ANTIVIRUS: Jurnal Ilmiah Teknik Informatika (p – ISSN: 1978 – 5232; e – ISSN: 2527 – 337X)
Vol. 18 No. 2 November 2024, pp. 177 – 190

SKEMA PENGAMANAN DATA DENGAN KOMBINASI ALGORITMA *ADVANCED ENCRYPTION STANDARD* (AES) DAN *RIVEST SHAMIR ALDEMAN* (RSA)

Diterima Redaksi: 4 Desember 2023; Revisi Akhir: 11 Juni 2024; Diterbitkan Online: 15 November 2024

Hedy Sholihat Ruhaedi[1)], Alam Rahmatulloh[2)]
[1, 2)] Informatika, Fakultas Teknik Universitas Siliwangi
[1, 2)] Mugarsari, Kec. Tamansari, Kota Tasikmalaya, Jawa Barat, Indonesia, kode pos: 46196
e-mail: 207006028@student.unsil.ac.id[1)], alam@unsil.ac.id[2)]

***Abstrak*:** *Pesatnya perkembangan data mendorong kebutuhan keamanan dalam proses pengiriman data, dengan tujuan mencegah akses pihak yang tidak diinginkan untuk melihat isi dari data yang dikirimkan. Dalam rangka mencapai tingkat keamanan yang optimal, penerapan kriptografi dapat menjadi alternatif. Kriptografi dengan algoritma AES atau RSA secara terpisah telah sering dimanfaatkan dalam melindungi keamanan pesan, baik berbentuk teks maupun citra digital. Dalam penelitian ini memanfaatkan kombinasi dari algoritma AES dan RSA. Penggabungan ini bertujuan agar algoritma RSA dapat berperan dalam proses enkripsi kunci rahasia, sementara algoritma AES berperan dalam mengamankan teks asli (plaintext). Pengirim dapat menerima ciphertext. Penerima melakukan dekripsi kunci rahasia dengan menggunakan algoritma RSA, dan setelah mendapatkan kunci rahasia, ciphertext didekripsi dengan menggunakan algoritma AES Bersama kunci rahasia yang telah didekripsi sebelumnya. Penelitian ini diharapkan dapat memberi keamanan yang lebih optimal dalam pengiriman informasi karena dalam prosesnya yang memanfaatkan kombinasi dari kedua algoritma tersebut. Pada penelitian ini dilakukan Enkripsi dan Dekripsi pada AES dan RSA. kemudian pada implementasi kombinasi algoritma AES dan RSA dibuatlah suatu program dari bahasa python. Proses dekripsi membutuhkan Waktu lebih lama dibandingkan proses enkripsi. Proses enkripsi menggunakan RSA membutuhkan waktu rata-rata 5,104 ms dan proses enkripsi menggunakan AES membutuhkan waktu rata-rata 4,543 ms. Sedangkan proses dekripsi menggunakan RSA membutuhkan waktu rata-rata 59,435 ms dan proses dekripsi menggunakan AES membutuhkan waktu rata-rata 2,925 ms.*

Kata Kunci— Ciphertext, Deskripsi, Enkripsi, Keamanan Pesan, Plaintext

***Abstract*:** *The rapid development of data drives the need for security in the data transmission process, with the aim of preventing access by unwanted parties to view the contents of the data sent. In order to achieve an optimal level of security, the application of cryptography can be an alternative. Cryptography with the AES or RSA algorithm separately has often been used to protect the security of messages, both in the form of text and digital images. This research uses a combination of the AES and RSA algorithms. This merger aims to ensure that the RSA algorithm can participate in the secret key encryption process, while the AES algorithm plays a role in securing the original text (plaintext). The sender can receive the ciphertext. The recipient decrypts the secret key using the RSA algorithm, and after obtaining the secret key, the ciphertext is decrypted using the AES algorithm along with the previously decrypted secret key. This research is expected to provide more optimal security in conveying information because the process utilizes a combination of the two algorithms. In this research, Encryption and Decryption were carried out on AES and RSA. then by implementing the combination of the AES and RSA algorithms a program was created in the Python language. The decryption process takes longer than the encryption process. The encryption process using RSA takes an average of 5,104 ms and the encryption process using AES takes an average of 4,543 ms. Meanwhile, the decryption process using RSA takes an average of 59.435 ms and the decryption process using AES takes an average of 2.925 ms.*

Keywords— Ciphertext, Description, Encryption, Message Security, Plaintext

DOI: https://doi.org/1035457/antivirus.v18i2.3369

I. PENDAHULUAN

PADA era digital saat ini yang semakin canggih, keamanan informasi menjadi aspek krusial dalam berbagai bentuk komunikasi. Aplikasi pesan sering digunakan untuk pertukaran pesan pribadi dan sensitif. Salah satu metode yang umum digunakan untuk memastikan kerahasiaan pesan selama pengiriman adalah enkripsi *end-to-end*. Dalam konteks ini, Dengan menggabungkan algoritma AES dan RSA diharapkan menjadi pilihan yang sangat efektif untuk menjaga keamanan sistem komunikasi dalam aplikasi pesan [1].

Keamanan data dengan algoritma AES-128 dapat dimanfaatkan untuk enkripsi dan dekripsi dokumen. Ukuran file adalah faktor penting karena dapat mempengaruhi lamanya proses tersebut. Variabel hasil menjadi patokan untuk proses kecepatan tergantung pada ukuran file. Selama kunci enkripsi simetris tidak bocor ke pihak luar, hasil enkripsi dapat dijamin aman. Jika file dienkripsi selanjutnya kembali seperti sebelum dienkripsi, penelitian ini dapat membuktikan hal itu. Proses enkripsi 7,1MB membutuhkan 3,3 detik, sedangkan 1,8MB membutuhkan 1 detik. Proses dekripsi 7,2MB membutuhkan 2,5 detik, dan 1,8MB membutuhkan 0,4 detik [2].

Dua metode pembacaan digunakan oleh algoritma RSA untuk melindungi data: enkripsi dan dekripsi. Enkripsi dapat membuat file menjadi tidak terbaca, sedangkan dekripsi dapat mengubah file yang tidak terbaca menjadi file asli. Kalimat sandi, atau passphrase, pada aplikasi ini sangat sensitif sehingga perlu diingat. Penulisan huruf harus dibedakan. Uji coba aplikasi pengamanan ini menemukan empat komponen keamanan: kerahasiaan, otentikasi, integritas data, dan nir-penyangkalan. Dalam penelitian, disarankan untuk mengambil perhatian pada keamanan file [3].

