PERBANDINGAN DAN ANALISIS PERFORMANSI ENKRIPSI-DEKRIPSI TEKS MENGGUNAKAN ALGORITMA AES DAN AES YANG TERMODIFIKASI BERBASIS ANDROID

COMPARATION AND ANALYSIS OF PERFORMANCE OF ENCRYPTION-DECRYPTION OF TEXT USING AES ALGORITHM AND MODIFIED AES BASED ON ANDROID

FADHLAN PUTRA[1] GELAR BUDIMAN[2] NUR ANDINI[3]

[1, 2, 3]Prodi S1 Teknik Telekomunikasi, Fakultas Teknik Elektro, Universitas Telkom

[1]fadlanp0@gmail.com [2]gelarbudiman@telkomuniversity.ac.id [3]nurandini@telkomuniversity.ac.id

Abstrak

Pada tugas akhir ini menggunakan dua algoritma yakni algoritma Rijndael atau AES *(Andvanced Encryption Standard)* dan AES termodifikasi. Penggunaan algoritma AES termodifikasi ini dimaksudkan untuk mengetahui kapabilitas performansi dari algortima tersebut. Apakah lebih baik dari algoritma AES original, mengingat algoritma AES telah digunakan lebih dari sepuluh tahun. Hasil yang diperoleh dari pengujian yang dilakukan pada tugas akhir ini, performansi dari sistem yang menggunakan algoritma AES hampir sama dengan performansi dari sistem yang menggunakan algoritma AES termodifikasi. Kedua sistem memiliki nilai *avalanche effect* yang hampir sama yakni berkisar 0.5 dan durasi waktu yang dibutuhkan untuk melakukan *brute force attack* yakni 2.6×10^{21} tahun. Namun berbeda waktu komputasi dan performansi *robustness* pada kedua sistem. Waktu komputasi pada sistem yang menggunakan algoritma AES lebih cepat dibandingkan sistem yang menggunakan algoritma AES termodifikasi.

Kata kunci: Android, enkripsi, dekripsi, AES.

Abstract

In this final project use two algorithms namely Rijndael or AES algorithm (Andvanced Encryption Standard) and modified AES. The use of the modified AES algorithm is intended to determine the performance capability of these algorithms. Is it better than the AES algorithm itself, given the AES algorithm has been used for more than ten years. The results obtained from tests performed in this final performance of the system using the AES algorithm is similar to the performance of a system that uses the AES algorithm modified. Both systems have values similar avalanche effect which ranges from 0.5 and the duration of time required to perform a brute force attack that is 2.6×10^{21} years. However, different computation time and robustness performance on both systems. Computing time on a system that uses the AES algorithm is faster than systems using the AES algorithm modified.

Keywords: Android, encryption, decryption, AES.

1. Pendahuluan

Untuk menjamin keamanan pertukaran informasi data dapat diatasi dengan implementasi kriptografi atau pengacakan terhadap informasi yang terdapat pada pesan. Sehingga pesan tersebut tidak dapat dipahami oleh pihak lain. Proses enkripsi-dekripsi memerlukan algoritma untuk mengatur pola pengacakan suatu informasi yang terdapat pada pesan. Ada banyak algoritma yang digunakan pada proses enkripsi-dekripsi, salah satunya algoritma Rijndael yang ditetapkan sebagai algoritma *Advanced Encrytpion Standard (*AES*)* oleh *National Institute of Standards and Technology* (NIST) pada awal abad dua satu. Walaupun sudah lebih dari sepuluh tahun algoritma tersebut digunakan untuk proses enkripsi-dekripsi, untuk membobol suatu pesan yang terenkripsi menggunakan algoritma AES masih sulit dilakukan. Algoritma AES telah banyak diimplementasikan di penelitian-penelitian sebelumnya ke dalam sistem keamanan informasi. Implementasi dapat berupa sistem untuk enkripsi informasi di berbagai *platform.* Adapun penelitian-penelitian sebelumnya antara lain Aplikasi Enkripsi SMS pada Handphone Berbasis Java dengan Menggunakan Algoritma Rijndael oleh Anisa J. Rahmah [10]**,** Analisis Perbandingan Performansi Sistem Keamanan pada Android Menggunakan Algoritma AES dan Twofish oleh Taufik Ardi Susilo[14], Analisa dan Implementasi Enkripsi-Dekripsi Suara Menggunakan Algoritma Rijndael oleh Iskan Susanto[13], Secure Chatting (Instant Messaging) Menggunakan Metode Enkripsi Blowfish, Twofish, dan AES oleh Angga Kusumah[5]. Namun pada tahun 2006, serangan terbaik terhadap algoritma *Rijndael* berhasil menembus putaran ke-7 untuk panjang kunci 128 bit, putaran ke-8 untuk panjang kunci 192 bit dan putaran ke-9 untuk kunci 256 bit[12].

