama penulisan skripsi.
7. **Semua pihak lainnya**, yang tidak dapat disebutkan satu per satu, atas bantuan dan dukungannya selama proses ini.
8. **Diri penulis sendiri**, atas perjuangan dan kerja keras hingga akhirnya skripsi ini selesai tepat waktu.

Penulis menyadari bahwa skripsi ini masih jauh dari kesempurnaan. Oleh karena itu, saran dan kritik yang membangun sangat diharapkan untuk memperbaiki skripsi ini ke depannya.

Terima kasih.

### Wassalamu’alaikum Warrahmatullahi Wabarakatuh.

Surabaya, 24 Desember 2024 Penulis

Emilia Ramona

# DAFTAR ISI

[LEMBAR PERSETUJUAN ii](#_bookmark0)

[ABSTRAK iii](#_bookmark1)

[ABSTRACT iv](#_bookmark2)

[KATA PENGANTAR v](#_bookmark3)

[DAFTAR ISI vii](#_bookmark4)

[DAFTAR TABEL ix](#_bookmark5)

[DAFTAR GAMBAR x](#_bookmark6)

[DAFTAR RUMUS xi](#_bookmark7)

[BAB I PENDAHULUAN 1](#_bookmark8)

* 1. [Latar Belakang 1](#_bookmark9)
  2. [Rumusan Masalah 4](#_bookmark10)
  3. [Tujuan Penelitian 4](#_bookmark11)
  4. [Batasan dan Asumsi Penelitian 5](#_bookmark12)
  5. [Manfaat Penelitian 5](#_bookmark13)
     1. [Manfaat Akademis 5](#_bookmark14)
     2. [Manfaat Ilmiah 5](#_bookmark15)
  6. [Sistematika Penulisan 6](#_bookmark16)

[BAB II LANDASAN TEORI 7](#_bookmark17)

* 1. [Literatur Terkait Teori 7](#_bookmark18)
  2. [Kajian Teori 10](#_bookmark20)
     1. [Kriptografi 10](#_bookmark21)
     2. [AES (Advanced Encryption Standard) 12](#_bookmark24)
     3. [Chacha20 20](#_bookmark32)
     4. [Entropi 22](#_bookmark35)

[BAB III METODOLOGI PENELITIAN 24](#_bookmark37)

* 1. [Sistematika Penyelesaian Masalah 24](#_bookmark38)
  2. [Alat dan Bahan 24](#_bookmark39)
     1. [Alat 24](#_bookmark40)
     2. [Bahan 25](#_bookmark41)
  3. [Rancangan Penelitian 25](#_bookmark42)
     1. [Studi Literatur 27](#_bookmark45)
     2. [Pengumpulan Data 27](#_bookmark46)
     3. [Implementasi Modifikasi AES 28](#_bookmark47)
     4. [Implementasi Kombinasi AES Modifikasi dan ChaCha20 31](#_bookmark49)
     5. [Implementasi Antarmuka Pengguna (UI) 36](#_bookmark52)
     6. [Pengujian Algoritma 38](#_bookmark54)
     7. [Analisis Hasil 42](#_bookmark59)

[DAFTAR PUSTAKA 44](#_bookmark61)

# DAFTAR TABEL

[Tabel II.1 Literatur Penelitian Terkait Kriptografi 7](#_bookmark19)

[Tabel II.2 Panjang Kunci 13](#_bookmark25)

[Tabel II.3 Tabel Substitusi (S-Box) 16](#_bookmark28)

[Tabel II.4 Transformasi ShiftRows 17](#_bookmark29)

[Tabel III.1 Jenis, Ukuran, dan Isi File untuk Pengujian 40](#_bookmark58)

[Tabel III.2 Evaluasi Hasil 43](#_bookmark60)

# DAFTAR GAMBAR

[Gambar II.1 Alur Proses Enkripsi Dan Dekripsi 11](#_bookmark22)

[Gambar II.2 Proses Kunci Simetris 11](#_bookmark23)

[Gambar II.3.Transformasi ShiftRows 15](#_bookmark27)

[Gambar II.4 Proses AES 19](#_bookmark31)

[Gambar III.1 Diagram Alur Penelitian 26](#_bookmark43)

[Gambar III.2 Diagram Proses Enkripsi AES 28](#_bookmark48)

[Gambar III.3 Diagram Proses Double Encrypt 32](#_bookmark50)

[Gambar III.4 Proses Dekripsi 36](#_bookmark51)

[Gambar III.5 Desain Antarmuka Pengguna Enkripsi dan Dekripsi 37](#_bookmark53)

# DAFTAR RUMUS

[(II-1) Rumus AddRoundKey 14](#_bookmark26)

[(II-2) Operasi Perkalian MixColumns 17](#_bookmark30)

[(II-3) Rumus Keystream 22](#_bookmark33)

[(II-4) Rumus Proses Enkripsi XOR pada Keystream 22](#_bookmark34)

[(II-5) Entropi 22](#_bookmark36)

[(III-1) Pengujian Waktu Enkripsi 39](#_bookmark55)

[(III-2) Pengujian Waktu Dekripsi 39](#_bookmark56)

[(III-3) Pengujian Waktu Kecepatan 39](#_bookmark57)

# BAB I PENDAHULUAN

### Latar Belakang

Semakin meningkatnya teknologi yang semakin canggih membawa sebuah perubahan yang cukup signifikan. Perubahan tersebut dikenal dengan istilah Revolusi Industri 4.0 yang merupakan sebuah kemajuan teknologi yang sering digunakan oleh banyak orang. Revolusi industri 4.0 adalah sebuah data yang digerakkan dengan adanya campur tangan komputer menggunakan metode machine learning atau AI (Alimuddin et al., 2023). Menurut Klaus Schawab revolusi Industri 4.0 memiliki tantangan yang harus dihadapi salah satunya seperti keamanan *cyber* dan privasi data.

Maka dari itu, dengan semakin pesatnya informasi teknologi maka perlunya untuk mengamankan data atau *file*. Keamanan data atau *file* merupakan salah satu langkah yang mudah untuk melindungi sebuah informasi bersifat penting yang berasal dari akses tidak sah. Keamanan *file* ini sangat diperlukan pada era Revolusi Industri 4.0, karena maraknya serangan *cyber* dan peretasan komputer yang bisa mengganggu keberlangsungan suatu industri. Menurut (Mulyana*, 2016*), keamanan *file* merupakan salah satu aksi pencegahan dari sebuah penganggu pengguna *file* atau pengguna yang memakai jaringan yang tidak berkewajiban memiliki akses (Soesanto et al., 2023). Dengan semakin pesatnya dunia teknologi ini membuat keamanan *file* menjadi sangat penting karena untuk melindungi privasi dan informasi dari pihak yang tidak memiliki akses.

Keamanan *file* adalah serangkaian metode atau teknik yang untuk melindungi *file* dari akses yang tidak sah, seperti pencurian atau kerusakan. Tujuannya untuk menjaga informasi dalam *file* tetap rahasia, aman, dan hanya dapat di akses oleh pihak yang berwenang. Oleh karena itu, upaya untuk memastikan agar *file* terjaga dengan menggunakan metode ChaCha20 dan *Advanced Encryption Standard* (AES). ChaCha20 merupakan metode enkripsi yang cepat dan aman. ChaCha20 dipopulerkan oleh Daniel J.

Bernstein sebagai varian dari algoritma. Dalam algoritma ChaCha20 sangat populer dalam pengimplementasi enkripsi dikarenakan percampuran antara kecepatan dan keamanannya yang dapat diunggulkan. Chaca20 mempunyai desain yang hampir sama dengan Salsa20, pada algoritma ChaCha20 menggunakan kunci 256-bit yang mempunyai 8 putaran (Kebande, 2023).

Sedangkan AES (*Advanced Encryption Standard)* adalah sebuah algoritma yang melakukan transformasi terhadap blok data yang menggunakan kunci enkripsi yang sama. AES bekerja pada sistem yang melibatkan operasi matematika dengan level tingkat sulit (Saripa, 2023). Proses kerja enkripsi dasar AES adalah *subbytes* (menggunakan tabel S-Box untuk menggabungkan *byte* pada blok matriks), *shift row* (digunakan untuk menggeser baris matriks), *MixColumns* (melakukan pergeseran dari kanan ke kiri), dan *AddRoundkey* (blok di XOR kan dengan kunci yang diperluas) (Aryanto et al., 2023). Pada proses *dekripsi* AES terdapat tahap-tahapan kebalikan dari proses enkripsi. Dengan memakai kunci yang sama untuk *dekripsi*, blok data yang telah terenkripsi dapat dipulihkan kembali ke bentuk awalnya. algoritma AES adalah jenis *block chipper* yang memiliki panjang kunci 128 bit, 192 bit dan 256 bit (Surbakti & Purwanto, 2024).

Algoritma Chacha20 dan AES merupakan suatu enkripsi yang dapat di digunakan dalam menjaga keamanan *file*. Pada algoritma AES terdapat kelebihan salah satunya adalah sifat karakteristiknya yang di mana selalu ada bidang khusus untuk bilangan prima, memicu kekuatan AES(Surbakti & Purwanto, 2024) . AES juga memiliki sebuah kelemahan salah satunya yaitu algoritma kriptografi yang menggunakan kunci simetris dalam proses pengiriman dan penerimanya data. Namun, penggunaan kunci simetris ini memiliki risiko lebih tinggi terhadap kebocoran, terutama seiring waktu (Hidayatulloh et al., 2023). Pada algoritma ChaCha20 terdapat keunggulan yaitu menggunakan teknik kriptografi ARX (*addition, rotation*, XOR) yang memberikan performa lebih cepat, kompleksitas lebih rendah, dan mampu menghilangkan serangan waktu Klik atau ketuk di sini untuk memasukkan teks.

Meskipun algoritma AES dan Chacha20 telah efektif untuk menjaga peningkatan keamanan *file* dan implementasinya yang cukup rumit sehingga membuat orang yang tidak memiliki akses tidak dapat mengakses *file* tersebut. Hal ini tidak dapat kita pungkiri bahwa semakin pesatnya dunia teknologi membuat banyak orang berbondong-bondong menciptakan pembaruan yang lebih efektif. Metode *double encrypt* ini menawarkan kombinasi yang mampu mengurangi kelemahan dari setiap algoritma jika digunakan secara bersamaan, serta meningkatkan ketahanan terhadap berbagai jenis serangan kriptografi. *Double encrypt* merupakan Teknik kriptografi yang melibatkan proses enkripsi secara dua kali, yang di mana memiliki dua kunci yang berbeda untuk lebih meningkatkan keamanan. *Double encrypt* ini bertujuan untuk meningkatkan keamanan *file* lebih tinggi dibandingkan hanya 1 metode saja (Pangestu et al., 2022).

Pada penelitian terlebih dahulu yang berjudul “Implementasi Sistem Keamanan *File* Menggunakan Algoritma AES untuk Mengamankan *File* Pribadi” yang diteliti oleh Saripa Universitas Negeri Makassar. Dalam penelitian tersebut menjelaskan tentang keamanan *file* dengan menggunakan metode algoritma AES. Di mana hasil penelitian ini menunjukkan bahwa algoritma AES telah berhasil mengamankan *file* yaitu dokumen Word, PDF, Excel, dan Power Poin, *file* yang dienkripsi menjadi tidak terbaca oleh pihak yang tidak memiliki akses sehingga dapat menjaga kerahasiaan data dan AES mampu menjaga keamanan data dengan baik.

Dalam perbedaan penelitian terdahulu dan penelitian saya akan menggunakan double encrypt untuk keamanan *file* dan juga saya akan menggunakan AES dengan menambahkan metode ChaCha20 dan melakukan modifikasi pada bagian AES. Modifikasi menggunakan penambahan *AddRoundkey* sebelum dengan bertujuan agar proses enkripsi akan lebih susah untuk dipecahkan. Dalam algoritma AES, langkah *AddRoundKey* merupakan komponen krusial yang menggabungkan data dengan kunci enkripsi menggunakan operasi XOR. Penelitian (Luong, 2023) oleh menunjukkan bahwa memodifikasi transformasi *AddRoundKey*

dapat meningkatkan keamanan algoritma dengan menciptakan hubungan yang lebih kompleks antara *plainteks* dan *cipherteks*. Penelitian ini menyoroti pentingnya penyesuaian langkah *AddRoundKey* untuk meningkatkan ketahanan algoritma AES terhadap serangan kriptografi, seperti *differential cryptanalysis* dan linear *cryptanalysis*.

Dengan mendasarkan pada penelitian tersebut, modifikasi algoritma AES melalui penambahan langkah *AddRoundKey* sebelum *MixColumns* bertujuan untuk lebih memperkuat difusi data pada tahap awal enkripsi. Penambahan ini diharapkan dapat meningkatkan kerandoman hasil enkripsi dan mengurangi kemungkinan eksploitasi pola oleh pihak yang tidak berwenang, sehingga keamanan algoritma menjadi lebih optimal. Berdasarkan dari latar belakang di atas penelitian ini memiliki tujuan untuk meningkatkan keamanan *file* agar lebih sulit untuk diretas atau diakses pihak yang tidak berwenang. Oleh karena itu penelitian ini dianggap penting karena maraknya kasus peretasan data oleh oknum yang tidak memiliki akses.

### Rumusan Masalah

1. Bagaimana implementasi dari modifikasi AES dengan penambahan *AddRoundkey* sebelum *MixColumns* dalam meningkatkan keamanan *file*?
2. Bagaimana performa kombinasi antara AES modifikasi dan ChaCha20 dalam hal waktu enrkipsi, waktu *dekripsi* , kecepatan enkripsi, dan kecepatan *dekripsi*?

### Tujuan Penelitian

1. Mampu mengidentifikasi *AddRoundkey* yang akan digunakan dalam menjaga keamanan *file* dan mengimplementasi kemampuan Chacha20 dan AES yang telah dimodifikasi dalam proses menjaga keamanan *file*.
2. Mampu mengevaluasi performa algoritma ChaCha20 dan AES yang telah dimodifikasi berdasarkan waktu enkripsi, waktu *dekripsi*, kecepatan enkripsi, dan kecepatan *dekripsi*, untuk menentukan efisiensi

dan efektivitas kombinasi kedua algoritma tersebut dalam menjaga keamanan *file*

### Batasan dan Asumsi Penelitian

1. Penelitian ini berfokus pada jurnal atau artikel ilmiah dengan batasan tahun 2021-2024 yang membahas tentang :
   * Keamanan *file*
   * Algoritma ChaCha20
   * *Advanced Encryption Standard* (AES)
   * Double encryption
2. Penelitian dilakukan dengan pengujian pada *file* dengan ukuran dan format sebagai berikut :
   * Format PDF dan Word dengan variasi ukuran *file* 100 KB - 100MB.
   * Format *file* teks (TXT) dengan ukuran maksimal 300 byte Batasan ukuran *file* digunakan untuk memastikan pengujian dilakukan secara efektif tanpa mengurangi kinerja proses enkripsi dan *dekripsi*.

### Manfaat Penelitian

### Manfaat Akademis

* + - 1. Memberikan inovasi terbaru mengenai pelindungan keamanan

*file* dengan menambahkan *AddRoundkey* pada algoritma AES.

* + - 1. Memudahkan peneliti lainnya dalam kinerja *AddRoundkey* pada algoritma AES yang berfungsi untuk keamanan *file***.**

### Manfaat Ilmiah

* + - 1. Menjadi sumber acuan dan sumber referensi bagi peneliti lainnya yang akan meneliti topik ChaCha20 dan AES modifikasi, terutama dalam konteks keamanan data dan performa enkripsi.
      2. Membuktikan hipotesis apakah AES yang dimodifikasi dengan tambahan *AddRoundkey* dapat meningkatkan performa keamanan dan efisiensi dibandingkan dengan AES standart, dan sekaligus menunjukkan kinerja kombinasi ChaCha20 dan AES dalam pengamanan *file* berbagai format dan ukuran.

### Sistematika Penulisan

## BAB 1 PENDAHULUAN

Menjelaskan latar belakang penelitian, rumusan masalah, tujuan penelitian, batasam penelitian, manfaat penelitian, serta sistematika penulisan secara umum.

## BAB 2 LANDASAN TEORI

Membahas teori-teori yang relavan dengan penelitian, termasuk algoritma ChaCha20, AES, konsep kriptografi, Serta tinjauan terhadap penelitian terdahulu yang terkait.

## BAB 3 METODOLOGI PENELITIAN

Menjelaskan metode yang digunakan dalam penelitian, termasuk algoritma yang diterapkan (modifikasi AES dan ChaCha20 dengan enkripsi ganda). Diuraikan juga alat dan bahan yang digunakan, serta tahapan penelitian yang meliputi implementasi algoritma, pengujian parameter kinerja (waktu dan kecepatan enkripsi, dekripsi, serta analisis entropi), dan evaluasi hasil pengujian pada berbagai jenis file (PDF, Word, TXT).

## BAB 4 IMPLEMENTASI DAN PENGUJIAN

Menyajikan hasil implementasi algoritma yang dikembangkan, pengujian algoritma terhadap berbagai jenis *file*, serta analisis data hasil pengujian**.**

## BAB 5 EVALUASI HASIL PENGUJIAN

Menyajikan hasil implementasi dari aplikasi yang telah dirancang, pengujian fungsionalitas sistem, serta evaluasi kinerja aplikasi terhadap berbagai skenario penggunaan. Analisis data hasil pengujian juga dilakukan untuk menilai keberhasilan aplikasi dalam mencapai tujuan penelitian.

## BAB 6 KESIMPULAN DAN SARAN

Merangkum temuan utama penelitian serta memberikan saran untuk pengembangan lebih lanjut di masa depan.

# BAB II LANDASAN TEORI

### Literatur Terkait Teori

Keamanan informasi menjadi salah satu isu yang paling mendesak di era digital saat ini, di mana data sensitif sering kali menjadi target serangan siber yang beragam. Dengan meningkatnya ancaman seperti serangan *man-in-the- middle*, *eavesdropping*, dan *DDoS*, penting bagi organisasi dan individu untuk menerapkan metode enkripsi yang efektif untuk melindungi data mereka. Berbagai algoritma enkripsi telah dikembangkan untuk memenuhi kebutuhan ini, dengan *Advanced Encryption Standard* (AES) dan ChaCha20 menjadi dua di antara yang paling banyak digunakan. Penelitian-penelitian terkini menunjukkan bahwa kedua algoritma ini tidak hanya menawarkan tingkat keamanan yang tinggi, tetapi juga dapat diimplementasikan dalam berbagai konteks, mulai dari pengamanan *file* hingga enkripsi konten *livestreaming*.

Pada [Tabel II.1 Literatur Penelitian Terkait Kriptografi](#_bookmark19) menyajikan ringkasan dari beberapa penelitian yang mengeksplorasi efektivitas dan penerapan algoritma AES dan ChaCha20 dalam berbagai aplikasi. Setiap penelitian memberikan wawasan tentang tujuan, hasil, dan persamaan dalam pendekatan yang digunakan, serta menyoroti inovasi yang dilakukan untuk meningkatkan keamanan data. Dengan memahami hasil-hasil ini, kita dapat lebih menghargai perkembangan teknologi enkripsi dan aplikasinya dalam menjaga kerahasiaan serta integritas informasi di dunia yang semakin terhubung.

### Kajian Teori

### Kriptografi

Menurut Menezes, Oorschot, dan Vantone (1996), Kriptografi adalah studi teknik matematika yang berkaitan dengan keamanan informasi seperti kerahasiaan, *autentikasi* keaslian data (Belvin Tannadi, 2022). Kriptografi dapat diartikan sebagai sebuah proses untuk melindungi data atau sebuah sistem yang bertujuan untuk melindungi informasi rahasia dengan cara yang tidak dipahami oleh pihak yang tidak berwenang. Dalam kriptografi yang meliputi proses enkripsi dan proses *dekripsi*. Di dalam Kriptografi terdapat tiga fungsi dasar yang meliputi :

* + - 1. Enkripsi merupakan proses data asli (*plainteks*) yang akan diubah menjadi data yang tidak bisa terbaca oleh pihak yang tidak berwenang (*chiperteks*).
      2. *Dekripsi* merupakan proses pengembalian dari data yang tidak bisa terbaca(*chiperteks*) menjadi data terbaca atau data asli (*plainteks*).
      3. Kunci merupakan proses yang dipakai untuk melakukan proses enkripsi dan *dekripsi*, terdapat 2 jenis kunci, yaitu kunci rahasia (*private key*) dan kunci publik (*public key*).

Pada [Gambar II.1 Alur Proses Enkripsi Dan *Dekripsi*](#_bookmark22)

memperlihatkan alur enkripsi dan *dekripsi.* Proses ini dimulai

dengan *plainteks* yang kemudian dienkripsi dengan menggunakan kunci menjadi *chiperteks*. *Chiperteks* tersebut kemudian *didekripsi* kembali menggunakan kunci yang sama untuk mendapatkan kembali *plainteks*.

*Gambar II.1 Alur Proses Enkripsi Dan Dekripsi*

Berikut algoritma Kriptografi yang berdasarkan kunci :

a. Algoritma Simetris

Algoritma simetris adalah algoritma yang merujuk pada metode enkripsi dan *dekripsi* yang di mana algoritma ini memiliki kunci tunggal atau sama. Sehingga dapat disimpulkan algoritma ini mewajibkan pengirim dan penerima harus menyetujui satu kunci yang digunakan dalam proses enkripsi dan *dekripsi* data. Namun, kelemahan dari algoritma ini yaitu tergantung pada rahasia kunci. Apabila kunci tersebut tersebar maka akan dengan mudah pihak yang tidak memiliki akses dapat melakukan proses enkripsi dan *dekripsi* data secara mudah.

*Gambar II.2 Proses Kunci Simetris*

Pada [Gambar II.2 Proses Kunci Simetris](#_bookmark23) menjelaskan proses enkripsi dan *dekripsi* menggunakan kunci simetris. Dimulai dengan pesan yang dikirimkan, kemudian dienkripsi menjadi kode yang tidak bisa dibaca menggunakan kunci simetris. Kode tersebut kemudian dikirim dan di *dekripsi* di sisi penerima menggunakan kunci simetris yang sama, sehingga pesan dapat dibaca kembali. Pada contoh gambar, pesan "Selamat Datang!" dienkripsi menjadi kode "9CDF4837 AB672019" menggunakan kunci simetris yang sama. Kemudian di *dekripsi* kembali ke pesan awal "Selamat Datang!" Kunci simetris merupakan kunci yang sama digunakan untuk enkripsi dan *dekripsi*, sehingga pengirim dan penerima pesan harus memiliki kunci yang sama agar dapat berkomunikasi.

### AES (Advanced Encryption Standard)

Advanced Encryption Standard (AES) adalah salah satu algoritma enkripsi yang paling terkenal dan banyak digunakan, serta terus dikembangkan sebagai pengganti algoritma Data Encryption Standard (DES). AES adalah algoritma enkripsi yang bekerja dengan memproses data dalam blok dengan ukuran minimal 128 bit, dan melalui beberapa tahapan untuk mengenkripsi data. Tujuan utama dari AES adalah untuk mengamankan data rahasia sehingga tidak dapat dibaca oleh pihak yang tidak berwenang (Wahyu Putra, 2023). Pada [Tabel II.2 Panjang Kunci](#_bookmark25) dalam proses enkripsi *Advanced Encryption Standard* (AES) menggunakan ukuran blok dengan panjang sebesar 126 bit, dengan pilihan panjang kunci yang bervariasi, yaitu 128 bit, 192 bit, dan 256 bit (Sekar Putri Ananda et al., 2022). Proses enkripsi dalam algoritma ini dilakukan melalui beberapa tahap yang dikenal sebagai ronde. Jumlah ronde yang digunakan bergantung pada panjang kunci yang akan dipilih. Berikut adalah hubungan panjang kunci dan banyaknya ronde.

Algoritma AES juga menawarkan tingkat keamanan yang sangat tinggi dikarenakan ukuran kunci dan jumlah ronde yang ditentukan. Ukuran kunci menentukan jumlah kemungkinan kombinasi kunci, sehingga membuat serangan *bruteforce* menjadi sangat sulit dilakukan. Di sisi lain, jumlah ronde yang besar dapat menghasilkan keamanan yang kuat, sehingga perubahan kecil pada *plainteks* dapat memengaruhi seluruh bagian *cipherteks*. Selain keunggulan keamanannya, AES juga memiliki efisiensi tinggi dalam hal waktu dan ruang. Algoritma ini dapat diimplementasikan baik pada perangkat keras maupun perangkat lunak, menjadikannya ideal untuk berbagai aplikasi, termasuk perlindungan data (Assyahid, 2024). Pada tahap awal enkripsi, data masukan yang telah disalin ke dalam matriks *state* akan menjalani transformasi *AddRoundkey*. Selanjutnya, matriks *state* akan melalui proses transformasi secara berulang, mencakup *SubBytes*, *ShiftRows*, *MixColumns*, dan *AddRoundkey*, dengan jumlah pengulangan sesuai dengan nilai Nr (Handoyo & Subakti, 2020). *Dekripsi* adalah proses mengembalikan *ciphertext* menjadi plaintext, atau mengubah data terenkripsi kembali ke bentuk aslinya. Pada proses *inverse cipher*, transformasi yang digunakan meliputi *InvShiftRows*, *InvSubBytes*, *InvMixColumns*, dan *AddRoundkey*. Berikut ini adalah cara kerja algoritma AES secara umum:

### Proses Enkripsi AES Modifikasi

* + - * 1. *AddRoundkey*

*AddRoundkey* adalah proses menggabungkan state masukan dengan kunci ronde dengan menggunakan operasi XOR. Dalam proses transformasi enkripsi dan *dekripsi* pada AES, dilakukan operasi penambahan kunci putaran (*round key*)

pada state menggunakan operasi XOR (Djong & Siswanto, 2022). Setiap kunci putaran terdiri dari *Nb word* di mana setiap *word* tersebut akan digabungkan dengan *word* atau kolom yang sesuai pada *state* . Secara matematis, hal ini (Putri et al., 2021).

Keterangan Algoritma dari *AddRoundkey* di mana :

Pada [Gambar II.3.Transformasi ShiftRows](#_bookmark27) di atas di mana S0,0 merupakan elemen di baris ke-0, kolom ke-0, S3,3 merupakan elemen di baris ke-3, kolom ke-3, dan 𝑊3,3 merupakan elemen di baris ke-3, kolom ke-3. Pada label S0,0 dan lainnya merupakan state saat ini, state ini representasi internal data yang sedang dienkripsi ataupun *didekripsi* dan setiap elemen adalah *byte* dari blok data input. Sedangkan pada label 𝑊0,0 merupakan kunci ronde yang dihasilkan dari proses *keyexpansion*, di mana ini bagian dari kunci utama yang dipecah untuk menjadi beberapa ronde dan setiap elemen adalah *byte* dari kunci untuk ronde tertentu. Operasi XOR dilakukan antara setiap *byte* pada *state* dengan *byte* kunci ronde yang sesuai. Hasil di atas dihasilkan dari operasi XOR yang akan digunakan untuk langkah transformasi berikutnya.

* + - * 1. *SubBytes*

Transformasi *transformasi byte* di mana setiap elemen dalam matriks *state* dipetakan menggunakan tabel substitusi yang dikenal sebagai S-Box. Setiap *byte* dalam matriks *state* direpresentasikan sebagai S’[r, c], yang merupakan elemen hasil dari perpotongan antara baris (x) dan kolom (y) (Kurnia Mahesa Naibaho et al., 2021). S-Box berupa matriks

berukuran 16×16 yang berisi nilai-nilai heksadesimal di mana nilai-nilai ini diperoleh melalui perhitungan tertentu (Hena & Kusumaningsih, 2024). Berikut ini tabel S-Box :

Pada [Tabel II.3 Tabel Substitusi (S-Box)](#_bookmark28) sebagai contoh, nilai heksadesimal {62} merujuk ke baris 7 dan kolom 3 dalam S-Box, yang berisi nilai {aa}. Dengan demikian, nilai

{62} dipetakan menjadi nilai {aa}.

* + - * 1. *ShiftRows*

Transformasi *ShiftRows* merupakan proses pergeseran *byte* di mana *byte* paling kiri dipindahkan ke posisi paling kanan (rotasi bit). Jumlah pergeseran dalam melakukan *ShiftRows* ini dilakukan perbedaan setiap baris dalam matriks *state*. Pada baris pertama, tidak ada pergeseran yang dilakukan. Kemudian pada baris kedua barulah mengalami pergeseran satu *byte* ke kiri, lalu baris ketiga digeser dua *byte* ke kiri, dan baris keempat digeser tiga *byte* ke kiri. Tujuan dari proses ini adalah untuk menciptakan *diffusion*(), yang memiliki tujuan untuk membantu menyebarkan efek transformasi non-linear ke seluruh baris dalam matriks *state* untuk proses selanjutnya (Cristy & Riandari, 2021). Proses pergeseran ini bertujuan untuk mencampur elemen-elemen dalam *state* sehingga memperkuat kerahasiaan data.