Keamanan data JAMKESMAS dapat ditingkatkan karena semua data disimpan dalam enkripsi, dan keaslian filter hanya dapat dilihat setelah file didekripsi. File dienkripsi tidak berubah, dan ekstensi berubah menjadi "AES". Dengan sistem keamanan ini, pihak desa dapat membantu menjaga data penting seperti pilihan JAMKESMAS [4].

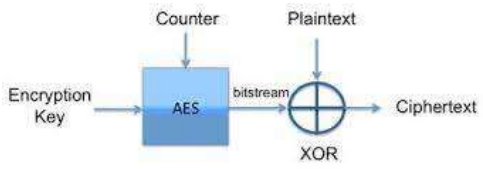
Penelitian sebelumnya hanya menggunakan algoritma AES atau RSA, jadi penelitian ini menggabungkan kedua algoritma untuk mendapatkan keamanan terbaik. Penelitian ini diharapkan dapat memberi keamanan yang lebih optimal dalam pengiriman informasi karena dalam prosesnya yang memanfaatkan kombinasi dari kedua algoritma tersebut. Pada penelitian ini dilakukan Enkripsi dan Dekripsi pada AES dan RSA. kemudian pada implementasi kombinasi algoritma AES dan RSA dibuatlah suatu program dari bahasa python.

II. TINJAUAN PUSTAKA

A. *Kriptografi*

Kriptografi merupakan disiplin matematika yang terkait dengan transformasi data untuk menghasilkan makna yang tidak dapat dipahami, dan mencegah penggunaan yang tidak sah [5]. Kita menyadari bahwa keamanan sangat penting untuk mencegah penyalahgunaan data. Kriptografi dapat memastikan kerahasiaan pesan dengan menciptakan kunci rahasia yang sulit dimengerti [6]. Melakukan enkripsi maupun dekripsi merupakan dua cara langkah yang dilakukan pada kriptografi. Penamaan *"plaintext"* mengacu pada pesan yang akan dienkripsi [7]. Hanya penerima yang dapat membuka data yang dienkripsi dengan menggunakan kunci tertentu. Plaintext dienkripsi atau didekripsi memerlukan penggunaan bentuk kunci [8].

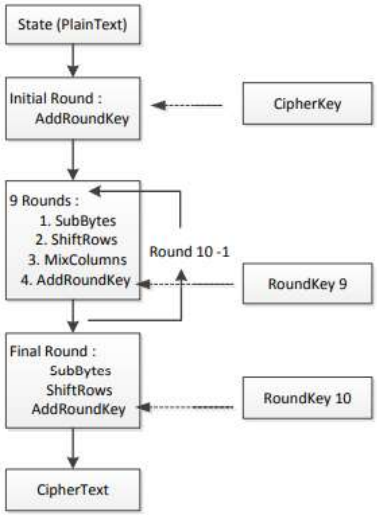
B. *AES*



Gambar 1. Proses Kriptografi Advanced Encryption Standard (AES)

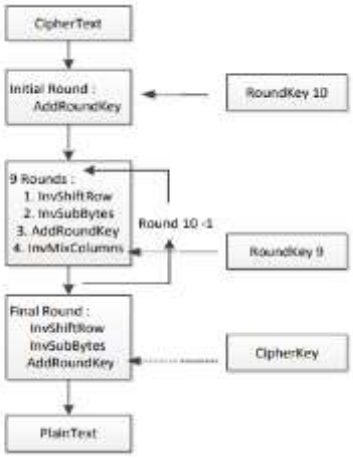
DOI: https://doi.org/1035457/antivirus.v18i2.3369

Pada awal proses enkripsi, status akan mengubah byte AddRoundKey. Kemudian, state akan mengubah SubBytes, ShiftRows, MixColumns, dan AddRoundKey sebanyak Nr, yang merupakan nilai round. Proses ini dikenal sebagai fungsi putaran. Pada putaran terakhir, proses berbeda dari putaran sebelumnya karena state tidak mengalami transformasi MixColumns. AES adalah algoritma kunci dalam mengamakan data. Sangat sulit untuk dipecahkan, dan ini adalah salah satu algoritma kunci yang paling aman. Algoritma AES adalah *chipertext* simetrik yang memiliki kemampuan untuk mengenkripsi dan dekripsi data [9]. Sementara enkripsi dapat berubah menjadi *ciphertext*, dekripsi mengubah data menjadi *plaintext*. Pengirim mengirimkan file ke server yang dienkripsi dengan menggunakan kunci rahasia AES. Ini mengenkripsi teks menjadi data *chiphertext*. Setelah itu, jika penerima memiliki kunci rahasia, mereka dapat membaca file. Kunci kriptografi digunakan oleh algoritma AES untuk mengenkrip dan dekripsi data [10].



Gambar 2. Diagram Alur Proses Enkripsi AES

Seperti pada Gambar 2. adalah beberapa jenis transformasi *byte* yang termasuk pada algoritma enkripsi standar Adavanced [11]. Pada tahap awal enkripsi, input disalin ke dalam state mengalami perubahan byte melalui proses yang dikenal sebagai AddRoundKey. Setelah tahap ini, state mengalami serangkaian transformasi, yang diulang sebanyak Nr. Proses ini, yang dikenal sebagai fungsi putaran, menciptakan langkah-langkah enkripsi dalam algoritma AES [12]. Berbeda dengan ronde sebelumnya, pada ronde terakhir, state tidak melakukan transformasi MixColumns [13].



Gambar 3. Diagram Alur Proses Deskripsi AES

DOI: https://doi.org/1035457/antivirus.v18i2.3369

Transformasi cipher dibalik dan digunakan pada arah berlawanan dalam membuat inverse cipher untuk algoritma AES [14]. Transformasi byte yang digunakan untuk invers cipher seperti dilihat pada Gambar 3. diatas [15].

C. RSA

Algoritma ini mengalikan dua bilangan prima besar, namun setelah kunci dibuat, bilangan prima yang asli tidak dibutuhkan lagi [16]. Algoritma ini tidak memerlukan pengiriman kunci privat. RSA termasuk ke dalam algoritma asimetri, karena memanfaatkan teknik kriptografi yang menggunakan kunci yang berbeda. Dalam algoritma RSA, kunci yang digunakan dapat dipublikasi atau diketahui banyak orang sehingga disebut juga algoritma kunci publik [17].

1. Enkripsi RSA

Menghitung chipertext c yang terkait pada n:

$$C = n^e \, mod \, N \; (1)$$

Penyelesaian perhitungan ini dapat dilakukan melalui metode *exponentiation by squaring*, suatu algoritma yang digunakan untuk menghitung nilai integer besar dengan efisien. Setelah itu, hasil perhitungan diterapkan pada variabel C dengan mengangkatnya ke pangkat B.