Semakin berkembangnya teknologi komputer, proses komputasi yang dilakukan komputer bahkan super-komputer, mengalami peningkatan pesat. Maka dari itu hanya masalah waktu saja, sampai algoritma AES dapat dibobol utuh. Sehingga hal ini yang mendasari lahirnya algoritma AES yang termodifikasi. Ada banyak teknik modifikasi pada algoritma AES yang diusulkan, salah satunya memodifikasi tiga transformasi. Namun teknik modifikasi yang digunakan menghasilkan nilai *avalanche effect* yang kecil. Sehingga pada tugas akhir ini, akan dibandingkan performansi dari implementasi algoritma AES dan AES termodifikasi yang diusulkan untuk proses enkripsi-dekripsi pesan teks pada *smartphone* Android.

Berdasarkan latar belakang tersebut, maka dapat dirumuskan suatu masalah yakni bagaimana mengamankan pesan teks dengan mengimplementasikan algoritma AES dan AES yang termodifikasi pada *smartphone* Android. Kemudian bagaimana hasil performansi dari perbandingan kedua algoritma yang digunakan dan mana yang lebih baik performansinya dengan berdasarkan parameter yang ditentukan. Bagaimana mengoptimalkan performansi dari algoritma AES termodifikasi refensi jika dilakukan pengujian *avalanche effect.*

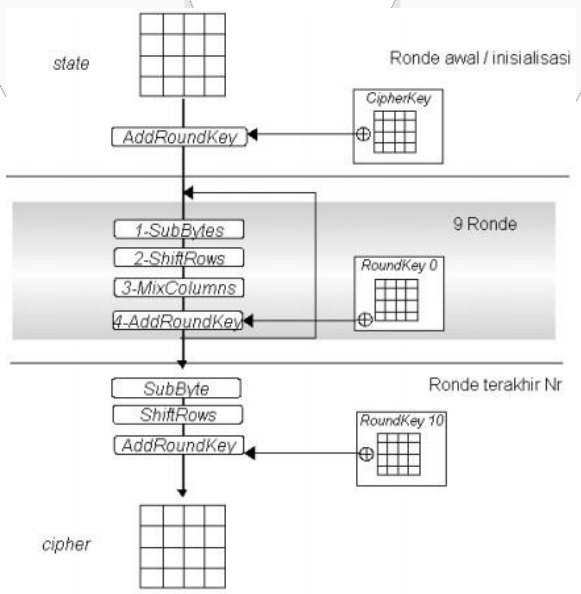
Tujuan dari penelitian tugas akhir ini yakni untuk mengimplementasikan algoritma AES dan AES yang termodifikasi untuk proses enkripsi-dekripsi pesan teks pada *smartphone* Android. Kemudian untuk menganalisis dan memberikan rekomendasi terhadap algoritma yang akan digunakan ke depannya dengan mempertimbangkan hasil perbandingan yang didapat. Melakukan optimasi terhadap algoritma AES termodifikasi referensi untuk menghasilkan nilai *avalanche effect* yang lebih tinggi dibandingkan algoritma AES termodifikasi referensi.

Pengerjaan tugas akhir ini dilakukan dengan menggunakan beberapa metode yakni mempelajari beberapa referensi buku dan situs internet yang berkaitan dengan topik yang dibahas pada tugas akhir ini. Kemudian hasil analisis data-data yang diperoleh dari hasil studi literatur didiskusikan dengan pembimbing guna memecahkan masalah yang sulit dihadapi. Melakukan implementasi terhadap algoritma AES dan AES termodifikasi untuk proses enkripsi-dekripsi pesan teks pada aplikasi Android. Melakukan pengujian terhadap hasil yang didapatkan sehingga sesuai dengan harapan serta melakukan analisis dari aplikasi yang dibuat. Melakukan pengujian sistem dan analisis terhadap hasil uji penelitian yang dilakukan. Pengambilan kesimpulan terhadap hasil analisis dan pembuatan laporan Tugas akhir dari seluruh kegiatan penelitian.

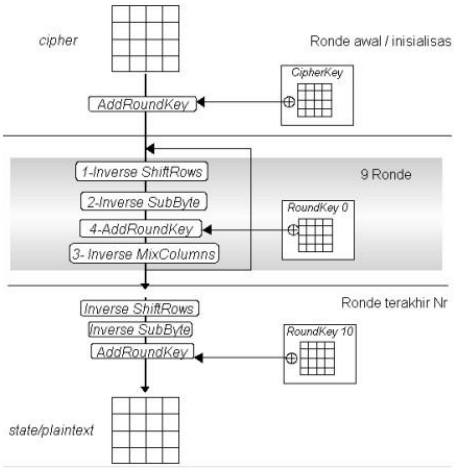
2. Dasar Teori

2.1 Algoritma AES

Algoritma Rijndael ditetapkan oleh *National Institute of Standards and Technology* (NIST) sebagai algoritma *Advanced Encryption Standard* (AES) pada bulan November tahun 2001 [6]. Algoritma AES bersifat simetris dan *cipher block*, dimana penggunaan kunci untuk enkripsi dan dekripsi sama dan membutuhkan waktu yang lebih singkat untuk menjalankan proses enkripsi dan dekripsi [2]. Karena bersifat *cipher block* sehingga *plaintext* diubah menjadi bilangan *byte* terlebih dahulu kemudian disusun menjadi matriks berukuran 4×4 yang disebut *state*. Rijndael mendukung berbagai variasi ukuran panjang kunci yakni 128, 192 dan 256 bit [9]. Proses yang dilakukan pada setiap rondenya identik sama (dari ronde ke-1 sampai dengan ronde ke Nr-1) kecuali untuk ronde Nr. Proses yang identik tersebut terdiri atas *SubBytes, ShiftRows, MixColumns dan AddRoundKey* [15]. Sedangkan pada ronde Nr, proses *MixColumns* tidak dilakukan. Tiap ronde memilki *roundkey* yang dihasilkan dari ekspansi dari kunci utama [1].