*MixColumns* adalah proses transformasi di mana setiap elemen dalam blok *cipher* dikalikan dengan matriks . Transformasi ini bertujuan untuk mencampur elemen- elemen dalam setiap kolom *state* berguna meningkatkan difusi data dan memperkuat keamanan enkripsi . Transformasi ini bekerja dengan membagi matriks atau *state* menjadi empat bagian, untuk setiap kolomnya (Widyawan & Imelda, 2021). Setiap kolom dalam matriks *state* diperlakukan sebagai *polinomial* dengan empat bagian dalam *Galois Field* (GF), pada persamaan [(II-2)](#_bookmark30) dilakukan Algoritma dari *AddRoundkey* di mana :

𝑋8 = Elemen dengan derajat ke-8 yang akan muncul sebagai 100000000 (1 diposisi ke 9 dari kanan) di bilangan biner.

X = Menunjukkan elemen dengan derajat ke 1 yang akan muncul sebagai 00000010 ( 1 di posisi ke 2 dari kanan

) dalam bilangan biner

1 = Elemen konstanta yang akan muncul sebagai

00000001 ( 1 diposisi ke 1 dari kanan), dimana elemen 1 ini menyatakan bahwa *polinomial* ini memiliki koefisien konstan.

Proses *MixColumns* juga dapat direpresentasikan sebagai operasi perkalian matriks, di mana setiap kolom matriks dikalikan dengan matriks tertentu dalam *Galois Field* (Cristy & Riandari, 2021).

* + - 1. **Proses *dekripsi* AES :**

Proses *dekripsi* merupakan kebalikan dari proses enkripsi. *Transformasi bytes* yang digunakan dalam *dekripsi* mencakup *AddRoundKey*, *InvShiftRows*, *InvSubBytes,* dan *InvMixColumns*. Pada iterasi pertama *dekripsi*, dilakukan *transformasi AddRoundKey, Inverse ShiftRows, dan Inverse SubBytes. Ciphertex*t kemudian akan mengalami transformasi *AddRoundKey* .

* + - * 1. *InvShiftRows*

Transformasi *byte* yang merupakan kebalikan dari proses *ShiftRows*. Dalam transformasi ini, bit-bit pada *state* digeser ke arah kanan, berbeda dengan *ShiftRows* yang melakukan pergeseran bit ke arah kiri.

* + - * 1. *InvSubBytes*

Transformasi *byte* yang menjadi kebalikan dari *SubBytes*. Dalam proses ini, setiap elemen pada *state* dipetakan menggunakan tabel substitusi *Inverse* S-Box.

* + - * 1. *InvMixColumns*

Setiap kolom dalam *state* dikalikan dengan matriks khusus yang digunakan dalam proses perkalian pada algoritma AES (Syahputra Sianipar et al., 2024)

*Gambar II.4 Proses AES*

Pada [Gambar II.4 Proses AES](#_bookmark31) sebelah kiri menunjukkan tahapan proses enkripsi pada algoritma AES Modifikasi. Proses dimulai dengan memasukkan plainteks, kemudian melalui serangkaian langkah seperti *AddRoundkey*, SubBytes, ShiftRows, MixColumns, dan diakhiri dengan *AddRoundkey* tambahan. Langkah ini diulang sebanyak Nr−1 ronde untuk meningkatkan keamanan data, menghasilkan cipherteks di akhir proses. Flowchart sebelah kanan menjelaskan tahapan proses *dekripsi*, yang merupakan kebalikan dari proses enkripsi. Proses dimulai dengan memasukkan cipherteks, kemudian melalui langkah-langkah seperti *AddRoundkey*, Inverse ShiftRows, Inverse SubBytes, dan Inverse MixColumns. Setelah semua ronde selesai, data dikembalikan ke bentuk plainteks asli.

### Chacha20

Algoritma ChaCha20 adalah salah satu jenis *stream cipher* yang dikembangkan sebagai varian dari algoritma Salsa20. Algoritma ini berbasis desain yang serupa dengan Salsa20, namun ChaCha20 meningkatkan tingkat *diffusion* pada setiap ronde untuk memperkuat keamanan proses enkripsi (Anindita & 18219086, 2023). Chacha20 beroperasi pada unit data 32-bit menggunakan kunci 256-bit. Algoritma ini menghasilkan *output* berupa *stream* kunci sepanjang 512-bit, yang kemudian akan di XOR kan dengan *plainteks* untuk menghasilkan *chiperteks*(Alyas & Abdullah, 2021b). ChaCha20 dimulai dengan membuat *state* 512-bit, yang berupa *array* yang berukuran 16x32-bit. State ini disusun dari empat konstanta 32 bit, kunci rahasia sepanjang 256-bit, penghitung blok 32-bit, serta nonce Ini menunjukkan struktur *state* ChaCha20 yang terdiri dari matriks 4x4 dengan 16 elemen 32-bit. Matriks ini diinisialisasi dengan empat konstanta 32-bit, kunci 256-bit yang dibagi menjadi 8 bagian, penghitung 32 blok 32-bit, dan Nonce 96-bit. Elemen-elemen ini digunakan untuk menghasilkan *keystream* yang kemudian di XOR kan dengan *plainteks* untuk menghasilkan *chiperteks*. Di mana *const* (konstanta) ini berisi empat nilai tetap 32-bit, *key* (kunci) yang terbagi 8 bagian 32-bit dari kunci rahasia 256-bit, *counter* (penghitung), dan Nonce (nilai acak) yang berukuran 96-bit.

Langkah – Langkah Algoritma ChaCha20 :

* + - 1. *Inisialisasi* State

Algoritma ChaCha20 dimulai dengan membangun sebuah state internal berbentuk matriks 4x4 (32-bit) (Barbero et al., 2022) . Matriks ini terdiri dari:

* + - * 1. Baris pertama: Berisi 4 konstanta tetap (128-bit), yang digunakan untuk mengenali algoritma.
        2. Baris kedua dan ketiga: Berisi kunci rahasia 256-bit (8 elemen 32-bit), yang digunakan untuk proses enkripsi.
        3. Baris keempat: Berisi kombinasi 64-bit counter dan 64-bit nonce (masing-masing terdiri dari 2 elemen 32-bit).
      1. *Quarter Round*

Operasi ini memodifikasi 4 elemen dari matriks ( a, b, c, d ) (Almukhlifi & Vora, 2020)

𝑎 += 𝑏; 𝑑 ^ = 𝑎; 𝑑 <<<= 16;

𝑐 += 𝑑; 𝑏 ^ = 𝑐; 𝑏 <<<= 12;

𝑎 += 𝑏; 𝑏 ^ = 𝑎; 𝑑 <<<= 8;

𝑐 += 𝑑; 𝑏 ^ = 𝑐; 𝑏 <<<= 7;

* + - 1. *Double Round*

*Double Round* adalah kombinasi dari dua *Quarter Round* yang dijalankan secara berurutan. Setiap iterasi *Double Round* mencakup:

* + - * 1. Operasi pada kolom matriks *(Column Round):*

melakukan operasi Quarter Round pada setiap kolom matriks 4x4. Tujuannya adalah untuk mencampur data antar elemen dalam kolom, sehingga meningkatkan difusi data secara vertikal.

* + - * 1. Operasi pada diagonal matriks (Diagonal Round):

Setelah operasi pada kolom selesai, langkah berikutnya adalah menerapkan Quarter Round pada elemen-elemen diagonal dalam matriks.

Proses Double Round dilakukan sebanyak 10 iterasi, menghasilkan total 20 ronde Quarter Round untuk setiap blok data yang diproses.

* + - 1. Finalisasi State
      2. *Keystream Generation*

Setelah 20 ronde selesai*, state* akhir ditambahkan kembali ke

### Entropi

Entropi dalam teori informasi digunakan untuk mengukur keacakan data, yang mencerminkan kekuatan algoritma enkripsi. Semakin tinggi nilai entropi, semakin acak data tersebut, yang berarti algoritma enkripsi lebih tahan terhadap serangan (Wahyudi et al., 2024). Entropi dihitung dengan menggunakan rumus pada

Dimana:

* 𝐻𝑚 : Menunjukkan entropi data, dimana tingkat keacakan dalam

# BAB III METODOLOGI PENELITIAN

### Sistematika Penyelesaian Masalah

Penelitian ini bersifat kuantitatif dengan tujuan mengevaluasi performa algoritma AES modifikasi dan kombinasi ChaCha20 dan AES modifikasi. Penilaian performa ini dilakukan berdasarkan waktu enkripsi, waktu *dekripsi*, kecepatan enkripsi, dan kecepatan *dekripsi*, yang akan dianalisis menggunakan data hasil pengujian pada berbagai jenis *file* dengan format PDF, Word dan TXT, yang hasilnya berbentuk tabel untuk mempermudah analisis perbandingan.

### Alat dan Bahan

### Alat

1. Perangkat Keras (Hardware) Laptop dengan spesifikasi :
   1. Prosesor : Intel *Core* I5
   2. RAM : 8 GB
   3. Penyimpanan : 256 GB SSD atau lebih
2. Perangkat Lunak (*Software*)
   1. *Python* :

Menggunakan *library cryptography, pycryptodome*, dan *library* lain yang mendukung algoritma AES dan ChaCha20. Berikut beberapa pustaka yang digunakan :

* + - PyCryptodome : Digunakan Iplementasi algoritma AES dan ChaCha20
    - Cryptography : Implementasi kemanan tambahan (jika nanti dibutuhkan )
    - Numpy : Digunakan untuk analisis data (perhitungan entropi)
    - Time : Mencatat waktu eksekusi algoritma, baik untuk enkripsi maupun dekripsi.
  1. *Text* Editor :

Visual Studio Code dan *Jupyter Notebook*

* 1. Sistem Operasi : Windows

### Bahan

1. Data set *file* :
   1. *File* PDF : Ukuran *file* 100 KB – 100 MB
   2. *File* Word : Ukuran *file* 100 KB – 100 MB
   3. *File* TXT : Ukuran *file* maksimal 300 *byte*
2. Literatur :

Jurnal atau artikel ilmiah tentang AES, ChaCha20 dan *Double Encrypt* (tahun 2020-2024)

1. *Spreadsheet* Analisis Data :

Format CSV (Comma Separated Values)

### Rancangan Penelitian

Pada penelitian ini, dilakukan tahapan yang dirancang secara sistematis untuk mencapai tujuan utama, yaitu Implementasi kombinasi Advanced Encryption Standard (AES) yang dimodifikasi dan ChaCha20. Proses penelitian dimulai dengan studi literatur untuk memahami konsep dasar dan pengembangan terbaru terkait algoritma kriptografi. Setelah itu, dilakukan pengumpulan data yang relevan sebagai dasar implementasi. Tahapan selanjutnya mencakup implementasi algoritma modifikasi AES serta kombinasi algoritma tersebut dengan ChaCha20. Untuk mengukur efektivitas dan efisiensi algoritma yang dikembangkan, pengujian dilakukan pada berbagai jenis *file* seperti PDF, Word, dan TXT. Analisis hasil dari pengujian ini memberikan dasar untuk menarik kesimpulan dan memberikan rekomendasi untuk pengembangan lebih lanjut.

*Gambar III.1 Diagram Alur Penelitian*

[Gambar III.1 Diagram Alur Penelitian](#_bookmark44) ini terdapat beberapa tahapan yaitu:

1. Mulai adalah tahap awal dari penelitian. Pada tahap ini menentukan tujuan dan menyusun langkah-langkah yang akan dilakukan dalam penelitian.
2. Studi literatur dilakukan pengumpulan referensi yang berkaitan dengan algoritma AES, ChaCha20 dan *Double Encrypt. Referensi* berupa jurnal, artikel ilmiah.
3. Pengumpulan data untuk mengumpulkan *file* atau data set yang akan digunakan untuk pengujian algoritma. Format *file* yang digunakan adalah PDF, Word dan TXT.
4. Implementasi modifikasi AES, pada tahap ini algoritma AES dimodifikasi dengan menambahkan langkah *AddRoundkey* sebelum *Mixcolumns*. Tujuannya adalah untuk meningkatkan keamanan dan kecepatan enkripsi dibandingkan dengan AES standar.
5. Implementasi kombinasi AES modifikasi dan ChaCha20, menggabungkan algoritma AES yang sudah dimodifikasi dengan ChaCha20 untuk melakukan *double encypt*. Di mana proses ini bertujuan untuk meningkatkan keamanan *file* lebih baik daripada menggunakan *single encrypt*.
6. Pengujian algoritma, untuk menguji performa algoritma yang sudah diimplementasikan, di mana data set yang dikumpulkan sebelumnya itu digunakan untuk uji coba. Parameter yang diukur pada pengujian ialah

waktu enkripsi dan *dekripsi,* kecepatan enkripsi dan *dekripsi*, dan hasil pengujian ini akan digunakan untuk mengevaluasi kinerja algoritma.

1. Analisis hasil, data pengujian yang sudah terkumpul akan dianalisis. Analisis dilakukan untuk membandingkan performa AES standar, AES yang sudah dimodifikasi, dan kombinasi AES modifikasi dan ChaCha20. Tujuannya sendiri untuk mengetahui algoritma mana yang lebih efisien.
2. Selanjutnya ke kesimpulan dan saran, pada tahap ini merangkum hasil penelitian termasuk kelebihan dan kekurangan algoritma yang sudah diuji. Saran juga untuk pengembangan penelitian lebih lanjut
3. Selesai

### Studi Literatur

Studi literatur dilakukan untuk memahami konsep dasar dan perkembangan terbaru terkait algoritma kriptografi yang digunakan dalam penelitian ini, yaitu Advanced Encryption Standard (AES) dan ChaCha20. Proses ini melibatkan pengumpulan referensi dari jurnal, paper, dan buku yang membahas metode enkripsi, keamanan data, serta teknik evaluasi seperti entropi dan analisis performa. Literatur yang dikaji juga mencakup penelitian sebelumnya tentang algoritma kriptografi untuk keamanan *file*, sehingga memberikan dasar teori yang kuat untuk modifikasi algoritma dan penerapan metode double encryption dalam penelitian ini.

### Pengumpulan Data

Data yang digunakan dalam penelitian ini dibuat sendiri dengan mengikuti ketentuan yang telah ditentukan. Format data meliputi *file* PDF, Word, dan TXT dengan ukuran bervariasi, mulai dari 100 KB hingga 100 MB untuk *file* PDF dan Word, serta hingga 300 *byte* untuk *file* TXT. Data ini dirancang sedemikian rupa agar sesuai dengan kebutuhan pengujian algoritma enkripsi, seperti mengukur waktu enkripsi, waktu *dekripsi*, kecepatan, dan tingkat keacakan menggunakan analisis entropi. Variasi format dan ukuran data ini

bertujuan untuk memastikan bahwa pengujian mencakup berbagai jenis isi *file*.

### Implementasi Modifikasi AES

Pengembangan algoritma *Advanced Encryption Standard* (AES) dilakukan dengan memodifikasi. Modifikasi dilakukan dengan menambahkan *AddRoundkey* sebelum *Mixcolumns*. Penambahan ini bertujuan untuk memperkuat keamanan dan meningkatkan kompleksitas pada proses enkripsi, sehingga lebih sulit untuk diretas dengan pihak tidak berwenang. Berikut adalah proses alur dari modifikasi algoritma AES.

*Gambar III.2 Diagram Proses Enkripsi AES*

Pada [Gambar III.2 Diagram Proses Enkripsi AES](#_bookmark48) ini digunakan untuk mengamankan *file* dengan melakukan tahap enkripsi terhadap data *plainteks* yang menggunakan kunci tertentu. Dalam penelitian ini, modifikasi dilakukan dengan menambahkan langkah *AddRoundkey* untuk meningkatkan tingkat keamanan dan kompleksitas enkripsi. Berikut langkah-langkah proses modifikasi AES .

1. Input *file* dan kunci

Proses enkripsi dimulai dengan memasukkan *file* input (*file* asli yang akan dienkripsi) dan kunci enkripsi. *File* input dibaca dalam mode biner, diubah menjadi *byte array*, lalu dipecah menjadi blok-blok data berukuran 16 *byte*. Jika panjang data tidak habis dibagi 16 *byte, padding* dilakukan dengan metode PKCS#7 untuk melengkapinya.

Contoh :

* 1. *File* input : contoh.pdf

Isi *file* dalam biner : [0x25, 0x50, 0x44, 0x46, ...]

*Byte array* : [37, 80, 68, 70, 10, 0, 45, 78, ...]

* 1. Kunci enkripsi :

Kunci : surabayaitelkom

ASCII : [115, 117, 114, 97, 98, 97, 121, 97, 105, 116, 116,

101, 108, 107, 111, 109]

* 1. Matriks data *file* :

|  |  |  |  |
| --- | --- | --- | --- |
| [37, | 80, | 68, | 70] |
| [10, | 0, | 45, | 78] |
| [255, | 100, | 32, | 12] |
| [80, | 50, | 60, | 72] |

1. *AddRoundkey*

Merupakan tahap awal proses enkripsi, di mana *plainteks* ini digabungkan dengan kunci awal yang menggunakan operasi logika yaitu XOR. Di mana blok data *file* digabungkan dengan kunci awal (W0-W3) menggunakan operasi XOR untuk menambahkan lapisan keamanan awal.

Contoh :

* 1. Proses XOR :

37 XOR 115 = 86

80 XOR 117 = 37

68 XOR 114 = 54

70 XOR 97 = 39

* 1. Matriks setelah *AddRoundkey* :

[86, 37, 54, 39]

[108, 97, 76, 47]

[150, 16, 84, 97]

[28, 89, 83, 37]

1. Proses ronde 1-9
   1. *SubBytes*

Pada proses ini, nantinya setiap *byte* pada matriks *state* ( data yang diperoleh *AddRounKey*) akan diubah menggunakan tabel substitusi yang biasa disebut dengan S-Box. *Tranformasi* ini bertujuan untuk meningkatkan keacakan data agar sulit diprediksi.

Contoh proses :

*Byte* 37 diganti dengan nilai dari tabel S-Box jadi 120

* 1. *ShiftRows*

Setiap baris dalam matriks *state* nantinya digeser ke kiri pada baris pertama tidak ada pergeseran, baris kedua di geser 1 *byte* ke kiri, baris ketiga ini digeser 2 *byte* ke kiri, dan terakhir baris keempat yang nantinya digeser 3 *byte* ke kiri. Contoh proses :

[244, 120, 99, 200] [244, 120, 99, 200]

[60, 107, 55, 175] [107, 55, 175, 60]

[12, 243, 172, 85] [172, 85, 12, 243]

[180, 97, 92, 65] [65, 180, 97, 92]

* 1. *AddRoundkey* (W4-W7)

Setelah *ShiftRows* dilakukan operasi XOR antara matriks *state* dengan kunci ronde berikutnya yaitu (W4-W7) Contoh proses :

*Byte* 244 XOR 115 = 135

*Byte* 120 XOR 117 = 13

Hasil *AddRounKey* : [135, 13, 17, 169]

[38, 98, 85, 12]

[55, 32, 82, 84]

[49, 53, 180, 193]

* 1. *MixColumns*

Setiap kolom pada matriks *state* dimodifikasi dengan operasi matematis berbasis matriks yang menggunakan bidang *Galois* (GF). Di mana proses ini bertujuan untuk mencampur elemen-elemen dalam kolom dan meningkatkan difusi data. [2, 3, 1, 1]

[1, 2, 3, 1]

[1, 1, 2, 3]

[3, 1, 1, 2]

* 1. *AddRoundkey* (W36-W39)

Dilakukan setelah *MixColums* untuk menambahkan lapisan keamanan dengan kunci ronde tambahan (W36-W39).

1. Pada ronde terakhir, hanya dilakukan tiga langkah yaitu

*SubBytes, ShiftRows*, dan *AddRoundkey* tanpa *MixColumns.*

Proses modifikasi algoritma AES ini digunakan untuk mengenkripsi *file* dengan menambahkan langkah *AddRoundkey* tambahan, yang meningkatkan keamanan dengan memberikan lapisan XOR tambahan pada setiap ronde enkripsi.

* + 1. **Implementasi Kombinasi AES Modifikasi dan ChaCha20** Algoritma AES modifikasi dan ChaCha20 dengan menerapkan *double encypt.* Proses ini memakai AES modifikasi dengan memberikan perlindungan berbasis blok *chiper* yang kuat dengan memodifikasi *AddRoundkey* tambahan sebelum *MixColumns,* sedangkan ChaCha20 sebagai *stream cipher* yang cepat dan aman, meningkatkan kompleksitas keseluruhan dengan memanfaatkan proses enkripsi berbasis *keystream.*

Pada metode ini, setiap *file* menggunakan 1 kunci enkripsi. Kunci yang digunakan memiliki panjang awal 128-bit, yang kemudian diperluas menjadi 256-bit pada ChaCha20 dengan menambahkan padding nol (0x00) jika panjangnya kurang dari 256-bit. Proses

dimulai dengan mengenkripsi *file* menggunakan AES Modifikasi untuk menghasilkan cipherteks awal dalam format matriks 4x4. Cipherteks ini kemudian diubah menjadi byte stream untuk kompatibilitas dengan ChaCha20. Selanjutnya, ChaCha20 melakukan proses enkripsi kedua dengan memanfaatkan kunci 256- bit, nonce sepanjang 96-bit, dan counter. Hasil akhir adalah cipherteks final yang lebih aman karena telah melalui dua lapisan enkripsi. Penggunaan double encryption dengan satu kunci untuk setiap *file* memberikan keamanan yang lebih tinggi, di mana kombinasi kedua algoritma menciptakan perlindungan yang kuat terhadap berbagai jenis serangan kriptografi.

*Gambar III.3 Diagram Proses Double Encrypt*

Pada [Gambar III.3 Diagram Proses Double Encrypt](#_bookmark50) menunjukkan langkah-langkah enkripsi ganda. Proses dimulai dengan mengenkripsi *plainteks* menggunakan AES modifikasi untuk menghasilkan *chiperteks* AES. *Chiperteks* ini kemudian dienkripsi ulang menggunakan ChaCha20, yang melibatkan matriks 4x4, proses *quarter-round,* dan operasi XOR dengan *keystream*. Algoritma AES modifikasi dan ChaCha20 menggunakan kunci yang sama, yaitu kunci AES sepanjang 128-bit. berikut adalah tahapan alur di atas :

1. Input *plainteks* dan *key* dan enkripsi AES modifikasi
2. Input *cipherteks* AES ke ChaCha20

*Chiperteks* yang dihasilkan dari proses enkripsi pertama diubah menjadi input untuk algoritma ChaCha20. Ini dilakukan untuk menerapkan enkripsi berlapis yaitu *double encryption,* sehingga keamanan data akan meningkat secara signifikan. *Chiperteks* AES yang biasanya dalam format matriks (4x4) ini diubah ke format *byte stream* untuk kompatibilitas dengan algoritma ChaCha20. Byte stream ini memungkinkan untuk memproses data, sesuai dengan *stream chiper*.

Contoh :

[115, 75, 189, 234, 201, 97, 58, 14, 245, 142, 12, 120, 53, 190,

201, 20] data ini sudah dalam bentuk *byte stream*.

1. Validasi panjang *key* (256-bit *Key*)

Ini untuk memastikan bahwa kunci enkripsi untuk ChaCha20 sesuai dengan spesifikasi algoritma, yaitu panjangnya harus tepat 256 bit (32 *byte*). Kunci ini untuk menghasilkan *keystream* yang beda dalam proses enkripsi. Jika panjang ke kurang dari 256 bit, ditambahkan *padding* berupa *byte* nol (0x00) untuk memenuhi panjang. Jika *key* yang lebih panjang, *key* akan dipotong hingga hanya menyisakan 256 bit. *Key* kemudian dibagi menjadi 8 bagian ( di mana masing-masing 32 bit) untuk digunakan dalam proses pembentukan matriks awal (*state* matriks)

Contoh :

* 1. *Key* input “surabayaittelkom”

[115, 117, 114, 97, 98, 97, 121, 97, 105, 116, 116, 101, 108,

107, 111, 109]

* 1. Panjang *key* adalah 16 *byte* (128 bit), sehingga tidak memenuhi syarat panjang 256 bit. Jadi ditambahkan *padding* nol (0x00) hingga mencapai 32 *byte:*

[115, 117, 114, 97, 98, 97, 121, 97, 105, 116, 116, 101, 108,

107, 111, 109, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

1. Membuat *Initial* State Matriks 4x4

Membentuk *state* matriks awal yang menjadi dasar operasi dalam ChaCha20. Matriks ini berisi kunci, *counter*, dan *nonce.* Matriks 4x4 dalam ChaCha20 terdiri 16 elemen (masing-masing 32 bit) Contoh :

* 1. 4 elemen konstanta tetap “*expand* 32-byte k”4 elemen 32 bit [1634760805, 857760878, 2036477234, 1797285236]
  2. Elemen kunci 256-bit dibagi menjadi 8 elemen 32-bit: [1936028945, 1885434721, 1701147239, 1647539057,

1225269481, 1226844037, 1214606441, 0]

* 1. 1 *Counter*: Nilai awal 0.
  2. 3 Elemen Nonce: Nilai *nonce* yang unik untuk sesi enkripsi: Contoh *nonce* : [12345678, 87654321, 11223344]
  3. Matriks yang dibentuk

[1634760805, 857760878, 2036477234, 1797285236]

[1936028945, 1885434721, 1701147239, 1647539057]

[1225269481, 1226844037, 1214606441, 0]

[0, 12345678, 87654321, 11223344]

Baris ke 1 konstanta, ke 2 *key,* ke 3 *key* dan ke 4 *counter* dan

*nonce*

1. Operasi Quarter Round (20 iterasi)

Melakukan proses pencampuran data di dalam matriks untuk meningkatkan kerandoman dan keamanan *keystream* yang dihasilkan. Quarter Round adalah inti dari algoritma ChaCha20. Quarter Round dilakukan pada 4 elemen tertentu dari matriks. Penjumlahan Modular (mod 2³²): Menambah elemen dalam matriks untuk mencampur data. Operasi XOR: Mencampur data antar elemen melalui operasi logika XOR. Rotasi Bit (ROTL): Menggeser bit elemen untuk mendistribusikan informasi secara acak. Setelah 20 iterasi (10 putaran double round), matriks akan diubah menjadi bentuk yang sangat acak.

1. *Counter* +=1

*Counter* bertambah untuk setiap blok data yang diproses, memastikan *keystream* yang dihasilkan unik untuk setiap blok. Setelah satu blok data selesai diproses, *counter* di matriks bertambah sebesar 1.

Contoh :

*Counter* baru : [1, 12345678, 87654321, 11223344]

1. XOR *keystream* dengan *chiperteks* AES

Menggabungkan *keystream* hasil ChaCha20 dengan *cipherteks AES* melalui operasi XOR untuk menghasilkan *final ciphertext*. XOR dipilih karena sifatnya yang sederhana namun aman dan mudah dibalik. *Keystream* yang dihasilkan dari matriks ChaCha20 di-XOR-kan dengan *cipherteks AES*.

Operasi XOR:

Jika *ciphertext* = *keystream* XOR *plaintext*, maka *dekripsi*

dilakukan dengan cara *plaintext* = *keystream* XOR *ciphertext*.

* + - 1. **Proses *Dekripsi* Double Encryption**

Proses *dekripsi* dilakukan untuk mengembalikan data yang telah terenkripsi ke bentuk aslinya atau *plainteks*. Di mana pada tahap ini, *chiperteks* hasil dari kombinasi algoritma AES modifikasi dan Chacha20 akan *didekrpsi* secara urut. Proses nantinya akan dimulai dengan *mendekripsi chiperteks* dengan ChaCha20 untuk mendapatkan kembali *chiperteks* AES modifikasi, kemudian dilanjutkan dengan *mendekripsi chiprteks* AES modifikasi untuk memperoleh *plainteks* asli.

Pada [Gambar III.4 Proses Dekripsi](#_bookmark51) Diagram alur di atas menjelaskan proses *dekripsi* dari kombinasi algoritma ChaCha20 dan modifikasi AES. Proses dimulai dengan memasukkan cipherteks hasil enkripsi sebagai input. Langkah pertama adalah men*dekripsi* cipherteks menggunakan algoritma ChaCha20 untuk menghasilkan keystream. Keystream ini kemudian dikombinasikan dengan cipherteks menggunakan operasi XOR untuk mendapatkan cipherteks AES.

Selanjutnya, proses ekspansi kunci dilakukan untuk menghasilkan kunci yang akan digunakan pada setiap ronde *dekripsi* AES. Proses *dekripsi* AES terdiri dari tiga tahapan utama: ronde terakhir (Round 10) yang melibatkan *AddRoundKey*, InvShiftRows, dan InvSubBytes; ronde utama (Round 9-1) yang melibatkan *AddRoundKey*, InvMixColumns, *AddRoundKey, InvShiftRows,* dan *InvSubBytes* tambahan; dan ronde awal (Round 0) yang menggunakan *AddRoundKey* (W0-W3). Hasil akhir dari seluruh proses ini adalah plainteks, yaitu data asli yang berhasil di*dekripsi*. Kombinasi kedua algoritma ini memastikan keamanan data yang lebih kuat melalui enkripsi ganda.