2. Dekripsi RSA

B telah menerima informasi dari A dan memiliki pengetahuan tentang kunci privat yang digunakan oleh B. Selanjutnya, B mengembalikan nilai n dengan mengikuti langkah-langkah berikut:

$$n = c^d \, mod \, N \; (2)$$

Melalui perhitungan tersebut, nilai n dihasilkan, memungkinkan B untuk mengembalikan pesan aslinya, yaitu m. Prosedur dekripsi bekerja karena

$$c^d = (c^d)^{ed} \; (mod \, N) \; (3)$$

Kemudian, karena $ed \equiv 1 \; (mod \, p-1)$ dan $ed \equiv 1 \; (mod \, q-1)$, hasil dari Fermat's little theorem.

$$n^{ed} \equiv n \; (mod \, p) \; (4)$$

dan

$$n^{ed} \equiv n \; (mod \, q) \; (5)$$

Dengan memanfaatkan *Chinese Remainder Theorem* pada bilangan prima beda pada p dan q, akan menghasilkan dua kondisi kongruen berlainan

$$n^{ed} \equiv n \; (mod \, pq) \; (6)$$

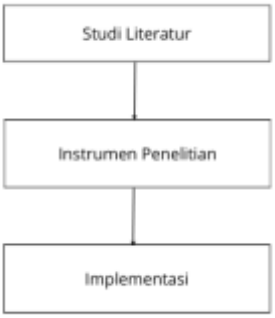
serta

$$c^d \equiv n \; (mod \, N) \; (7)$$

III. METODE PENELITIAN

Metode penelitian dalam penelitian ini menggunakan pendekatan hipotesis dan eksperimental dengan menggunakan data secara random atau acak.

DOI: https://doi.org/1035457/antivirus.v18i2.3369



Gambar 4. Tahapan Penelitian

A. Studi Literatur

Mencari literatur dengan mencari buku referensi seperti artikel, jurnal, dan internet mengenai topik yang dibahas yaitu algoritma AES dan RSA.

B. Instrumen Penelitian

Perangkat lunak (*Software*) yang dibutuhkan dalam penelitian ini sebagai berikut :

1) Sistem operasi windows 11
2) Visual Studio Code

Perangkat Keras (*Hardware*) yang dibutuhkan dalam penelitian ini sebagai berikut :

1) AMD Ryzen 5 5500U with Radeon Graphics 2.10 GHz
2) RAM 16,0 GB
3) System type base 64-bit operating system, x64-based processor

C. Implementasi

Untuk membuktikan kinerja gabungan dari algoritma AES dan RSA, maka dibuatlah suatu program dari bahasa pemrograman python untuk penerapan algoritma tersebut. Tujuan dari penggabungan algoritma tersebut adalah terciptanya keamanan yang lebih baik.

IV. HASIL DAN PEMBAHASAN

Penelitian ini memberikan penjelasan tentang skema pengamanan data. Pokok bahasan mencakup hasil dari setiap langkah mengamankan data dengan menggunakan kombinasi dari algoritma AES dan RSA.

A. Enkripsi dan Dekripsi AES

```
sbox = [
            0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67,
            0x2b, 0xfe, 0xd7, 0xab, 0x76, 0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59,
            0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0, 0xb7,
            0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1,
            0x71, 0xd8, 0x31, 0x15, 0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05,
            0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75, 0x09, 0x83,
            0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29,
            0xe3, 0x2f, 0x84, 0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b,
            0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf, 0xd0, 0xef, 0xaa,
            0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c,
            0x9f, 0xa8, 0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc,
            0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2, 0xcd, 0x0c, 0x13, 0xec,
            0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19,
            0x73, 0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee,
            0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb, 0xe0, 0x32, 0x3a, 0x0a, 0x49,
            0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79,
            0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4,
            0xea, 0x65, 0x7a, 0xae, 0x08, 0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6,
            0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a, 0x70,
            0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9,
```


