JIKUM
JURNAL ILMU KOMPUTER

E-ISSN: 9999-9999
Volume 1, **Number** 1, Mei 2025
https://doi.org/10.62671/jikum.vxix.xxxx

Implementasi Algoritma Advanced Encryption Standard (AES) Secara Manual Menggunakan Python

Farhan Yusri[1*], Agil Muttaqin[2], Abdul Hafiz As Syamil[3], Mahadika Nafis Luqman[4], Citra[5], Mursyid[6], M.Fajar Sidiq[7], Fathar Arsa Siregar[8]
[1,2,3,4,5,6,7,8]Universitas Malikussaleh, Indonesia
[1]farhan.220170125@mhs.unimal.ac.id, [2]agil.220170133@mhs.unimal.ac.id, [3]abdul.220170160@mhs.unimal.ac.id,
[4]mahadika.220170110@mhs.unimal.ac.id,[5]citra.210170063@mhs.unimal.ac.id,
[6]mursyid.220170069@mhs.unimal.ac.id, [7]fajar.220170209@mhs.unimal.ac.id, [8]fathar.220170122@mhs.unimal.ac.id

ABSTRACT

Algoritma Advanced Encryption Standard (AES) merupakan salah satu algoritma kriptografi simetris yang paling banyak digunakan dalam menjaga kerahasiaan data digital. Penelitian ini bertujuan untuk memahami struktur dan cara kerja internal algoritma AES melalui implementasi manual menggunakan bahasa pemrograman Python. Proses enkripsi dilakukan secara bertahap, mencakup tahapan utama AES: SubBytes, ShiftRows, MixColumns, AddRoundKey, dan Key Expansion. Setiap tahapan direalisasikan dalam bentuk fungsi Python dan diuji menggunakan data input yang direpresentasikan dalam bentuk matriks. Hasil implementasi menunjukkan bahwa proses enkripsi berhasil membentuk blok cipher melalui transformasi bertahap sesuai dengan standar algoritma AES. Penelitian ini memberikan pemahaman yang lebih dalam mengenai konsep-konsep kriptografi modern serta pentingnya enkripsi dalam perlindungan data.

Kata Kunci: *Advanced Encryption Standard* (AES), Kriptografi Simetris, Python, Deskripsi Data, Enkripsi Data

PENDAHULUAN

Perkembangan teknologi informasi yang pesat menyebabkan meningkatnya kebutuhan terhadap keamanan data, terutama dalam sistem komunikasi digital. Salah satu pendekatan utama dalam menjaga kerahasiaan dan integritas informasi adalah melalui kriptografi. Algoritma kriptografi yang handal tidak hanya menjamin kerahasiaan data, namun juga mampu menghadapi berbagai jenis serangan. Advanced Encryption Standard (AES) telah menjadi standar enkripsi global yang digunakan secara luas di berbagai sektor, mulai dari pemerintahan hingga industri teknologi informasi. AES dipilih karena struktur algoritmanya yang efisien dan kemampuannya dalam menjaga keamanan data bahkan terhadap serangan kriptografi modern. Meski tersedia banyak pustaka (library) pemrograman untuk mengimplementasikan AES, memahami proses kerjanya secara manual sangat penting dalam bidang akademik dan praktis. Penelitian ini berfokus pada implementasi manual algoritma AES menggunakan bahasa pemrograman Python. Dengan memecah setiap tahapan algoritma ke dalam fungsi-fungsi terpisah, diharapkan pengguna dapat memperoleh wawasan mendalam mengenai prinsip kerja masing-masing komponen dalam proses enkripsi AES.

TINJAUAN PUSTAKA

Kriptografi dan Keamanan Data

Kriptografi adalah cabang ilmu komputer dan matematika yang mempelajari teknik-teknik untuk menjaga keamanan informasi, terutama dalam konteks komunikasi digital. Menurut Stallings (2017), kriptografi modern bertujuan untuk menjaga kerahasiaan (confidentiality), keutuhan (integrity), otentikasi (authentication), dan non-repudiation suatu pesan atau data. Dua jenis utama dalam kriptografi adalah kriptografi simetris dan asimetris. Kriptografi simetris menggunakan satu kunci rahasia yang sama untuk proses enkripsi dan dekripsi data. Sistem ini dikenal cepat dan efisien, sehingga banyak digunakan dalam pengamanan data internal dan komunikasi skala besar (Katz & Lindell, 2014). Salah satu algoritma kriptografi simetris yang paling banyak digunakan adalah **AES (Advanced Encryption Standard**).