*Gambar III.4 Proses Dekripsi*

### Implementasi Antarmuka Pengguna (UI)

Antarmuka pengguna (UI) dirancang untuk mempermudah interaksi, berfungsi sebagai media interaktif yang memungkinkan pengguna untuk memilih file, memasukkan kunci enkripsi, serta memulai proses enkripsi dan dekripsi dengan mudah. Implementasinya menggunakan Flask sebagai framework backend untuk menangani

proses enkripsi dan dekripsi, sedangkan HTML digunakan untuk membuat form input file dan kunci enkripsi. CSS digunakan untuk mempercantik tampilan agar antarmuka lebih menarik, jelas, dan mudah digunakan oleh pengguna dalam proses enkripsi dan dekripsi file menggunakan kombinasi algoritma ChaCha20 dan AES yang telah dimodifikasi.

*Gambar III.5 Desain Antarmuka Pengguna Enkripsi dan Dekripsi*

Pada [Gambar III.5 Desain Antarmuka Pengguna Enkripsi dan](#_bookmark53) [Dekripsi](#_bookmark53) Gambar tersebut menampilkan desain antarmuka pengguna (UI) dari sebuah aplikasi berbasis web yang digunakan untuk enkripsi dan *dekripsi file*. Antarmuka ini memiliki beberapa elemen input pengguna untuk mengunggah *file* dan memilih opsi enkripsi atau *dekripsi*. Terdapat tiga kolom utama pada desain ini: pertama, Pilih *File*, untuk pengguna memilih *file* yang akan diproses. Kedua, Enkripsi atau *Dekripsi*, yang memberikan pilihan kepada pengguna apakah *file* yang dipilih akan dienkripsi atau di *dekripsi*, yang merupakan opsi penting untuk menentukan jenis operasi yang akan dilakukan pada *file* tersebut. Ketiga, Key, di mana pengguna diminta untuk memasukkan kunci yang diperlukan untuk proses enkripsi atau *dekripsi*. Kunci ini memainkan peran penting dalam menjaga keamanan data yang diproses. Di bagian bawah antarmuka, terdapat

tombol Proses yang akan menjalankan operasi sesuai dengan pilihan pengguna, baik itu enkripsi maupun *dekripsi*.

### Pengujian Algoritma

Menjelaskan pengujian yang dilakukan untuk mengevaluasi kinerja dan keamanan algoritma berdasarkan parameter seperti entropi, waktu enkripsi dan *dekripsi*, serta kecapatan.

### Pengujian Entropi

Pengujian kekuatan algoritma dilakukan dengan mengukur tingkat keacakan data hasil enkripsi menggunakan analisis entropi. Keacakan data merupakan indikator penting dalam menentukan seberapa baik algoritma enkripsi menyamarkan pola asli dari data. Semakin tinggi tingkat keacakan, semakin sulit bagi pihak yang tidak berwenang untuk menganalisis pola atau memprediksi isi data terenkripsi. Pengujian ini dilakukan dengan menghitung nilai entropi menggunakan distribusi probabilitas setiap simbol dalam cipherteks. Nilai entropi yang mendekati maksimum menunjukkan bahwa distribusi simbol dalam cipherteks hampir merata, menandakan bahwa algoritma telah berhasil menghasilkan data yang acak. Data dengan keacakan tinggi lebih tahan terhadap serangan berbasis statistik, sehingga pengujian ini memberikan evaluasi kuantitatif terhadap kekuatan keamanan algoritma yang diuji. Hasil pengujian dibandingkan antara AES Standar, AES Modifikasi, Chacha20 dan Double Encryption untuk menunjukkan efektivitas modifikasi dalam meningkatkan keamanan.

### Pengujian Waktu dan Kecepatan

Menguji algoritma AES modifikasi dan *Double encrypt* AES modifikasi dan ChaCha20. Pengujian ini dilakukan untuk menilai kinerja algoritma dalam berbagai parameter. Penelitian ini akan menganalisis parameter pengujian menggunakan matriks dan formula berikut :

* + - * 1. Waktu enkripsi :

mengukur waktu yang dibutuhkan untuk mengenkripsi data dari

*plainteks* menjadi *chiperteks* dijelaskan dalam persamaan [(III-1)](#_bookmark55).

|  |  |
| --- | --- |
| 𝐸𝑡 = 𝐸𝑁𝑡 – 𝑆𝑇𝑡 | (III-1) |

Di mana:

𝐸𝑡 = waktu enkripsi dalam mili detik (ms)

𝐸𝑁𝑡 = waktu selesai enkripsi

𝑆𝑇𝑡 = waktu mulai enkripsi

Waktu ini akan dihitung untuk setiap *file* yang diuji, dan hasilnya nanti akan memberikan gambaran mengenai kecepatan algoritma dalam melakukan proses enkripsi.

* + - * 1. Waktu *dekripsi* :

Mengukur waktu yang dibutuhkan untuk *mendekripsi* data dari

*chiperteks* menjadi *plainteks* dijelaskan pada persamaan [(III-2)](#_bookmark56).

Dimana:

𝐷𝑡 = waktu *dekripsi* (ms)

𝐸𝑁𝑡 = waktu selesai *dekripsi*

𝑆𝑇𝑡 = waktu mulai *dekripsi*

Hasil waktu *dekripsi* ini juga perlu untuk masing-masing algoritma karena ini menganalisis perbandingan kinerja antara algoritma enkripsi.

* + - * 1. Kecepatan :

Menunjukkan jumlah data dalam *kilobyte* yang dapat dienkripsi

Et = waktu enkripsi atau *dekripsi* (s)

### Pengujian pada Berbagai Jenis *File*

*File* PDF yang digunakan dalam pengujian dapat berisi elemen seperti teks, gambar, dan tabel yang semuanya akan dienkripsi secara utuh karena algoritma bekerja pada level data biner. *File* PDF dengan ukuran 100 KB, 1 MB, 10 MB, hingga 100 MB diuji untuk memastikan algoritma dapat bekerja pada dokumen berukuran kecil hingga besar. Hal serupa juga berlaku untuk *file* Word dengan ukuran yang sama, meskipun memiliki struktur internal yang lebih kompleks dibandingkan PDF. *File* TXT dengan ukuran maksimal 300 *byte*, yang biasanya hanya berisi teks sederhana, juga dapat dienkripsi ChaCha20 (*double encrypt*). Di mana pengujian ini bertujuan untuk

mengukur kinerja algoritma. Berikut ini adalah contoh hasil pengujian terhadap *file* PDF :

*Tabel III.2 Evaluasi Hasil*

Pada [Tabel III.2 Evaluasi Hasil](#_bookmark60) menunjukkan hasil evaluasi simulasi kinerja algoritma enkripsi dan *dekripsi*. pada Nilai-nilai tersebut memberikan gambaran awal mengenai performa beberapa metode, termasuk AES Standar, ChaCha20, AES Modifikasi, dan kombinasi AES Modifikasi dengan ChaCha20. Hasil ini digunakan sebagai contoh output yang diharapkan dari model algoritma yang akan dirancang, ini belum merupakan data dari eksperimen nyata.

# DAFTAR PUSTAKA

Alimuddin, A., Juntak, J. N. S., Jusnita, R. A. E., Murniawaty, I., & Wono, H. Y. (2023). Teknologi dalam pendidikan: Membantu siswa beradaptasi dengan revolusi industri 4.0. *Journal on Education*, *5*(4), 11777–11790.

Almukhlifi, R., & Vora, P. L. (2020). Linear cryptanalysis of reduced-round simon using super rounds. *Cryptography*, *4*(1), 1–34. https://doi.org/10.3390/cryptography4010009

Alyas, H. H., & Abdullah, A. A. (2021). Enhancement the ChaCha20 Encryption Algorithm Based on Chaotic Maps. *Lecture Notes in Networks and Systems*, *201*, 91–107. https://doi.org/10.1007/978-981-16-0666-3\_10

Anindita, M., & 18219086, A. A. (2023). *Analisis Perbandingan Algoritma AES dan ChaCha20-Poly1305 dalam Enkripsi Konten Livestreaming*.

Aryanto, M. B., Tahir, M., Devita, S. I., Mustofa, Z. N., Ainiyah, Q., & Sundoro,

S. (2023). Implementasi Enkrip Dan Dekrip File Menggunakan Metode Advance Encryption Standard (AES-128). *Jurnal Ilmiah Sistem Informasi Dan Ilmu Komputer*, *3*(1), 89–104.

Assyahid, M. H. (2024). *Implementasi interpolasi pixel wadah citra dan enkripsi aes pada steganografi least significan bit (LSB)*. https://repository.uinjkt.ac.id/dspace/handle/123456789/76727

Barbero, S., Bazzanella, D., & Bellini, E. (2022). Rotational Cryptanalysis on ChaCha Stream Cipher. *Symmetry*, *14*(6). https://doi.org/10.3390/sym14061087

Belvin Tannadi. (2022). *Ilmu Crypto* (Digital). Jakarta : Elex Media Komputindo.

Cristy, N., & Riandari, F. (2021). *Niolinda Cristy 1 , Fristi Riandari 2 [Implementasi Metode Advanced Encryption Standard (AES 128 Bit) Untuk Mengamankan Data Keuangan*. *4*(2), 75.

Djong, H. S., & Siswanto, S. (2022). Implementasi Kriptografi Dengan Menggunakan Metode RC4 dan AES-256 Untuk Mengamankan File Dokumen pada PT Varnion Technology Semesta. *Prosiding Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI)*, *1*(1), 149–158.

Handoyo, J., & Subakti, Y. M. (2020). Keamanan Dokumen Menggunakan Algoritma Advanced Encryption Standard (AES). *Jurnal SITECH : Sistem Informasi Dan Teknologi*, *3*(2), 143–152. https://jurnal.umk.ac.id/index.php/sitech/article/view/5865

Hena, N., & Kusumaningsih, D. (2024). Pengamanan File Pada Sistem Master Vendor Berbasis WEB Menggunakan Algoritma AES Pada Trinitiland. *Prosiding Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI)*, *3*(2), 30–37.

Hidayatulloh, N. W., Tahir, M., Amalia, H., Basyar, N. A., Prianggara, A. F., & Yasin, M. (2023). Mengenal Advance Encryption Standard (AES) sebagai Algoritma Kriptografi dalam Mengamankan Data. *Digital Transformation Technology*, *3*(1), 1–10.

Kebande, V. R. (2023). Extended-Chacha20 Stream Cipher With Enhanced Quarter Round Function. *IEEE Access*, *11*, 114220–114237. https://doi.org/10.1109/ACCESS.2023.3324612

Kim, B. G., & Wong, D. (2023). Smart Contract-based Secure Verifiable Random Function using ChaCha20 Sequence in Blockchain. *ACM International Conference Proceeding Series*, 41–51. https://doi.org/10.1145/3638025.3638028

Kurnia Mahesa Naibaho, A., Putri, A., & Fikri Husaini, Y. (2021). Analisis Sistem Keamanan Informasi Chat Online dengan Menggunakan Algoritma AES. In *Ilmu Komputer* (Vol. 6, Issue 2).

Luong, T. T. (2023). Strengthening AES Security Through Key-Dependent ShiftRow and AddRoundKey Transformations Utilizing Permutation. In *IJACSA) International Journal of Advanced Computer Science and Applications* (Vol. 14, Issue 11). [www.ijacsa.thesai.org](http://www.ijacsa.thesai.org/)

Muhammed, R. K., Rashid Aziz, R., Hassan, A. A., Aladdin, A. M., Saydah, S. J., Rashid, T. A., & Hassan, A. (2024). *Comparative Analysis of AES, Blowfish, Twofish, Salsa20, and ChaCha20 for Image Encryption*.

Olivia Putri Irine Irawan, B., Tahir, M., Ayu Windrastuti, N., Yurina Cholili, D., Mulaikah, D., Batsul Mushofi Septian wachid, A., & Pendidikan Informatika,

P. (2023). Implementasi Kriptografi Pada Keamanan Data Menggunakan Algoritma Advance Encryption Standard (AES). *Jurnal Simantec*, *11*(2), 167–

174. https://doi.org/10.21107/SIMANTEC.V11I2.20034

Pangestu, G. Y., Hadiana, A. I., & Sabrina, P. N. (2022). Kriptografi Untuk Enkripsi Ganda Pada Gambar Menggunakan Algoritma AES (Advanced Encryption Standard) Dan RC5 (Rivest Code 5). *Vol*, *1*, 25–32.

Putri, A. E., Kartikadewi, A., & Rosyid, L. A. A. (2021). Implementasi Kriptografi dengan Algoritma Advanced Encryption Standard (AES) 128 Bit dan Steganografi menggunakan Metode End of File (EOF) Berbasis Java Desktop

pada Dinas Pendidikan Kabupaten Tangerang. *Appl. Inf. Syst. Manag*, *3*(2), 69–78.

Saripa, S. (2023). Implementation Implementation of a File Security System Using the AES Algorithm to Secure Personal Files: Implementasi Sistem Keamanan File Menggunakan Algoritma AES untuk Mengamankan File Pribadi. *Progressive Information, Security, Computer, and Embedded System*, *1*(2), 138–148.

Sekar Putri Ananda, Saepul Lukman, & Irfan. (2022). *Analisa Metode Kriptogra Modern Advance Encryption Standar (AES) 128 Bit dalam Mengenkripsi dan Mendekripsi File Dokumen Digital*. https://doi.org/10.32409/jikstik.21.3.2973

Soesanto, E., Purba, L. M., Aprilia, B., Putra, D. R., & Putri, S. D. (2023). Implementasi Objek Vital, Pengamanan File dan Pengamanan Cyber di PT Pertamina. *IJM: Indonesian Journal of Multidisciplinary*, *1*(1).

Surbakti, P. S., & Purwanto, P. (2024). Implementasi Kriptografi Dengan Menggunakan Algoritma AES-128 Untuk Enkripsi dan Dekripsi File Dokumen Berbasis Web Pada Law Office Erdi Surbakti, SH & Rekan. *Prosiding Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI)*, *3*(1), 1–8.

Syahputra Sianipar, J., Budi Nuugroho, N., Mariami, I., Informasi, S., & Triguna Dharma, S. (2024). Pengamanan Data Gaji Karyawan Dengan Menggunakan Metode Advanced Encryption Standard (AES). *Jurnal Sistem Informasi Triguna Dharma (JURSI TGD)*, *3*(1), 35–45. https://doi.org/10.53513/JURSI.V3I1.5653

Wahyu Putra, M. R. F. T. H. (2023, July 2). *Implementasi Algoritma Advanced Encryption Standard Untuk Kemanan Dokumen*. https://journal.grahamitra.id/index.php/jurikti/article/view/55/181

Wahyudi, E. N., Ardhianto, E., Handoko, W. T., Murti, H., Supriyanto, E., Lestariningsih, E., & Redjeki, R. S. (2024). Peningkatan Keamanan Data Melalui Teknik Super Enkripsi Menggunakan Algoritma Vigenere dan Caesar. *Jurnal Informatika Polinema*, *10*(3), 315–322. https://doi.org/10.33795/JIP.V10I3.5131

Widyawan, D., & Imelda, I. (2021). Pengamanan File Menggunakan Kriptografi Dengan Metode Aes-128 Berbasis Web Di Komite Nasional Keselamatan Transportasi. *SKANIKA: Sistem Komputer Dan Teknik Informatika*, *4*(1), 15– 22.

**IMPLEMENTASI KOMBINASI CHACHA20 DAN MODIFIKASI AES MELALUI *ADDROUNDKEY* PADA**

**STUDI KASUS KEAMANAN *FILE***

**Proposal Tugas Akhir Mata Kuliah Penulisan Proposal**

## (CAK4EAB2)

Disusun oleh:

## EMILIA RAMONA 1203210017

**PROGRAM STUDI SARJANA INFORMATIKA DIREKTORAT KAMPUS SURABAYA**

**UNIVERSITAS TELKOM SURABAYA TAHUN 2025**

# ABSTRAK

Keamanan data menjadi tantangan besar di era Revolusi Industri 4.0, di mana informasi penting perlu dilindungi dari akses tidak sah. Ancaman serangan siber dan peretasan mengharuskan pengembangan solusi keamanan yang lebih efektif. Penelitian ini bertujuan untuk mengimplementasikan dan mengevaluasi kombinasi algoritma ChaCha20 dan modifikasi *Advanced Encryption* Standard (AES) dengan penambahan langkah *AddRoundKey* sebelum *MixColumns* untuk meningkatkan keamanan *file*. Metode yang digunakan adalah enkripsi ganda, dengan AES modifikasi sebagai *block cipher* yang kuat dan ChaCha20 sebagai *stream cipher* yang cepat. Penelitian ini mencakup implementasi algoritma dengan menggunakan pustaka kriptografi untuk menguji kinerja algoritma melalui parameter seperti waktu enkripsi, waktu *dekripsi*, kecepatan enkripsi, kecepatan *dekripsi*, dan tingkat keacakan data menggunakan analisis entropi. *File* dengan berbagai format dan ukuran, PDF, Word, dan TXT, digunakan sebagai data pengujian untuk mengevaluasi efisiensi algoritma. Selain itu, antarmuka pengguna berbasis web dirancang untuk mempermudah proses enkripsi dan *dekripsi*, melibatkan langkah- langkah seperti pemilihan *file* dan *input* kunci enkripsi. Penelitian ini diharapkan menunjukkan bahwa kombinasi AES modifikasi dan ChaCha20 mampu meningkatkan keamanan data serta memberikan efisiensi yang lebih baik dalam perlindungan *file*.

**Kata Kunci**: Keamanan *File*, *Advanced Encryption Standard* (AES), ChaCha20, Enkripsi Ganda, *AddRoundkey*, *Entropi*.

# ABSTRACT

*Data security has become a major challenge in the era of the Industrial Revolution 4.0, where critical information must be protected from unauthorized access. Cyberattacks and hacking threats demand the development of more effective security solutions. This research aims to implement and evaluate the combination of the ChaCha20 algorithm and a modified Advanced Encryption Standard (AES) by adding an AddRoundKey step before MixColumns to enhance file security. The method used is double encryption, with the modified AES as a robust block cipher and ChaCha20 as a fast stream cipher. This study involves implementing the algorithms using cryptographic libraries to test their performance through parameters such as encryption time, decryption time, encryption speed, decryption speed, and data randomness level using entropy analysis. Files of various formats and sizes, including PDF, Word, and TXT, are utilized as test data to evaluate the efficiency of the algorithms. Additionally, a web-based user interface is designed to simplify the encryption and decryption process, involving steps such as file selection and encryption key input. This research is expected to demonstrate that the combination of modified AES and ChaCha20 can enhance data security and provide better efficiency in file protection.*

***Keywords****: File Security, Advanced Encryption Standard (AES), ChaCha20, Double Encryption, AddRoundkey, Entropy*

# KATA PENGANTAR

Assalamu’alaikum Warrahmatullahi Wabarakatuh, puja puji syukur ke hadirat Allah SWT yang telah memberikan nikmat dan karunia-Nya yang sangat luar biasa kepada hamba – hambanya. Tidak lupa shalawat serta salam kita curahkan atas bimbingan nabi kita nabi Muhammad SAW yang telah membawa cahaya islam ke dalam dunia ini dan menjadi panutan bagi setiap umatnya hingga akhir zaman.

Setelah melalui berbagai proses kerja keras, penyusunan skripsi yang berjudul “Implementasi Kombinasi ChaCha20 dan Modifikasi AES melalui *AddRoundkey* Pada Studi Kasus Keamanan *File*” akhirnya dapat terselesaikan. Selain sebagai salah satu syarat memperoleh gelar S-Kom, penulis berharap juga skripsi ini dapat bermanfaat bagi semua pihak yang berkepentingan. Oleh karena itu, penulis ingin mengungkapkan rasa terima kasih kepada pihak-pihak yang telah memberikan dukungan dan doa sehingga penulis dapat menyelesaikan skripsi ini tepat pada waktunya. Beberapa pihak tersebut antara lain:

1. **Kepada Mama** yang sangat penulis cintai yang tidak pernah lupa selalu memberikan dukungannya dan doa agar diberikan kemudahan dan kelancaran dalam penyusunan skripsi ini sehingga penulis dapat menyelesaikan studi Informatika di Universitas IT Telkom Surabaya.
2. **Kepada Alm.Ibu Suhartatik**, yang penulis cintai dan sayangi, yang telah memberikan motivasi untuk saya.
3. **Kepada Saudara Kembar**, yang selalu mendukung, terima kasih atas bantuan dan waktunya dalam memberikan masukan terkait susunan skripsi ini.
4. **Keluarga Besar penulis** yang telah memberikan motivasi dan kepercayaan besar sehingga penulis bisa mewujudkan impian untuk meraih gelar S-Kom.
5. **Dosen Pembimbing**, Pak Rizky Fenaldo Maulana, S.Kom., M.Kom. dan Tanzila Mustaqim, S.Kom., M.Kom. Yang telah memberikan solusi, waktu, dan bimbingan penuh kesabaran selama proses penyusunan skripsi ini.
6. **Teman-teman dekat**, termasuk Jus’an, Erica Dyah, Qori Emalia, Ardina Firdatana, dan Elma Saskia yang selalu memberikan semangat, mendengarkan keluh kesah, serta menemani dalam berbagai kondisi selama penulisan skripsi.
7. **Semua pihak lainnya**, yang tidak dapat disebutkan satu per satu, atas bantuan dan dukungannya selama proses ini.
8. **Diri penulis sendiri**, atas perjuangan dan kerja keras hingga akhirnya skripsi ini selesai tepat waktu.

Penulis menyadari bahwa skripsi ini masih jauh dari kesempurnaan. Oleh karena itu, saran dan kritik yang membangun sangat diharapkan untuk memperbaiki skripsi ini ke depannya.

Terima kasih.

### Wassalamu’alaikum Warrahmatullahi Wabarakatuh.

Surabaya, 24 Desember 2024 Penulis

Emilia Ramona

# DAFTAR ISI

[LEMBAR PERSETUJUAN ii](#_bookmark0)

[ABSTRAK iii](#_bookmark1)

[ABSTRACT iv](#_bookmark2)

[KATA PENGANTAR v](#_bookmark3)

[DAFTAR ISI vii](#_bookmark4)

[DAFTAR TABEL ix](#_bookmark5)

[DAFTAR GAMBAR x](#_bookmark6)

[DAFTAR RUMUS xi](#_bookmark7)

[BAB I PENDAHULUAN 1](#_bookmark8)

* 1. [Latar Belakang 1](#_bookmark9)
  2. [Rumusan Masalah 4](#_bookmark10)
  3. [Tujuan Penelitian 4](#_bookmark11)
  4. [Batasan dan Asumsi Penelitian 5](#_bookmark12)
  5. [Manfaat Penelitian 5](#_bookmark13)
     1. [Manfaat Akademis 5](#_bookmark14)
     2. [Manfaat Ilmiah 5](#_bookmark15)
  6. [Sistematika Penulisan 6](#_bookmark16)

[BAB II LANDASAN TEORI 7](#_bookmark17)

* 1. [Literatur Terkait Teori 7](#_bookmark18)
  2. [Kajian Teori 10](#_bookmark20)
     1. [Kriptografi 10](#_bookmark21)
     2. [AES (Advanced Encryption Standard) 12](#_bookmark24)
     3. [Chacha20 20](#_bookmark32)
     4. [Entropi 22](#_bookmark35)

[BAB III METODOLOGI PENELITIAN 24](#_bookmark37)

* 1. [Sistematika Penyelesaian Masalah 24](#_bookmark38)
  2. [Alat dan Bahan 24](#_bookmark39)
     1. [Alat 24](#_bookmark40)
     2. [Bahan 25](#_bookmark41)
  3. [Rancangan Penelitian 25](#_bookmark42)
     1. [Studi Literatur 27](#_bookmark45)
     2. [Pengumpulan Data 27](#_bookmark46)
     3. [Implementasi Modifikasi AES 28](#_bookmark47)
     4. [Implementasi Kombinasi AES Modifikasi dan ChaCha20 31](#_bookmark49)
     5. [Implementasi Antarmuka Pengguna (UI) 36](#_bookmark52)
     6. [Pengujian Algoritma 38](#_bookmark54)
     7. [Analisis Hasil 42](#_bookmark59)

[DAFTAR PUSTAKA 44](#_bookmark61)

# DAFTAR TABEL

[Tabel II.1 Literatur Penelitian Terkait Kriptografi 7](#_bookmark19)

[Tabel II.2 Panjang Kunci 13](#_bookmark25)

[Tabel II.3 Tabel Substitusi (S-Box) 16](#_bookmark28)

[Tabel II.4 Transformasi ShiftRows 17](#_bookmark29)

[Tabel III.1 Jenis, Ukuran, dan Isi File untuk Pengujian 40](#_bookmark58)

[Tabel III.2 Evaluasi Hasil 43](#_bookmark60)

# DAFTAR GAMBAR

[Gambar II.1 Alur Proses Enkripsi Dan Dekripsi 11](#_bookmark22)

[Gambar II.2 Proses Kunci Simetris 11](#_bookmark23)

[Gambar II.3.Transformasi ShiftRows 15](#_bookmark27)

[Gambar II.4 Proses AES 19](#_bookmark31)

[Gambar III.1 Diagram Alur Penelitian 26](#_bookmark43)

[Gambar III.2 Diagram Proses Enkripsi AES 28](#_bookmark48)

[Gambar III.3 Diagram Proses Double Encrypt 32](#_bookmark50)

[Gambar III.4 Proses Dekripsi 36](#_bookmark51)

[Gambar III.5 Desain Antarmuka Pengguna Enkripsi dan Dekripsi 37](#_bookmark53)

# DAFTAR RUMUS

[(II-1) Rumus AddRoundKey 14](#_bookmark26)

[(II-2) Operasi Perkalian MixColumns 17](#_bookmark30)

[(II-3) Rumus Keystream 22](#_bookmark33)

[(II-4) Rumus Proses Enkripsi XOR pada Keystream 22](#_bookmark34)

[(II-5) Entropi 22](#_bookmark36)

[(III-1) Pengujian Waktu Enkripsi 39](#_bookmark55)

[(III-2) Pengujian Waktu Dekripsi 39](#_bookmark56)

[(III-3) Pengujian Waktu Kecepatan 39](#_bookmark57)

# BAB I PENDAHULUAN

### Latar Belakang

Semakin meningkatnya teknologi yang semakin canggih membawa sebuah perubahan yang cukup signifikan. Perubahan tersebut dikenal dengan istilah Revolusi Industri 4.0 yang merupakan sebuah kemajuan teknologi yang sering digunakan oleh banyak orang. Revolusi industri 4.0 adalah sebuah data yang digerakkan dengan adanya campur tangan komputer menggunakan metode machine learning atau AI (Alimuddin et al., 2023). Menurut Klaus Schawab revolusi Industri 4.0 memiliki tantangan yang harus dihadapi salah satunya seperti keamanan *cyber* dan privasi data.

Maka dari itu, dengan semakin pesatnya informasi teknologi maka perlunya untuk mengamankan data atau *file*. Keamanan data atau *file* merupakan salah satu langkah yang mudah untuk melindungi sebuah informasi bersifat penting yang berasal dari akses tidak sah. Keamanan *file* ini sangat diperlukan pada era Revolusi Industri 4.0, karena maraknya serangan *cyber* dan peretasan komputer yang bisa mengganggu keberlangsungan suatu industri. Menurut (Mulyana*, 2016*), keamanan *file* merupakan salah satu aksi pencegahan dari sebuah penganggu pengguna *file* atau pengguna yang memakai jaringan yang tidak berkewajiban memiliki akses (Soesanto et al., 2023). Dengan semakin pesatnya dunia teknologi ini membuat keamanan *file* menjadi sangat penting karena untuk melindungi privasi dan informasi dari pihak yang tidak memiliki akses.

Keamanan *file* adalah serangkaian metode atau teknik yang untuk melindungi *file* dari akses yang tidak sah, seperti pencurian atau kerusakan. Tujuannya untuk menjaga informasi dalam *file* tetap rahasia, aman, dan hanya dapat di akses oleh pihak yang berwenang. Oleh karena itu, upaya untuk memastikan agar *file* terjaga dengan menggunakan metode ChaCha20 dan *Advanced Encryption Standard* (AES). ChaCha20 merupakan metode enkripsi yang cepat dan aman. ChaCha20 dipopulerkan oleh Daniel J.

Bernstein sebagai varian dari algoritma. Dalam algoritma ChaCha20 sangat populer dalam pengimplementasi enkripsi dikarenakan percampuran antara kecepatan dan keamanannya yang dapat diunggulkan. Chaca20 mempunyai desain yang hampir sama dengan Salsa20, pada algoritma ChaCha20 menggunakan kunci 256-bit yang mempunyai 8 putaran (Kebande, 2023).