DOI: https://doi.org/1035457/antivirus.v18i2.3369

```
            0x86, 0xc1, 0x1d, 0x9e, 0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e,
            0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf, 0x8c, 0xa1,
            0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0,
            0x54, 0xbb, 0x16]

        # Rijndael Inverted S-box
rsbox = [
            0x52, 0x09, 0x6a, 0xd5, 0x30, 0x36, 0xa5, 0x38, 0xbf, 0x40, 0xa3,
            0x9e, 0x81, 0xf3, 0xd7, 0xfb, 0x7c, 0xe3, 0x39, 0x82, 0x9b, 0x2f,
            0xff, 0x87, 0x34, 0x8e, 0x43, 0x44, 0xc4, 0xde, 0xe9, 0xcb, 0x54,
            0x7b, 0x94, 0x32, 0xa6, 0xc2, 0x23, 0x3d, 0xee, 0x4c, 0x95, 0x0b,
            0x42, 0xfa, 0xc3, 0x4e, 0x08, 0x2e, 0xa1, 0x66, 0x28, 0xd9, 0x24,
            0xb2, 0x76, 0x5b, 0xa2, 0x49, 0x6d, 0x8b, 0xd1, 0x25, 0x72, 0xf8,
            0xf6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xd4, 0xa4, 0x5c, 0xcc, 0x5d,
            0x65, 0xb6, 0x92, 0x6c, 0x70, 0x48, 0x50, 0xfd, 0xed, 0xb9, 0xda,
            0x5e, 0x15, 0x46, 0x57, 0xa7, 0x8d, 0x9d, 0x84, 0x90, 0xd8, 0xab,
            0x00, 0x8c, 0xbc, 0xd3, 0x0a, 0xf7, 0xe4, 0x58, 0x05, 0xb8, 0xb3,
            0x45, 0x06, 0xd0, 0x2c, 0x1e, 0x8f, 0xca, 0x3f, 0x0f, 0x02, 0xc1,
            0xaf, 0xbd, 0x03, 0x01, 0x13, 0x8a, 0x6b, 0x3a, 0x91, 0x11, 0x41,
            0x4f, 0x67, 0xdc, 0xea, 0x97, 0xf2, 0xcf, 0xce, 0xf0, 0xb4, 0xe6,
            0x73, 0x96, 0xac, 0x74, 0x22, 0xe7, 0xad, 0x35, 0x85, 0xe2, 0xf9,
            0x37, 0xe8, 0x1c, 0x75, 0xdf, 0x6e, 0x47, 0xf1, 0x1a, 0x71, 0x1d,
            0x29, 0xc5, 0x89, 0x6f, 0xb7, 0x62, 0x0e, 0xaa, 0x18, 0xbe, 0x1b,
            0xfc, 0x56, 0x3e, 0x4b, 0xc6, 0xd2, 0x79, 0x20, 0x9a, 0xdb, 0xc0,
            0xfe, 0x78, 0xcd, 0x5a, 0xf4, 0x1f, 0xdd, 0xa8, 0x33, 0x88, 0x07,
            0xc7, 0x31, 0xb1, 0x12, 0x10, 0x59, 0x27, 0x80, 0xec, 0x5f, 0x60,
            0x51, 0x7f, 0xa9, 0x19, 0xb5, 0x4a, 0x0d, 0x2d, 0xe5, 0x7a, 0x9f,
            0x93, 0xc9, 0x9c, 0xef, 0xa0, 0xe0, 0x3b, 0x4d, 0xae, 0x2a, 0xf5,
            0xb0, 0xc8, 0xeb, 0xbb, 0x3c, 0x83, 0x53, 0x99, 0x61, 0x17, 0x2b,
            0x04, 0x7e, 0xba, 0x77, 0xd6, 0x26, 0xe1, 0x69, 0x14, 0x63, 0x55,
            0x21, 0x0c, 0x7d]

 rcon = [0x00, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1B, 0x36]

 mul2=[ 0x00,0x02,0x04,0x06,0x08,0x0a,0x0c,0x0e,0x10,0x12,0x14,0x16,0x18,0x1a,0x1c,0x1e,
        0x20,0x22,0x24,0x26,0x28,0x2a,0x2c,0x2e,0x30,0x32,0x34,0x36,0x38,0x3a,0x3c,0x3e,
        0x40,0x42,0x44,0x46,0x48,0x4a,0x4c,0x4e,0x50,0x52,0x54,0x56,0x58,0x5a,0x5c,0x5e,
        0x60,0x62,0x64,0x66,0x68,0x6a,0x6c,0x6e,0x70,0x72,0x74,0x76,0x78,0x7a,0x7c,0x7e,
        0x80,0x82,0x84,0x86,0x88,0x8a,0x8c,0x8e,0x90,0x92,0x94,0x96,0x98,0x9a,0x9c,0x9e,
        0xa0,0xa2,0xa4,0xa6,0xa8,0xaa,0xac,0xae,0xb0,0xb2,0xb4,0xb6,0xb8,0xba,0xbc,0xbe,
        0xc0,0xc2,0xc4,0xc6,0xc8,0xca,0xcc,0xce,0xd0,0xd2,0xd4,0xd6,0xd8,0xda,0xdc,0xde,
        0xe0,0xe2,0xe4,0xe6,0xe8,0xea,0xec,0xee,0xf0,0xf2,0xf4,0xf6,0xf8,0xfa,0xfc,0xfe,
        0x1b,0x19,0x1f,0x1d,0x13,0x11,0x17,0x15,0x0b,0x09,0x0f,0x0d,0x03,0x01,0x07,0x05,
        0x3b,0x39,0x3f,0x3d,0x33,0x31,0x37,0x35,0x2b,0x29,0x2f,0x2d,0x23,0x21,0x27,0x25,
        0x5b,0x59,0x5f,0x5d,0x53,0x51,0x57,0x55,0x4b,0x49,0x4f,0x4d,0x43,0x41,0x47,0x45,
        0x7b,0x79,0x7f,0x7d,0x73,0x71,0x77,0x75,0x6b,0x69,0x6f,0x6d,0x63,0x61,0x67,0x65,
        0x9b,0x99,0x9f,0x9d,0x93,0x91,0x97,0x95,0x8b,0x89,0x8f,0x8d,0x83,0x81,0x87,0x85,
        0xbb,0xb9,0xbf,0xbd,0xb3,0xb1,0xb7,0xb5,0xab,0xa9,0xaf,0xad,0xa3,0xa1,0xa7,0xa5,
        0xdb,0xd9,0xdf,0xdd,0xd3,0xd1,0xd7,0xd5,0xcb,0xc9,0xcf,0xcd,0xc3,0xc1,0xc7,0xc5,
        0xfb,0xf9,0xff,0xfd,0xf3,0xf1,0xf7,0xf5,0xeb,0xe9,0xef,0xed,0xe3,0xe1,0xe7,0xe5]

 mul3=[ 0x00,0x03,0x06,0x05,0x0c,0x0f,0x0a,0x09,0x18,0x1b,0x1e,0x1d,0x14,0x17,0x12,0x11,
        0x30,0x33,0x36,0x35,0x3c,0x3f,0x3a,0x39,0x28,0x2b,0x2e,0x2d,0x24,0x27,0x22,0x21,
        0x60,0x63,0x66,0x65,0x6c,0x6f,0x6a,0x69,0x78,0x7b,0x7e,0x7d,0x74,0x77,0x72,0x71,
        0x50,0x53,0x56,0x55,0x5c,0x5f,0x5a,0x59,0x48,0x4b,0x4e,0x4d,0x44,0x47,0x42,0x41,
        0xc0,0xc3,0xc6,0xc5,0xcc,0xcf,0xca,0xc9,0xd8,0xdb,0xde,0xdd,0xd4,0xd7,0xd2,0xd1,
        0xf0,0xf3,0xf6,0xf5,0xfc,0xff,0xfa,0xf9,0xe8,0xeb,0xee,0xed,0xe4,0xe7,0xe2,0xe1,
        0xa0,0xa3,0xa6,0xa5,0xac,0xaf,0xaa,0xa9,0xb8,0xbb,0xbe,0xbd,0xb4,0xb7,0xb2,0xb1,
        0x90,0x93,0x96,0x95,0x9c,0x9f,0x9a,0x99,0x88,0x8b,0x8e,0x8d,0x84,0x87,0x82,0x81,
        0x9b,0x98,0x9d,0x9e,0x97,0x94,0x91,0x92,0x83,0x80,0x85,0x86,0x8f,0x8c,0x89,0x8a,
        0xab,0xa8,0xad,0xae,0xa7,0xa4,0xa1,0xa2,0xb3,0xb0,0xb5,0xb6,0xbf,0xbc,0xb9,0xba,
        0xfb,0xf8,0xfd,0xfe,0xf7,0xf4,0xf1,0xf2,0xe3,0xe0,0xe5,0xe6,0xef,0xec,0xe9,0xea,
        0xcb,0xc8,0xcd,0xce,0xc7,0xc4,0xc1,0xc2,0xd3,0xd0,0xd5,0xd6,0xdf,0xdc,0xd9,0xda,
        0x5b,0x58,0x5d,0x5e,0x57,0x54,0x51,0x52,0x43,0x40,0x45,0x46,0x4f,0x4c,0x49,0x4a,
        0x6b,0x68,0x6d,0x6e,0x67,0x64,0x61,0x62,0x73,0x70,0x75,0x76,0x7f,0x7c,0x79,0x7a,
        0x3b,0x38,0x3d,0x3e,0x37,0x34,0x31,0x32,0x23,0x20,0x25,0x26,0x2f,0x2c,0x29,0x2a,
        0x0b,0x08,0x0d,0x0e,0x07,0x04,0x01,0x02,0x13,0x10,0x15,0x16,0x1f,0x1c,0x19,0x1a]

 def keyExpansionCore(x,i):
     #one left shift
     t=x[0]
```


