Analisis Performa Modifikasi KSA dan PRGA pada RC4 dengan Vigenere Cipher untuk Enkripsi dan Dekripsi Citra Digital

Josua Adriel Sinabutar
Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
18221065@std.stei.itb.ac.id

***Abstract*— In this digital era, information can be transmitted in real-time through various communication technologies. However, alongside this convenience, the protection of access and security of transmitted information has become crucial. Various methods have been developed to address these security issues, including cryptography. One commonly used encryption algorithm in cryptography is RC4 (Rivest Cipher 4). Despite its advantages, RC4 has several vulnerabilities, particularly in its Key Scheduling Algorithm (KSA) and Pseudo-Random Generation Algorithm (PRGA), making it unsuitable for high-security applications. To mitigate these issues, modifications can be made to strengthen the security by mixing algorithms. This paper explores the combination of RC4 with the Vigenere Cipher, a polyalphabetic substitution cipher. The performance analysis focuses on comparing the original RC4 algorithm with the modified version that incorporates the Vigenere Cipher, specifically for digital images and videos encryption.**

Keywords—RC4, Modified RC4, Vigenere Cipher, Digital Image, Digital Video

I. Pendahuluan

Di era digital yang semakin canggih ini, pertukaran data berlangsung dengan cepat dan masif. Setiap orang dapat mengirimkan informasi melalui teknologi komunikasi secara real-time. Jenis data yang dapat dikirimkan pun berkembang pesat seiring kemajuan teknologi informasi. Mulai dari teks singkat, suara, dokumen, hingga citra digital, semua dapat ditransmisikan dengan mudah saat ini. Namun, di balik kemudahan ini, perlindungan terhadap akses dan keamanan informasi yang dikirimkan menjadi sangat krusial. Informasi yang dikirimkan melalui internet rentan terhadap perubahan, akses ilegal, atau pemalsuan oleh pihak-pihak yang tidak bertanggung jawab.

Berbagai metode pengamanan informasi telah ditemukan, seperti biometrik, steganografi, dan kriptografi untuk mengatasi masalah ini. Di antara metode-metode tersebut, kriptografi menjadi salah satu cara paling efektif untuk memastikan kerahasiaan data dari pihak eksternal. Dalam dunia kriptografi, salah satu algoritma enkripsi yang umum digunakan adalah adalah RC4 (Rivest Cipher 4). RC4, yang dikembangkan oleh Ron Rivest pada tahun 1987, adalah algoritma stream cipher yang menggunakan kunci variabel untuk menghasilkan byte stream pseudo-acak [1]. *Byte stream* ini kemudian digabungkan dengan teks asli untuk menghasilkan *ciphertext* yang terenkripsi. Popularitas RC4 meningkat pesat karena kesederhanaannya dan kecepatan prosesnya, yang membuatnya cocok untuk berbagai aplikasi seperti komunikasi nirkabel dan protokol keamanan internet [2]. Meskipun demikian, RC4 memiliki beberapa kelemahan yang dapat dieksploitasi, khususnya dalam tahap *Key Scheduling Algorithm* (KSA) dan *Pseudo-Random Generation Algorithm* (PRGA) [2]. Kelemahan ini membuat RC4 tidak lagi direkomendasikan untuk aplikasi keamanan yang tinggi.

Untuk menangani hal ini, berbagai upaya modifikasi dapat dilakukan, seperti memperkuat algoritma pencampuran, meningkatkan entropi kunci, menambahkan status internal, dan menggabungkan RC4 dengan algoritma enkripsi lain. Makalah ini akan mengeksplorasi penggabungan algoritma RC4 dengan sebuah algoritma enkripsi lain, yaitu Vigenere Cipher. Vigenere Cipher adalah teknik enkripsi *polyalphabetic* yang menggunakan variasi dalam pengkodean karakter berdasarkan kunci tertentu [3]. Makalah ini akan melakukan analisis performa algoritma enkripsi RC4 yang asli dengan algoritma yang telah digabungkan dengan Vigenere Cipher, khususnya pada jenis masukan citra digital.