Sedangkan AES (*Advanced Encryption Standard)* adalah sebuah algoritma yang melakukan transformasi terhadap blok data yang menggunakan kunci enkripsi yang sama. AES bekerja pada sistem yang melibatkan operasi matematika dengan level tingkat sulit (Saripa, 2023). Proses kerja enkripsi dasar AES adalah *subbytes* (menggunakan tabel S-Box untuk menggabungkan *byte* pada blok matriks), *shift row* (digunakan untuk menggeser baris matriks), *MixColumns* (melakukan pergeseran dari kanan ke kiri), dan *AddRoundkey* (blok di XOR kan dengan kunci yang diperluas) (Aryanto et al., 2023). Pada proses *dekripsi* AES terdapat tahap-tahapan kebalikan dari proses enkripsi. Dengan memakai kunci yang sama untuk *dekripsi*, blok data yang telah terenkripsi dapat dipulihkan kembali ke bentuk awalnya. algoritma AES adalah jenis *block chipper* yang memiliki panjang kunci 128 bit, 192 bit dan 256 bit (Surbakti & Purwanto, 2024).

Algoritma Chacha20 dan AES merupakan suatu enkripsi yang dapat di digunakan dalam menjaga keamanan *file*. Pada algoritma AES terdapat kelebihan salah satunya adalah sifat karakteristiknya yang di mana selalu ada bidang khusus untuk bilangan prima, memicu kekuatan AES(Surbakti & Purwanto, 2024) . AES juga memiliki sebuah kelemahan salah satunya yaitu algoritma kriptografi yang menggunakan kunci simetris dalam proses pengiriman dan penerimanya data. Namun, penggunaan kunci simetris ini memiliki risiko lebih tinggi terhadap kebocoran, terutama seiring waktu (Hidayatulloh et al., 2023). Pada algoritma ChaCha20 terdapat keunggulan yaitu menggunakan teknik kriptografi ARX (*addition, rotation*, XOR) yang memberikan performa lebih cepat, kompleksitas lebih rendah, dan mampu menghilangkan serangan waktu Klik atau ketuk di sini untuk memasukkan teks.

Meskipun algoritma AES dan Chacha20 telah efektif untuk menjaga peningkatan keamanan *file* dan implementasinya yang cukup rumit sehingga membuat orang yang tidak memiliki akses tidak dapat mengakses *file* tersebut. Hal ini tidak dapat kita pungkiri bahwa semakin pesatnya dunia teknologi membuat banyak orang berbondong-bondong menciptakan pembaruan yang lebih efektif. Metode *double encrypt* ini menawarkan kombinasi yang mampu mengurangi kelemahan dari setiap algoritma jika digunakan secara bersamaan, serta meningkatkan ketahanan terhadap berbagai jenis serangan kriptografi. *Double encrypt* merupakan Teknik kriptografi yang melibatkan proses enkripsi secara dua kali, yang di mana memiliki dua kunci yang berbeda untuk lebih meningkatkan keamanan. *Double encrypt* ini bertujuan untuk meningkatkan keamanan *file* lebih tinggi dibandingkan hanya 1 metode saja (Pangestu et al., 2022).

Pada penelitian terlebih dahulu yang berjudul “Implementasi Sistem Keamanan *File* Menggunakan Algoritma AES untuk Mengamankan *File* Pribadi” yang diteliti oleh Saripa Universitas Negeri Makassar. Dalam penelitian tersebut menjelaskan tentang keamanan *file* dengan menggunakan metode algoritma AES. Di mana hasil penelitian ini menunjukkan bahwa algoritma AES telah berhasil mengamankan *file* yaitu dokumen Word, PDF, Excel, dan Power Poin, *file* yang dienkripsi menjadi tidak terbaca oleh pihak yang tidak memiliki akses sehingga dapat menjaga kerahasiaan data dan AES mampu menjaga keamanan data dengan baik.

Dalam perbedaan penelitian terdahulu dan penelitian saya akan menggunakan double encrypt untuk keamanan *file* dan juga saya akan menggunakan AES dengan menambahkan metode ChaCha20 dan melakukan modifikasi pada bagian AES. Modifikasi menggunakan penambahan *AddRoundkey* sebelum dengan bertujuan agar proses enkripsi akan lebih susah untuk dipecahkan. Dalam algoritma AES, langkah *AddRoundKey* merupakan komponen krusial yang menggabungkan data dengan kunci enkripsi menggunakan operasi XOR. Penelitian (Luong, 2023) oleh menunjukkan bahwa memodifikasi transformasi *AddRoundKey*

dapat meningkatkan keamanan algoritma dengan menciptakan hubungan yang lebih kompleks antara *plainteks* dan *cipherteks*. Penelitian ini menyoroti pentingnya penyesuaian langkah *AddRoundKey* untuk meningkatkan ketahanan algoritma AES terhadap serangan kriptografi, seperti *differential cryptanalysis* dan linear *cryptanalysis*.

Dengan mendasarkan pada penelitian tersebut, modifikasi algoritma AES melalui penambahan langkah *AddRoundKey* sebelum *MixColumns* bertujuan untuk lebih memperkuat difusi data pada tahap awal enkripsi. Penambahan ini diharapkan dapat meningkatkan kerandoman hasil enkripsi dan mengurangi kemungkinan eksploitasi pola oleh pihak yang tidak berwenang, sehingga keamanan algoritma menjadi lebih optimal. Berdasarkan dari latar belakang di atas penelitian ini memiliki tujuan untuk meningkatkan keamanan *file* agar lebih sulit untuk diretas atau diakses pihak yang tidak berwenang. Oleh karena itu penelitian ini dianggap penting karena maraknya kasus peretasan data oleh oknum yang tidak memiliki akses.

### Rumusan Masalah

1. Bagaimana implementasi dari modifikasi AES dengan penambahan *AddRoundkey* sebelum *MixColumns* dalam meningkatkan keamanan *file*?
2. Bagaimana performa kombinasi antara AES modifikasi dan ChaCha20 dalam hal waktu enrkipsi, waktu *dekripsi* , kecepatan enkripsi, dan kecepatan *dekripsi*?

### Tujuan Penelitian

1. Mampu mengidentifikasi *AddRoundkey* yang akan digunakan dalam menjaga keamanan *file* dan mengimplementasi kemampuan Chacha20 dan AES yang telah dimodifikasi dalam proses menjaga keamanan *file*.
2. Mampu mengevaluasi performa algoritma ChaCha20 dan AES yang telah dimodifikasi berdasarkan waktu enkripsi, waktu *dekripsi*, kecepatan enkripsi, dan kecepatan *dekripsi*, untuk menentukan efisiensi

dan efektivitas kombinasi kedua algoritma tersebut dalam menjaga keamanan *file*

### Batasan dan Asumsi Penelitian

1. Penelitian ini berfokus pada jurnal atau artikel ilmiah dengan batasan tahun 2021-2024 yang membahas tentang :
   * Keamanan *file*
   * Algoritma ChaCha20
   * *Advanced Encryption Standard* (AES)
   * Double encryption
2. Penelitian dilakukan dengan pengujian pada *file* dengan ukuran dan format sebagai berikut :
   * Format PDF dan Word dengan variasi ukuran *file* 100 KB - 100MB.
   * Format *file* teks (TXT) dengan ukuran maksimal 300 byte Batasan ukuran *file* digunakan untuk memastikan pengujian dilakukan secara efektif tanpa mengurangi kinerja proses enkripsi dan *dekripsi*.

### Manfaat Penelitian

### Manfaat Akademis

* + - 1. Memberikan inovasi terbaru mengenai pelindungan keamanan

*file* dengan menambahkan *AddRoundkey* pada algoritma AES.

* + - 1. Memudahkan peneliti lainnya dalam kinerja *AddRoundkey* pada algoritma AES yang berfungsi untuk keamanan *file***.**

### Manfaat Ilmiah

* + - 1. Menjadi sumber acuan dan sumber referensi bagi peneliti lainnya yang akan meneliti topik ChaCha20 dan AES modifikasi, terutama dalam konteks keamanan data dan performa enkripsi.
      2. Membuktikan hipotesis apakah AES yang dimodifikasi dengan tambahan *AddRoundkey* dapat meningkatkan performa keamanan dan efisiensi dibandingkan dengan AES standart, dan sekaligus menunjukkan kinerja kombinasi ChaCha20 dan AES dalam pengamanan *file* berbagai format dan ukuran.

### Sistematika Penulisan

## BAB 1 PENDAHULUAN

Menjelaskan latar belakang penelitian, rumusan masalah, tujuan penelitian, batasam penelitian, manfaat penelitian, serta sistematika penulisan secara umum.

## BAB 2 LANDASAN TEORI

Membahas teori-teori yang relavan dengan penelitian, termasuk algoritma ChaCha20, AES, konsep kriptografi, Serta tinjauan terhadap penelitian terdahulu yang terkait.

## BAB 3 METODOLOGI PENELITIAN

Menjelaskan metode yang digunakan dalam penelitian, termasuk algoritma yang diterapkan (modifikasi AES dan ChaCha20 dengan enkripsi ganda). Diuraikan juga alat dan bahan yang digunakan, serta tahapan penelitian yang meliputi implementasi algoritma, pengujian parameter kinerja (waktu dan kecepatan enkripsi, dekripsi, serta analisis entropi), dan evaluasi hasil pengujian pada berbagai jenis file (PDF, Word, TXT).

## BAB 4 IMPLEMENTASI DAN PENGUJIAN

Menyajikan hasil implementasi algoritma yang dikembangkan, pengujian algoritma terhadap berbagai jenis *file*, serta analisis data hasil pengujian**.**

## BAB 5 EVALUASI HASIL PENGUJIAN

Menyajikan hasil implementasi dari aplikasi yang telah dirancang, pengujian fungsionalitas sistem, serta evaluasi kinerja aplikasi terhadap berbagai skenario penggunaan. Analisis data hasil pengujian juga dilakukan untuk menilai keberhasilan aplikasi dalam mencapai tujuan penelitian.

## BAB 6 KESIMPULAN DAN SARAN

Merangkum temuan utama penelitian serta memberikan saran untuk pengembangan lebih lanjut di masa depan.

# BAB II LANDASAN TEORI

### Literatur Terkait Teori

Keamanan informasi menjadi salah satu isu yang paling mendesak di era digital saat ini, di mana data sensitif sering kali menjadi target serangan siber yang beragam. Dengan meningkatnya ancaman seperti serangan *man-in-the- middle*, *eavesdropping*, dan *DDoS*, penting bagi organisasi dan individu untuk menerapkan metode enkripsi yang efektif untuk melindungi data mereka. Berbagai algoritma enkripsi telah dikembangkan untuk memenuhi kebutuhan ini, dengan *Advanced Encryption Standard* (AES) dan ChaCha20 menjadi dua di antara yang paling banyak digunakan. Penelitian-penelitian terkini menunjukkan bahwa kedua algoritma ini tidak hanya menawarkan tingkat keamanan yang tinggi, tetapi juga dapat diimplementasikan dalam berbagai konteks, mulai dari pengamanan *file* hingga enkripsi konten *livestreaming*.

Pada [Tabel II.1 Literatur Penelitian Terkait Kriptografi](#_bookmark19) menyajikan ringkasan dari beberapa penelitian yang mengeksplorasi efektivitas dan penerapan algoritma AES dan ChaCha20 dalam berbagai aplikasi. Setiap penelitian memberikan wawasan tentang tujuan, hasil, dan persamaan dalam pendekatan yang digunakan, serta menyoroti inovasi yang dilakukan untuk meningkatkan keamanan data. Dengan memahami hasil-hasil ini, kita dapat lebih menghargai perkembangan teknologi enkripsi dan aplikasinya dalam menjaga kerahasiaan serta integritas informasi di dunia yang semakin terhubung.

### Kajian Teori

### Kriptografi

Menurut Menezes, Oorschot, dan Vantone (1996), Kriptografi adalah studi teknik matematika yang berkaitan dengan keamanan informasi seperti kerahasiaan, *autentikasi* keaslian data (Belvin Tannadi, 2022). Kriptografi dapat diartikan sebagai sebuah proses untuk melindungi data atau sebuah sistem yang bertujuan untuk melindungi informasi rahasia dengan cara yang tidak dipahami oleh pihak yang tidak berwenang. Dalam kriptografi yang meliputi proses enkripsi dan proses *dekripsi*. Di dalam Kriptografi terdapat tiga fungsi dasar yang meliputi :

* + - 1. Enkripsi merupakan proses data asli (*plainteks*) yang akan diubah menjadi data yang tidak bisa terbaca oleh pihak yang tidak berwenang (*chiperteks*).
      2. *Dekripsi* merupakan proses pengembalian dari data yang tidak bisa terbaca(*chiperteks*) menjadi data terbaca atau data asli (*plainteks*).
      3. Kunci merupakan proses yang dipakai untuk melakukan proses enkripsi dan *dekripsi*, terdapat 2 jenis kunci, yaitu kunci rahasia (*private key*) dan kunci publik (*public key*).

Pada [Gambar II.1 Alur Proses Enkripsi Dan *Dekripsi*](#_bookmark22)

memperlihatkan alur enkripsi dan *dekripsi.* Proses ini dimulai

dengan *plainteks* yang kemudian dienkripsi dengan menggunakan kunci menjadi *chiperteks*. *Chiperteks* tersebut kemudian *didekripsi* kembali menggunakan kunci yang sama untuk mendapatkan kembali *plainteks*.

*Gambar II.1 Alur Proses Enkripsi Dan Dekripsi*

Berikut algoritma Kriptografi yang berdasarkan kunci :

a. Algoritma Simetris

Algoritma simetris adalah algoritma yang merujuk pada metode enkripsi dan *dekripsi* yang di mana algoritma ini memiliki kunci tunggal atau sama. Sehingga dapat disimpulkan algoritma ini mewajibkan pengirim dan penerima harus menyetujui satu kunci yang digunakan dalam proses enkripsi dan *dekripsi* data. Namun, kelemahan dari algoritma ini yaitu tergantung pada rahasia kunci. Apabila kunci tersebut tersebar maka akan dengan mudah pihak yang tidak memiliki akses dapat melakukan proses enkripsi dan *dekripsi* data secara mudah.

*Gambar II.2 Proses Kunci Simetris*

Pada [Gambar II.2 Proses Kunci Simetris](#_bookmark23) menjelaskan proses enkripsi dan *dekripsi* menggunakan kunci simetris. Dimulai dengan pesan yang dikirimkan, kemudian dienkripsi menjadi kode yang tidak bisa dibaca menggunakan kunci simetris. Kode tersebut kemudian dikirim dan di *dekripsi* di sisi penerima menggunakan kunci simetris yang sama, sehingga pesan dapat dibaca kembali. Pada contoh gambar, pesan "Selamat Datang!" dienkripsi menjadi kode "9CDF4837 AB672019" menggunakan kunci simetris yang sama. Kemudian di *dekripsi* kembali ke pesan awal "Selamat Datang!" Kunci simetris merupakan kunci yang sama digunakan untuk enkripsi dan *dekripsi*, sehingga pengirim dan penerima pesan harus memiliki kunci yang sama agar dapat berkomunikasi.

### AES (Advanced Encryption Standard)

Advanced Encryption Standard (AES) adalah salah satu algoritma enkripsi yang paling terkenal dan banyak digunakan, serta terus dikembangkan sebagai pengganti algoritma Data Encryption Standard (DES). AES adalah algoritma enkripsi yang bekerja dengan memproses data dalam blok dengan ukuran minimal 128 bit, dan melalui beberapa tahapan untuk mengenkripsi data. Tujuan utama dari AES adalah untuk mengamankan data rahasia sehingga tidak dapat dibaca oleh pihak yang tidak berwenang (Wahyu Putra, 2023). Pada [Tabel II.2 Panjang Kunci](#_bookmark25) dalam proses enkripsi *Advanced Encryption Standard* (AES) menggunakan ukuran blok dengan panjang sebesar 126 bit, dengan pilihan panjang kunci yang bervariasi, yaitu 128 bit, 192 bit, dan 256 bit (Sekar Putri Ananda et al., 2022). Proses enkripsi dalam algoritma ini dilakukan melalui beberapa tahap yang dikenal sebagai ronde. Jumlah ronde yang digunakan bergantung pada panjang kunci yang akan dipilih. Berikut adalah hubungan panjang kunci dan banyaknya ronde.

Algoritma AES juga menawarkan tingkat keamanan yang sangat tinggi dikarenakan ukuran kunci dan jumlah ronde yang ditentukan. Ukuran kunci menentukan jumlah kemungkinan kombinasi kunci, sehingga membuat serangan *bruteforce* menjadi sangat sulit dilakukan. Di sisi lain, jumlah ronde yang besar dapat menghasilkan keamanan yang kuat, sehingga perubahan kecil pada *plainteks* dapat memengaruhi seluruh bagian *cipherteks*. Selain keunggulan keamanannya, AES juga memiliki efisiensi tinggi dalam hal waktu dan ruang. Algoritma ini dapat diimplementasikan baik pada perangkat keras maupun perangkat lunak, menjadikannya ideal untuk berbagai aplikasi, termasuk perlindungan data (Assyahid, 2024). Pada tahap awal enkripsi, data masukan yang telah disalin ke dalam matriks *state* akan menjalani transformasi *AddRoundkey*. Selanjutnya, matriks *state* akan melalui proses transformasi secara berulang, mencakup *SubBytes*, *ShiftRows*, *MixColumns*, dan *AddRoundkey*, dengan jumlah pengulangan sesuai dengan nilai Nr (Handoyo & Subakti, 2020). *Dekripsi* adalah proses mengembalikan *ciphertext* menjadi plaintext, atau mengubah data terenkripsi kembali ke bentuk aslinya. Pada proses *inverse cipher*, transformasi yang digunakan meliputi *InvShiftRows*, *InvSubBytes*, *InvMixColumns*, dan *AddRoundkey*. Berikut ini adalah cara kerja algoritma AES secara umum:

### Proses Enkripsi AES Modifikasi

* + - * 1. *AddRoundkey*

*AddRoundkey* adalah proses menggabungkan state masukan dengan kunci ronde dengan menggunakan operasi XOR. Dalam proses transformasi enkripsi dan *dekripsi* pada AES, dilakukan operasi penambahan kunci putaran (*round key*)

pada state menggunakan operasi XOR (Djong & Siswanto, 2022). Setiap kunci putaran terdiri dari *Nb word* di mana setiap *word* tersebut akan digabungkan dengan *word* atau kolom yang sesuai pada *state* . Secara matematis, hal ini (Putri et al., 2021).

Keterangan Algoritma dari *AddRoundkey* di mana :

Pada [Gambar II.3.Transformasi ShiftRows](#_bookmark27) di atas di mana S0,0 merupakan elemen di baris ke-0, kolom ke-0, S3,3 merupakan elemen di baris ke-3, kolom ke-3, dan 𝑊3,3 merupakan elemen di baris ke-3, kolom ke-3. Pada label S0,0 dan lainnya merupakan state saat ini, state ini representasi internal data yang sedang dienkripsi ataupun *didekripsi* dan setiap elemen adalah *byte* dari blok data input. Sedangkan pada label 𝑊0,0 merupakan kunci ronde yang dihasilkan dari proses *keyexpansion*, di mana ini bagian dari kunci utama yang dipecah untuk menjadi beberapa ronde dan setiap elemen adalah *byte* dari kunci untuk ronde tertentu. Operasi XOR dilakukan antara setiap *byte* pada *state* dengan *byte* kunci ronde yang sesuai. Hasil di atas dihasilkan dari operasi XOR yang akan digunakan untuk langkah transformasi berikutnya.

* + - * 1. *SubBytes*

Transformasi *transformasi byte* di mana setiap elemen dalam matriks *state* dipetakan menggunakan tabel substitusi yang dikenal sebagai S-Box. Setiap *byte* dalam matriks *state* direpresentasikan sebagai S’[r, c], yang merupakan elemen hasil dari perpotongan antara baris (x) dan kolom (y) (Kurnia Mahesa Naibaho et al., 2021). S-Box berupa matriks

berukuran 16×16 yang berisi nilai-nilai heksadesimal di mana nilai-nilai ini diperoleh melalui perhitungan tertentu (Hena & Kusumaningsih, 2024). Berikut ini tabel S-Box :

Pada [Tabel II.3 Tabel Substitusi (S-Box)](#_bookmark28) sebagai contoh, nilai heksadesimal {62} merujuk ke baris 7 dan kolom 3 dalam S-Box, yang berisi nilai {aa}. Dengan demikian, nilai

{62} dipetakan menjadi nilai {aa}.

* + - * 1. *ShiftRows*

Transformasi *ShiftRows* merupakan proses pergeseran *byte* di mana *byte* paling kiri dipindahkan ke posisi paling kanan (rotasi bit). Jumlah pergeseran dalam melakukan *ShiftRows* ini dilakukan perbedaan setiap baris dalam matriks *state*. Pada baris pertama, tidak ada pergeseran yang dilakukan. Kemudian pada baris kedua barulah mengalami pergeseran satu *byte* ke kiri, lalu baris ketiga digeser dua *byte* ke kiri, dan baris keempat digeser tiga *byte* ke kiri. Tujuan dari proses ini adalah untuk menciptakan *diffusion*(), yang memiliki tujuan untuk membantu menyebarkan efek transformasi non-linear ke seluruh baris dalam matriks *state* untuk proses selanjutnya (Cristy & Riandari, 2021). Proses pergeseran ini bertujuan untuk mencampur elemen-elemen dalam *state* sehingga memperkuat kerahasiaan data.

*MixColumns* adalah proses transformasi di mana setiap elemen dalam blok *cipher* dikalikan dengan matriks . Transformasi ini bertujuan untuk mencampur elemen- elemen dalam setiap kolom *state* berguna meningkatkan difusi data dan memperkuat keamanan enkripsi . Transformasi ini bekerja dengan membagi matriks atau *state* menjadi empat bagian, untuk setiap kolomnya (Widyawan & Imelda, 2021). Setiap kolom dalam matriks *state* diperlakukan sebagai *polinomial* dengan empat bagian dalam *Galois Field* (GF), pada persamaan [(II-2)](#_bookmark30) dilakukan Algoritma dari *AddRoundkey* di mana :

𝑋8 = Elemen dengan derajat ke-8 yang akan muncul sebagai 100000000 (1 diposisi ke 9 dari kanan) di bilangan biner.

X = Menunjukkan elemen dengan derajat ke 1 yang akan muncul sebagai 00000010 ( 1 di posisi ke 2 dari kanan

) dalam bilangan biner

1 = Elemen konstanta yang akan muncul sebagai

00000001 ( 1 diposisi ke 1 dari kanan), dimana elemen 1 ini menyatakan bahwa *polinomial* ini memiliki koefisien konstan.

Proses *MixColumns* juga dapat direpresentasikan sebagai operasi perkalian matriks, di mana setiap kolom matriks dikalikan dengan matriks tertentu dalam *Galois Field* (Cristy & Riandari, 2021).

* + - 1. **Proses *dekripsi* AES :**

Proses *dekripsi* merupakan kebalikan dari proses enkripsi. *Transformasi bytes* yang digunakan dalam *dekripsi* mencakup *AddRoundKey*, *InvShiftRows*, *InvSubBytes,* dan *InvMixColumns*. Pada iterasi pertama *dekripsi*, dilakukan *transformasi AddRoundKey, Inverse ShiftRows, dan Inverse SubBytes. Ciphertex*t kemudian akan mengalami transformasi *AddRoundKey* .

* + - * 1. *InvShiftRows*

Transformasi *byte* yang merupakan kebalikan dari proses *ShiftRows*. Dalam transformasi ini, bit-bit pada *state* digeser ke arah kanan, berbeda dengan *ShiftRows* yang melakukan pergeseran bit ke arah kiri.

* + - * 1. *InvSubBytes*

Transformasi *byte* yang menjadi kebalikan dari *SubBytes*. Dalam proses ini, setiap elemen pada *state* dipetakan menggunakan tabel substitusi *Inverse* S-Box.

* + - * 1. *InvMixColumns*

Setiap kolom dalam *state* dikalikan dengan matriks khusus yang digunakan dalam proses perkalian pada algoritma AES (Syahputra Sianipar et al., 2024)

*Gambar II.4 Proses AES*

Pada [Gambar II.4 Proses AES](#_bookmark31) sebelah kiri menunjukkan tahapan proses enkripsi pada algoritma AES Modifikasi. Proses dimulai dengan memasukkan plainteks, kemudian melalui serangkaian langkah seperti *AddRoundkey*, SubBytes, ShiftRows, MixColumns, dan diakhiri dengan *AddRoundkey* tambahan. Langkah ini diulang sebanyak Nr−1 ronde untuk meningkatkan keamanan data, menghasilkan cipherteks di akhir proses. Flowchart sebelah kanan menjelaskan tahapan proses *dekripsi*, yang merupakan kebalikan dari proses enkripsi. Proses dimulai dengan memasukkan cipherteks, kemudian melalui langkah-langkah seperti *AddRoundkey*, Inverse ShiftRows, Inverse SubBytes, dan Inverse MixColumns. Setelah semua ronde selesai, data dikembalikan ke bentuk plainteks asli.

### Chacha20

Algoritma ChaCha20 adalah salah satu jenis *stream cipher* yang dikembangkan sebagai varian dari algoritma Salsa20. Algoritma ini berbasis desain yang serupa dengan Salsa20, namun ChaCha20 meningkatkan tingkat *diffusion* pada setiap ronde untuk memperkuat keamanan proses enkripsi (Anindita & 18219086, 2023). Chacha20 beroperasi pada unit data 32-bit menggunakan kunci 256-bit. Algoritma ini menghasilkan *output* berupa *stream* kunci sepanjang 512-bit, yang kemudian akan di XOR kan dengan *plainteks* untuk menghasilkan *chiperteks*(Alyas & Abdullah, 2021b). ChaCha20 dimulai dengan membuat *state* 512-bit, yang berupa *array* yang berukuran 16x32-bit. State ini disusun dari empat konstanta 32 bit, kunci rahasia sepanjang 256-bit, penghitung blok 32-bit, serta nonce Ini menunjukkan struktur *state* ChaCha20 yang terdiri dari matriks 4x4 dengan 16 elemen 32-bit. Matriks ini diinisialisasi dengan empat konstanta 32-bit, kunci 256-bit yang dibagi menjadi 8 bagian, penghitung 32 blok 32-bit, dan Nonce 96-bit. Elemen-elemen ini digunakan untuk menghasilkan *keystream* yang kemudian di XOR kan dengan *plainteks* untuk menghasilkan *chiperteks*. Di mana *const* (konstanta) ini berisi empat nilai tetap 32-bit, *key* (kunci) yang terbagi 8 bagian 32-bit dari kunci rahasia 256-bit, *counter* (penghitung), dan Nonce (nilai acak) yang berukuran 96-bit.

Langkah – Langkah Algoritma ChaCha20 :

* + - 1. *Inisialisasi* State

Algoritma ChaCha20 dimulai dengan membangun sebuah state internal berbentuk matriks 4x4 (32-bit) (Barbero et al., 2022) . Matriks ini terdiri dari:

* + - * 1. Baris pertama: Berisi 4 konstanta tetap (128-bit), yang digunakan untuk mengenali algoritma.
        2. Baris kedua dan ketiga: Berisi kunci rahasia 256-bit (8 elemen 32-bit), yang digunakan untuk proses enkripsi.
        3. Baris keempat: Berisi kombinasi 64-bit counter dan 64-bit nonce (masing-masing terdiri dari 2 elemen 32-bit).
      1. *Quarter Round*

Operasi ini memodifikasi 4 elemen dari matriks ( a, b, c, d ) (Almukhlifi & Vora, 2020)

𝑎 += 𝑏; 𝑑 ^ = 𝑎; 𝑑 <<<= 16;

𝑐 += 𝑑; 𝑏 ^ = 𝑐; 𝑏 <<<= 12;

𝑎 += 𝑏; 𝑏 ^ = 𝑎; 𝑑 <<<= 8;

𝑐 += 𝑑; 𝑏 ^ = 𝑐; 𝑏 <<<= 7;

* + - 1. *Double Round*

*Double Round* adalah kombinasi dari dua *Quarter Round* yang dijalankan secara berurutan. Setiap iterasi *Double Round* mencakup:

* + - * 1. Operasi pada kolom matriks *(Column Round):*

melakukan operasi Quarter Round pada setiap kolom matriks 4x4. Tujuannya adalah untuk mencampur data antar elemen dalam kolom, sehingga meningkatkan difusi data secara vertikal.

* + - * 1. Operasi pada diagonal matriks (Diagonal Round):

Setelah operasi pada kolom selesai, langkah berikutnya adalah menerapkan Quarter Round pada elemen-elemen diagonal dalam matriks.

Proses Double Round dilakukan sebanyak 10 iterasi, menghasilkan total 20 ronde Quarter Round untuk setiap blok data yang diproses.