DOI: https://doi.org/1035457/antivirus.v18i2.3369

```
     x[0]=x[1]
     x[1]=x[2]
     x[2]=x[3]
     x[3]=t

     x[0]=sbox[x[0]]
     x[1]=sbox[x[1]]
     x[2]=sbox[x[2]]
     x[3]=sbox[x[3]]

def keyExpansion():

def subBytes(state):
    for i in range(16):
        state[i]=sbox[state[i]]

def shiftRows(state):
    tmp[0]=state[0]
    tmp[1]=state[5]
    tmp[2]=state[10]
    tmp[3]=state[15]
    tmp[4]=state[4]
    tmp[5]=state[9]
    tmp[6]=state[14]
    tmp[7]=state[3]
    tmp[8]=state[8]
    tmp[9]=state[13]
    tmp[10]=state[2]
    tmp[11]=state[7]
    tmp[12]=state[12]
    tmp[13]=state[1]
    tmp[14]=state[6]
    tmp[15]=state[11]
    for i in range(16):
        state[i]=tmp[i]

def mixColumns():
    tmp[0] = (mul2[state[0]] ^ mul3[state[1]] ^ state[2] ^ state[3])
    tmp[1] = (state[0] ^ mul2[state[1]] ^ mul3[state[2]] ^ state[3])
    tmp[2] = (state[0] ^ state[1] ^ mul2[state[2]] ^ mul3[state[3]])
    tmp[3] = (mul3[state[0]] ^ state[1] ^ state[2] ^ mul2[state[3]])

    tmp[4] = (mul2[state[4]] ^ mul3[state[5]] ^ state[6] ^ state[7])
    tmp[5] = (state[4] ^ mul2[state[5]] ^ mul3[state[6]] ^ state[7])
    tmp[6] = (state[4] ^ state[5] ^ mul2[state[6]] ^ mul3[state[7]])
    tmp[7] = (mul3[state[4]] ^ state[5] ^ state[6] ^ mul2[state[7]])

    tmp[8] = (mul2[state[8]] ^ mul3[state[9]] ^ state[10] ^ state[11])
    tmp[9] = (state[8] ^ mul2[state[9]] ^ mul3[state[10]] ^ state[11])
    tmp[10] = (state[8] ^ state[9] ^ mul2[state[10]] ^ mul3[state[11]])
    tmp[11] = (mul3[state[8]] ^ state[9] ^ state[10] ^ mul2[state[11]])

    tmp[12] = (mul2[state[12]] ^ mul3[state[13]] ^ state[14] ^ state[15])
    tmp[13] = (state[12] ^ mul2[state[13]] ^ mul3[state[14]] ^ state[15])
    tmp[14] = (state[12] ^ state[13] ^ mul2[state[14]] ^ mul3[state[15]])
    tmp[15] = (mul3[state[12]] ^ state[13] ^ state[14] ^ mul2[state[15]])

    for i in range(16):
        state[i]=tmp[i]

def addRoundKey(state, roundKey):
    for i in range(16):
        state[i]^=roundKey[i]

def encrypt(message, key):
    for i in range(16):
        state[i]=message[i]

    numberOfRounds = 1
    keyExpansion()
    addRoundKey(state,key)
```


DOI: https://doi.org/1035457/antivirus.v18i2.3369

```
    for round in range(numberOfRounds):
        subBytes(state)
        shiftRows(state)
        mixColumns()
        addRoundKey(state,key)
    #Final round
    subBytes(state)
    shiftRows(state)
    addRoundKey(state, key)

def main():
    message=input("Enter a message: ")
    key=[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16]
    encrypt(message,key)

if __name__ == "__main__":
    main()
```

Gambar 5. Enkripsi dan Deskripsi AES

Telah menghasilkan enkripsi dan *ciphertext* dengan hasil deskripsi *plaintext* "data ini akan kita enkripsi" kembali ke semula.

B. Enkripsi dan Dekripsi RSA

Dalam menentukan bilangan prima dari p dan q memperoleh nilai n dengan rumus : $n = p * q$ sehingga memperoleh nilai m dengan rumus : $m = (p-1) * (q-1)$. Syarat untuk memperoleh nilai e, $e = e > 1\ and\ GCD(m,e) = 1$ harus dapat memenuhi syarat nilai e, $e = GCD(120{,}17) = 1$. Syarat untuk menentukan nilai d dengan : $d = (d * e)\ mod\ m = 1$ harus dapat memenuhi nilai d, $d = (473 * 17)\ mod\ 120 = 1$. Sehingga diperoleh kunci public = (e,n) | (17,143) dan kunci rahasia = (d,n) | (473,143).

Selanjutnya dilakukan proses Enkripsi dan Deskripsi pada kata "BIRU":

Tabel 1. Proses Enkripsi dan Deskripsi

Text	ASCII (A)	Proses Enkripsi (X) $C = A \wedge e\ mod\ n$	Proses Deskripsi (Y) $Y = C \wedge d\ mod\ n$
B	66	$= (6 \wedge 17)\ mod\ 143 = 41$ $= (6 \wedge 17)\ mod\ 143 = 41$ = 41.41	$= (41 \wedge 473)\ mod\ 143 = 6$ $= (41 \wedge 473)\ mod\ 143 = 6$ = 66 → B
I	73	$= (7 \wedge 17)\ mod\ 143 = 50$ $= (3 \wedge 17)\ mod\ 143 = 9$ = 50.9	$= (50 \wedge 473)\ mod\ 143 = 7$ $= (9 \wedge 473)\ mod\ 143 = 3$ = 73 → I
R	82	$= (8 \wedge 17)\ mod\ 143 = 112$ $= (2 \wedge 17)\ mod\ 143 = 84$ = 41.112	$= (112 \wedge 473)\ mod\ 143 = 8$ $= (84 \wedge 473)\ mod\ 143 = 2$ = 82 → R
U	85	$= (8 \wedge 17)\ mod\ 143 = 112$ $= (5 \wedge 17)\ mod\ 143 = 135$ = 112.135	$= (112 \wedge 473)\ mod\ 143 = 8$ $= (135 \wedge 473)\ mod\ 143 = 5$ = 85 → U