Proses enkripsi dan dekripsi citra digital membutuhkan pendekatan yang cukup berbeda. Struktur data citra juga berbeda dengan teks biasa karena piksel-piksel yang berdekatan biasanya memiliki nilai yang saling terkait, sehingga enkripsi citra digital menimbulkan tantangan unik yang memerlukan pendekatan khusus [4]. Oleh karena itu, Penulis tertarik untuk mengevaluasi bagaimana modifikasi pada KSA dan PRGA dari RC4, dikombinasikan dengan Vigenere Cipher, dapat meningkatkan keamanan dan performa dalam enkripsi dan dekripsi citra digital. Penulis akan memfokuskan analisis pada dampak modifikasi KSA dan PGRA terhadap performa keamanan dan kecepatan kerja algoritma.

II. LANDASAN TEORI

A. Rivest Cipher 4 (RC4)

Rivest Cipher 4 (RC4) adalah algoritma enkripsi stream cipher yang dirancang oleh Ron Rivest pada tahun 1987 [1]. Algoritma ini menjadi salah satu cipher yang paling banyak digunakan dalam sejarah kriptografi karena kesederhanaan dan kecepatan prosesnya. RC4 telah digunakan secara luas dalam berbagai aplikasi keamanan seperti protokol *Secure Sockets Layer* (SSL), *Transport Layer Security* (TLS), dan jaringan nirkabel WEP (*Wired Equivalent Privacy*).

Secara garis besar, algortima RC4 terdiri dari dua komponen utama, yaitu:

1. Key Scheduling Algorithm (KSA)

KSA adalah proses inisialisasi yang menggunakan kunci enkripsi untuk menghasilkan array permutasi awal dari *byte* (S-box). S-box adalah *array* berukuran 256-*byte* yang mengandung nilai-nilai dari 0 hingga 255 dalam urutan tertentu. KSA bertanggung jawab untuk memadukan kunci awal dengan S-box untuk membuat permutasi yang digunakan dalam PRGA [1].

Proses KSA dimulai dengan menginisialisasi *array* S dengan nilai *default* (0 hingga 255) dan kemudian menggunakan kunci untuk mengacak nilai dalam *array* tersebut. Proses ini memastikan bahwa S-box yang dihasilkan tergantung pada kunci yang digunakan, sehingga menghasilkan keunikan dalam setiap enkripsi yang dilakukan.

2. *Pseudo-Random Generation Algorithm* (PRGA)

PRGA adalah proses yang menghasilkan *stream* bit pseudo-acak menggunakan S-box yang telah diinisialisasi oleh KSA [1]. Algoritma ini mengubah *array* S secara terus-menerus dan menggunakan nilai-nilai dari *array* ini untuk menghasilkan *byte pseudo-random* yang kemudian digabungkan dengan teks asli untuk membuat *ciphertext*. PRGA beroperasi dengan cara memilih *byte* dari S-box berdasarkan indeks yang dihitung menggunakan variabel penunjuk. Kemudian *byte* yang dipilih digabungkan untuk menghasilkan keluaran *byte* yang digabungkan dengan teks asli.

Secara runtut, algoritma RC4 terdiri atas tiga langkah, yaitu:

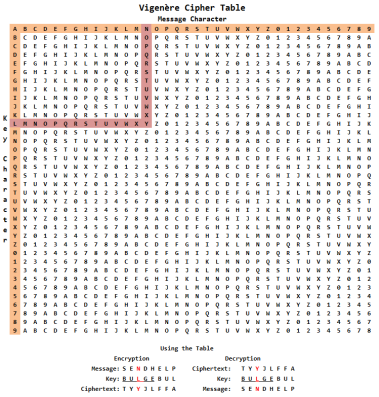
1. Inisialisasi: Dimulai dengan kunci yang panjangnya bisa bervariasi, array S-box diinisialisasi dengan nilai 0 hingga 255.
2. KSA: Kunci digunakan untuk mengacak nilai dalam *array* S sehingga menghasilkan permutasi yang tergantung pada kunci.
3. PRGA: Setelah S-box terinisialisasi, PRGA menghasilkan *stream byte pseudo-random* yang digunakan untuk mengenkripsi atau mendekripsi data.

B. Vigenere Cipher

Vigenere Cipher adalah salah satu metode enkripsi klasik yang menggunakan pendekatan *polyalphabetic* substitution cipher [3]. Dikembangkan pada abad ke-16 oleh Blaise de Vigenère, cipher ini dikenal karena kesederhanaan dan efektivitasnya dalam menyembunyikan pesan dari analisis frekuensi yang digunakan pada cipher substitusi *monoalphabetic* seperti Caesar Cipher.