* + - 1. Finalisasi State
      2. *Keystream Generation*

Setelah 20 ronde selesai*, state* akhir ditambahkan kembali ke

### Entropi

Entropi dalam teori informasi digunakan untuk mengukur keacakan data, yang mencerminkan kekuatan algoritma enkripsi. Semakin tinggi nilai entropi, semakin acak data tersebut, yang berarti algoritma enkripsi lebih tahan terhadap serangan (Wahyudi et al., 2024). Entropi dihitung dengan menggunakan rumus pada

Dimana:

* 𝐻𝑚 : Menunjukkan entropi data, dimana tingkat keacakan dalam

# BAB III METODOLOGI PENELITIAN

### Sistematika Penyelesaian Masalah

Penelitian ini bersifat kuantitatif dengan tujuan mengevaluasi performa algoritma AES modifikasi dan kombinasi ChaCha20 dan AES modifikasi. Penilaian performa ini dilakukan berdasarkan waktu enkripsi, waktu *dekripsi*, kecepatan enkripsi, dan kecepatan *dekripsi*, yang akan dianalisis menggunakan data hasil pengujian pada berbagai jenis *file* dengan format PDF, Word dan TXT, yang hasilnya berbentuk tabel untuk mempermudah analisis perbandingan.

### Alat dan Bahan

### Alat

1. Perangkat Keras (Hardware) Laptop dengan spesifikasi :
   1. Prosesor : Intel *Core* I5
   2. RAM : 8 GB
   3. Penyimpanan : 256 GB SSD atau lebih
2. Perangkat Lunak (*Software*)
   1. *Python* :

Menggunakan *library cryptography, pycryptodome*, dan *library* lain yang mendukung algoritma AES dan ChaCha20. Berikut beberapa pustaka yang digunakan :

* + - PyCryptodome : Digunakan Iplementasi algoritma AES dan ChaCha20
    - Cryptography : Implementasi kemanan tambahan (jika nanti dibutuhkan )
    - Numpy : Digunakan untuk analisis data (perhitungan entropi)
    - Time : Mencatat waktu eksekusi algoritma, baik untuk enkripsi maupun dekripsi.
  1. *Text* Editor :

Visual Studio Code dan *Jupyter Notebook*

* 1. Sistem Operasi : Windows

### Bahan

1. Data set *file* :
   1. *File* PDF : Ukuran *file* 100 KB – 100 MB
   2. *File* Word : Ukuran *file* 100 KB – 100 MB
   3. *File* TXT : Ukuran *file* maksimal 300 *byte*
2. Literatur :

Jurnal atau artikel ilmiah tentang AES, ChaCha20 dan *Double Encrypt* (tahun 2020-2024)

1. *Spreadsheet* Analisis Data :

Format CSV (Comma Separated Values)

### Rancangan Penelitian

Pada penelitian ini, dilakukan tahapan yang dirancang secara sistematis untuk mencapai tujuan utama, yaitu Implementasi kombinasi Advanced Encryption Standard (AES) yang dimodifikasi dan ChaCha20. Proses penelitian dimulai dengan studi literatur untuk memahami konsep dasar dan pengembangan terbaru terkait algoritma kriptografi. Setelah itu, dilakukan pengumpulan data yang relevan sebagai dasar implementasi. Tahapan selanjutnya mencakup implementasi algoritma modifikasi AES serta kombinasi algoritma tersebut dengan ChaCha20. Untuk mengukur efektivitas dan efisiensi algoritma yang dikembangkan, pengujian dilakukan pada berbagai jenis *file* seperti PDF, Word, dan TXT. Analisis hasil dari pengujian ini memberikan dasar untuk menarik kesimpulan dan memberikan rekomendasi untuk pengembangan lebih lanjut.

*Gambar III.1 Diagram Alur Penelitian*

[Gambar III.1 Diagram Alur Penelitian](#_bookmark44) ini terdapat beberapa tahapan yaitu:

1. Mulai adalah tahap awal dari penelitian. Pada tahap ini menentukan tujuan dan menyusun langkah-langkah yang akan dilakukan dalam penelitian.
2. Studi literatur dilakukan pengumpulan referensi yang berkaitan dengan algoritma AES, ChaCha20 dan *Double Encrypt. Referensi* berupa jurnal, artikel ilmiah.
3. Pengumpulan data untuk mengumpulkan *file* atau data set yang akan digunakan untuk pengujian algoritma. Format *file* yang digunakan adalah PDF, Word dan TXT.
4. Implementasi modifikasi AES, pada tahap ini algoritma AES dimodifikasi dengan menambahkan langkah *AddRoundkey* sebelum *Mixcolumns*. Tujuannya adalah untuk meningkatkan keamanan dan kecepatan enkripsi dibandingkan dengan AES standar.
5. Implementasi kombinasi AES modifikasi dan ChaCha20, menggabungkan algoritma AES yang sudah dimodifikasi dengan ChaCha20 untuk melakukan *double encypt*. Di mana proses ini bertujuan untuk meningkatkan keamanan *file* lebih baik daripada menggunakan *single encrypt*.
6. Pengujian algoritma, untuk menguji performa algoritma yang sudah diimplementasikan, di mana data set yang dikumpulkan sebelumnya itu digunakan untuk uji coba. Parameter yang diukur pada pengujian ialah

waktu enkripsi dan *dekripsi,* kecepatan enkripsi dan *dekripsi*, dan hasil pengujian ini akan digunakan untuk mengevaluasi kinerja algoritma.

1. Analisis hasil, data pengujian yang sudah terkumpul akan dianalisis. Analisis dilakukan untuk membandingkan performa AES standar, AES yang sudah dimodifikasi, dan kombinasi AES modifikasi dan ChaCha20. Tujuannya sendiri untuk mengetahui algoritma mana yang lebih efisien.
2. Selanjutnya ke kesimpulan dan saran, pada tahap ini merangkum hasil penelitian termasuk kelebihan dan kekurangan algoritma yang sudah diuji. Saran juga untuk pengembangan penelitian lebih lanjut
3. Selesai

### Studi Literatur

Studi literatur dilakukan untuk memahami konsep dasar dan perkembangan terbaru terkait algoritma kriptografi yang digunakan dalam penelitian ini, yaitu Advanced Encryption Standard (AES) dan ChaCha20. Proses ini melibatkan pengumpulan referensi dari jurnal, paper, dan buku yang membahas metode enkripsi, keamanan data, serta teknik evaluasi seperti entropi dan analisis performa. Literatur yang dikaji juga mencakup penelitian sebelumnya tentang algoritma kriptografi untuk keamanan *file*, sehingga memberikan dasar teori yang kuat untuk modifikasi algoritma dan penerapan metode double encryption dalam penelitian ini.

### Pengumpulan Data

Data yang digunakan dalam penelitian ini dibuat sendiri dengan mengikuti ketentuan yang telah ditentukan. Format data meliputi *file* PDF, Word, dan TXT dengan ukuran bervariasi, mulai dari 100 KB hingga 100 MB untuk *file* PDF dan Word, serta hingga 300 *byte* untuk *file* TXT. Data ini dirancang sedemikian rupa agar sesuai dengan kebutuhan pengujian algoritma enkripsi, seperti mengukur waktu enkripsi, waktu *dekripsi*, kecepatan, dan tingkat keacakan menggunakan analisis entropi. Variasi format dan ukuran data ini

bertujuan untuk memastikan bahwa pengujian mencakup berbagai jenis isi *file*.

### Implementasi Modifikasi AES

Pengembangan algoritma *Advanced Encryption Standard* (AES) dilakukan dengan memodifikasi. Modifikasi dilakukan dengan menambahkan *AddRoundkey* sebelum *Mixcolumns*. Penambahan ini bertujuan untuk memperkuat keamanan dan meningkatkan kompleksitas pada proses enkripsi, sehingga lebih sulit untuk diretas dengan pihak tidak berwenang. Berikut adalah proses alur dari modifikasi algoritma AES.

*Gambar III.2 Diagram Proses Enkripsi AES*

Pada [Gambar III.2 Diagram Proses Enkripsi AES](#_bookmark48) ini digunakan untuk mengamankan *file* dengan melakukan tahap enkripsi terhadap data *plainteks* yang menggunakan kunci tertentu. Dalam penelitian ini, modifikasi dilakukan dengan menambahkan langkah *AddRoundkey* untuk meningkatkan tingkat keamanan dan kompleksitas enkripsi. Berikut langkah-langkah proses modifikasi AES .

1. Input *file* dan kunci

Proses enkripsi dimulai dengan memasukkan *file* input (*file* asli yang akan dienkripsi) dan kunci enkripsi. *File* input dibaca dalam mode biner, diubah menjadi *byte array*, lalu dipecah menjadi blok-blok data berukuran 16 *byte*. Jika panjang data tidak habis dibagi 16 *byte, padding* dilakukan dengan metode PKCS#7 untuk melengkapinya.

Contoh :

* 1. *File* input : contoh.pdf

Isi *file* dalam biner : [0x25, 0x50, 0x44, 0x46, ...]

*Byte array* : [37, 80, 68, 70, 10, 0, 45, 78, ...]

* 1. Kunci enkripsi :

Kunci : surabayaitelkom

ASCII : [115, 117, 114, 97, 98, 97, 121, 97, 105, 116, 116,

101, 108, 107, 111, 109]

* 1. Matriks data *file* :

|  |  |  |  |
| --- | --- | --- | --- |
| [37, | 80, | 68, | 70] |
| [10, | 0, | 45, | 78] |
| [255, | 100, | 32, | 12] |
| [80, | 50, | 60, | 72] |

1. *AddRoundkey*

Merupakan tahap awal proses enkripsi, di mana *plainteks* ini digabungkan dengan kunci awal yang menggunakan operasi logika yaitu XOR. Di mana blok data *file* digabungkan dengan kunci awal (W0-W3) menggunakan operasi XOR untuk menambahkan lapisan keamanan awal.

Contoh :

* 1. Proses XOR :

37 XOR 115 = 86

80 XOR 117 = 37

68 XOR 114 = 54

70 XOR 97 = 39

* 1. Matriks setelah *AddRoundkey* :

[86, 37, 54, 39]

[108, 97, 76, 47]

[150, 16, 84, 97]

[28, 89, 83, 37]

1. Proses ronde 1-9
   1. *SubBytes*

Pada proses ini, nantinya setiap *byte* pada matriks *state* ( data yang diperoleh *AddRounKey*) akan diubah menggunakan tabel substitusi yang biasa disebut dengan S-Box. *Tranformasi* ini bertujuan untuk meningkatkan keacakan data agar sulit diprediksi.

Contoh proses :

*Byte* 37 diganti dengan nilai dari tabel S-Box jadi 120

* 1. *ShiftRows*

Setiap baris dalam matriks *state* nantinya digeser ke kiri pada baris pertama tidak ada pergeseran, baris kedua di geser 1 *byte* ke kiri, baris ketiga ini digeser 2 *byte* ke kiri, dan terakhir baris keempat yang nantinya digeser 3 *byte* ke kiri. Contoh proses :

[244, 120, 99, 200] [244, 120, 99, 200]

[60, 107, 55, 175] [107, 55, 175, 60]

[12, 243, 172, 85] [172, 85, 12, 243]

[180, 97, 92, 65] [65, 180, 97, 92]

* 1. *AddRoundkey* (W4-W7)

Setelah *ShiftRows* dilakukan operasi XOR antara matriks *state* dengan kunci ronde berikutnya yaitu (W4-W7) Contoh proses :

*Byte* 244 XOR 115 = 135

*Byte* 120 XOR 117 = 13

Hasil *AddRounKey* : [135, 13, 17, 169]

[38, 98, 85, 12]

[55, 32, 82, 84]

[49, 53, 180, 193]

* 1. *MixColumns*

Setiap kolom pada matriks *state* dimodifikasi dengan operasi matematis berbasis matriks yang menggunakan bidang *Galois* (GF). Di mana proses ini bertujuan untuk mencampur elemen-elemen dalam kolom dan meningkatkan difusi data. [2, 3, 1, 1]

[1, 2, 3, 1]

[1, 1, 2, 3]

[3, 1, 1, 2]

* 1. *AddRoundkey* (W36-W39)

Dilakukan setelah *MixColums* untuk menambahkan lapisan keamanan dengan kunci ronde tambahan (W36-W39).

1. Pada ronde terakhir, hanya dilakukan tiga langkah yaitu

*SubBytes, ShiftRows*, dan *AddRoundkey* tanpa *MixColumns.*

Proses modifikasi algoritma AES ini digunakan untuk mengenkripsi *file* dengan menambahkan langkah *AddRoundkey* tambahan, yang meningkatkan keamanan dengan memberikan lapisan XOR tambahan pada setiap ronde enkripsi.

* + 1. **Implementasi Kombinasi AES Modifikasi dan ChaCha20** Algoritma AES modifikasi dan ChaCha20 dengan menerapkan *double encypt.* Proses ini memakai AES modifikasi dengan memberikan perlindungan berbasis blok *chiper* yang kuat dengan memodifikasi *AddRoundkey* tambahan sebelum *MixColumns,* sedangkan ChaCha20 sebagai *stream cipher* yang cepat dan aman, meningkatkan kompleksitas keseluruhan dengan memanfaatkan proses enkripsi berbasis *keystream.*

Pada metode ini, setiap *file* menggunakan 1 kunci enkripsi. Kunci yang digunakan memiliki panjang awal 128-bit, yang kemudian diperluas menjadi 256-bit pada ChaCha20 dengan menambahkan padding nol (0x00) jika panjangnya kurang dari 256-bit. Proses

dimulai dengan mengenkripsi *file* menggunakan AES Modifikasi untuk menghasilkan cipherteks awal dalam format matriks 4x4. Cipherteks ini kemudian diubah menjadi byte stream untuk kompatibilitas dengan ChaCha20. Selanjutnya, ChaCha20 melakukan proses enkripsi kedua dengan memanfaatkan kunci 256- bit, nonce sepanjang 96-bit, dan counter. Hasil akhir adalah cipherteks final yang lebih aman karena telah melalui dua lapisan enkripsi. Penggunaan double encryption dengan satu kunci untuk setiap *file* memberikan keamanan yang lebih tinggi, di mana kombinasi kedua algoritma menciptakan perlindungan yang kuat terhadap berbagai jenis serangan kriptografi.

*Gambar III.3 Diagram Proses Double Encrypt*

Pada [Gambar III.3 Diagram Proses Double Encrypt](#_bookmark50) menunjukkan langkah-langkah enkripsi ganda. Proses dimulai dengan mengenkripsi *plainteks* menggunakan AES modifikasi untuk menghasilkan *chiperteks* AES. *Chiperteks* ini kemudian dienkripsi ulang menggunakan ChaCha20, yang melibatkan matriks 4x4, proses *quarter-round,* dan operasi XOR dengan *keystream*. Algoritma AES modifikasi dan ChaCha20 menggunakan kunci yang sama, yaitu kunci AES sepanjang 128-bit. berikut adalah tahapan alur di atas :

1. Input *plainteks* dan *key* dan enkripsi AES modifikasi
2. Input *cipherteks* AES ke ChaCha20

*Chiperteks* yang dihasilkan dari proses enkripsi pertama diubah menjadi input untuk algoritma ChaCha20. Ini dilakukan untuk menerapkan enkripsi berlapis yaitu *double encryption,* sehingga keamanan data akan meningkat secara signifikan. *Chiperteks* AES yang biasanya dalam format matriks (4x4) ini diubah ke format *byte stream* untuk kompatibilitas dengan algoritma ChaCha20. Byte stream ini memungkinkan untuk memproses data, sesuai dengan *stream chiper*.

Contoh :

[115, 75, 189, 234, 201, 97, 58, 14, 245, 142, 12, 120, 53, 190,

201, 20] data ini sudah dalam bentuk *byte stream*.

1. Validasi panjang *key* (256-bit *Key*)

Ini untuk memastikan bahwa kunci enkripsi untuk ChaCha20 sesuai dengan spesifikasi algoritma, yaitu panjangnya harus tepat 256 bit (32 *byte*). Kunci ini untuk menghasilkan *keystream* yang beda dalam proses enkripsi. Jika panjang ke kurang dari 256 bit, ditambahkan *padding* berupa *byte* nol (0x00) untuk memenuhi panjang. Jika *key* yang lebih panjang, *key* akan dipotong hingga hanya menyisakan 256 bit. *Key* kemudian dibagi menjadi 8 bagian ( di mana masing-masing 32 bit) untuk digunakan dalam proses pembentukan matriks awal (*state* matriks)

Contoh :

* 1. *Key* input “surabayaittelkom”

[115, 117, 114, 97, 98, 97, 121, 97, 105, 116, 116, 101, 108,

107, 111, 109]

* 1. Panjang *key* adalah 16 *byte* (128 bit), sehingga tidak memenuhi syarat panjang 256 bit. Jadi ditambahkan *padding* nol (0x00) hingga mencapai 32 *byte:*

[115, 117, 114, 97, 98, 97, 121, 97, 105, 116, 116, 101, 108,

107, 111, 109, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

1. Membuat *Initial* State Matriks 4x4

Membentuk *state* matriks awal yang menjadi dasar operasi dalam ChaCha20. Matriks ini berisi kunci, *counter*, dan *nonce.* Matriks 4x4 dalam ChaCha20 terdiri 16 elemen (masing-masing 32 bit) Contoh :

* 1. 4 elemen konstanta tetap “*expand* 32-byte k”4 elemen 32 bit [1634760805, 857760878, 2036477234, 1797285236]
  2. Elemen kunci 256-bit dibagi menjadi 8 elemen 32-bit: [1936028945, 1885434721, 1701147239, 1647539057,

1225269481, 1226844037, 1214606441, 0]

* 1. 1 *Counter*: Nilai awal 0.
  2. 3 Elemen Nonce: Nilai *nonce* yang unik untuk sesi enkripsi: Contoh *nonce* : [12345678, 87654321, 11223344]
  3. Matriks yang dibentuk

[1634760805, 857760878, 2036477234, 1797285236]

[1936028945, 1885434721, 1701147239, 1647539057]

[1225269481, 1226844037, 1214606441, 0]

[0, 12345678, 87654321, 11223344]

Baris ke 1 konstanta, ke 2 *key,* ke 3 *key* dan ke 4 *counter* dan

*nonce*

1. Operasi Quarter Round (20 iterasi)

Melakukan proses pencampuran data di dalam matriks untuk meningkatkan kerandoman dan keamanan *keystream* yang dihasilkan. Quarter Round adalah inti dari algoritma ChaCha20. Quarter Round dilakukan pada 4 elemen tertentu dari matriks. Penjumlahan Modular (mod 2³²): Menambah elemen dalam matriks untuk mencampur data. Operasi XOR: Mencampur data antar elemen melalui operasi logika XOR. Rotasi Bit (ROTL): Menggeser bit elemen untuk mendistribusikan informasi secara acak. Setelah 20 iterasi (10 putaran double round), matriks akan diubah menjadi bentuk yang sangat acak.

1. *Counter* +=1

*Counter* bertambah untuk setiap blok data yang diproses, memastikan *keystream* yang dihasilkan unik untuk setiap blok. Setelah satu blok data selesai diproses, *counter* di matriks bertambah sebesar 1.

Contoh :

*Counter* baru : [1, 12345678, 87654321, 11223344]

1. XOR *keystream* dengan *chiperteks* AES

Menggabungkan *keystream* hasil ChaCha20 dengan *cipherteks AES* melalui operasi XOR untuk menghasilkan *final ciphertext*. XOR dipilih karena sifatnya yang sederhana namun aman dan mudah dibalik. *Keystream* yang dihasilkan dari matriks ChaCha20 di-XOR-kan dengan *cipherteks AES*.

Operasi XOR:

Jika *ciphertext* = *keystream* XOR *plaintext*, maka *dekripsi*

dilakukan dengan cara *plaintext* = *keystream* XOR *ciphertext*.

* + - 1. **Proses *Dekripsi* Double Encryption**

Proses *dekripsi* dilakukan untuk mengembalikan data yang telah terenkripsi ke bentuk aslinya atau *plainteks*. Di mana pada tahap ini, *chiperteks* hasil dari kombinasi algoritma AES modifikasi dan Chacha20 akan *didekrpsi* secara urut. Proses nantinya akan dimulai dengan *mendekripsi chiperteks* dengan ChaCha20 untuk mendapatkan kembali *chiperteks* AES modifikasi, kemudian dilanjutkan dengan *mendekripsi chiprteks* AES modifikasi untuk memperoleh *plainteks* asli.

Pada [Gambar III.4 Proses Dekripsi](#_bookmark51) Diagram alur di atas menjelaskan proses *dekripsi* dari kombinasi algoritma ChaCha20 dan modifikasi AES. Proses dimulai dengan memasukkan cipherteks hasil enkripsi sebagai input. Langkah pertama adalah men*dekripsi* cipherteks menggunakan algoritma ChaCha20 untuk menghasilkan keystream. Keystream ini kemudian dikombinasikan dengan cipherteks menggunakan operasi XOR untuk mendapatkan cipherteks AES.

Selanjutnya, proses ekspansi kunci dilakukan untuk menghasilkan kunci yang akan digunakan pada setiap ronde *dekripsi* AES. Proses *dekripsi* AES terdiri dari tiga tahapan utama: ronde terakhir (Round 10) yang melibatkan *AddRoundKey*, InvShiftRows, dan InvSubBytes; ronde utama (Round 9-1) yang melibatkan *AddRoundKey*, InvMixColumns, *AddRoundKey, InvShiftRows,* dan *InvSubBytes* tambahan; dan ronde awal (Round 0) yang menggunakan *AddRoundKey* (W0-W3). Hasil akhir dari seluruh proses ini adalah plainteks, yaitu data asli yang berhasil di*dekripsi*. Kombinasi kedua algoritma ini memastikan keamanan data yang lebih kuat melalui enkripsi ganda.

*Gambar III.4 Proses Dekripsi*

### Implementasi Antarmuka Pengguna (UI)

Antarmuka pengguna (UI) dirancang untuk mempermudah interaksi, berfungsi sebagai media interaktif yang memungkinkan pengguna untuk memilih file, memasukkan kunci enkripsi, serta memulai proses enkripsi dan dekripsi dengan mudah. Implementasinya menggunakan Flask sebagai framework backend untuk menangani

proses enkripsi dan dekripsi, sedangkan HTML digunakan untuk membuat form input file dan kunci enkripsi. CSS digunakan untuk mempercantik tampilan agar antarmuka lebih menarik, jelas, dan mudah digunakan oleh pengguna dalam proses enkripsi dan dekripsi file menggunakan kombinasi algoritma ChaCha20 dan AES yang telah dimodifikasi.

*Gambar III.5 Desain Antarmuka Pengguna Enkripsi dan Dekripsi*

Pada [Gambar III.5 Desain Antarmuka Pengguna Enkripsi dan](#_bookmark53) [Dekripsi](#_bookmark53) Gambar tersebut menampilkan desain antarmuka pengguna (UI) dari sebuah aplikasi berbasis web yang digunakan untuk enkripsi dan *dekripsi file*. Antarmuka ini memiliki beberapa elemen input pengguna untuk mengunggah *file* dan memilih opsi enkripsi atau *dekripsi*. Terdapat tiga kolom utama pada desain ini: pertama, Pilih *File*, untuk pengguna memilih *file* yang akan diproses. Kedua, Enkripsi atau *Dekripsi*, yang memberikan pilihan kepada pengguna apakah *file* yang dipilih akan dienkripsi atau di *dekripsi*, yang merupakan opsi penting untuk menentukan jenis operasi yang akan dilakukan pada *file* tersebut. Ketiga, Key, di mana pengguna diminta untuk memasukkan kunci yang diperlukan untuk proses enkripsi atau *dekripsi*. Kunci ini memainkan peran penting dalam menjaga keamanan data yang diproses. Di bagian bawah antarmuka, terdapat

tombol Proses yang akan menjalankan operasi sesuai dengan pilihan pengguna, baik itu enkripsi maupun *dekripsi*.

### Pengujian Algoritma

Menjelaskan pengujian yang dilakukan untuk mengevaluasi kinerja dan keamanan algoritma berdasarkan parameter seperti entropi, waktu enkripsi dan *dekripsi*, serta kecapatan.

### Pengujian Entropi

Pengujian kekuatan algoritma dilakukan dengan mengukur tingkat keacakan data hasil enkripsi menggunakan analisis entropi. Keacakan data merupakan indikator penting dalam menentukan seberapa baik algoritma enkripsi menyamarkan pola asli dari data. Semakin tinggi tingkat keacakan, semakin sulit bagi pihak yang tidak berwenang untuk menganalisis pola atau memprediksi isi data terenkripsi. Pengujian ini dilakukan dengan menghitung nilai entropi menggunakan distribusi probabilitas setiap simbol dalam cipherteks. Nilai entropi yang mendekati maksimum menunjukkan bahwa distribusi simbol dalam cipherteks hampir merata, menandakan bahwa algoritma telah berhasil menghasilkan data yang acak. Data dengan keacakan tinggi lebih tahan terhadap serangan berbasis statistik, sehingga pengujian ini memberikan evaluasi kuantitatif terhadap kekuatan keamanan algoritma yang diuji. Hasil pengujian dibandingkan antara AES Standar, AES Modifikasi, Chacha20 dan Double Encryption untuk menunjukkan efektivitas modifikasi dalam meningkatkan keamanan.

### Pengujian Waktu dan Kecepatan

Menguji algoritma AES modifikasi dan *Double encrypt* AES modifikasi dan ChaCha20. Pengujian ini dilakukan untuk menilai kinerja algoritma dalam berbagai parameter. Penelitian ini akan menganalisis parameter pengujian menggunakan matriks dan formula berikut :

* + - * 1. Waktu enkripsi :

mengukur waktu yang dibutuhkan untuk mengenkripsi data dari

*plainteks* menjadi *chiperteks* dijelaskan dalam persamaan [(III-1)](#_bookmark55).

|  |  |
| --- | --- |
| 𝐸𝑡 = 𝐸𝑁𝑡 – 𝑆𝑇𝑡 | (III-1) |

Di mana:

𝐸𝑡 = waktu enkripsi dalam mili detik (ms)

𝐸𝑁𝑡 = waktu selesai enkripsi

𝑆𝑇𝑡 = waktu mulai enkripsi

Waktu ini akan dihitung untuk setiap *file* yang diuji, dan hasilnya nanti akan memberikan gambaran mengenai kecepatan algoritma dalam melakukan proses enkripsi.

* + - * 1. Waktu *dekripsi* :

Mengukur waktu yang dibutuhkan untuk *mendekripsi* data dari

*chiperteks* menjadi *plainteks* dijelaskan pada persamaan [(III-2)](#_bookmark56).

Dimana:

𝐷𝑡 = waktu *dekripsi* (ms)

𝐸𝑁𝑡 = waktu selesai *dekripsi*

𝑆𝑇𝑡 = waktu mulai *dekripsi*

Hasil waktu *dekripsi* ini juga perlu untuk masing-masing algoritma karena ini menganalisis perbandingan kinerja antara algoritma enkripsi.

* + - * 1. Kecepatan :

Menunjukkan jumlah data dalam *kilobyte* yang dapat dienkripsi

Et = waktu enkripsi atau *dekripsi* (s)

### Pengujian pada Berbagai Jenis *File*

*File* PDF yang digunakan dalam pengujian dapat berisi elemen seperti teks, gambar, dan tabel yang semuanya akan dienkripsi secara utuh karena algoritma bekerja pada level data biner. *File* PDF dengan ukuran 100 KB, 1 MB, 10 MB, hingga 100 MB diuji untuk memastikan algoritma dapat bekerja pada dokumen berukuran kecil hingga besar. Hal serupa juga berlaku untuk *file* Word dengan ukuran yang sama, meskipun memiliki struktur internal yang lebih kompleks dibandingkan PDF. *File* TXT dengan ukuran maksimal 300 *byte*, yang biasanya hanya berisi teks sederhana, juga dapat dienkripsi ChaCha20 (*double encrypt*). Di mana pengujian ini bertujuan untuk

mengukur kinerja algoritma. Berikut ini adalah contoh hasil pengujian terhadap *file* PDF :

*Tabel III.2 Evaluasi Hasil*

Pada [Tabel III.2 Evaluasi Hasil](#_bookmark60) menunjukkan hasil evaluasi simulasi kinerja algoritma enkripsi dan *dekripsi*. pada Nilai-nilai tersebut memberikan gambaran awal mengenai performa beberapa metode, termasuk AES Standar, ChaCha20, AES Modifikasi, dan kombinasi AES Modifikasi dengan ChaCha20. Hasil ini digunakan sebagai contoh output yang diharapkan dari model algoritma yang akan dirancang, ini belum merupakan data dari eksperimen nyata.

# DAFTAR PUSTAKA

Alimuddin, A., Juntak, J. N. S., Jusnita, R. A. E., Murniawaty, I., & Wono, H. Y. (2023). Teknologi dalam pendidikan: Membantu siswa beradaptasi dengan revolusi industri 4.0. *Journal on Education*, *5*(4), 11777–11790.