Gambar 1 Proses Enkripsi AES-128 [15]

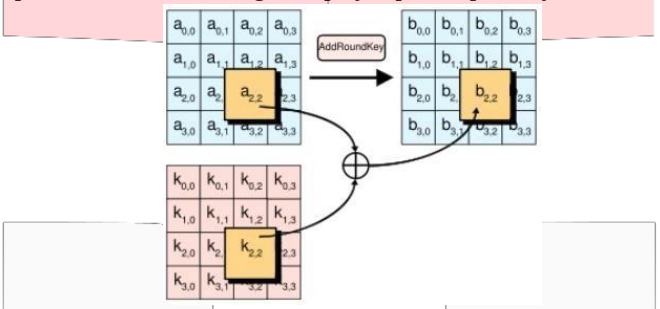


Gambar 2 Proses Dekripsi AES-128 [15]

Proses yang dilakukan menggunakan algoritma AES yakni:

1. *AddRoundKey*

   Dalam tahap *AddRoundKey*, setiap byte dari matriks *state* dilakukan operasi XOR dengan setiap byte dari *roundkey* pada *state* keluaran proses *MixColumns*. Proses ini juga dilakukan di awal ronde dengan melakukan operasi XOR tiap *byte* pada matriks *state* dengan tiap *byte* pada *cipherkey* [15].

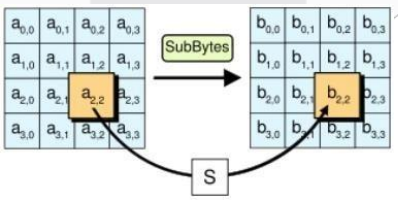


Gambar 3 Proses AddRoundKey [15]

Inverse atau kebalikan dari tahap *AddRoundKey* adalah operasi XOR antara byte-byte matriks *state* yang disusun dari *ciphertext* dengan byte-byte *roundkey* yang dibangkitkan sebelumnya. *Roundkey* yang digunakan di setiap iterasinya berkebalikan dengan *roundkey* yang ada pada proses enkripsi. *Inverse* dari transformasi ini digunakan untuk proses dekripsi[3].

2. *SubBytes*

   Setiap *byte-byte* dalam matriks *state* disubtitusikan dengan *byte-byte* yang terdapat pada tabel S-box. S-box dirancang dengan rumusan matematika agar menghilangkan kecurigaan akan ditanamnya *backdoor* pada tabel S-box [12]. S-box diturunkan dari fungsi invers dalam GF(2^8), yang diketahui mempunyai sifat non-linieritas yang bagus. Operasi ini akan memberikan prinsip non-linieritas pada *ciphertext* [15]. Subtitusi dilakukan bertujuan untuk mengaburkan hubungan antara *plaintext* dengan *ciphertext* (dalam hal ini disebut *confussion*) [3].

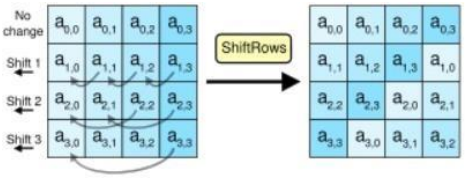


Gambar 4 Proses SubBytes [15]

Sedangkan pada *inverse SubBytes* menggunakan tabel *inverse* S-Box untuk melakukan subtitusi pada matriks. *Inverse* ini digunakan untuk proses dekripsi *ciphertext.* Cara subtitusi yang dilakukan untuk dekripsi *ciphertext* sama dengan saat *plaintext* dienkripsi.

3. *ShiftRows*

   Pada transformasi *ShiftRows*, baris-baris pada matriks *state* digeser ke kiri secara berputar. Namun jumlah pergeseran yang dilakukan berbeda, tergantung untuk setiap barisnya. Baris pertama tidak terjadi pergeseran. Setiap *byte* dari baris kedua pada matriks *state* digeser satu *byte* ke kiri. Selanjutnya baris ketiga dan keempat masing-masing digeser ke kiri sebanyak dua dan tiga byte [15]. Proses ini bertujuan untuk menghasilkan *diffusion* yakni dengan menyebarkan pengaruh transformasi nonlinear pada baris-baris matriks *state* untuk putaran selanjutnya [11].