DOI: https://doi.org/1035457/antivirus.v18i2.3369

C. Implementasi Kombinasi

Untuk membuktikan kinerja gabungan dari algoritma AES dan RSA, maka dibuatlah suatu program dari bahasa pemrograman python untuk penerapan algoritma tersebut sebagaimana pada Gambar 5.

```
import secrets
import random
import sys
from Crypto.Cipher import AES
from Crypto import Random

def gcd(a, b):
        '''Euclid's algorithm '''
        while b != 0:
            temp=a % b
            a=b
            b=temp
        return a

def multiplicativeInverse(a, b):
        """Euclid's extended algorithm"""
        x = 0
        y = 1
        lx = 1
        ly = 0
        oa = a
        ob = b
        while b != 0:
            q = a // b
            (a, b) = (b, a % b)
            (x, lx) = ((lx - (q * x)), x)
            (y, ly) = ((ly - (q * y)), y)
        if lx < 0:
            lx += ob
        if ly < 0:
            ly += oa
        return lx

def generatePrime(keysize):
    while True:
        num = random.randrange(2**(keysize-1), 2**(keysize))
        if isPrime(num):
            return num

def isPrime(num):
    if (num < 2):
        return False # 0, 1, and negative numbers are not prime
    lowPrimes = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71,
73, 79, 83, 89,
                 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167,
173, 179, 181, 191,
                 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271,
277, 281, 283, 293,
                 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389,
397, 401, 409, 419,
                 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503,
509, 521, 523, 541,
                 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631,
641, 643, 647, 653,
                 659, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 743, 751, 757,
761, 769, 773, 787,
                 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863, 877, 881, 883,
887, 907, 911, 919,
                 929, 937, 941, 947, 953, 967, 971, 977, 983, 991, 997]

    if num in lowPrimes:
        return True
```


DOI: https://doi.org/1035457/antivirus.v18i2.3369

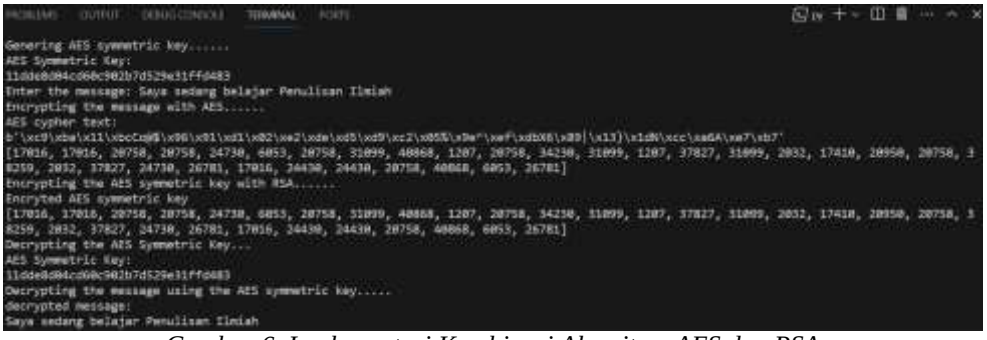
```
    for prime in lowPrimes:
        if (num % prime == 0):
            return False

    return millerRabin(num)

def millerRabin(n, k = 7):
    if n < 6:
        return [False, False, True, True, False, True][n]
    elif n & 1 == 0:
        return False
    else:
      s, d = 0, n - 1
      while d & 1 == 0:
         s, d = s + 1, d >> 1
      for a in random.sample(range(2, min(n - 2, sys.maxsize)), min(n - 4, k)):
         x = pow(a, d, n)
         if x != 1 and x + 1 != n:
            for r in range(1, s):
               x = pow(x, 2, n)
               if x == 1:
                  return False
               elif x == n - 1:
                  a = 0
                  break
            if a:
               return False
      return True

def KeyGeneration(size=8):

    #1)Generate 2 large random primes p,q (same size)
    p=generatePrime(size)
    q=generatePrime(size)
    if not (isPrime(p) and isPrime(q)):
        raise ValueError('Both numbers must be prime.')
    elif p == q:
        raise ValueError('p and q cannot be equal')
    #2)compute n=pq and phi=(p-1)(q-1)
    n = p * q
    phi = (p-1) * (q-1)

    #3) select random integer "e" (1<e<phi) such that gcd(e,phi)=1
    e = random.randrange(1, phi)
    g = gcd(e, phi)
    while g != 1:
        e = random.randrange(1, phi)
        g = gcd(e, phi)

    #4)Use Extended Euclid's Algorithm to compute another unique integer "d" (1<d<phi) such
that e.d≡1(mod phi)
    d = multiplicativeInverse(e, phi)

    #5)Return public and private keys
    #Public key is (e, n) and private key is (d, n)
    return ((n, e), (d, n))

def encrypt(pk, plaintext):
    #1) obtain (n,e)
    n, e = pk
    #2)message space [0,n-1]
    #3)compute c=m^e(mod n)
    c = [(ord(char) ** e) % n for char in plaintext]
    print(c)
    #4) send "C" to the other party
    return c

def decrypt(pk, ciphertext):
    d, n = pk
    #5)m=c^d (mod n)
```


DOI: https://doi.org/1035457/antivirus.v18i2.3369

```
    m = [chr((char ** d) % n) for char in ciphertext]
    return m

def encryptAES(cipherAESe,plainText):
    return cipherAESe.encrypt(plainText.encode("utf-8"))

def decryptAES(cipherAESd,cipherText):
    dec= cipherAESd.decrypt(cipherText).decode('utf-8')
    return dec

def main():
    #To encrypt a message addressed to Alice in a hybrid crypto-system, Bob does the
following:
    print("**********************************************************************")
    print("**********************************************************************")
    print("Welcome to the hybrid cryptographic scheme demostration...")
    print("We're going to encrypt and decrypt a message using AES and RSA")
    print("**********************************************************************")
    print("**********************************************************************")
    #To encrypt a message addressed to Alice in a hybrid crypto-system, Bob does the
following:
    #1. Obtains Alice's public key.
    print("Genering RSA public and Privite keys......")
    pub,pri=KeyGeneration()

    #2. Generates a fresh symmetric key for the data encapsulation scheme.
    print("Genering AES symmetric key......")
    key = secrets.token_hex(16)
    print("AES Symmetric Key: ")
    print(key)
    KeyAES=key.encode('utf-8')

    #3. Encrypts the message under the data encapsulation scheme, using the symmetric key
just generated.
    plainText = input("Enter the message: ")
    cipherAESe = AES.new(KeyAES,AES.MODE_GCM)
    nonce = cipherAESe.nonce
    print("Encrypting the message with AES......")
    cipherText=encryptAES(cipherAESe,plainText)
    print("AES cypher text: ")
    print(cipherText)

    #4. Encrypt the symmetric key under the key encapsulation scheme, using Alice's public
key.
    cipherKey=encrypt(pub,key)
    print("Encrypting the AES symmetric key with RSA......")
    print("Encryted AES symmetric key")
    print(cipherKey)
    #5. Send both of these encryptions to Alice.
    #Sending.........

    #To decrypt this hybrid cipher-text, Alice does the following:

    #1. Uses her private key to decrypt the symmetric key contained in the key encapsulation
segment.
    decriptedKey=''.join(decrypt(pri,cipherKey))
    print("Decrypting the AES Symmetric Key...")
    print("AES Symmetric Key:")
    print(decriptedKey)

    #2. Uses this symmetric key to decrypt the message contained in the data encapsulation
segment.
    decriptedKey=decriptedKey.encode('utf-8')
    cipherAESd = AES.new(decriptedKey, AES.MODE_GCM, nonce=nonce)
    decrypted=decryptAES(cipherAESd,cipherText)
    print("Decrypting the message using the AES symmetric key.....")
    print("decrypted message: ")
    print(decrypted)

    input('Press ENTER to exit')
```