Almukhlifi, R., & Vora, P. L. (2020). Linear cryptanalysis of reduced-round simon using super rounds. *Cryptography*, *4*(1), 1–34. https://doi.org/10.3390/cryptography4010009

Alyas, H. H., & Abdullah, A. A. (2021). Enhancement the ChaCha20 Encryption Algorithm Based on Chaotic Maps. *Lecture Notes in Networks and Systems*, *201*, 91–107. https://doi.org/10.1007/978-981-16-0666-3\_10

Anindita, M., & 18219086, A. A. (2023). *Analisis Perbandingan Algoritma AES dan ChaCha20-Poly1305 dalam Enkripsi Konten Livestreaming*.

Aryanto, M. B., Tahir, M., Devita, S. I., Mustofa, Z. N., Ainiyah, Q., & Sundoro,

S. (2023). Implementasi Enkrip Dan Dekrip File Menggunakan Metode Advance Encryption Standard (AES-128). *Jurnal Ilmiah Sistem Informasi Dan Ilmu Komputer*, *3*(1), 89–104.

Assyahid, M. H. (2024). *Implementasi interpolasi pixel wadah citra dan enkripsi aes pada steganografi least significan bit (LSB)*. https://repository.uinjkt.ac.id/dspace/handle/123456789/76727

Barbero, S., Bazzanella, D., & Bellini, E. (2022). Rotational Cryptanalysis on ChaCha Stream Cipher. *Symmetry*, *14*(6). https://doi.org/10.3390/sym14061087

Belvin Tannadi. (2022). *Ilmu Crypto* (Digital). Jakarta : Elex Media Komputindo.

Cristy, N., & Riandari, F. (2021). *Niolinda Cristy 1 , Fristi Riandari 2 [Implementasi Metode Advanced Encryption Standard (AES 128 Bit) Untuk Mengamankan Data Keuangan*. *4*(2), 75.

Djong, H. S., & Siswanto, S. (2022). Implementasi Kriptografi Dengan Menggunakan Metode RC4 dan AES-256 Untuk Mengamankan File Dokumen pada PT Varnion Technology Semesta. *Prosiding Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI)*, *1*(1), 149–158.

Handoyo, J., & Subakti, Y. M. (2020). Keamanan Dokumen Menggunakan Algoritma Advanced Encryption Standard (AES). *Jurnal SITECH : Sistem Informasi Dan Teknologi*, *3*(2), 143–152. https://jurnal.umk.ac.id/index.php/sitech/article/view/5865

Hena, N., & Kusumaningsih, D. (2024). Pengamanan File Pada Sistem Master Vendor Berbasis WEB Menggunakan Algoritma AES Pada Trinitiland. *Prosiding Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI)*, *3*(2), 30–37.

Hidayatulloh, N. W., Tahir, M., Amalia, H., Basyar, N. A., Prianggara, A. F., & Yasin, M. (2023). Mengenal Advance Encryption Standard (AES) sebagai Algoritma Kriptografi dalam Mengamankan Data. *Digital Transformation Technology*, *3*(1), 1–10.

Kebande, V. R. (2023). Extended-Chacha20 Stream Cipher With Enhanced Quarter Round Function. *IEEE Access*, *11*, 114220–114237. https://doi.org/10.1109/ACCESS.2023.3324612

Kim, B. G., & Wong, D. (2023). Smart Contract-based Secure Verifiable Random Function using ChaCha20 Sequence in Blockchain. *ACM International Conference Proceeding Series*, 41–51. https://doi.org/10.1145/3638025.3638028

Kurnia Mahesa Naibaho, A., Putri, A., & Fikri Husaini, Y. (2021). Analisis Sistem Keamanan Informasi Chat Online dengan Menggunakan Algoritma AES. In *Ilmu Komputer* (Vol. 6, Issue 2).

Luong, T. T. (2023). Strengthening AES Security Through Key-Dependent ShiftRow and AddRoundKey Transformations Utilizing Permutation. In *IJACSA) International Journal of Advanced Computer Science and Applications* (Vol. 14, Issue 11). [www.ijacsa.thesai.org](http://www.ijacsa.thesai.org/)

Muhammed, R. K., Rashid Aziz, R., Hassan, A. A., Aladdin, A. M., Saydah, S. J., Rashid, T. A., & Hassan, A. (2024). *Comparative Analysis of AES, Blowfish, Twofish, Salsa20, and ChaCha20 for Image Encryption*.

Olivia Putri Irine Irawan, B., Tahir, M., Ayu Windrastuti, N., Yurina Cholili, D., Mulaikah, D., Batsul Mushofi Septian wachid, A., & Pendidikan Informatika,

P. (2023). Implementasi Kriptografi Pada Keamanan Data Menggunakan Algoritma Advance Encryption Standard (AES). *Jurnal Simantec*, *11*(2), 167–

174. https://doi.org/10.21107/SIMANTEC.V11I2.20034

Pangestu, G. Y., Hadiana, A. I., & Sabrina, P. N. (2022). Kriptografi Untuk Enkripsi Ganda Pada Gambar Menggunakan Algoritma AES (Advanced Encryption Standard) Dan RC5 (Rivest Code 5). *Vol*, *1*, 25–32.

Putri, A. E., Kartikadewi, A., & Rosyid, L. A. A. (2021). Implementasi Kriptografi dengan Algoritma Advanced Encryption Standard (AES) 128 Bit dan Steganografi menggunakan Metode End of File (EOF) Berbasis Java Desktop

pada Dinas Pendidikan Kabupaten Tangerang. *Appl. Inf. Syst. Manag*, *3*(2), 69–78.

Saripa, S. (2023). Implementation Implementation of a File Security System Using the AES Algorithm to Secure Personal Files: Implementasi Sistem Keamanan File Menggunakan Algoritma AES untuk Mengamankan File Pribadi. *Progressive Information, Security, Computer, and Embedded System*, *1*(2), 138–148.

Sekar Putri Ananda, Saepul Lukman, & Irfan. (2022). *Analisa Metode Kriptogra Modern Advance Encryption Standar (AES) 128 Bit dalam Mengenkripsi dan Mendekripsi File Dokumen Digital*. https://doi.org/10.32409/jikstik.21.3.2973

Soesanto, E., Purba, L. M., Aprilia, B., Putra, D. R., & Putri, S. D. (2023). Implementasi Objek Vital, Pengamanan File dan Pengamanan Cyber di PT Pertamina. *IJM: Indonesian Journal of Multidisciplinary*, *1*(1).

Surbakti, P. S., & Purwanto, P. (2024). Implementasi Kriptografi Dengan Menggunakan Algoritma AES-128 Untuk Enkripsi dan Dekripsi File Dokumen Berbasis Web Pada Law Office Erdi Surbakti, SH & Rekan. *Prosiding Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI)*, *3*(1), 1–8.

Syahputra Sianipar, J., Budi Nuugroho, N., Mariami, I., Informasi, S., & Triguna Dharma, S. (2024). Pengamanan Data Gaji Karyawan Dengan Menggunakan Metode Advanced Encryption Standard (AES). *Jurnal Sistem Informasi Triguna Dharma (JURSI TGD)*, *3*(1), 35–45. https://doi.org/10.53513/JURSI.V3I1.5653

Wahyu Putra, M. R. F. T. H. (2023, July 2). *Implementasi Algoritma Advanced Encryption Standard Untuk Kemanan Dokumen*. https://journal.grahamitra.id/index.php/jurikti/article/view/55/181

Wahyudi, E. N., Ardhianto, E., Handoko, W. T., Murti, H., Supriyanto, E., Lestariningsih, E., & Redjeki, R. S. (2024). Peningkatan Keamanan Data Melalui Teknik Super Enkripsi Menggunakan Algoritma Vigenere dan Caesar. *Jurnal Informatika Polinema*, *10*(3), 315–322. https://doi.org/10.33795/JIP.V10I3.5131

Widyawan, D., & Imelda, I. (2021). Pengamanan File Menggunakan Kriptografi Dengan Metode Aes-128 Berbasis Web Di Komite Nasional Keselamatan Transportasi. *SKANIKA: Sistem Komputer Dan Teknik Informatika*, *4*(1), 15– 22.

**IMPLEMENTASI KOMBINASI CHACHA20 DAN MODIFIKASI AES MELALUI *ADDROUNDKEY* PADA**

**STUDI KASUS KEAMANAN *FILE***

**Proposal Tugas Akhir Mata Kuliah Penulisan Proposal**

## (CAK4EAB2)

Disusun oleh:

## EMILIA RAMONA 1203210017

**PROGRAM STUDI SARJANA INFORMATIKA DIREKTORAT KAMPUS SURABAYA**

**UNIVERSITAS TELKOM SURABAYA TAHUN 2025**

# ABSTRAK

Keamanan data menjadi tantangan besar di era Revolusi Industri 4.0, di mana informasi penting perlu dilindungi dari akses tidak sah. Ancaman serangan siber dan peretasan mengharuskan pengembangan solusi keamanan yang lebih efektif. Penelitian ini bertujuan untuk mengimplementasikan dan mengevaluasi kombinasi algoritma ChaCha20 dan modifikasi *Advanced Encryption* Standard (AES) dengan penambahan langkah *AddRoundKey* sebelum *MixColumns* untuk meningkatkan keamanan *file*. Metode yang digunakan adalah enkripsi ganda, dengan AES modifikasi sebagai *block cipher* yang kuat dan ChaCha20 sebagai *stream cipher* yang cepat. Penelitian ini mencakup implementasi algoritma dengan menggunakan pustaka kriptografi untuk menguji kinerja algoritma melalui parameter seperti waktu enkripsi, waktu *dekripsi*, kecepatan enkripsi, kecepatan *dekripsi*, dan tingkat keacakan data menggunakan analisis entropi. *File* dengan berbagai format dan ukuran, PDF, Word, dan TXT, digunakan sebagai data pengujian untuk mengevaluasi efisiensi algoritma. Selain itu, antarmuka pengguna berbasis web dirancang untuk mempermudah proses enkripsi dan *dekripsi*, melibatkan langkah- langkah seperti pemilihan *file* dan *input* kunci enkripsi. Penelitian ini diharapkan menunjukkan bahwa kombinasi AES modifikasi dan ChaCha20 mampu meningkatkan keamanan data serta memberikan efisiensi yang lebih baik dalam perlindungan *file*.

**Kata Kunci**: Keamanan *File*, *Advanced Encryption Standard* (AES), ChaCha20, Enkripsi Ganda, *AddRoundkey*, *Entropi*.

# ABSTRACT

*Data security has become a major challenge in the era of the Industrial Revolution 4.0, where critical information must be protected from unauthorized access. Cyberattacks and hacking threats demand the development of more effective security solutions. This research aims to implement and evaluate the combination of the ChaCha20 algorithm and a modified Advanced Encryption Standard (AES) by adding an AddRoundKey step before MixColumns to enhance file security. The method used is double encryption, with the modified AES as a robust block cipher and ChaCha20 as a fast stream cipher. This study involves implementing the algorithms using cryptographic libraries to test their performance through parameters such as encryption time, decryption time, encryption speed, decryption speed, and data randomness level using entropy analysis. Files of various formats and sizes, including PDF, Word, and TXT, are utilized as test data to evaluate the efficiency of the algorithms. Additionally, a web-based user interface is designed to simplify the encryption and decryption process, involving steps such as file selection and encryption key input. This research is expected to demonstrate that the combination of modified AES and ChaCha20 can enhance data security and provide better efficiency in file protection.*

***Keywords****: File Security, Advanced Encryption Standard (AES), ChaCha20, Double Encryption, AddRoundkey, Entropy*

# KATA PENGANTAR

Assalamu’alaikum Warrahmatullahi Wabarakatuh, puja puji syukur ke hadirat Allah SWT yang telah memberikan nikmat dan karunia-Nya yang sangat luar biasa kepada hamba – hambanya. Tidak lupa shalawat serta salam kita curahkan atas bimbingan nabi kita nabi Muhammad SAW yang telah membawa cahaya islam ke dalam dunia ini dan menjadi panutan bagi setiap umatnya hingga akhir zaman.

Setelah melalui berbagai proses kerja keras, penyusunan skripsi yang berjudul “Implementasi Kombinasi ChaCha20 dan Modifikasi AES melalui *AddRoundkey* Pada Studi Kasus Keamanan *File*” akhirnya dapat terselesaikan. Selain sebagai salah satu syarat memperoleh gelar S-Kom, penulis berharap juga skripsi ini dapat bermanfaat bagi semua pihak yang berkepentingan. Oleh karena itu, penulis ingin mengungkapkan rasa terima kasih kepada pihak-pihak yang telah memberikan dukungan dan doa sehingga penulis dapat menyelesaikan skripsi ini tepat pada waktunya. Beberapa pihak tersebut antara lain:

1. **Kepada Mama** yang sangat penulis cintai yang tidak pernah lupa selalu memberikan dukungannya dan doa agar diberikan kemudahan dan kelancaran dalam penyusunan skripsi ini sehingga penulis dapat menyelesaikan studi Informatika di Universitas IT Telkom Surabaya.
2. **Kepada Alm.Ibu Suhartatik**, yang penulis cintai dan sayangi, yang telah memberikan motivasi untuk saya.
3. **Kepada Saudara Kembar**, yang selalu mendukung, terima kasih atas bantuan dan waktunya dalam memberikan masukan terkait susunan skripsi ini.
4. **Keluarga Besar penulis** yang telah memberikan motivasi dan kepercayaan besar sehingga penulis bisa mewujudkan impian untuk meraih gelar S-Kom.
5. **Dosen Pembimbing**, Pak Rizky Fenaldo Maulana, S.Kom., M.Kom. dan Tanzila Mustaqim, S.Kom., M.Kom. Yang telah memberikan solusi, waktu, dan bimbingan penuh kesabaran selama proses penyusunan skripsi ini.
6. **Teman-teman dekat**, termasuk Jus’an, Erica Dyah, Qori Emalia, Ardina Firdatana, dan Elma Saskia yang selalu memberikan semangat, mendengarkan keluh kesah, serta menemani dalam berbagai kondisi selama penulisan skripsi.
7. **Semua pihak lainnya**, yang tidak dapat disebutkan satu per satu, atas bantuan dan dukungannya selama proses ini.
8. **Diri penulis sendiri**, atas perjuangan dan kerja keras hingga akhirnya skripsi ini selesai tepat waktu.

Penulis menyadari bahwa skripsi ini masih jauh dari kesempurnaan. Oleh karena itu, saran dan kritik yang membangun sangat diharapkan untuk memperbaiki skripsi ini ke depannya.

Terima kasih.

### Wassalamu’alaikum Warrahmatullahi Wabarakatuh.

Surabaya, 24 Desember 2024 Penulis

Emilia Ramona

# DAFTAR ISI

[LEMBAR PERSETUJUAN ii](#_bookmark0)

[ABSTRAK iii](#_bookmark1)

[ABSTRACT iv](#_bookmark2)

[KATA PENGANTAR v](#_bookmark3)

[DAFTAR ISI vii](#_bookmark4)

[DAFTAR TABEL ix](#_bookmark5)

[DAFTAR GAMBAR x](#_bookmark6)

[DAFTAR RUMUS xi](#_bookmark7)

[BAB I PENDAHULUAN 1](#_bookmark8)

* 1. [Latar Belakang 1](#_bookmark9)
  2. [Rumusan Masalah 4](#_bookmark10)
  3. [Tujuan Penelitian 4](#_bookmark11)
  4. [Batasan dan Asumsi Penelitian 5](#_bookmark12)
  5. [Manfaat Penelitian 5](#_bookmark13)
     1. [Manfaat Akademis 5](#_bookmark14)
     2. [Manfaat Ilmiah 5](#_bookmark15)
  6. [Sistematika Penulisan 6](#_bookmark16)

[BAB II LANDASAN TEORI 7](#_bookmark17)

* 1. [Literatur Terkait Teori 7](#_bookmark18)
  2. [Kajian Teori 10](#_bookmark20)
     1. [Kriptografi 10](#_bookmark21)
     2. [AES (Advanced Encryption Standard) 12](#_bookmark24)
     3. [Chacha20 20](#_bookmark32)
     4. [Entropi 22](#_bookmark35)

[BAB III METODOLOGI PENELITIAN 24](#_bookmark37)

* 1. [Sistematika Penyelesaian Masalah 24](#_bookmark38)
  2. [Alat dan Bahan 24](#_bookmark39)
     1. [Alat 24](#_bookmark40)
     2. [Bahan 25](#_bookmark41)
  3. [Rancangan Penelitian 25](#_bookmark42)
     1. [Studi Literatur 27](#_bookmark45)
     2. [Pengumpulan Data 27](#_bookmark46)
     3. [Implementasi Modifikasi AES 28](#_bookmark47)
     4. [Implementasi Kombinasi AES Modifikasi dan ChaCha20 31](#_bookmark49)
     5. [Implementasi Antarmuka Pengguna (UI) 36](#_bookmark52)
     6. [Pengujian Algoritma 38](#_bookmark54)
     7. [Analisis Hasil 42](#_bookmark59)

[DAFTAR PUSTAKA 44](#_bookmark61)

# DAFTAR TABEL

[Tabel II.1 Literatur Penelitian Terkait Kriptografi 7](#_bookmark19)

[Tabel II.2 Panjang Kunci 13](#_bookmark25)

[Tabel II.3 Tabel Substitusi (S-Box) 16](#_bookmark28)

[Tabel II.4 Transformasi ShiftRows 17](#_bookmark29)

[Tabel III.1 Jenis, Ukuran, dan Isi File untuk Pengujian 40](#_bookmark58)

[Tabel III.2 Evaluasi Hasil 43](#_bookmark60)

# DAFTAR GAMBAR

[Gambar II.1 Alur Proses Enkripsi Dan Dekripsi 11](#_bookmark22)

[Gambar II.2 Proses Kunci Simetris 11](#_bookmark23)

[Gambar II.3.Transformasi ShiftRows 15](#_bookmark27)

[Gambar II.4 Proses AES 19](#_bookmark31)

[Gambar III.1 Diagram Alur Penelitian 26](#_bookmark43)

[Gambar III.2 Diagram Proses Enkripsi AES 28](#_bookmark48)

[Gambar III.3 Diagram Proses Double Encrypt 32](#_bookmark50)

[Gambar III.4 Proses Dekripsi 36](#_bookmark51)

[Gambar III.5 Desain Antarmuka Pengguna Enkripsi dan Dekripsi 37](#_bookmark53)

# DAFTAR RUMUS

[(II-1) Rumus AddRoundKey 14](#_bookmark26)

[(II-2) Operasi Perkalian MixColumns 17](#_bookmark30)

[(II-3) Rumus Keystream 22](#_bookmark33)

[(II-4) Rumus Proses Enkripsi XOR pada Keystream 22](#_bookmark34)

[(II-5) Entropi 22](#_bookmark36)

[(III-1) Pengujian Waktu Enkripsi 39](#_bookmark55)

[(III-2) Pengujian Waktu Dekripsi 39](#_bookmark56)

[(III-3) Pengujian Waktu Kecepatan 39](#_bookmark57)

# BAB I PENDAHULUAN

### Latar Belakang

Semakin meningkatnya teknologi yang semakin canggih membawa sebuah perubahan yang cukup signifikan. Perubahan tersebut dikenal dengan istilah Revolusi Industri 4.0 yang merupakan sebuah kemajuan teknologi yang sering digunakan oleh banyak orang. Revolusi industri 4.0 adalah sebuah data yang digerakkan dengan adanya campur tangan komputer menggunakan metode machine learning atau AI (Alimuddin et al., 2023). Menurut Klaus Schawab revolusi Industri 4.0 memiliki tantangan yang harus dihadapi salah satunya seperti keamanan *cyber* dan privasi data.

Maka dari itu, dengan semakin pesatnya informasi teknologi maka perlunya untuk mengamankan data atau *file*. Keamanan data atau *file* merupakan salah satu langkah yang mudah untuk melindungi sebuah informasi bersifat penting yang berasal dari akses tidak sah. Keamanan *file* ini sangat diperlukan pada era Revolusi Industri 4.0, karena maraknya serangan *cyber* dan peretasan komputer yang bisa mengganggu keberlangsungan suatu industri. Menurut (Mulyana*, 2016*), keamanan *file* merupakan salah satu aksi pencegahan dari sebuah penganggu pengguna *file* atau pengguna yang memakai jaringan yang tidak berkewajiban memiliki akses (Soesanto et al., 2023). Dengan semakin pesatnya dunia teknologi ini membuat keamanan *file* menjadi sangat penting karena untuk melindungi privasi dan informasi dari pihak yang tidak memiliki akses.

Keamanan *file* adalah serangkaian metode atau teknik yang untuk melindungi *file* dari akses yang tidak sah, seperti pencurian atau kerusakan. Tujuannya untuk menjaga informasi dalam *file* tetap rahasia, aman, dan hanya dapat di akses oleh pihak yang berwenang. Oleh karena itu, upaya untuk memastikan agar *file* terjaga dengan menggunakan metode ChaCha20 dan *Advanced Encryption Standard* (AES). ChaCha20 merupakan metode enkripsi yang cepat dan aman. ChaCha20 dipopulerkan oleh Daniel J.

Bernstein sebagai varian dari algoritma. Dalam algoritma ChaCha20 sangat populer dalam pengimplementasi enkripsi dikarenakan percampuran antara kecepatan dan keamanannya yang dapat diunggulkan. Chaca20 mempunyai desain yang hampir sama dengan Salsa20, pada algoritma ChaCha20 menggunakan kunci 256-bit yang mempunyai 8 putaran (Kebande, 2023).

Sedangkan AES (*Advanced Encryption Standard)* adalah sebuah algoritma yang melakukan transformasi terhadap blok data yang menggunakan kunci enkripsi yang sama. AES bekerja pada sistem yang melibatkan operasi matematika dengan level tingkat sulit (Saripa, 2023). Proses kerja enkripsi dasar AES adalah *subbytes* (menggunakan tabel S-Box untuk menggabungkan *byte* pada blok matriks), *shift row* (digunakan untuk menggeser baris matriks), *MixColumns* (melakukan pergeseran dari kanan ke kiri), dan *AddRoundkey* (blok di XOR kan dengan kunci yang diperluas) (Aryanto et al., 2023). Pada proses *dekripsi* AES terdapat tahap-tahapan kebalikan dari proses enkripsi. Dengan memakai kunci yang sama untuk *dekripsi*, blok data yang telah terenkripsi dapat dipulihkan kembali ke bentuk awalnya. algoritma AES adalah jenis *block chipper* yang memiliki panjang kunci 128 bit, 192 bit dan 256 bit (Surbakti & Purwanto, 2024).

Algoritma Chacha20 dan AES merupakan suatu enkripsi yang dapat di digunakan dalam menjaga keamanan *file*. Pada algoritma AES terdapat kelebihan salah satunya adalah sifat karakteristiknya yang di mana selalu ada bidang khusus untuk bilangan prima, memicu kekuatan AES(Surbakti & Purwanto, 2024) . AES juga memiliki sebuah kelemahan salah satunya yaitu algoritma kriptografi yang menggunakan kunci simetris dalam proses pengiriman dan penerimanya data. Namun, penggunaan kunci simetris ini memiliki risiko lebih tinggi terhadap kebocoran, terutama seiring waktu (Hidayatulloh et al., 2023). Pada algoritma ChaCha20 terdapat keunggulan yaitu menggunakan teknik kriptografi ARX (*addition, rotation*, XOR) yang memberikan performa lebih cepat, kompleksitas lebih rendah, dan mampu menghilangkan serangan waktu Klik atau ketuk di sini untuk memasukkan teks.

Meskipun algoritma AES dan Chacha20 telah efektif untuk menjaga peningkatan keamanan *file* dan implementasinya yang cukup rumit sehingga membuat orang yang tidak memiliki akses tidak dapat mengakses *file* tersebut. Hal ini tidak dapat kita pungkiri bahwa semakin pesatnya dunia teknologi membuat banyak orang berbondong-bondong menciptakan pembaruan yang lebih efektif. Metode *double encrypt* ini menawarkan kombinasi yang mampu mengurangi kelemahan dari setiap algoritma jika digunakan secara bersamaan, serta meningkatkan ketahanan terhadap berbagai jenis serangan kriptografi. *Double encrypt* merupakan Teknik kriptografi yang melibatkan proses enkripsi secara dua kali, yang di mana memiliki dua kunci yang berbeda untuk lebih meningkatkan keamanan. *Double encrypt* ini bertujuan untuk meningkatkan keamanan *file* lebih tinggi dibandingkan hanya 1 metode saja (Pangestu et al., 2022).

Pada penelitian terlebih dahulu yang berjudul “Implementasi Sistem Keamanan *File* Menggunakan Algoritma AES untuk Mengamankan *File* Pribadi” yang diteliti oleh Saripa Universitas Negeri Makassar. Dalam penelitian tersebut menjelaskan tentang keamanan *file* dengan menggunakan metode algoritma AES. Di mana hasil penelitian ini menunjukkan bahwa algoritma AES telah berhasil mengamankan *file* yaitu dokumen Word, PDF, Excel, dan Power Poin, *file* yang dienkripsi menjadi tidak terbaca oleh pihak yang tidak memiliki akses sehingga dapat menjaga kerahasiaan data dan AES mampu menjaga keamanan data dengan baik.

Dalam perbedaan penelitian terdahulu dan penelitian saya akan menggunakan double encrypt untuk keamanan *file* dan juga saya akan menggunakan AES dengan menambahkan metode ChaCha20 dan melakukan modifikasi pada bagian AES. Modifikasi menggunakan penambahan *AddRoundkey* sebelum dengan bertujuan agar proses enkripsi akan lebih susah untuk dipecahkan. Dalam algoritma AES, langkah *AddRoundKey* merupakan komponen krusial yang menggabungkan data dengan kunci enkripsi menggunakan operasi XOR. Penelitian (Luong, 2023) oleh menunjukkan bahwa memodifikasi transformasi *AddRoundKey*

dapat meningkatkan keamanan algoritma dengan menciptakan hubungan yang lebih kompleks antara *plainteks* dan *cipherteks*. Penelitian ini menyoroti pentingnya penyesuaian langkah *AddRoundKey* untuk meningkatkan ketahanan algoritma AES terhadap serangan kriptografi, seperti *differential cryptanalysis* dan linear *cryptanalysis*.

Dengan mendasarkan pada penelitian tersebut, modifikasi algoritma AES melalui penambahan langkah *AddRoundKey* sebelum *MixColumns* bertujuan untuk lebih memperkuat difusi data pada tahap awal enkripsi. Penambahan ini diharapkan dapat meningkatkan kerandoman hasil enkripsi dan mengurangi kemungkinan eksploitasi pola oleh pihak yang tidak berwenang, sehingga keamanan algoritma menjadi lebih optimal. Berdasarkan dari latar belakang di atas penelitian ini memiliki tujuan untuk meningkatkan keamanan *file* agar lebih sulit untuk diretas atau diakses pihak yang tidak berwenang. Oleh karena itu penelitian ini dianggap penting karena maraknya kasus peretasan data oleh oknum yang tidak memiliki akses.

### Rumusan Masalah

1. Bagaimana implementasi dari modifikasi AES dengan penambahan *AddRoundkey* sebelum *MixColumns* dalam meningkatkan keamanan *file*?
2. Bagaimana performa kombinasi antara AES modifikasi dan ChaCha20 dalam hal waktu enrkipsi, waktu *dekripsi* , kecepatan enkripsi, dan kecepatan *dekripsi*?

### Tujuan Penelitian

1. Mampu mengidentifikasi *AddRoundkey* yang akan digunakan dalam menjaga keamanan *file* dan mengimplementasi kemampuan Chacha20 dan AES yang telah dimodifikasi dalam proses menjaga keamanan *file*.
2. Mampu mengevaluasi performa algoritma ChaCha20 dan AES yang telah dimodifikasi berdasarkan waktu enkripsi, waktu *dekripsi*, kecepatan enkripsi, dan kecepatan *dekripsi*, untuk menentukan efisiensi

dan efektivitas kombinasi kedua algoritma tersebut dalam menjaga keamanan *file*

### Batasan dan Asumsi Penelitian

1. Penelitian ini berfokus pada jurnal atau artikel ilmiah dengan batasan tahun 2021-2024 yang membahas tentang :
   * Keamanan *file*
   * Algoritma ChaCha20
   * *Advanced Encryption Standard* (AES)
   * Double encryption
2. Penelitian dilakukan dengan pengujian pada *file* dengan ukuran dan format sebagai berikut :
   * Format PDF dan Word dengan variasi ukuran *file* 100 KB - 100MB.
   * Format *file* teks (TXT) dengan ukuran maksimal 300 byte Batasan ukuran *file* digunakan untuk memastikan pengujian dilakukan secara efektif tanpa mengurangi kinerja proses enkripsi dan *dekripsi*.

### Manfaat Penelitian

### Manfaat Akademis

* + - 1. Memberikan inovasi terbaru mengenai pelindungan keamanan

*file* dengan menambahkan *AddRoundkey* pada algoritma AES.