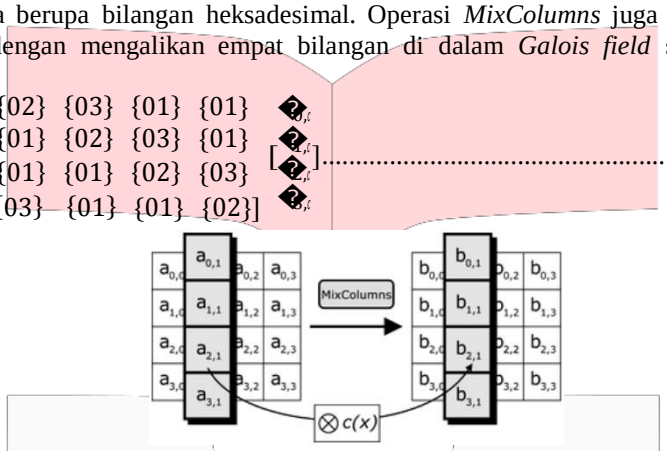
Gambar 5 Proses ShiftRows [15]

Untuk proses dekripsinya dilakukan proses *Inverse* dari transformasi *ShiftRows*. Pada proses ini, byte-byte pada tiga baris terakhir diputar dengan jumlah putaran yang sama saat proses enkripsi, hanya saja dengan arah kebalikannya yakni arah ke kanan.

4. *MixColumns*

Pada proses *MixColumns*, tiap kolom dari matriks *state* dilakukan operasi perkalian. Hal ini bertujuan untuk menyebarkan pengaruh setiap bit *plaintext* dan *cipherkey* terhadap *ciphertext* yang dihasilkan, pada arah kolom matriks *state* [3]. Setiap kolom matriks *state* diperlakukan sebagai polinomial empat suku dalam *Galois field*. Kemudian dikalikan dengan a(x) modulo (x^4+1), dimana a(x)= {03}x^3+{01}x^2+{01}x+{02} dengan konstantanya berupa bilangan heksadesimal. Operasi *MixColumns* juga dapat dipandang sebagai perkalian matriks, dengan mengalikan empat bilangan di dalam *Galois field* seperti yang ditunjukkan berikut[12].

$$\begin{bmatrix} \square_{0,c} \\ \square_{1,c} \\ \square_{2,c} \\ \square_{3,c} \end{bmatrix} = \begin{matrix} \{02\} & \{03\} & \{01\} & \{01\} \\ \{01\} & \{02\} & \{03\} & \{01\} \\ \{01\} & \{01\} & \{02\} & \{03\} \\ \{03\} & \{01\} & \{01\} & \{02\} \end{matrix} \begin{bmatrix} \square_{0,c} \\ \square_{1,c} \\ \square_{2,c} \\ \square_{3,c} \end{bmatrix} \quad \text{.....(2.1)}$$



Gambar 6 Proses MixColumns [12]

Untuk melakukan dekripsi pesan, dilakukan *inverse* dari transformasi *MixColumns* yakni mengalikan setiap kolom pada matriks *state* dengan polinom b(x) mod (x^4+1). Setiap kolom diperlakukan sebagai polinom empat suku pada GF (2^8). Sedangkan, polinom b(x) yang ditetapkan yakni b(x)= {0B}x^3+{0D}x^2+{09}x+{0E}, dengan tiap konstantanya merupakan bilangan heksadesimal [3]. *Inverse MixColumns* dapat dipandang sebagai perkalian matriks seperti ditujukan pada persamaan berikut:

$$\begin{bmatrix} \square_{0,c} \\ \square_{1,c} \\ \square_{2,c} \\ \square_{3,c} \end{bmatrix} = \begin{matrix} \{0\square\} & \{0\square\} & \{0\square\} & \{09\} \\ \{09\} & \{0\square\} & \{0\square\} & \{0\square\} \\ \{0\square\} & \{09\} & \{0\square\} & \{0\square\} \end{matrix} \begin{bmatrix} \square_{0,c} \\ \square_{1,c} \\ \square_{2,c} \\ \square_{3,c} \end{bmatrix}$$

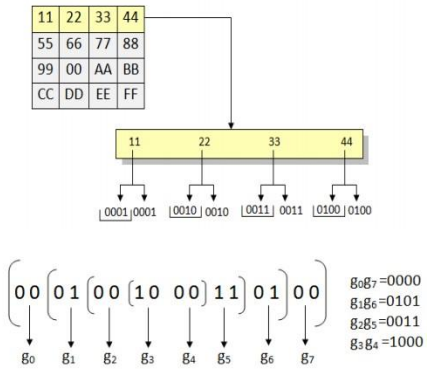
2.2 Algoritma AES Termodifikasi

Algoritma AES merupakan algoritma kriptografi selain mudah dalam implementasinya juga cukup handal hingga saat ini. Hingga tahun 2006 serangan terbaik terhadap algoritma *Rijndael* hanya berhasil menembus putaran ke-7 untuk panjang kunci 128 bit, putaran ke-8 untuk panjang kunci 192 bit dan putaran ke-9 untuk panjang kunci 256 bit [12]. Dengan melihat jumlah putaran yang berhasil ditembus, bukanlah tidak mungkin suatu hari algoritma ini dapat ditembus. Sehingga banyak peneliti di bidang kriptografi melakukan peremajaan untuk sistem enkripsi dan dekripsi menggunakan algoritma AES. Salah satunya dengan melakukan modifikasi pada algoritma AES. Modifikasi algoritma AES ini dilakukan di berbagai proses enkripsi seperti modifikasi terhadap pembangkitan kunci dan modifikasi lainnya [4]. Pada tugas akhir ini, proses *AddRoundKey* dan *Key Scheduling* sama seperti algoritma AES asli tetap tidak berubah, namun modifikasi dilakukan pada proses *SubBytes*, *ShiftRows serta MixColumns*.