Vigenere Cipher mengandalkan penggunaan kunci untuk mengenkripsi dan mendekripsi pesan. Kunci ini adalah kata atau frase yang menentukan pola substitusi huruf dalam pesan asli. Dengan menggunakan kunci yang lebih panjang dan lebih kompleks, Vigenere Cipher dapat mengaburkan pola dalam teks yang dienkripsi, membuatnya lebih tahan terhadap analisis kriptografi sederhana.

Proses enkripsi dan dekripsi pada Vigenere Cipher melibatkan penggunaan tabel yang disebut Vigenere Table atau Vigenere Square. Tabel ini adalah matriks 26x26 huruf yang menampilkan semua kemungkinan substitusi Caesar untuk setiap huruf dalam alfabet.



Gambar 1. Tabel Vigenere

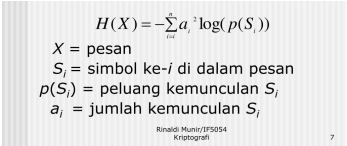
Proses enkripsi dan dekripsi dialkukan dengan mencari titik temu antara karakter *plaintext* dengan karakter kunci yang dipilih. Karakter dari tabel yang menjadi titik temu kedua nilai tersebut akan menjadi keluaran algoritma sebagai *ciphertext*. Proses ini diulang terus hingga seluruh pesan terenkripsi dengan baik atau sebaliknya.

C. Alat Ukur Algoritma Kriptografi

1. Tingkat Kerahasiaan (*Secrecy*)
   a. Entropi

      Entropi dalam kriptografi adalah ukuran dari ketidakpastian atau keacakan dalam sebuah sistem [6]. Entropi mengukur jumlah informasi yang tidak dapat diprediksi dalam suatu pesan atau keystream yang digunakan dalam enkripsi. Semakin tinggi entropi, semakin sulit untuk memprediksi informasi tersebut, dan karenanya semakin tinggi tingkat kerahasiaan [6].

Entropi, biasanya dinyatakan dalam bit, dihitung berdasarkan distribusi probabilitas dari simbol-simbol yang terdapat dalam pesan atau keystream. Formula umum untuk entropi (H) adalah sebagai berikut.



Gambar 2. Rumus Entropi [6]

b. *Byte Distribution*

Byte distribution mengacu pada frekuensi kemunculan berbagai nilai *byte* (0-255) dalam data terenkripsi atau *keystream*. Distribusi yang merata menunjukkan bahwa setiap byte memiliki kemungkinan yang sama untuk muncul, yang merupakan karakteristik yang diinginkan dalam sistem kriptografi yang kuat.

Distribusi *byte* dapat divisualisasikan melalui histogram, di mana setiap bar mewakili frekuensi kemunculan *byte* tertentu dalam data. Idealnya, histogram untuk data yang teracak dengan baik akan mendekati garis yang merata di seluruh nilai *byte* [7].

Untuk menganalisis distribusi *byte*, data terenkripsi atau keystream dapat diperiksa untuk memastikan tidak ada bias yang signifikan terhadap nilai *byte* tertentu. Misalnya, jika byte tertentu muncul lebih sering dari yang lain, ini bisa menjadi tanda bahwa data tidak teracak dengan baik dan mungkin rentan terhadap serangan.

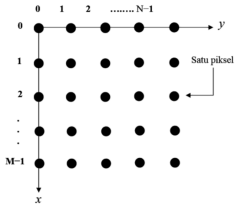
2. Tingkat Performa (*Performance*)

Tingkat performa (*performance*) merujuk pada efisiensi dan efektivitas suatu algoritma enkripsi dalam proses pengamanan data. Hal ini mencakup kecepatan dan efisiensi dalam enkripsi dan dekripsi. Evaluasi performa dari teknik *cipher* sangat penting untuk menentukan kelayakan dan efisiensinya dalam berbagai aplikasi.

Pada makalah ini, tingkat performa akan dihitung berdasarkan kecepatan enkripsi dan dekripsi algoritma *cipher*. Kecepatan menjadi penting saat data masukan memiliki volume yang besar. Tingkat performa dapat dinyatakan baik jika durasi kerja algoritma semakin kecil.