DOI: https://doi.org/1035457/antivirus.v18i2.3369

```
if __name__ == "__main__":
    main()
```



Gambar 6. Implementasi Kombinasi Algoritma AES dan RSA

Telah dibuat kunci publik dan pribadi dan kunci tanda tangan. kunci simestrisnya yang (AES *Symetric*). memasukan pesan "Saya sedang belajar Penulisan Ilmiah" setelah diketik, dapat melihat *chipertext* dan mengenkripsi kunci simetris. Kunci simetris terenkripsi dengan baik, selanjutnya mendeskripsi kunci simetris sehingga mendapatkan kunci asli. Kunci deskripsi nya akan sama dengan kunci asli. Kemudian deskripsi menghasilkan pesan yang dimasukkan sebelumnya. Sehingga dari semua pengujian kombinasi algoritma AES dan RSA menghasilkan hasil yang lebih baik karena memiliki tipe enkripsi yang berbeda yaitu simetris dan asimetris, yang dapat digunakan untuk mengenkripsi dan mendekripsi data dengan aman.

D. Pengujian Data

Sebelum dilakukan pengujian data, dilakukan dahulu pengujian terhadap pembangkitan kunci privat dan kunci publik.

Pembangkitan kunci menggunakan empat jenis bit yaitu 512, 1024, 2048, dan 4096 bit. Hasil pengujian pembangkitan kunci ditunjukkan pada Tabel 3 Hasil Pengujian Pembangkitan Kunci.

Tabel 2. Hasil Pengujian Pembangkitan Kunci

Bit	Waktu Proses (ms)	Hasil
512	110,28	Menghasilkan kunci publik sebanyak 114 karakter dan kunci privat 398 karakter
1024	189,286	Menghasilkan kunci publik sebanyak 224 karakter dan kunci privat 726 karakter
2048	480,28	Menghasilkan kunci publik sebanyak 299 karakter dan kunci privat 1497 karakter
4096	17861,561	Menghasilkan kunci publik sebanyak 677 karakter dan kunci privat 3103 karakter

Berdasarkan hasil pengujian diperoleh perbedaan yang cukup jauh antara 2048 bit dengan 4096 bit yaitu 480,28 ms dan 17861,561 ms sehingga jenis bit yang akan digunakan dalam proses ini adalah 2048 bit.

Setelah dilakukan pengujian pembangkitan kunci, dilakukan pengujian data yang akan digunakan. Hasil pengujian data ditunjukkan pada tabel 3. dan 4 sebagai berikut.

Tabel 3. Hasil Pengujian Enkripsi Data

Data ke-	Waktu Enkripsi RSA (ms)	Waktu Enkripsi AES (ms)	Hasil
1	4,678	9,881	Text yang diterima ‘Saya belajar penulisan ilmiah’
2	4,765	1,98	Text yang diterima ‘Belajar penulisan ilmiah’
3	5,87	1,77	Text yang diterima ‘Penulisan’
Rata - rata	5,104	4,543	

DOI: https://doi.org/1035457/antivirus.v18i2.3369

Berdasarkan hasil pengujian, proses enkripsi secret key dengan algoritma RSA dan proses enkripsi plaintext menggunakan algoritma AES. Rata-rata waktu yang digunakan pada proses enkripsi RSA enkripsi yaitu 5,104 ms dan 4,543 ms pada proses enkripsi AES.

Tabel 4. Hasil Pengujian Dekripsi Data

Data ke-	Waktu Dekripsi RSA (ms)	Waktu Dekripsi AES (ms)	Hasil
1	48,276	2,996	Text yang diterima 'Saya belajar penulisan ilmiah'
2	59,254	3,88	Text yang diterima 'Belajar penulisan ilmiah'
3	70,775	1,9	Text yang diterima 'Penulisan'
Rata - rata	59,435	2,925	

Berdasarkan hasil pengujian, rata-rata waktu yang digunakan pada proses dekripsi RSA yaitu 59,435 ms, lebih lambat dibandingkan dengan proses enkripsi. Sedangkan pada proses dekripsi AES lebih cepat dibandingkan proses enkripsi yaitu 2,925 ms.

V. KESIMPULAN DAN SARAN

Algoritma RSA umumnya digunakan untuk melindungi kunci simetris, sementara AES digunakan untuk mengamankan isi pesan. Kunci teks, kunci rahasia terenkripsi, dan kunci pribadi akan diberikan kepada pengirim. Pertama, penerima melakukan dekripsi kunci dengan algoritma RSA, selanjutnya algoritma AES untuk mendekripsi *ciphertext* menggunakan kunci rahasia. Proses ini diulang, dengan penerima sekali lagi mendekripsi kunci rahasia dengan algoritma RSA sebelum melakukan dekripsi *ciphertext* dengan menggunakan algoritma AES serta kunci rahasia. Kombinasi algoritma AES dan RSA menghasilkan hasil yang lebih baik karena memiliki tipe enkripsi yang berbeda yaitu simetris dan asimetris, yang dapat digunakan untuk mengenkripsi dan mendekripsi data dengan aman.

Studi menunjukkan bahwa mengenkripsi dan mendekripsi data dengan aman dengan kombinasi tipe enkripsi simetris dan asimetris AES dan RSA. Ini karena kunci pribadi yang digunakan untuk mendapatkan data dalam pesan rahasia dibutuhkan dua kali untuk proses dekripsi. Pembangkitan kunci 2048 bit yang ideal digunakan membutuhkan waktu lebih lama daripada enkripsi; RSA membutuhkan waktu rata-rata 5,104 ms dan AES membutuhkan waktu rata-rata 4,543 ms, sedangkan dekripsi menggunakan RSA membutuhkan waktu rata-rata 59,435 ms dan AES membutuhkan waktu rata-rata 2,925 ms.