1. *SubBytes* Termodifikasi

Baris pertama dan kedua dari matriks *cipherkey* masing-masing dikonversikan ke bilangan biner. Ambil 4 bit pertama dari setiap 8 bit seperti pada gambar 2.7. Kelompokkan bit-bit tersebut menjadi 4 kelompok yakni g0g7, g1g6, g2g5, g3g4. Kelompok g0, g1, g2 dan g3 masing-masing menunjukkan baris pada matriks *state* yang akan disubtitusi. Sedangkan kelompok g7, g6, g5 dan g4 masing-masing menunjukkan kolom pada matriks *state* yang akan disubtitusi. Operasi substitusi yang dilakukan sama seperti proses *SubBytes* pada

algoritma AES yang asli yakni menggunakan tabel S-box. Demikian pula ulangi seluruh proses untuk baris kedua matriks kunci *cipherkey*. Sehingga subsitusi yang terjadi maksimal sebanyak 8 *byte* [8].
Cipherkey: 11223344556677889900AABBCCDDEEFF

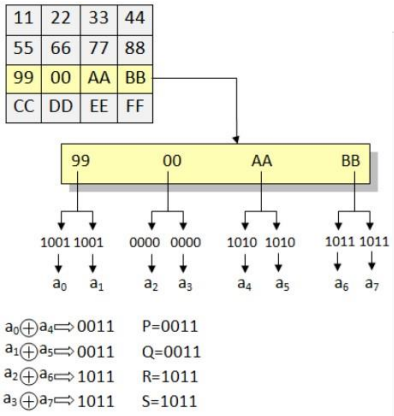


Gambar 7 Proses SubBytes Termodifikasi [8]

Untuk proses dekripsi menggunakan baris pertama dan kedua dari matriks *cipherkey*. Operasi yang dilakukan di putaran *inverse SubBytes* sama seperti saat proses enkripsi. Urutan substitusi diambil dari dalam ke luar yang berarti koordinat disubstitusi dalam urutan M (g3, g4), M (g2, g5), M (g1, g6) dan M (g0, g7). Kemudian subtitusi pada matriks *state* menggunakan tabel *inverse* S-Box.

2. *ShiftRows* Termodifikasi

Baris ketiga dari matriks *cipherkey* diubah menjadi bilangan biner. Kemudian dikelompokkan menjadi 8 kelompok, masing-masing kelompok memiliki 4 bit. Bit-bit a0 dilakukan operasi XOR dengan bit-bit a4 untuk mendapatkan 4 bit hasil biner, P. Demikian pula, Q, R dan S dihasilkan dari kelompok yang tersisa yaitu [a1,a5], [a2,a6] dan [a3,a7]. Bit-bit P dan R digunakan untuk mengidentifikasi nomor baris pada matriks *state* yang akan digeser. Sedangkan bit-bit Q dan S menentukan jumlah siklus pergeseran ke kiri. Dua bit pertama P dan R menentukan jumlah baris yang harus digeser secara berputar ke kiri. Kemudian bit yang bernilai 1 pada Q dan S dijumlahkan dan dikurangi 1 untuk memberikan jumlah siklus pergeseran untuk baris yang diwakili oleh P dan R [8]. Jika dua bit pertama (misal a1 dan a2) mewakili baris yang sama dimana sebelumnya telah digeser, maka baris yang digeser adalah baris selanjutnya.



Gambar 8 Proses ShiftRows Termodifikasi [8]

Jumlah maksimum baris digeser yakni dua baris. Operasi cermin dilakukan pada baris yang tersisa. Dalam operasi ini, tiap data dikonversi ke dalam bilangan biner dan dibaca dari kanan ke kiri, terakhir dikembalikan ke nilai heksadesimal. Contoh proses yang dilakukan.

- Misalkan *byte* yang tidak digeser, 58 (Heksadesimal) = 0101 1000 (bentuk biner)
- Baca dari kanan ke kiri sehingga didapatkan 0001 1010 = 1A (Heksadesimal)

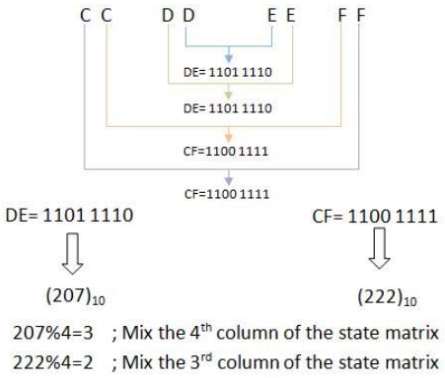
Untuk proses dekripsi, bit dari baris ketiga dari matriks kunci dikelompokkan seperti yang dijelaskan pada proses *ShiftRows* termodifikasi di atas. Dalam tahap ini baris yang ditunjukkan oleh bit-bit P dan R menentukan baris pada matriks *state* yang akan digeser kanan. Baris yang tidak digeser, kemudian dicerminkan [8].