Sejarah Evolusi Algoritma AES

Algoritma AES dikembangkan sebagai pengganti Data Encryption Standard (DES) yang dinilai tidak lagi cukup aman karena panjang kunci yang hanya 56-bit. Pada tahun 2001, National Institute of Standards and Technology (NIST) secara resmi menetapkan algoritma Rijndael sebagai standar AES setelah melalui kompetisi internasional terbuka (NIST, 2001). Rijndael, yang dikembangkan oleh Vincent Rijmen dan Joan Daemen, dipilih karena memiliki struktur yang sederhana namun kuat dalam menahan berbagai jenis serangan kriptografi seperti serangan diferensial dan linier. AES bekerja pada blok data berukuran 128 bit dan mendukung panjang kunci 128, 192, atau 256 bit. Dengan struktur yang terdiri dari 10 hingga 14 putaran (rounds) tergantung panjang kunci, AES menjadi algoritma ideal untuk digunakan dperangkat keras dan perangkat lunak modern.

JIKUM
JURNAL ILMU KOMPUTER
E-ISSN: 9999-9999
Volume 1, **Number** 1, Mei 2025
https://doi.org/10.62671/jikum.vxix.xxxx

Struktur Komponen Algoritma AES

Menurut Daemen dan Rijmen (2002), algoritma **Advanced Encryption Standard (AES)** dirancang dengan struktur arsitektur yang dikenal sebagai **Substitution-Permutation Network (SPN)**. Struktur ini merupakan kombinasi sistematis dari proses substitusi dan permutasi yang bertujuan untuk menciptakan keamanan kriptografi yang tinggi melalui prinsip confusion (kerancuan) dan diffusion (penyebaran informasi).

1. SubBytes

Merupakan proses substitusi non-linear di mana setiap byte dari blok data digantikan menggunakan sebuah tabel pengganti yang disebut **S-Box**. Proses ini memperkenalkan elemen confusion ke dalam data, sehingga hubungan antara plaintext dan ciphertext menjadi lebih kompleks dan sulit ditebak.

2. ShiftRows

Pada tahap ini, baris-baris dalam matriks data hasil dari SubBytes digeser secara sistematis. Baris pertama tetap, baris kedua digeser satu byte ke kiri, baris ketiga dua byte, dan baris keempat tiga byte. Pergeseran berfungsi menyebarkan pengaruh byte dalam struktur data, meningkatkan sifat diffusion sehingga perubahan kecil pada input dapat memengaruhi output secara signifikan.

3. MixColumns

Tahapan ini mencampurkan data antar kolom menggunakan operasi matematika dalam **medan hingga GF(2^8)**. Setiap kolom dari matriks dianggap sebagai polinomial dan dikalikan dengan polinomial tetap. Proses ini dirancang untuk memperkuat difusi lebih lanjut, menyebarkan efek dari setiap byte ke seluruh blok data.

4. AddRoundKey

Pross ini melibatkan penggabungan data dengan kunci sesi (round key) yang dihasilkan dari proses ekspansi kunci. Penggabungan dilakukan melalui operasi bitwise XOR, yang sederhana namun efektif dalam mengacak data, serta menjamin bahwa hasil enkripsi sangat bergantung pada nilai kunci yang digunakan.

5. Key Expansion

Sebelum digunakan dalam tahapan AddRoundKey, kunci utama perlu diperluas menjadi beberapa kunci sesi atau **round key** melalui algoritma khusus yang menggabungkan rotasi byte, substitusi byte, dan penambahan konstanta round. Proses ini memastikan setiap ronde enkripsi menggunakan kunci yang unik namun masih berkaitan dengan kunci utama, menambah lapisan kompleksitas dalam enkripsi.