* + - 1. Memudahkan peneliti lainnya dalam kinerja *AddRoundkey* pada algoritma AES yang berfungsi untuk keamanan *file***.**

### Manfaat Ilmiah

* + - 1. Menjadi sumber acuan dan sumber referensi bagi peneliti lainnya yang akan meneliti topik ChaCha20 dan AES modifikasi, terutama dalam konteks keamanan data dan performa enkripsi.
      2. Membuktikan hipotesis apakah AES yang dimodifikasi dengan tambahan *AddRoundkey* dapat meningkatkan performa keamanan dan efisiensi dibandingkan dengan AES standart, dan sekaligus menunjukkan kinerja kombinasi ChaCha20 dan AES dalam pengamanan *file* berbagai format dan ukuran.

### Sistematika Penulisan

## BAB 1 PENDAHULUAN

Menjelaskan latar belakang penelitian, rumusan masalah, tujuan penelitian, batasam penelitian, manfaat penelitian, serta sistematika penulisan secara umum.

## BAB 2 LANDASAN TEORI

Membahas teori-teori yang relavan dengan penelitian, termasuk algoritma ChaCha20, AES, konsep kriptografi, Serta tinjauan terhadap penelitian terdahulu yang terkait.

## BAB 3 METODOLOGI PENELITIAN

Menjelaskan metode yang digunakan dalam penelitian, termasuk algoritma yang diterapkan (modifikasi AES dan ChaCha20 dengan enkripsi ganda). Diuraikan juga alat dan bahan yang digunakan, serta tahapan penelitian yang meliputi implementasi algoritma, pengujian parameter kinerja (waktu dan kecepatan enkripsi, dekripsi, serta analisis entropi), dan evaluasi hasil pengujian pada berbagai jenis file (PDF, Word, TXT).

## BAB 4 IMPLEMENTASI DAN PENGUJIAN

Menyajikan hasil implementasi algoritma yang dikembangkan, pengujian algoritma terhadap berbagai jenis *file*, serta analisis data hasil pengujian**.**

## BAB 5 EVALUASI HASIL PENGUJIAN

Menyajikan hasil implementasi dari aplikasi yang telah dirancang, pengujian fungsionalitas sistem, serta evaluasi kinerja aplikasi terhadap berbagai skenario penggunaan. Analisis data hasil pengujian juga dilakukan untuk menilai keberhasilan aplikasi dalam mencapai tujuan penelitian.

## BAB 6 KESIMPULAN DAN SARAN

Merangkum temuan utama penelitian serta memberikan saran untuk pengembangan lebih lanjut di masa depan.

# BAB II LANDASAN TEORI

### Literatur Terkait Teori

Keamanan informasi menjadi salah satu isu yang paling mendesak di era digital saat ini, di mana data sensitif sering kali menjadi target serangan siber yang beragam. Dengan meningkatnya ancaman seperti serangan *man-in-the- middle*, *eavesdropping*, dan *DDoS*, penting bagi organisasi dan individu untuk menerapkan metode enkripsi yang efektif untuk melindungi data mereka. Berbagai algoritma enkripsi telah dikembangkan untuk memenuhi kebutuhan ini, dengan *Advanced Encryption Standard* (AES) dan ChaCha20 menjadi dua di antara yang paling banyak digunakan. Penelitian-penelitian terkini menunjukkan bahwa kedua algoritma ini tidak hanya menawarkan tingkat keamanan yang tinggi, tetapi juga dapat diimplementasikan dalam berbagai konteks, mulai dari pengamanan *file* hingga enkripsi konten *livestreaming*.

Pada [Tabel II.1 Literatur Penelitian Terkait Kriptografi](#_bookmark19) menyajikan ringkasan dari beberapa penelitian yang mengeksplorasi efektivitas dan penerapan algoritma AES dan ChaCha20 dalam berbagai aplikasi. Setiap penelitian memberikan wawasan tentang tujuan, hasil, dan persamaan dalam pendekatan yang digunakan, serta menyoroti inovasi yang dilakukan untuk meningkatkan keamanan data. Dengan memahami hasil-hasil ini, kita dapat lebih menghargai perkembangan teknologi enkripsi dan aplikasinya dalam menjaga kerahasiaan serta integritas informasi di dunia yang semakin terhubung.

### Kajian Teori

### Kriptografi

Menurut Menezes, Oorschot, dan Vantone (1996), Kriptografi adalah studi teknik matematika yang berkaitan dengan keamanan informasi seperti kerahasiaan, *autentikasi* keaslian data (Belvin Tannadi, 2022). Kriptografi dapat diartikan sebagai sebuah proses untuk melindungi data atau sebuah sistem yang bertujuan untuk melindungi informasi rahasia dengan cara yang tidak dipahami oleh pihak yang tidak berwenang. Dalam kriptografi yang meliputi proses enkripsi dan proses *dekripsi*. Di dalam Kriptografi terdapat tiga fungsi dasar yang meliputi :

* + - 1. Enkripsi merupakan proses data asli (*plainteks*) yang akan diubah menjadi data yang tidak bisa terbaca oleh pihak yang tidak berwenang (*chiperteks*).
      2. *Dekripsi* merupakan proses pengembalian dari data yang tidak bisa terbaca(*chiperteks*) menjadi data terbaca atau data asli (*plainteks*).
      3. Kunci merupakan proses yang dipakai untuk melakukan proses enkripsi dan *dekripsi*, terdapat 2 jenis kunci, yaitu kunci rahasia (*private key*) dan kunci publik (*public key*).

Pada [Gambar II.1 Alur Proses Enkripsi Dan *Dekripsi*](#_bookmark22)

memperlihatkan alur enkripsi dan *dekripsi.* Proses ini dimulai

dengan *plainteks* yang kemudian dienkripsi dengan menggunakan kunci menjadi *chiperteks*. *Chiperteks* tersebut kemudian *didekripsi* kembali menggunakan kunci yang sama untuk mendapatkan kembali *plainteks*.

*Gambar II.1 Alur Proses Enkripsi Dan Dekripsi*

Berikut algoritma Kriptografi yang berdasarkan kunci :

a. Algoritma Simetris

Algoritma simetris adalah algoritma yang merujuk pada metode enkripsi dan *dekripsi* yang di mana algoritma ini memiliki kunci tunggal atau sama. Sehingga dapat disimpulkan algoritma ini mewajibkan pengirim dan penerima harus menyetujui satu kunci yang digunakan dalam proses enkripsi dan *dekripsi* data. Namun, kelemahan dari algoritma ini yaitu tergantung pada rahasia kunci. Apabila kunci tersebut tersebar maka akan dengan mudah pihak yang tidak memiliki akses dapat melakukan proses enkripsi dan *dekripsi* data secara mudah.

*Gambar II.2 Proses Kunci Simetris*

Pada [Gambar II.2 Proses Kunci Simetris](#_bookmark23) menjelaskan proses enkripsi dan *dekripsi* menggunakan kunci simetris. Dimulai dengan pesan yang dikirimkan, kemudian dienkripsi menjadi kode yang tidak bisa dibaca menggunakan kunci simetris. Kode tersebut kemudian dikirim dan di *dekripsi* di sisi penerima menggunakan kunci simetris yang sama, sehingga pesan dapat dibaca kembali. Pada contoh gambar, pesan "Selamat Datang!" dienkripsi menjadi kode "9CDF4837 AB672019" menggunakan kunci simetris yang sama. Kemudian di *dekripsi* kembali ke pesan awal "Selamat Datang!" Kunci simetris merupakan kunci yang sama digunakan untuk enkripsi dan *dekripsi*, sehingga pengirim dan penerima pesan harus memiliki kunci yang sama agar dapat berkomunikasi.

### AES (Advanced Encryption Standard)

Advanced Encryption Standard (AES) adalah salah satu algoritma enkripsi yang paling terkenal dan banyak digunakan, serta terus dikembangkan sebagai pengganti algoritma Data Encryption Standard (DES). AES adalah algoritma enkripsi yang bekerja dengan memproses data dalam blok dengan ukuran minimal 128 bit, dan melalui beberapa tahapan untuk mengenkripsi data. Tujuan utama dari AES adalah untuk mengamankan data rahasia sehingga tidak dapat dibaca oleh pihak yang tidak berwenang (Wahyu Putra, 2023). Pada [Tabel II.2 Panjang Kunci](#_bookmark25) dalam proses enkripsi *Advanced Encryption Standard* (AES) menggunakan ukuran blok dengan panjang sebesar 126 bit, dengan pilihan panjang kunci yang bervariasi, yaitu 128 bit, 192 bit, dan 256 bit (Sekar Putri Ananda et al., 2022). Proses enkripsi dalam algoritma ini dilakukan melalui beberapa tahap yang dikenal sebagai ronde. Jumlah ronde yang digunakan bergantung pada panjang kunci yang akan dipilih. Berikut adalah hubungan panjang kunci dan banyaknya ronde.

Algoritma AES juga menawarkan tingkat keamanan yang sangat tinggi dikarenakan ukuran kunci dan jumlah ronde yang ditentukan. Ukuran kunci menentukan jumlah kemungkinan kombinasi kunci, sehingga membuat serangan *bruteforce* menjadi sangat sulit dilakukan. Di sisi lain, jumlah ronde yang besar dapat menghasilkan keamanan yang kuat, sehingga perubahan kecil pada *plainteks* dapat memengaruhi seluruh bagian *cipherteks*. Selain keunggulan keamanannya, AES juga memiliki efisiensi tinggi dalam hal waktu dan ruang. Algoritma ini dapat diimplementasikan baik pada perangkat keras maupun perangkat lunak, menjadikannya ideal untuk berbagai aplikasi, termasuk perlindungan data (Assyahid, 2024). Pada tahap awal enkripsi, data masukan yang telah disalin ke dalam matriks *state* akan menjalani transformasi *AddRoundkey*. Selanjutnya, matriks *state* akan melalui proses transformasi secara berulang, mencakup *SubBytes*, *ShiftRows*, *MixColumns*, dan *AddRoundkey*, dengan jumlah pengulangan sesuai dengan nilai Nr (Handoyo & Subakti, 2020). *Dekripsi* adalah proses mengembalikan *ciphertext* menjadi plaintext, atau mengubah data terenkripsi kembali ke bentuk aslinya. Pada proses *inverse cipher*, transformasi yang digunakan meliputi *InvShiftRows*, *InvSubBytes*, *InvMixColumns*, dan *AddRoundkey*. Berikut ini adalah cara kerja algoritma AES secara umum:

### Proses Enkripsi AES Modifikasi

* + - * 1. *AddRoundkey*

*AddRoundkey* adalah proses menggabungkan state masukan dengan kunci ronde dengan menggunakan operasi XOR. Dalam proses transformasi enkripsi dan *dekripsi* pada AES, dilakukan operasi penambahan kunci putaran (*round key*)

pada state menggunakan operasi XOR (Djong & Siswanto, 2022). Setiap kunci putaran terdiri dari *Nb word* di mana setiap *word* tersebut akan digabungkan dengan *word* atau kolom yang sesuai pada *state* . Secara matematis, hal ini (Putri et al., 2021).

Keterangan Algoritma dari *AddRoundkey* di mana :

Pada [Gambar II.3.Transformasi ShiftRows](#_bookmark27) di atas di mana S0,0 merupakan elemen di baris ke-0, kolom ke-0, S3,3 merupakan elemen di baris ke-3, kolom ke-3, dan 𝑊3,3 merupakan elemen di baris ke-3, kolom ke-3. Pada label S0,0 dan lainnya merupakan state saat ini, state ini representasi internal data yang sedang dienkripsi ataupun *didekripsi* dan setiap elemen adalah *byte* dari blok data input. Sedangkan pada label 𝑊0,0 merupakan kunci ronde yang dihasilkan dari proses *keyexpansion*, di mana ini bagian dari kunci utama yang dipecah untuk menjadi beberapa ronde dan setiap elemen adalah *byte* dari kunci untuk ronde tertentu. Operasi XOR dilakukan antara setiap *byte* pada *state* dengan *byte* kunci ronde yang sesuai. Hasil di atas dihasilkan dari operasi XOR yang akan digunakan untuk langkah transformasi berikutnya.

* + - * 1. *SubBytes*

Transformasi *transformasi byte* di mana setiap elemen dalam matriks *state* dipetakan menggunakan tabel substitusi yang dikenal sebagai S-Box. Setiap *byte* dalam matriks *state* direpresentasikan sebagai S’[r, c], yang merupakan elemen hasil dari perpotongan antara baris (x) dan kolom (y) (Kurnia Mahesa Naibaho et al., 2021). S-Box berupa matriks

berukuran 16×16 yang berisi nilai-nilai heksadesimal di mana nilai-nilai ini diperoleh melalui perhitungan tertentu (Hena & Kusumaningsih, 2024). Berikut ini tabel S-Box :

Pada [Tabel II.3 Tabel Substitusi (S-Box)](#_bookmark28) sebagai contoh, nilai heksadesimal {62} merujuk ke baris 7 dan kolom 3 dalam S-Box, yang berisi nilai {aa}. Dengan demikian, nilai

{62} dipetakan menjadi nilai {aa}.

* + - * 1. *ShiftRows*

Transformasi *ShiftRows* merupakan proses pergeseran *byte* di mana *byte* paling kiri dipindahkan ke posisi paling kanan (rotasi bit). Jumlah pergeseran dalam melakukan *ShiftRows* ini dilakukan perbedaan setiap baris dalam matriks *state*. Pada baris pertama, tidak ada pergeseran yang dilakukan. Kemudian pada baris kedua barulah mengalami pergeseran satu *byte* ke kiri, lalu baris ketiga digeser dua *byte* ke kiri, dan baris keempat digeser tiga *byte* ke kiri. Tujuan dari proses ini adalah untuk menciptakan *diffusion*(), yang memiliki tujuan untuk membantu menyebarkan efek transformasi non-linear ke seluruh baris dalam matriks *state* untuk proses selanjutnya (Cristy & Riandari, 2021). Proses pergeseran ini bertujuan untuk mencampur elemen-elemen dalam *state* sehingga memperkuat kerahasiaan data.

*MixColumns* adalah proses transformasi di mana setiap elemen dalam blok *cipher* dikalikan dengan matriks . Transformasi ini bertujuan untuk mencampur elemen- elemen dalam setiap kolom *state* berguna meningkatkan difusi data dan memperkuat keamanan enkripsi . Transformasi ini bekerja dengan membagi matriks atau *state* menjadi empat bagian, untuk setiap kolomnya (Widyawan & Imelda, 2021). Setiap kolom dalam matriks *state* diperlakukan sebagai *polinomial* dengan empat bagian dalam *Galois Field* (GF), pada persamaan [(II-2)](#_bookmark30) dilakukan Algoritma dari *AddRoundkey* di mana :

𝑋8 = Elemen dengan derajat ke-8 yang akan muncul sebagai 100000000 (1 diposisi ke 9 dari kanan) di bilangan biner.

X = Menunjukkan elemen dengan derajat ke 1 yang akan muncul sebagai 00000010 ( 1 di posisi ke 2 dari kanan

) dalam bilangan biner

1 = Elemen konstanta yang akan muncul sebagai

00000001 ( 1 diposisi ke 1 dari kanan), dimana elemen 1 ini menyatakan bahwa *polinomial* ini memiliki koefisien konstan.

Proses *MixColumns* juga dapat direpresentasikan sebagai operasi perkalian matriks, di mana setiap kolom matriks dikalikan dengan matriks tertentu dalam *Galois Field* (Cristy & Riandari, 2021).

* + - 1. **Proses *dekripsi* AES :**

Proses *dekripsi* merupakan kebalikan dari proses enkripsi. *Transformasi bytes* yang digunakan dalam *dekripsi* mencakup *AddRoundKey*, *InvShiftRows*, *InvSubBytes,* dan *InvMixColumns*. Pada iterasi pertama *dekripsi*, dilakukan *transformasi AddRoundKey, Inverse ShiftRows, dan Inverse SubBytes. Ciphertex*t kemudian akan mengalami transformasi *AddRoundKey* .

* + - * 1. *InvShiftRows*

Transformasi *byte* yang merupakan kebalikan dari proses *ShiftRows*. Dalam transformasi ini, bit-bit pada *state* digeser ke arah kanan, berbeda dengan *ShiftRows* yang melakukan pergeseran bit ke arah kiri.

* + - * 1. *InvSubBytes*

Transformasi *byte* yang menjadi kebalikan dari *SubBytes*. Dalam proses ini, setiap elemen pada *state* dipetakan menggunakan tabel substitusi *Inverse* S-Box.

* + - * 1. *InvMixColumns*

Setiap kolom dalam *state* dikalikan dengan matriks khusus yang digunakan dalam proses perkalian pada algoritma AES (Syahputra Sianipar et al., 2024)

*Gambar II.4 Proses AES*

Pada [Gambar II.4 Proses AES](#_bookmark31) sebelah kiri menunjukkan tahapan proses enkripsi pada algoritma AES Modifikasi. Proses dimulai dengan memasukkan plainteks, kemudian melalui serangkaian langkah seperti *AddRoundkey*, SubBytes, ShiftRows, MixColumns, dan diakhiri dengan *AddRoundkey* tambahan. Langkah ini diulang sebanyak Nr−1 ronde untuk meningkatkan keamanan data, menghasilkan cipherteks di akhir proses. Flowchart sebelah kanan menjelaskan tahapan proses *dekripsi*, yang merupakan kebalikan dari proses enkripsi. Proses dimulai dengan memasukkan cipherteks, kemudian melalui langkah-langkah seperti *AddRoundkey*, Inverse ShiftRows, Inverse SubBytes, dan Inverse MixColumns. Setelah semua ronde selesai, data dikembalikan ke bentuk plainteks asli.

### Chacha20

Algoritma ChaCha20 adalah salah satu jenis *stream cipher* yang dikembangkan sebagai varian dari algoritma Salsa20. Algoritma ini berbasis desain yang serupa dengan Salsa20, namun ChaCha20 meningkatkan tingkat *diffusion* pada setiap ronde untuk memperkuat keamanan proses enkripsi (Anindita & 18219086, 2023). Chacha20 beroperasi pada unit data 32-bit menggunakan kunci 256-bit. Algoritma ini menghasilkan *output* berupa *stream* kunci sepanjang 512-bit, yang kemudian akan di XOR kan dengan *plainteks* untuk menghasilkan *chiperteks*(Alyas & Abdullah, 2021b). ChaCha20 dimulai dengan membuat *state* 512-bit, yang berupa *array* yang berukuran 16x32-bit. State ini disusun dari empat konstanta 32 bit, kunci rahasia sepanjang 256-bit, penghitung blok 32-bit, serta nonce Ini menunjukkan struktur *state* ChaCha20 yang terdiri dari matriks 4x4 dengan 16 elemen 32-bit. Matriks ini diinisialisasi dengan empat konstanta 32-bit, kunci 256-bit yang dibagi menjadi 8 bagian, penghitung 32 blok 32-bit, dan Nonce 96-bit. Elemen-elemen ini digunakan untuk menghasilkan *keystream* yang kemudian di XOR kan dengan *plainteks* untuk menghasilkan *chiperteks*. Di mana *const* (konstanta) ini berisi empat nilai tetap 32-bit, *key* (kunci) yang terbagi 8 bagian 32-bit dari kunci rahasia 256-bit, *counter* (penghitung), dan Nonce (nilai acak) yang berukuran 96-bit.

Langkah – Langkah Algoritma ChaCha20 :

* + - 1. *Inisialisasi* State

Algoritma ChaCha20 dimulai dengan membangun sebuah state internal berbentuk matriks 4x4 (32-bit) (Barbero et al., 2022) . Matriks ini terdiri dari:

* + - * 1. Baris pertama: Berisi 4 konstanta tetap (128-bit), yang digunakan untuk mengenali algoritma.
        2. Baris kedua dan ketiga: Berisi kunci rahasia 256-bit (8 elemen 32-bit), yang digunakan untuk proses enkripsi.
        3. Baris keempat: Berisi kombinasi 64-bit counter dan 64-bit nonce (masing-masing terdiri dari 2 elemen 32-bit).
      1. *Quarter Round*

Operasi ini memodifikasi 4 elemen dari matriks ( a, b, c, d ) (Almukhlifi & Vora, 2020)

𝑎 += 𝑏; 𝑑 ^ = 𝑎; 𝑑 <<<= 16;

𝑐 += 𝑑; 𝑏 ^ = 𝑐; 𝑏 <<<= 12;

𝑎 += 𝑏; 𝑏 ^ = 𝑎; 𝑑 <<<= 8;

𝑐 += 𝑑; 𝑏 ^ = 𝑐; 𝑏 <<<= 7;

* + - 1. *Double Round*

*Double Round* adalah kombinasi dari dua *Quarter Round* yang dijalankan secara berurutan. Setiap iterasi *Double Round* mencakup:

* + - * 1. Operasi pada kolom matriks *(Column Round):*

melakukan operasi Quarter Round pada setiap kolom matriks 4x4. Tujuannya adalah untuk mencampur data antar elemen dalam kolom, sehingga meningkatkan difusi data secara vertikal.

* + - * 1. Operasi pada diagonal matriks (Diagonal Round):

Setelah operasi pada kolom selesai, langkah berikutnya adalah menerapkan Quarter Round pada elemen-elemen diagonal dalam matriks.

Proses Double Round dilakukan sebanyak 10 iterasi, menghasilkan total 20 ronde Quarter Round untuk setiap blok data yang diproses.

* + - 1. Finalisasi State
      2. *Keystream Generation*

Setelah 20 ronde selesai*, state* akhir ditambahkan kembali ke

### Entropi

Entropi dalam teori informasi digunakan untuk mengukur keacakan data, yang mencerminkan kekuatan algoritma enkripsi. Semakin tinggi nilai entropi, semakin acak data tersebut, yang berarti algoritma enkripsi lebih tahan terhadap serangan (Wahyudi et al., 2024). Entropi dihitung dengan menggunakan rumus pada

Dimana:

* 𝐻𝑚 : Menunjukkan entropi data, dimana tingkat keacakan dalam

# BAB III METODOLOGI PENELITIAN

### Sistematika Penyelesaian Masalah

Penelitian ini bersifat kuantitatif dengan tujuan mengevaluasi performa algoritma AES modifikasi dan kombinasi ChaCha20 dan AES modifikasi. Penilaian performa ini dilakukan berdasarkan waktu enkripsi, waktu *dekripsi*, kecepatan enkripsi, dan kecepatan *dekripsi*, yang akan dianalisis menggunakan data hasil pengujian pada berbagai jenis *file* dengan format PDF, Word dan TXT, yang hasilnya berbentuk tabel untuk mempermudah analisis perbandingan.

### Alat dan Bahan

### Alat

1. Perangkat Keras (Hardware) Laptop dengan spesifikasi :
   1. Prosesor : Intel *Core* I5
   2. RAM : 8 GB
   3. Penyimpanan : 256 GB SSD atau lebih
2. Perangkat Lunak (*Software*)
   1. *Python* :

Menggunakan *library cryptography, pycryptodome*, dan *library* lain yang mendukung algoritma AES dan ChaCha20. Berikut beberapa pustaka yang digunakan :

* + - PyCryptodome : Digunakan Iplementasi algoritma AES dan ChaCha20
    - Cryptography : Implementasi kemanan tambahan (jika nanti dibutuhkan )
    - Numpy : Digunakan untuk analisis data (perhitungan entropi)
    - Time : Mencatat waktu eksekusi algoritma, baik untuk enkripsi maupun dekripsi.
  1. *Text* Editor :

Visual Studio Code dan *Jupyter Notebook*

* 1. Sistem Operasi : Windows

### Bahan

1. Data set *file* :
   1. *File* PDF : Ukuran *file* 100 KB – 100 MB
   2. *File* Word : Ukuran *file* 100 KB – 100 MB
   3. *File* TXT : Ukuran *file* maksimal 300 *byte*
2. Literatur :

Jurnal atau artikel ilmiah tentang AES, ChaCha20 dan *Double Encrypt* (tahun 2020-2024)

1. *Spreadsheet* Analisis Data :

Format CSV (Comma Separated Values)

### Rancangan Penelitian

Pada penelitian ini, dilakukan tahapan yang dirancang secara sistematis untuk mencapai tujuan utama, yaitu Implementasi kombinasi Advanced Encryption Standard (AES) yang dimodifikasi dan ChaCha20. Proses penelitian dimulai dengan studi literatur untuk memahami konsep dasar dan pengembangan terbaru terkait algoritma kriptografi. Setelah itu, dilakukan pengumpulan data yang relevan sebagai dasar implementasi. Tahapan selanjutnya mencakup implementasi algoritma modifikasi AES serta kombinasi algoritma tersebut dengan ChaCha20. Untuk mengukur efektivitas dan efisiensi algoritma yang dikembangkan, pengujian dilakukan pada berbagai jenis *file* seperti PDF, Word, dan TXT. Analisis hasil dari pengujian ini memberikan dasar untuk menarik kesimpulan dan memberikan rekomendasi untuk pengembangan lebih lanjut.

*Gambar III.1 Diagram Alur Penelitian*

[Gambar III.1 Diagram Alur Penelitian](#_bookmark44) ini terdapat beberapa tahapan yaitu:

1. Mulai adalah tahap awal dari penelitian. Pada tahap ini menentukan tujuan dan menyusun langkah-langkah yang akan dilakukan dalam penelitian.
2. Studi literatur dilakukan pengumpulan referensi yang berkaitan dengan algoritma AES, ChaCha20 dan *Double Encrypt. Referensi* berupa jurnal, artikel ilmiah.
3. Pengumpulan data untuk mengumpulkan *file* atau data set yang akan digunakan untuk pengujian algoritma. Format *file* yang digunakan adalah PDF, Word dan TXT.
4. Implementasi modifikasi AES, pada tahap ini algoritma AES dimodifikasi dengan menambahkan langkah *AddRoundkey* sebelum *Mixcolumns*. Tujuannya adalah untuk meningkatkan keamanan dan kecepatan enkripsi dibandingkan dengan AES standar.
5. Implementasi kombinasi AES modifikasi dan ChaCha20, menggabungkan algoritma AES yang sudah dimodifikasi dengan ChaCha20 untuk melakukan *double encypt*. Di mana proses ini bertujuan untuk meningkatkan keamanan *file* lebih baik daripada menggunakan *single encrypt*.
6. Pengujian algoritma, untuk menguji performa algoritma yang sudah diimplementasikan, di mana data set yang dikumpulkan sebelumnya itu digunakan untuk uji coba. Parameter yang diukur pada pengujian ialah

waktu enkripsi dan *dekripsi,* kecepatan enkripsi dan *dekripsi*, dan hasil pengujian ini akan digunakan untuk mengevaluasi kinerja algoritma.

1. Analisis hasil, data pengujian yang sudah terkumpul akan dianalisis. Analisis dilakukan untuk membandingkan performa AES standar, AES yang sudah dimodifikasi, dan kombinasi AES modifikasi dan ChaCha20. Tujuannya sendiri untuk mengetahui algoritma mana yang lebih efisien.
2. Selanjutnya ke kesimpulan dan saran, pada tahap ini merangkum hasil penelitian termasuk kelebihan dan kekurangan algoritma yang sudah diuji. Saran juga untuk pengembangan penelitian lebih lanjut
3. Selesai

### Studi Literatur

Studi literatur dilakukan untuk memahami konsep dasar dan perkembangan terbaru terkait algoritma kriptografi yang digunakan dalam penelitian ini, yaitu Advanced Encryption Standard (AES) dan ChaCha20. Proses ini melibatkan pengumpulan referensi dari jurnal, paper, dan buku yang membahas metode enkripsi, keamanan data, serta teknik evaluasi seperti entropi dan analisis performa. Literatur yang dikaji juga mencakup penelitian sebelumnya tentang algoritma kriptografi untuk keamanan *file*, sehingga memberikan dasar teori yang kuat untuk modifikasi algoritma dan penerapan metode double encryption dalam penelitian ini.

### Pengumpulan Data

Data yang digunakan dalam penelitian ini dibuat sendiri dengan mengikuti ketentuan yang telah ditentukan. Format data meliputi *file* PDF, Word, dan TXT dengan ukuran bervariasi, mulai dari 100 KB hingga 100 MB untuk *file* PDF dan Word, serta hingga 300 *byte* untuk *file* TXT. Data ini dirancang sedemikian rupa agar sesuai dengan kebutuhan pengujian algoritma enkripsi, seperti mengukur waktu enkripsi, waktu *dekripsi*, kecepatan, dan tingkat keacakan menggunakan analisis entropi. Variasi format dan ukuran data ini

bertujuan untuk memastikan bahwa pengujian mencakup berbagai jenis isi *file*.

### Implementasi Modifikasi AES

Pengembangan algoritma *Advanced Encryption Standard* (AES) dilakukan dengan memodifikasi. Modifikasi dilakukan dengan menambahkan *AddRoundkey* sebelum *Mixcolumns*. Penambahan ini bertujuan untuk memperkuat keamanan dan meningkatkan kompleksitas pada proses enkripsi, sehingga lebih sulit untuk diretas dengan pihak tidak berwenang. Berikut adalah proses alur dari modifikasi algoritma AES.