3. *MixColumns* Termodifikasi

Jika pada algoritma AES termodifikasi referensi menggunakan baris keempat matriks *cipherkey*, maka pada algoritma AES termodifikasi yang diusulkan menggunakan baris keempat pada matriks *roundkey*, kemudian dikonversikan ke dalam bilangan biner dan dikelompokkan seperti yang ditunjukkan pada gambar 2.9. Kelompok ini masing-masing dikonversi ke nilai desimal dan dilakukan operasi modulus-4 pada masing-masing kelompok. Sisanya (r) akan selalu terletak pada kisaran 0 sampai 3 atau $0 \leq r \leq 3$ yang masing-masing menunjukkan kolom pertama, kedua, ketiga dan keempat pada matriks *state*. Proses yang dilakukan pada kolom yang dipilih sama seperti saat proses *MixColumn* di algoritma AES yang asli. Jumlah maksimum kolom yang diproses dalam langkah ini adalah empat [8].

Untuk proses dekripsnya, proses yang dijalankan sama dengan proses *MixColumns* termodifikasi tetapi dalam arah yang sebaliknya. Proses ini menggunakan baris terakhir dari matriks *roundkey*. Operasi dilakukan

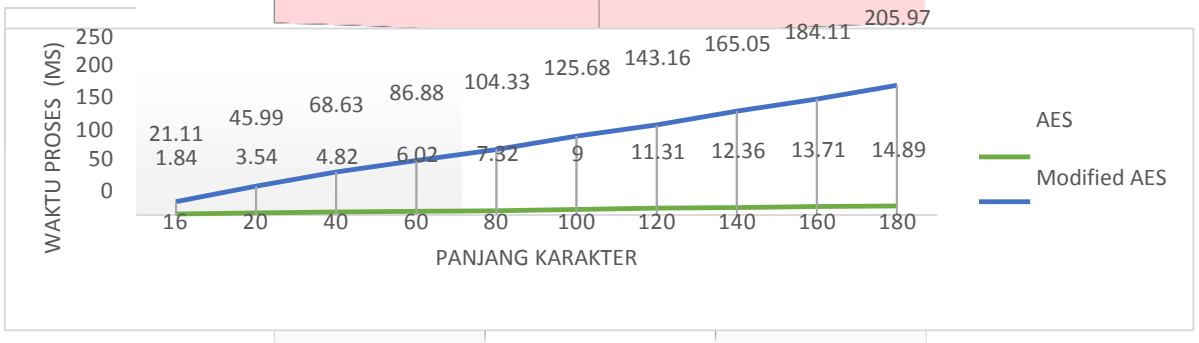
pada matriks data persis dengan urutan terbalik seperti yang proses enkripsi.



Gambar 9 Proses MixColumns Termodifikasi [8]

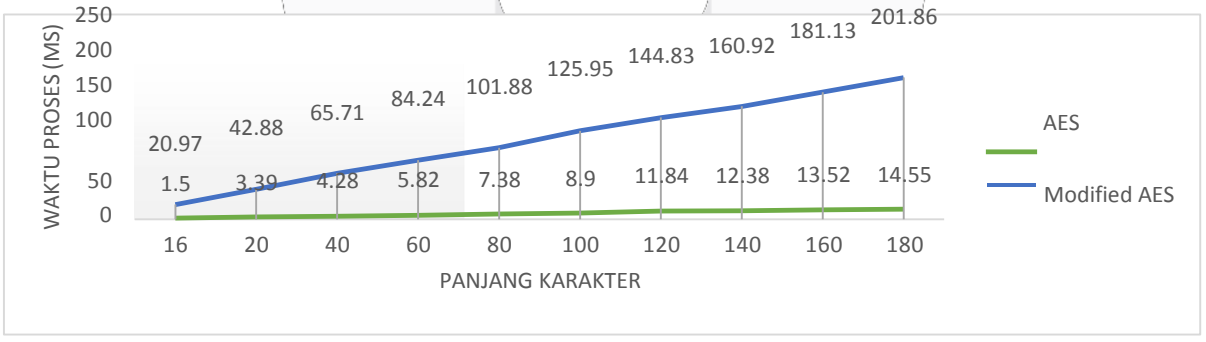
3. Pembahasan

Pada gambar 10 menjelaskan waktu komputasi yang dibutuhkan oleh kedua sistem untuk melakukan proses enkripsi pesan. Berdasarkan pada data tersebut didapatkan bahwa proses enkripsi menggunakan algoritma AES lebih cepat dibandingkan algoritma AES Termodifikasi. Serta panjang pesan yang dienkripsi berpengaruh terhadap semakin lama waktu yang dibutuhkan oleh kedua sistem untuk melakukan enkripsi pesan.



Gambar 10 Perbandingan Waktu Enkripsi AES dengan AES Termodifikasi

Sedangkan waktu komputasi yang dibutuhkan oleh kedua sistem untuk melakukan proses dekripsi pesan hampir sama dengan saat enkripsi pesan. Sistem yang menggunakan algoritma AES lebih cepat dalam dekripsi pesan dibandingkan sistem yang menggunakan algoritma AES termodifikasi. Serta panjang pesan yang didekripsi berpengaruh terhadap semakin lamanya waktu yang dibutuhkan oleh sistem untuk melakukan dekripsi pesan.