1.4. Implementasi Manual AES dengan Python

Implementasi algoritma AES secara manual di lingkungan pemrograman seperti Python merupakan langkah penting untuk memahami cara kerja internal algoritma. Dalam studi oleh Mohd, Sahibuddin & Sulaiman (2020), implementasi mandiri algoritma kriptografi memberikan wawasan yang lebih dalam terhadap struktur enkripsi dibandingkan hanya menggunakan library pihak ketiga seperti **PyCryptodome** atau **cryptography**. Dengan menggunakan Python, setiap langkah AES seperti **SubBytes**, **ShiftRows**, **MixColumns**, dan **AddRoundKey** dapat diuraikan menjadi fungsi yang merepresentasikan masing operasi transformasi. Penggunaan operasi seperti XOR (^), penggeseran bit (>> dan <<), serta perkalian dalam medan GF(2^8) (xtime) merupakan inti dari manipulasi data dalam AES.

1.5 Peran AES dalam Keamanan Modern

Saat ini, **AES (Advanced Encryption Standard)** telah menjadi standar industri yang digunakan secara luas dalam berbagai aplikasi keamanan digital di seluruh dunia. Keandalannya dalam menjaga kerahasiaan data membuat algoritma ini diadopsi dalam sistem-sistem krusial yang memerlukan proteksi tinggi terhadap informasi yang bersifat sensitif. Penggunaan AES tidak terbatas hanya pada lembaga pemerintah atau sektor keuangan, tetapi juga telah menyebar ke berbagai bidang teknologi informasi yang bersifat umum maupun khusus. Beberapa contoh penerapannya meliputi:

1. Enkripsi file dan basis data

AES digunakan untuk mengenkripsi file individual maupun seluruh sistem file agar tidak dapat diakses oleh pihak yang tidak berwenang. Contoh implementasinya dapat ditemukan pada aplikasi seperti **BitLocker** (fitur enkripsi drive milik Microsoft Windows) dan **VeraCrypt** (aplikasi open-source enkripsi disk penuh dan kontainer terenkripsi). AES memastikan bahwa bahkan jika media penyimpanan dicuri atau hilang, data tetap tidak dapat dibaca tanpa kunci yang benar.

2. Protokol komunikasi

Dalam dunia jaringan komputer, AES merupakan komponen penting dari berbagai protokol komunikasi yang aman seperti **HTTPS**, **SSH**, dan **VPN**. Pada protokol **HTTPS**, yang digunakan dalam hampir semua komunikasi web modern, AES berperan dalam mengenkripsi data antara browser dan server agar tidak dapat diintip atau dimanipulasi oleh pihak ketiga. Demikian pula, dalam **Virtual Private Network (VPN)**, AES menjaga integritas dan kerahasiaan data yang ditransmisikan melalui jaringan publik.

3. Enkripsi end-to-end

JIKUM
JURNAL ILMU KOMPUTER
E-ISSN: 9999-9999
Volume 1, **Number** 1, Mei 2025
https://doi.org/10.62671/jikum.vxix.xxxx

Banyak aplikasi pesan modern seperti **WhatsApp**, **Signal**, dan **Telegram** menerapkan enkripsi end-to-end menggunakan AES untuk memastikan hanya pengirim dan penerima pesan yang dapat membaca isi komunikasi. Bahkan pihak pengelola aplikasi tidak dapat mengakses pesan yang dienkripsi dengan mekanisme ini. AES digunakan bersama algoritma lain seperti Diffie-Hellman untuk pertukaran kunci.

METODE PENELITIAN

Penelitian ini menggunakan pendekatan implementatif dengan menyusun kembali setiap tahapan algoritma AES secara manual dalam bahasa Python. Tahapan-tahapan utama yang diimplementasikan meliputi:

1. SubBytes – melakukan substitusi byte dalam blok data menggunakan S-Box.
2. ShiftRows – menggeser setiap baris dalam matriks data untuk meningkatkan difusi.
3. MixColumns – mencampur nilai antar kolom dengan operasi dalam medan hingga GF(2^8).
4. AddRoundKey – menggabungkan data dengan round key melalui operasi XOR.
5. Key Expansion – memperluas kunci utama menjadi beberapa round key dengan operasi RotWord, SubWord, dan konstanta Rcon.