*Gambar III.2 Diagram Proses Enkripsi AES*

Pada [Gambar III.2 Diagram Proses Enkripsi AES](#_bookmark48) ini digunakan untuk mengamankan *file* dengan melakukan tahap enkripsi terhadap data *plainteks* yang menggunakan kunci tertentu. Dalam penelitian ini, modifikasi dilakukan dengan menambahkan langkah *AddRoundkey* untuk meningkatkan tingkat keamanan dan kompleksitas enkripsi. Berikut langkah-langkah proses modifikasi AES .

1. Input *file* dan kunci

Proses enkripsi dimulai dengan memasukkan *file* input (*file* asli yang akan dienkripsi) dan kunci enkripsi. *File* input dibaca dalam mode biner, diubah menjadi *byte array*, lalu dipecah menjadi blok-blok data berukuran 16 *byte*. Jika panjang data tidak habis dibagi 16 *byte, padding* dilakukan dengan metode PKCS#7 untuk melengkapinya.

Contoh :

* 1. *File* input : contoh.pdf

Isi *file* dalam biner : [0x25, 0x50, 0x44, 0x46, ...]

*Byte array* : [37, 80, 68, 70, 10, 0, 45, 78, ...]

* 1. Kunci enkripsi :

Kunci : surabayaitelkom

ASCII : [115, 117, 114, 97, 98, 97, 121, 97, 105, 116, 116,

101, 108, 107, 111, 109]

* 1. Matriks data *file* :

|  |  |  |  |
| --- | --- | --- | --- |
| [37, | 80, | 68, | 70] |
| [10, | 0, | 45, | 78] |
| [255, | 100, | 32, | 12] |
| [80, | 50, | 60, | 72] |

1. *AddRoundkey*

Merupakan tahap awal proses enkripsi, di mana *plainteks* ini digabungkan dengan kunci awal yang menggunakan operasi logika yaitu XOR. Di mana blok data *file* digabungkan dengan kunci awal (W0-W3) menggunakan operasi XOR untuk menambahkan lapisan keamanan awal.

Contoh :

* 1. Proses XOR :

37 XOR 115 = 86

80 XOR 117 = 37

68 XOR 114 = 54

70 XOR 97 = 39

* 1. Matriks setelah *AddRoundkey* :

[86, 37, 54, 39]

[108, 97, 76, 47]

[150, 16, 84, 97]

[28, 89, 83, 37]

1. Proses ronde 1-9
   1. *SubBytes*

Pada proses ini, nantinya setiap *byte* pada matriks *state* ( data yang diperoleh *AddRounKey*) akan diubah menggunakan tabel substitusi yang biasa disebut dengan S-Box. *Tranformasi* ini bertujuan untuk meningkatkan keacakan data agar sulit diprediksi.

Contoh proses :

*Byte* 37 diganti dengan nilai dari tabel S-Box jadi 120

* 1. *ShiftRows*

Setiap baris dalam matriks *state* nantinya digeser ke kiri pada baris pertama tidak ada pergeseran, baris kedua di geser 1 *byte* ke kiri, baris ketiga ini digeser 2 *byte* ke kiri, dan terakhir baris keempat yang nantinya digeser 3 *byte* ke kiri. Contoh proses :

[244, 120, 99, 200] [244, 120, 99, 200]

[60, 107, 55, 175] [107, 55, 175, 60]

[12, 243, 172, 85] [172, 85, 12, 243]

[180, 97, 92, 65] [65, 180, 97, 92]

* 1. *AddRoundkey* (W4-W7)

Setelah *ShiftRows* dilakukan operasi XOR antara matriks *state* dengan kunci ronde berikutnya yaitu (W4-W7) Contoh proses :

*Byte* 244 XOR 115 = 135

*Byte* 120 XOR 117 = 13

Hasil *AddRounKey* : [135, 13, 17, 169]

[38, 98, 85, 12]

[55, 32, 82, 84]

[49, 53, 180, 193]

* 1. *MixColumns*

Setiap kolom pada matriks *state* dimodifikasi dengan operasi matematis berbasis matriks yang menggunakan bidang *Galois* (GF). Di mana proses ini bertujuan untuk mencampur elemen-elemen dalam kolom dan meningkatkan difusi data. [2, 3, 1, 1]

[1, 2, 3, 1]

[1, 1, 2, 3]

[3, 1, 1, 2]

* 1. *AddRoundkey* (W36-W39)

Dilakukan setelah *MixColums* untuk menambahkan lapisan keamanan dengan kunci ronde tambahan (W36-W39).

1. Pada ronde terakhir, hanya dilakukan tiga langkah yaitu

*SubBytes, ShiftRows*, dan *AddRoundkey* tanpa *MixColumns.*

Proses modifikasi algoritma AES ini digunakan untuk mengenkripsi *file* dengan menambahkan langkah *AddRoundkey* tambahan, yang meningkatkan keamanan dengan memberikan lapisan XOR tambahan pada setiap ronde enkripsi.

* + 1. **Implementasi Kombinasi AES Modifikasi dan ChaCha20** Algoritma AES modifikasi dan ChaCha20 dengan menerapkan *double encypt.* Proses ini memakai AES modifikasi dengan memberikan perlindungan berbasis blok *chiper* yang kuat dengan memodifikasi *AddRoundkey* tambahan sebelum *MixColumns,* sedangkan ChaCha20 sebagai *stream cipher* yang cepat dan aman, meningkatkan kompleksitas keseluruhan dengan memanfaatkan proses enkripsi berbasis *keystream.*

Pada metode ini, setiap *file* menggunakan 1 kunci enkripsi. Kunci yang digunakan memiliki panjang awal 128-bit, yang kemudian diperluas menjadi 256-bit pada ChaCha20 dengan menambahkan padding nol (0x00) jika panjangnya kurang dari 256-bit. Proses

dimulai dengan mengenkripsi *file* menggunakan AES Modifikasi untuk menghasilkan cipherteks awal dalam format matriks 4x4. Cipherteks ini kemudian diubah menjadi byte stream untuk kompatibilitas dengan ChaCha20. Selanjutnya, ChaCha20 melakukan proses enkripsi kedua dengan memanfaatkan kunci 256- bit, nonce sepanjang 96-bit, dan counter. Hasil akhir adalah cipherteks final yang lebih aman karena telah melalui dua lapisan enkripsi. Penggunaan double encryption dengan satu kunci untuk setiap *file* memberikan keamanan yang lebih tinggi, di mana kombinasi kedua algoritma menciptakan perlindungan yang kuat terhadap berbagai jenis serangan kriptografi.

*Gambar III.3 Diagram Proses Double Encrypt*

Pada [Gambar III.3 Diagram Proses Double Encrypt](#_bookmark50) menunjukkan langkah-langkah enkripsi ganda. Proses dimulai dengan mengenkripsi *plainteks* menggunakan AES modifikasi untuk menghasilkan *chiperteks* AES. *Chiperteks* ini kemudian dienkripsi ulang menggunakan ChaCha20, yang melibatkan matriks 4x4, proses *quarter-round,* dan operasi XOR dengan *keystream*. Algoritma AES modifikasi dan ChaCha20 menggunakan kunci yang sama, yaitu kunci AES sepanjang 128-bit. berikut adalah tahapan alur di atas :

1. Input *plainteks* dan *key* dan enkripsi AES modifikasi
2. Input *cipherteks* AES ke ChaCha20

*Chiperteks* yang dihasilkan dari proses enkripsi pertama diubah menjadi input untuk algoritma ChaCha20. Ini dilakukan untuk menerapkan enkripsi berlapis yaitu *double encryption,* sehingga keamanan data akan meningkat secara signifikan. *Chiperteks* AES yang biasanya dalam format matriks (4x4) ini diubah ke format *byte stream* untuk kompatibilitas dengan algoritma ChaCha20. Byte stream ini memungkinkan untuk memproses data, sesuai dengan *stream chiper*.

Contoh :

[115, 75, 189, 234, 201, 97, 58, 14, 245, 142, 12, 120, 53, 190,

201, 20] data ini sudah dalam bentuk *byte stream*.

1. Validasi panjang *key* (256-bit *Key*)

Ini untuk memastikan bahwa kunci enkripsi untuk ChaCha20 sesuai dengan spesifikasi algoritma, yaitu panjangnya harus tepat 256 bit (32 *byte*). Kunci ini untuk menghasilkan *keystream* yang beda dalam proses enkripsi. Jika panjang ke kurang dari 256 bit, ditambahkan *padding* berupa *byte* nol (0x00) untuk memenuhi panjang. Jika *key* yang lebih panjang, *key* akan dipotong hingga hanya menyisakan 256 bit. *Key* kemudian dibagi menjadi 8 bagian ( di mana masing-masing 32 bit) untuk digunakan dalam proses pembentukan matriks awal (*state* matriks)

Contoh :

* 1. *Key* input “surabayaittelkom”

[115, 117, 114, 97, 98, 97, 121, 97, 105, 116, 116, 101, 108,

107, 111, 109]

* 1. Panjang *key* adalah 16 *byte* (128 bit), sehingga tidak memenuhi syarat panjang 256 bit. Jadi ditambahkan *padding* nol (0x00) hingga mencapai 32 *byte:*

[115, 117, 114, 97, 98, 97, 121, 97, 105, 116, 116, 101, 108,

107, 111, 109, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

1. Membuat *Initial* State Matriks 4x4

Membentuk *state* matriks awal yang menjadi dasar operasi dalam ChaCha20. Matriks ini berisi kunci, *counter*, dan *nonce.* Matriks 4x4 dalam ChaCha20 terdiri 16 elemen (masing-masing 32 bit) Contoh :

* 1. 4 elemen konstanta tetap “*expand* 32-byte k”4 elemen 32 bit [1634760805, 857760878, 2036477234, 1797285236]
  2. Elemen kunci 256-bit dibagi menjadi 8 elemen 32-bit: [1936028945, 1885434721, 1701147239, 1647539057,

1225269481, 1226844037, 1214606441, 0]

* 1. 1 *Counter*: Nilai awal 0.
  2. 3 Elemen Nonce: Nilai *nonce* yang unik untuk sesi enkripsi: Contoh *nonce* : [12345678, 87654321, 11223344]
  3. Matriks yang dibentuk

[1634760805, 857760878, 2036477234, 1797285236]

[1936028945, 1885434721, 1701147239, 1647539057]

[1225269481, 1226844037, 1214606441, 0]

[0, 12345678, 87654321, 11223344]

Baris ke 1 konstanta, ke 2 *key,* ke 3 *key* dan ke 4 *counter* dan

*nonce*

1. Operasi Quarter Round (20 iterasi)

Melakukan proses pencampuran data di dalam matriks untuk meningkatkan kerandoman dan keamanan *keystream* yang dihasilkan. Quarter Round adalah inti dari algoritma ChaCha20. Quarter Round dilakukan pada 4 elemen tertentu dari matriks. Penjumlahan Modular (mod 2³²): Menambah elemen dalam matriks untuk mencampur data. Operasi XOR: Mencampur data antar elemen melalui operasi logika XOR. Rotasi Bit (ROTL): Menggeser bit elemen untuk mendistribusikan informasi secara acak. Setelah 20 iterasi (10 putaran double round), matriks akan diubah menjadi bentuk yang sangat acak.

1. *Counter* +=1

*Counter* bertambah untuk setiap blok data yang diproses, memastikan *keystream* yang dihasilkan unik untuk setiap blok. Setelah satu blok data selesai diproses, *counter* di matriks bertambah sebesar 1.

Contoh :

*Counter* baru : [1, 12345678, 87654321, 11223344]

1. XOR *keystream* dengan *chiperteks* AES

Menggabungkan *keystream* hasil ChaCha20 dengan *cipherteks AES* melalui operasi XOR untuk menghasilkan *final ciphertext*. XOR dipilih karena sifatnya yang sederhana namun aman dan mudah dibalik. *Keystream* yang dihasilkan dari matriks ChaCha20 di-XOR-kan dengan *cipherteks AES*.

Operasi XOR:

Jika *ciphertext* = *keystream* XOR *plaintext*, maka *dekripsi*

dilakukan dengan cara *plaintext* = *keystream* XOR *ciphertext*.

* + - 1. **Proses *Dekripsi* Double Encryption**

Proses *dekripsi* dilakukan untuk mengembalikan data yang telah terenkripsi ke bentuk aslinya atau *plainteks*. Di mana pada tahap ini, *chiperteks* hasil dari kombinasi algoritma AES modifikasi dan Chacha20 akan *didekrpsi* secara urut. Proses nantinya akan dimulai dengan *mendekripsi chiperteks* dengan ChaCha20 untuk mendapatkan kembali *chiperteks* AES modifikasi, kemudian dilanjutkan dengan *mendekripsi chiprteks* AES modifikasi untuk memperoleh *plainteks* asli.

Pada [Gambar III.4 Proses Dekripsi](#_bookmark51) Diagram alur di atas menjelaskan proses *dekripsi* dari kombinasi algoritma ChaCha20 dan modifikasi AES. Proses dimulai dengan memasukkan cipherteks hasil enkripsi sebagai input. Langkah pertama adalah men*dekripsi* cipherteks menggunakan algoritma ChaCha20 untuk menghasilkan keystream. Keystream ini kemudian dikombinasikan dengan cipherteks menggunakan operasi XOR untuk mendapatkan cipherteks AES.

Selanjutnya, proses ekspansi kunci dilakukan untuk menghasilkan kunci yang akan digunakan pada setiap ronde *dekripsi* AES. Proses *dekripsi* AES terdiri dari tiga tahapan utama: ronde terakhir (Round 10) yang melibatkan *AddRoundKey*, InvShiftRows, dan InvSubBytes; ronde utama (Round 9-1) yang melibatkan *AddRoundKey*, InvMixColumns, *AddRoundKey, InvShiftRows,* dan *InvSubBytes* tambahan; dan ronde awal (Round 0) yang menggunakan *AddRoundKey* (W0-W3). Hasil akhir dari seluruh proses ini adalah plainteks, yaitu data asli yang berhasil di*dekripsi*. Kombinasi kedua algoritma ini memastikan keamanan data yang lebih kuat melalui enkripsi ganda.

*Gambar III.4 Proses Dekripsi*

### Implementasi Antarmuka Pengguna (UI)

Antarmuka pengguna (UI) dirancang untuk mempermudah interaksi, berfungsi sebagai media interaktif yang memungkinkan pengguna untuk memilih file, memasukkan kunci enkripsi, serta memulai proses enkripsi dan dekripsi dengan mudah. Implementasinya menggunakan Flask sebagai framework backend untuk menangani

proses enkripsi dan dekripsi, sedangkan HTML digunakan untuk membuat form input file dan kunci enkripsi. CSS digunakan untuk mempercantik tampilan agar antarmuka lebih menarik, jelas, dan mudah digunakan oleh pengguna dalam proses enkripsi dan dekripsi file menggunakan kombinasi algoritma ChaCha20 dan AES yang telah dimodifikasi.

*Gambar III.5 Desain Antarmuka Pengguna Enkripsi dan Dekripsi*

Pada [Gambar III.5 Desain Antarmuka Pengguna Enkripsi dan](#_bookmark53) [Dekripsi](#_bookmark53) Gambar tersebut menampilkan desain antarmuka pengguna (UI) dari sebuah aplikasi berbasis web yang digunakan untuk enkripsi dan *dekripsi file*. Antarmuka ini memiliki beberapa elemen input pengguna untuk mengunggah *file* dan memilih opsi enkripsi atau *dekripsi*. Terdapat tiga kolom utama pada desain ini: pertama, Pilih *File*, untuk pengguna memilih *file* yang akan diproses. Kedua, Enkripsi atau *Dekripsi*, yang memberikan pilihan kepada pengguna apakah *file* yang dipilih akan dienkripsi atau di *dekripsi*, yang merupakan opsi penting untuk menentukan jenis operasi yang akan dilakukan pada *file* tersebut. Ketiga, Key, di mana pengguna diminta untuk memasukkan kunci yang diperlukan untuk proses enkripsi atau *dekripsi*. Kunci ini memainkan peran penting dalam menjaga keamanan data yang diproses. Di bagian bawah antarmuka, terdapat

tombol Proses yang akan menjalankan operasi sesuai dengan pilihan pengguna, baik itu enkripsi maupun *dekripsi*.

### Pengujian Algoritma

Menjelaskan pengujian yang dilakukan untuk mengevaluasi kinerja dan keamanan algoritma berdasarkan parameter seperti entropi, waktu enkripsi dan *dekripsi*, serta kecapatan.

### Pengujian Entropi

Pengujian kekuatan algoritma dilakukan dengan mengukur tingkat keacakan data hasil enkripsi menggunakan analisis entropi. Keacakan data merupakan indikator penting dalam menentukan seberapa baik algoritma enkripsi menyamarkan pola asli dari data. Semakin tinggi tingkat keacakan, semakin sulit bagi pihak yang tidak berwenang untuk menganalisis pola atau memprediksi isi data terenkripsi. Pengujian ini dilakukan dengan menghitung nilai entropi menggunakan distribusi probabilitas setiap simbol dalam cipherteks. Nilai entropi yang mendekati maksimum menunjukkan bahwa distribusi simbol dalam cipherteks hampir merata, menandakan bahwa algoritma telah berhasil menghasilkan data yang acak. Data dengan keacakan tinggi lebih tahan terhadap serangan berbasis statistik, sehingga pengujian ini memberikan evaluasi kuantitatif terhadap kekuatan keamanan algoritma yang diuji. Hasil pengujian dibandingkan antara AES Standar, AES Modifikasi, Chacha20 dan Double Encryption untuk menunjukkan efektivitas modifikasi dalam meningkatkan keamanan.

### Pengujian Waktu dan Kecepatan

Menguji algoritma AES modifikasi dan *Double encrypt* AES modifikasi dan ChaCha20. Pengujian ini dilakukan untuk menilai kinerja algoritma dalam berbagai parameter. Penelitian ini akan menganalisis parameter pengujian menggunakan matriks dan formula berikut :

* + - * 1. Waktu enkripsi :

mengukur waktu yang dibutuhkan untuk mengenkripsi data dari

*plainteks* menjadi *chiperteks* dijelaskan dalam persamaan [(III-1)](#_bookmark55).

|  |  |
| --- | --- |
| 𝐸𝑡 = 𝐸𝑁𝑡 – 𝑆𝑇𝑡 | (III-1) |

Di mana:

𝐸𝑡 = waktu enkripsi dalam mili detik (ms)

𝐸𝑁𝑡 = waktu selesai enkripsi

𝑆𝑇𝑡 = waktu mulai enkripsi

Waktu ini akan dihitung untuk setiap *file* yang diuji, dan hasilnya nanti akan memberikan gambaran mengenai kecepatan algoritma dalam melakukan proses enkripsi.

* + - * 1. Waktu *dekripsi* :

Mengukur waktu yang dibutuhkan untuk *mendekripsi* data dari

*chiperteks* menjadi *plainteks* dijelaskan pada persamaan [(III-2)](#_bookmark56).

Dimana:

𝐷𝑡 = waktu *dekripsi* (ms)

𝐸𝑁𝑡 = waktu selesai *dekripsi*

𝑆𝑇𝑡 = waktu mulai *dekripsi*

Hasil waktu *dekripsi* ini juga perlu untuk masing-masing algoritma karena ini menganalisis perbandingan kinerja antara algoritma enkripsi.

* + - * 1. Kecepatan :

Menunjukkan jumlah data dalam *kilobyte* yang dapat dienkripsi

Et = waktu enkripsi atau *dekripsi* (s)

### Pengujian pada Berbagai Jenis *File*

*File* PDF yang digunakan dalam pengujian dapat berisi elemen seperti teks, gambar, dan tabel yang semuanya akan dienkripsi secara utuh karena algoritma bekerja pada level data biner. *File* PDF dengan ukuran 100 KB, 1 MB, 10 MB, hingga 100 MB diuji untuk memastikan algoritma dapat bekerja pada dokumen berukuran kecil hingga besar. Hal serupa juga berlaku untuk *file* Word dengan ukuran yang sama, meskipun memiliki struktur internal yang lebih kompleks dibandingkan PDF. *File* TXT dengan ukuran maksimal 300 *byte*, yang biasanya hanya berisi teks sederhana, juga dapat dienkripsi ChaCha20 (*double encrypt*). Di mana pengujian ini bertujuan untuk

mengukur kinerja algoritma. Berikut ini adalah contoh hasil pengujian terhadap *file* PDF :

*Tabel III.2 Evaluasi Hasil*

Pada [Tabel III.2 Evaluasi Hasil](#_bookmark60) menunjukkan hasil evaluasi simulasi kinerja algoritma enkripsi dan *dekripsi*. pada Nilai-nilai tersebut memberikan gambaran awal mengenai performa beberapa metode, termasuk AES Standar, ChaCha20, AES Modifikasi, dan kombinasi AES Modifikasi dengan ChaCha20. Hasil ini digunakan sebagai contoh output yang diharapkan dari model algoritma yang akan dirancang, ini belum merupakan data dari eksperimen nyata.

# DAFTAR PUSTAKA

Alimuddin, A., Juntak, J. N. S., Jusnita, R. A. E., Murniawaty, I., & Wono, H. Y. (2023). Teknologi dalam pendidikan: Membantu siswa beradaptasi dengan revolusi industri 4.0. *Journal on Education*, *5*(4), 11777–11790.

Almukhlifi, R., & Vora, P. L. (2020). Linear cryptanalysis of reduced-round simon using super rounds. *Cryptography*, *4*(1), 1–34. https://doi.org/10.3390/cryptography4010009

Alyas, H. H., & Abdullah, A. A. (2021). Enhancement the ChaCha20 Encryption Algorithm Based on Chaotic Maps. *Lecture Notes in Networks and Systems*, *201*, 91–107. https://doi.org/10.1007/978-981-16-0666-3\_10

Anindita, M., & 18219086, A. A. (2023). *Analisis Perbandingan Algoritma AES dan ChaCha20-Poly1305 dalam Enkripsi Konten Livestreaming*.

Aryanto, M. B., Tahir, M., Devita, S. I., Mustofa, Z. N., Ainiyah, Q., & Sundoro,

S. (2023). Implementasi Enkrip Dan Dekrip File Menggunakan Metode Advance Encryption Standard (AES-128). *Jurnal Ilmiah Sistem Informasi Dan Ilmu Komputer*, *3*(1), 89–104.

Assyahid, M. H. (2024). *Implementasi interpolasi pixel wadah citra dan enkripsi aes pada steganografi least significan bit (LSB)*. https://repository.uinjkt.ac.id/dspace/handle/123456789/76727

Barbero, S., Bazzanella, D., & Bellini, E. (2022). Rotational Cryptanalysis on ChaCha Stream Cipher. *Symmetry*, *14*(6). https://doi.org/10.3390/sym14061087

Belvin Tannadi. (2022). *Ilmu Crypto* (Digital). Jakarta : Elex Media Komputindo.

Cristy, N., & Riandari, F. (2021). *Niolinda Cristy 1 , Fristi Riandari 2 [Implementasi Metode Advanced Encryption Standard (AES 128 Bit) Untuk Mengamankan Data Keuangan*. *4*(2), 75.

Djong, H. S., & Siswanto, S. (2022). Implementasi Kriptografi Dengan Menggunakan Metode RC4 dan AES-256 Untuk Mengamankan File Dokumen pada PT Varnion Technology Semesta. *Prosiding Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI)*, *1*(1), 149–158.

Handoyo, J., & Subakti, Y. M. (2020). Keamanan Dokumen Menggunakan Algoritma Advanced Encryption Standard (AES). *Jurnal SITECH : Sistem Informasi Dan Teknologi*, *3*(2), 143–152. https://jurnal.umk.ac.id/index.php/sitech/article/view/5865

Hena, N., & Kusumaningsih, D. (2024). Pengamanan File Pada Sistem Master Vendor Berbasis WEB Menggunakan Algoritma AES Pada Trinitiland. *Prosiding Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI)*, *3*(2), 30–37.

Hidayatulloh, N. W., Tahir, M., Amalia, H., Basyar, N. A., Prianggara, A. F., & Yasin, M. (2023). Mengenal Advance Encryption Standard (AES) sebagai Algoritma Kriptografi dalam Mengamankan Data. *Digital Transformation Technology*, *3*(1), 1–10.

Kebande, V. R. (2023). Extended-Chacha20 Stream Cipher With Enhanced Quarter Round Function. *IEEE Access*, *11*, 114220–114237. https://doi.org/10.1109/ACCESS.2023.3324612

Kim, B. G., & Wong, D. (2023). Smart Contract-based Secure Verifiable Random Function using ChaCha20 Sequence in Blockchain. *ACM International Conference Proceeding Series*, 41–51. https://doi.org/10.1145/3638025.3638028

Kurnia Mahesa Naibaho, A., Putri, A., & Fikri Husaini, Y. (2021). Analisis Sistem Keamanan Informasi Chat Online dengan Menggunakan Algoritma AES. In *Ilmu Komputer* (Vol. 6, Issue 2).

Luong, T. T. (2023). Strengthening AES Security Through Key-Dependent ShiftRow and AddRoundKey Transformations Utilizing Permutation. In *IJACSA) International Journal of Advanced Computer Science and Applications* (Vol. 14, Issue 11). [www.ijacsa.thesai.org](http://www.ijacsa.thesai.org/)

Muhammed, R. K., Rashid Aziz, R., Hassan, A. A., Aladdin, A. M., Saydah, S. J., Rashid, T. A., & Hassan, A. (2024). *Comparative Analysis of AES, Blowfish, Twofish, Salsa20, and ChaCha20 for Image Encryption*.

Olivia Putri Irine Irawan, B., Tahir, M., Ayu Windrastuti, N., Yurina Cholili, D., Mulaikah, D., Batsul Mushofi Septian wachid, A., & Pendidikan Informatika,

P. (2023). Implementasi Kriptografi Pada Keamanan Data Menggunakan Algoritma Advance Encryption Standard (AES). *Jurnal Simantec*, *11*(2), 167–

174. https://doi.org/10.21107/SIMANTEC.V11I2.20034

Pangestu, G. Y., Hadiana, A. I., & Sabrina, P. N. (2022). Kriptografi Untuk Enkripsi Ganda Pada Gambar Menggunakan Algoritma AES (Advanced Encryption Standard) Dan RC5 (Rivest Code 5). *Vol*, *1*, 25–32.

Putri, A. E., Kartikadewi, A., & Rosyid, L. A. A. (2021). Implementasi Kriptografi dengan Algoritma Advanced Encryption Standard (AES) 128 Bit dan Steganografi menggunakan Metode End of File (EOF) Berbasis Java Desktop

pada Dinas Pendidikan Kabupaten Tangerang. *Appl. Inf. Syst. Manag*, *3*(2), 69–78.

Saripa, S. (2023). Implementation Implementation of a File Security System Using the AES Algorithm to Secure Personal Files: Implementasi Sistem Keamanan File Menggunakan Algoritma AES untuk Mengamankan File Pribadi. *Progressive Information, Security, Computer, and Embedded System*, *1*(2), 138–148.

Sekar Putri Ananda, Saepul Lukman, & Irfan. (2022). *Analisa Metode Kriptogra Modern Advance Encryption Standar (AES) 128 Bit dalam Mengenkripsi dan Mendekripsi File Dokumen Digital*. https://doi.org/10.32409/jikstik.21.3.2973

Soesanto, E., Purba, L. M., Aprilia, B., Putra, D. R., & Putri, S. D. (2023). Implementasi Objek Vital, Pengamanan File dan Pengamanan Cyber di PT Pertamina. *IJM: Indonesian Journal of Multidisciplinary*, *1*(1).

Surbakti, P. S., & Purwanto, P. (2024). Implementasi Kriptografi Dengan Menggunakan Algoritma AES-128 Untuk Enkripsi dan Dekripsi File Dokumen Berbasis Web Pada Law Office Erdi Surbakti, SH & Rekan. *Prosiding Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI)*, *3*(1), 1–8.

Syahputra Sianipar, J., Budi Nuugroho, N., Mariami, I., Informasi, S., & Triguna Dharma, S. (2024). Pengamanan Data Gaji Karyawan Dengan Menggunakan Metode Advanced Encryption Standard (AES). *Jurnal Sistem Informasi Triguna Dharma (JURSI TGD)*, *3*(1), 35–45. https://doi.org/10.53513/JURSI.V3I1.5653

Wahyu Putra, M. R. F. T. H. (2023, July 2). *Implementasi Algoritma Advanced Encryption Standard Untuk Kemanan Dokumen*. https://journal.grahamitra.id/index.php/jurikti/article/view/55/181

Wahyudi, E. N., Ardhianto, E., Handoko, W. T., Murti, H., Supriyanto, E., Lestariningsih, E., & Redjeki, R. S. (2024). Peningkatan Keamanan Data Melalui Teknik Super Enkripsi Menggunakan Algoritma Vigenere dan Caesar. *Jurnal Informatika Polinema*, *10*(3), 315–322. https://doi.org/10.33795/JIP.V10I3.5131

Widyawan, D., & Imelda, I. (2021). Pengamanan File Menggunakan Kriptografi Dengan Metode Aes-128 Berbasis Web Di Komite Nasional Keselamatan Transportasi. *SKANIKA: Sistem Komputer Dan Teknik Informatika*, *4*(1), 15– 22.