Gambar 11 Perbandingan Waktu Dekripsi AES dengan AES Termodifikasi

Analisis dilakukan dengan menghitung hasil perhitungan matematis perhitungan kunci yang digunakan. *Brute force attack* dikatakan sukses jika berhasil mencoba separuh dari kemungkinan kunci yang digunakan. Kedua algoritma menggunakan kunci dengan panjang yang sama yakni 128 bit. Sehingga kemungkinan kunci yang digunakan sebanyak 2^{128} kunci atau $3.4x10^{38}$ kunci. Kemudian asumsi kecepatan komputasi komputer yakni $2x10^{9}$ komputasi/detik, maka durasi yang dibutuhkan untuk melakukan *brute force attack* yaitu:

$$\frac{2^{127}}{2 \times 10^{9} \times 3600 \times 24 \times 365} = \frac{1.7 \times 10^{38}}{6.3 \times 10^{16}} = 2.6 \times 10^{21}$$

Pada tabel 1, menunjukkan pengaruh *input* berubah 1 bit baik pada *plaintext* maupun *cipherkey* yang berubah 1 bit. Berdasarkan pada data grafik, menunjukkan bahwa performansi ketiga sistem baik sistem yang menggunakan algoritma AES maupun kedua algoritma AES termodifikasi menunjukkan performansi yang hampir sama. Hal ini

ditunjukkan dengan nilai *avalanche effect* yang dihasilkan oleh kedua sistem memiliki nilai berkisar 50% dari total 128 bit. Artinya bahwa setiap perubahan 1 bit pada input akan menghasilkan perubahan bit setidaknya sebanyak 50% dari 128 bit.

Tabel 1 Perbandingan Performansi

Parameter		*AES*	*Modified AES Referenced*	*Modified AES Proposed*
Avalanche Effect	1 *Bit of Plaintext Changed*	51.03%	11.52%	48.73%
	1 *Bit of Cipherkey Changed*	50.24%	49.56%	49.02%
Robustness Test		49.90%	21.67%	25.57%

Untuk pengujian *robustness*, performansi sistem yang menggunakan algoritma AES termodifikasi referensi lebih baik dengan menghasilkan nilai BER (*Bit Error* Rate) yang lebih kecil yakni sebesar 21.67%. Pengujian *robustness* mengambil sampel sebanyak 15 kali, dengan jumlah bit yang *error* meningkat secara liniar. Hasil yang didapatkan menunjukkan bahwa jumlah bit yang berubah pada *ciphertext* tidak mempengaruhi seberapa banyak bit yang berubah saat dilakukan dekripsi. Namun ini hanya berlaku untuk blok yang mengalami perubahan bit. Karena pada tugas akhir ini, aplikasi dibangun dengan operasi blok mode ECB. Dimana blok-blok dieksekusi secara independen. Sehingga jika terjadi perubahan 1 bit pada blok pertama, maka pada blok kedua dan seterusnya tidak mengalami perubahan hanya pada blok yang terkait saja yang mengalami perubahan.

4. Kesimpulan

Berdasarkan waktu enkripsi, sistem yang menggunakan algoritma AES lebih cepat dalam melakukan enkripsi terhadap pesan dibandingkan dengan sistem yang menggunakan algoritma AES termodifikasi. Sehingga algoritma AES lebih cocok untuk mengamankan sistem komunikasi yang bersifat *real-time* seperti *video call,* telepon dan lain-lain. Sedangkan algoritma AES termodifikasi lebih cocok untuk mengamankan media informasi yang bersifat *non-real-time* seperti mengamankan *file* atau dokumen-dokumen penting, data-data perusahaan maupun data-data pribadi penting lainnya. Berdasarkan durasi waktu yang dibutuhkan untuk melakukan *brute force attack*, kedua algoritma ini tahan dan aman terhadap *brute force attack*. Karena menggunakan panjang kunci yang sama, sehingga membutuhkan waktu 2.6×10^{21} tahun untuk dapat mencoba semua kemungkinan kunci yang digunakan. Telah berhasil melakukan optimasi performansi untuk nilai *avalanche effect* pada sistem yang menggunakan algoritma AES termodifikasi referensi dengan mengusulkan penggunaan baris keempat dari matriks *roundkey* untuk melakukan transformasi *MixColumns*. Setelah dilakukan optimasi pada algoritma AES termodifikasi, performansi sistem yang menggunakan algoritma AES dan AES termodifikasi tidak jauh berbeda jika dilihat dari nilai *avalanche effect* yang didapat. Dimana nilai *avalanche effect* berkisar 50%. Performansi sistem yang menggunakan algoritma AES tidak lebih baik ketika dilakukan pengujian *robustness* dengan menghasilkan BER yang lebih tinggi dibandingkan sistem yang menggunakan algoritma AES termodifikasi.

DAFTAR PUSTAKA

[1] Bagus Satrio Waluyu Putro, Aris Sugiharto, Sukmawati Nur Endah. 2010. "KRIPTOGRAFI CITRA DIGITAL DENGAN ALGORITMA RIJNDAEL DAN TRANSFORMASI WAVELET DISKRIT HAAR." *SEMINAR NASIONAL ILMU KOMPUTER UNIVERSITAS DIPONEGORO.* Semarang: Universitas Diponegoro. 209-215.