Setiap fungsi diuji menggunakan input berbentuk matriks 4x4 yang merepresentasikan blok data 128-bit. Proses enkripsi berdasarkan struktur Substitution-Permutation Network (SPN) yang merupakan dasar AES.

HASIL DAN PEMBAHASAN

Penelitian bertujuan menerapkan algoritma kriptografi **AES (Advanced Encryption Standard)** secara manual menggunakan Python sebagai bahasa pemrograman. Proses implementasi dilakukan dalam beberapa tahapan utama yang mengikuti prosedur standar dari AES, yaitu: **SubBytes**, **ShiftRows**, **MixColumns**, **AddRoundKey**, dan **Key Expansion**. Setiap proses diuji input untuk memastikan fungsionalitas program secara logis dan teknis.

2.1. SubBytes (Substitusi Non-Linear)

Tahapan SubBytes melakukan substitusi byte pada blok data (state) dengan menggunakan S-Box (Substitution Box) AES. Ini berfungsi untuk menghasilkan non-linearitas yang tinggi, sebagai bagian dari prinsip confusion dalam kriptografi.

Input State	state = [[0x32, 0x88, 0x31, 0xe0], [0x43, 0x5a, 0x31, 0x37], [0xf6, 0x30, 0x98, 0x07], [0xa8, 0x8d, 0xa2, 0x34]]

Setiap byte pada matriks **state** dikonversi melalui **S-Box**. Misalnya, byte **0x32** memiliki:	1. Nilai row = **0x3** (3) 2. Nilai col = **0x2** (2)

Maka nilai substitusinya diambil dari **sbox[3] [2]**.	[[0xb8, 0x1a, 0x23, 0x9c], [0x3f, 0x7d, 0x23, 0x55], [0x49, 0xeb, 0x83, 0x2f], [0x63, 0xd2, 0x5e, 0x18]]

2.2. ShiftRows (Pergeseran Baris)

ShiftRows bertugas menggeser setiap baris dalam blok **state** ke kiri berdasarkan indeks barisnya. Langkah ini meningkatkan difusi antar byte dalam blok.

Input	[[0xb8, 0x1a, 0x23, 0x9c], [0x3f, 0x7d, 0x23, 0x55], [0x49, 0xeb, 0x83, 0x2f], [0x63, 0xd2, 0x5e, 0x18]

JIKUM
JURNAL ILMU KOMPUTER
E-ISSN: 9999-9999
Volume 1, **Number** 1, Mei 2025
https://doi.org/10.62671/jikum.vxix.xxxx

	]

Cara Penyelesaian	1. Baris ke-1 tidak digeser. 2. Baris ke-2 digeser satu langkah ke kiri: [0x7d, 0x23, 0x55, 0x3f] 3. Baris ke-3 digeser dua langkah: [0x83, 0x2f, 0x49, 0xeb] 4. Baris ke-4 digeser tiga langkah: [0x18, 0x63, 0xd2, 0x5e]

Hasil ShiftRows	[[0xb8, 0x1a, 0x23, 0x9c], [0x7d, 0x23, 0x55, 0x3f], [0x83, 0x2f, 0x49, 0xeb], [0x18, 0x63, 0xd2, 0x5e]]

2.3. MixColumns (Pencampuran Kolom)

MixColumns mencampurkan setiap kolom pada blok state dengan menggunakan operasi aritmetika dalam medan hingga GF(2^8) untuk menghasilkan difusi tambahan antar elemen.

Input Kolom pertama sebelum MixColumns:	[0xb8, 0x7d, 0x83, 0x18]

Cara Penyelesaian Kolom ini dikalikan dengan matriks konstan berikut:	[2, 3, 1, 1] [1, 2, 3, 1] [1, 1, 2, 3] [3, 1, 1, 2]

Dengan operasi XOR dan fungsi **xtime** untuk menggandakan nilai di GF(2^8).	[[0x04, 0x66, 0x81, 0xe5], [0xe0, 0xcb, 0x19, 0x9a], [0x48, 0xf8, 0xd3, 0x7a], [0x28, 0x06, 0x26, 0x4c]]

2.4. AddRoundKey (Penambahan Kunci Sementara)

Melakukan operasi XOR antara **state** saat ini dengan kunci bulat (round key) yang dihasilkan dari proses ekspansi kunci.