Penelitian ini memerlukan penelitian lebih lanjut untuk memperbaikinya. Keamanan dengan menggunakan kombinasi algoritma yang lebih banyak daripada menggunakan AES dan RSA secara bersamaan adalah salah satu alternatif yang dapat dicoba. Selain itu, penelitian ini dapat diperluas untuk mencakup pengamanan media serta berformat lainnya. Oleh karena itu, peningkatan fitur ini dapat meningkatkan keamanan dan fleksibilitas aplikasi kriptografi yang sedang dikembangkan.

DAFTAR PUSTAKA

[1] I. Juniarmi, "Analisis Keamanan Data pada Aplikasi Chatting Menggunakan Enkripsi End-to-End," *Technol. J. J. Inform.*, vol. 1, no. 2, pp. 3046–9163, 2024, [Online]. Available: https://doi.org/10.62872/ppr42775

[2] A. Tumanggor, H. Rumapea, A. Silalahi, and H. Artikel, "Implementasi Algoritma Advance Encryption Standard (AES) Pada Keamanan Dokumen Keuangan (Studi Kasus : CV.Multikreasi Bersama)," *Methotika J. Ilm. Tek. Inform.*, vol. 3, no. 1, pp. 83–90, 2023, [Online]. Available: http://ojs.fikom-methodist.net/index.php/methotika

[3] D. Alfiani Fauzan and A. Fathurrozi, "Perancangan Aplikasi Pengamanan Data Menggunakan Algoritma RSA (Rivest Shamir Adleman) dan AES (Advanced Encryption Standard) Berbasis Web," *J. Inf. Inf. Secur.*, vol. 4, no. 1, pp. 2722–4058, 2023, [Online]. Available: http://ejurnal.ubharajaya.ac.id/index.php/jiforty

[4] F. Bibiola, T. U. Kalsum, and H. Alamsyah, "Penerapan Algoritma Advance Encryption

DOI: https://doi.org/1035457/antivirus.v18i2.3369

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.

[5] K. Nathassa, I. Saputra, and ..., "Implementasi Teknik Enkripsi Q-Chiper Pada Keamanan Data Transkrip Nilai Siswa," *KOMIK (Konferensi ...*, vol. 6, no. November, pp. 689–695, 2023, doi: 10.30865/komik.v6i1.5777.

[6] M. Yudistira and Ramadhan, "Tinjauan Yuridis Terhadap Efektivitas Penanganan Kejahatan Siber Terkait Pencurian Data Pribadi Menurut Undang-Undang No. 27 Tahun 2022 Oleh Kominfo," *Unes Law Rev.*, vol. 5, no. 4, pp. 3802–3815, 2023, [Online]. Available: https://doi.org/10.31933/unesrev.v5i4

[7] S. F. R. Salsabila, Asep Id Hadiana, and Fajri Rakhmat Umbara, "Penerapan Kriptografi Advanced Encryption Standard (AES) dan Steganografi Spread Spectrum Untuk Mengamankan Pesan Dalam Gambar," *J. Informatics Commun. Technol.*, vol. 5, no. 2, pp. 196–209, 2023, [Online]. Available: https://ejournal.akademitelkom.ac.id/j_ict/index.php/j_ict/article/view/216

[8] T. T. Trisnawati, S. Yurinanda, W. Syafmen, and C. Multahadah, "Penerapan Algoritma Rivest-Shamir-Adleman (RSA) pada Enkripsi Uniform Resource Locator (URL) Website untuk Keamanan Data," *Euler J. Ilm. Mat. Sains dan Teknol.*, vol. 11, no. 2, pp. 205–215, 2023, doi: 10.37905/euler.v11i2.21169.

[9] F. Diny Hermawati and M. Tahir, "Keamanan E-Voting Di Indonesia Melalui Pemanfaatan Kriptografi Pada Sistem AES (Advance Encryption Standard)," *Jaya Abadi Amroin*, vol. 2, no. 2, pp. 45–56, 2023.

[10] S. Oktaviani, F. Rizky, and I. Gunawan, "Analisis Keamanan Data Dengan Menggunakan Kriptografi Modern Algoritma Advance Encryption Standar (AES)," *J. Media Inform.*, vol. 4, no. 2, pp. 97–101, 2023, doi: 10.55338/jumin.v4i2.435.

[11] W. Putra, M. R. Fahlevi, and A. T. Hidayat, "Implementasi Algoritma Advanced Encryption Standard Untuk Kemanan Dokumen," *J. Ilmu Komputer, Teknol. Dan Inf.*, vol. 1, no. 2, pp. 76–83, 2023, doi: 10.62866/jurikti.v1i2.55.

[12] A. T. Pandya and J. C. Chandra, "Aplikasi Keamanan File Menggunakan Algoritme Kriptografi Aes 128 Berbasis Web Pada Pilar Medical Center," *Semin. Nas. Mhs. Fak. Teknol. Inf.*, vol. 2, no. 2, pp. 36–45, 2023.

[13] J. S. Sianipar, N. B. Nuugroho, and I. Mariami, "Pengamanan Data Gaji Karyawan Dengan Menggunakan Metode Advanced Encryption Standard (AES)," *J. Sist. Inf. Tgd*, vol. 3, no. 1, pp. 35–45, 2024, [Online]. Available: https://doi.org/10.53513/jursi.v3i1.5653

[14] R. H. Sukarna, A. M. Januriana, . H., and M. Hilman, "Implementasi Algoritma Enkripsi Pada Sistem Informasi Manajemen Helpdesk Berbasis Desktop," *J. Inform. dan Ris.*, vol. 1, no. 2, pp. 31–39, 2023, doi: 10.36308/iris.v1i2.535.

[15] D. Indra, G. Hts, M. Ridho Aldizar, and G. Hutasuhut, "Perbandingan Algoritma Kriptografi Simetris Dan Asimetris," *UNESJournal Inf. Syst.*, vol. 8, no. 1, pp. 42–47, 2023, [Online]. Available: https://fe.ekasakti.org/index.php/UJIS

[16] Z. D. Ladunniyyah *et al.*, "Atas Ring Dedekind Untuk Mengamankan Pesan," 2023.

[17] A. Eko and S. Informatika, "Keamanan Pesan Teks Dengan Metode Enkripsi Dan Dekripsi Menggunakan Algoritma Rsa (Rivest Shamir Adleman) Berbasis Android," *Teknologipintar.org*, vol. 3, no. 2, p. 1, 2023.

DOI: https://doi.org/1035457/antivirus.v18i2.3369