[2] Dusimar, Arnes. t.thn. "IMPLEMENTASI PENGAMANAN PESAN SMS DENGAN MANAJEMEN KUNCI PADA ANDROID."

[3] Hanifah, Fadhilah. 2012. *Aplikasi Algoritma Rijndael Dalam Pengamanan Citra Digital.* Depok: Universitas Indonesia.

[4] Ibtihal Mohamed Abdullateef Fadul, Tariq Mohamed Hassan Ahmed. 2013. "Enhanced Security of Rijndael Algorithm using Two Secret Keys." *International Journal of Security and Its Applications* 127-134.

[5] Kusumah, Angga. 2012. *Secure Chatting (Instant Messaging) Menggunakan Metode Enkripsi Blowfish, Twofish, dan AES.* Bandung: Institut Teknologi Telkom.

[6] Munir, Rinaldi. 2004. *Bahan Kuliah IF5054 Kriptografi Advanced Encryption Standard (AES).* Bandung: Institut Teknologi Bandung.

[7] P., Andi Kurniawan Dwi. 2012. "Penerapan Algoritma Vigenere Cipher pada Aplikasi SMS." *Makalah IF3058 Kriptografi.*

[8] Priyanka Pimpale, Rohan Rayarikar, Sanket Upadhyay. 2011. "Modification to AES Algorithm for Complex Encryption." *IJCSNS Internation Journal of Computer Science and Network Security* 183-186.

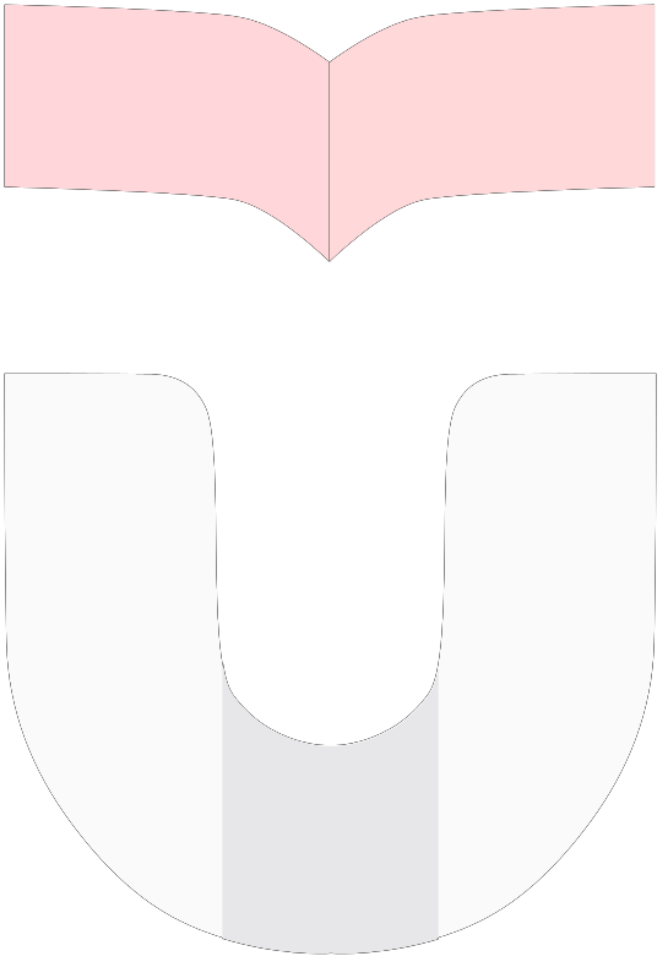
[9] R. Kristoforus JB, Stefanus Aditya BP. 2012. "IMPLEMENTASI ALGORITMA RIJNDAEL UNTUK ENKRIPSI DAN DEKRIPSI PADA CITRA DIGITAL." *Seminar Nasional Aplikasi Teknologi Informasi 2012.*

[10] Rahmah, Anisa J. 2007. *Aplikasi Enkripsi SMS pada Handphone Berbasis Java dengan Menggunakan Algoritma Rijndael.* Bandung: Sekolah Tinggi Teknologi Telkom.

[11] Satria, Eko. 2009. *Studi Algoritma RIJNDAEL Dalam Sistem Keamanan Data.* Medan: Universitas Sumatra Utara.

[12] Surian, Didi. 2006. "ALGORITMA KRIPTOGRAFI AES RIJNDAEL." *TESLA* 97-101.

[13] Susanto, Iskan. 2009. *ANALISA DAN IMPLEMENTASI ENKRIPSI-DEKRIPSI SUARA MENGGUNAKAN ALGORITMA RIJNDAEL.* Bandung: Institut Teknologi Telkom.

[14] Susilo, Taufik Ardi. 2013. *Analisis Perbandingan Performansi Sistem Keamanan Pada Android Menggunakan Algoritma AES dan Algoritma Twofish.* Bandung: Institut Teknologi Telkom.

[15] Y., Virgan R. 2007. *APLIKASI ENKRIPSI DAN DEKRIPSI* . Makalah Seminar Tugas Akhir, Semarang: Universitas Diponegoro.