Input	state = [[0x04, 0x66, 0x81, 0xe5], [0xe0, 0xcb, 0x19, 0x9a], [0x48, 0xf8, 0xd3, 0x7a], [0x28, 0x06, 0x26, 0x4c]] round_key = [[0xa0, 0x88, 0x23, 0x2a], [0xfa, 0x54, 0xa3, 0x6c], [0xfe, 0x2c, 0x39, 0x76], [0x17, 0xb1, 0x39, 0x05]]

Cara Penyelesaian
Lakukan XOR untuk setiap elemen:
	def add_round_key(state, round_key):
	for i in range(4):
	for j in range(4):
	state[i][j] ^= round_key[i][j]
	return state

JIKUM
JURNAL ILMU KOMPUTER

E-ISSN: 9999-9999
Volume 1, **Number** 1, Mei 2025
https://doi.org/10.62671/jikum.vxix.xxxx

Hasil AddRoundKey	[[0xa4, 0xee, 0xa2, 0xcf], [0x1a, 0x9f, 0xba, 0xf6], [0xb6, 0xd4, 0xea, 0x0c], [0x3f, 0xb7, 0x1f, 0x49]]

2.5. Key Expansion (Ekspansi Kunci)

Membuat kunci untuk setiap ronde dari kunci utama. Kunci-kunci ini akan digunakan dalam AddRoundKey pada setiap ronde.

Cara Penyelesaian
Menggunakan operasi **RotWord**, **SubWord**, dan konstanta **Rcon**. Misalnya, untuk kunci 128-bit:
	def key_expansion(key):
	# Ekspansi hingga 44 word (4 word × 11 round)
	expanded_key = ...
	return expanded_key

Hasil Key Schedule	Round 0: [0x2b, 0x28, 0xab, 0x09, ...] Round 1: [0xa0, 0x88, 0x23, 0x2a, ...]

KESIMPULAN

Implementasi manual algoritma AES dalam bahasa Python memberikan pemahaman yang lebih dalam terhadap cara kerja dan struktur internal algoritma enkripsi modern. Setiap tahapan dalam proses enkripsi, mulai dari substitusi hingga ekspansi kunci, dapat direpresentasikan dengan jelas melalui pendekatan pemrograman. Hasil akhir dari enkripsi menunjukkan keberhasilan proses transformasi data plaintext menjadi ciphertext yang aman. Penelitian ini membuktikan bahwa implementasi manual dapat menjadi alat pembelajaran yang efektif untuk memahami kriptografi, khususnya algoritma AES, dan penting dalam pengembangan sistem keamanan data digital yang handal.

REFERENSI

A. E. Standard, K. Dokumen, and B. B. Testing, “1,2 1* , 2,” no. November 2018, pp. 1044–1052, 2024.

F. Bibiola, T. U. Kalsum, and H. Alamsyah, “Penerapan Algoritma Advance Encryption Standard (AES) Untuk Pengamanan File Pada Aplikasi Berbasis WEB,” *J. Surya Energy*, vol. 8, no. 1, p. 35, 2023, doi: 10.32502/jse.v8i1.6461.

J. Handoyo and Y. M. Subakti, “Keamanan Dokumen Menggunakan Algoritma Advanced Encryption Standard (Aes),” *J. SITECH Sist. Inf. dan Teknol.*, vol. 3, no. 2, pp. 143–152, 2020, doi: 10.24176/sitech.v3i2.5865.

R. V. H. Chandra, A. Kusyanti, and M. Data, “Analisis Performa Proses Enkripsi dan Dekripsi Menggunakan Algoritme AES-128 Pada Berbagai Format File,” *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 3, no. 1, pp. 481–486, 2019, [Online]. Available: http://j-ptiik.ub.ac.id.

Tarisa Auliya Ramadhani, A. Fajaryanto Cobantoro, and S. Sugianti, “Implementasi Algoritma Advanced Encryption Standard 128 untuk Pengamanan Database Sistem Registrasi Pasien,” *J. Inform. Polinema*, vol. 10, no. 4, pp. 521–526, 2024, doi: 10.33795/jip.v10i4.5619.