D. Citra Digital

Citra adalah suatu gambaran atau kemiripan dari sebuah objek [5]. Citra sebagai objek terbagi menjadi dua, yaitu citra analog dan citra digital. Citra analog adalah gambaran yang terbentuk dari sinyal kontinu sedangkan citra digital adalah gambaran yang terbentuk dari proses sampling dan kuantisasi komputer. Sistem sampling membagi citra analog menjadi sejumlah baris dan kolom sehingga sinyal kontinu dapat direpresentasikan dalam bentuk diskrit. Pertemuan antara baris dan kolom inilah yang disebut sebagai piksel. Sedangkan sistem kuantisasi melakukan pengubahan intensitas analog menjadi intensitas diskrit yang memungkinkan proses gradasi warna pada tiap piksel. Kedua konsep inilah yang memproses sinyal kontinu citra analog menjadi representasi citra digital dalam bentuk matriks dengan sejumlah baris dan kolom serta intensitas warna yang sesuai.



Gambar 3. Matriks representasi citra digital [5]

Seperti ilustrasi diatas, citra digital direpresentasikan sebagai matriks, di mana setiap elemen matriks merepresentasikan nilai piksel. Untuk citra grayscale, matriks hanya memiliki satu nilai per piksel, sedangkan untuk citra berwarna (RGB), ada tiga nilai per piksel (merah, hijau, biru) [5]. Dalam proses enkripsi citra digital, nilai dari tiap piksel inilah yang akan menjadi masukan dari fungsi enkripsi yang digunakan. Algoritma enkripsi akan mengacaukan nilai tiap piksel sehingga menghasilkan ciphertext yang tidak dapat dibaca oleh pihak yang tidak berwenang. Pada proses dekripsi, nilai ciphertext tiap piksel dikembalikan ke posisi semula sehingga citra digital dapat dimengerti kembali.

III. Metode Penelitian

Dalam penelitian ini, penulis akan membandingkan tingkat entropi dan kecepatan dari algoritma RC4 asli dengan RC4 yang telah dimodifikasi dengan Vigenere Cipher. Penelitian dilakukan melalui perangkat lunak *website* yang dibangun menggunakan kerangka kerja Next.js dan bahasa pemrograman TypeScript. Performa diukur dengan melakukan percobaan sebanyak empat kali kemudian diukur rata-ratanya.

Pengukuran dilakukan menggunakan *laptop* Penulis dengan spesifikasi sebagai berikut:

- OS: macOS Ventura 13.5.1
- Chip: Apple M2
- RAM: 8 GB
- Jumlah Core: 8

Dalam proses pengujian, Penulis mengimplementasikan metode pengujian dalam bentuk sebagai berikut.

1. Metode Pengkuran Kerahasiaan (*Secrecy*)

Pengukuran kerahasiaan akan dilakukan dengan fungsi countByteDistribution dan fungsi calculateEntropy yang keduanya menerima data dalam Uint8Array. Kedua fungsi tersebut akan digunakan untuk menghitung distribusi *byte* dari data hasil enkripsi citra digital. Setelah itu, entropi ditampilkan ke layar dalam satuan bit dan *byte distribution* ditampilkan dalam bentuk histogram. Berikut adalah kode implementasi fungsi tersebut.

Tabel 1. Algoritma countByteDistribution dan calculateEntropy

```
const countByteDistribution = (data: Uint8Array): number[] => {
  const distribution = Array(256).fill(0);
  data.forEach((byte) => {
    distribution[byte]++;
  });
  return distribution;
};

const calculateEntropy = (data: Uint8Array): number => {
  const frequency: { [key: string]: number } = {};
  for (let i = 0; i < data.length; i++) {
    const char = data[i];
    frequency[char] = (frequency[char] || 0) + 1;
  }

  const dataSize = data.length;
  let entropy = 0;
  for (const char in frequency) {
    const p = frequency[char] / dataSize;
    entropy -= p * Math.log2(p);
  }

  return entropy;
};
```

2. Metode Pengukuran Performa (Performance)

Pengukuran performa kecepatan akan dilakukan dengan menghitung durasi yang dibutuhkan dalam mengeksekusi proses enkripsi dan dekripsi citra digital. Perhitungan durasi dilakukan dengan mencatat waktu awal dan akhir dari tiap proses enkripsi dan dekripsi. Waktu akhir akan dikurangi dengan waktu awal yang menghasilkan durasi proses dan ditampilkan ke layar. Berikut adalah implementasi kode fungsi pengukuran kecepatan.

Tabel 2. Algoritma encryptFile

```
const encryptFile = () => {
    if (!file || !key) {
      alert("Please select a file and enter a key."); // Jika file atau key
kosong
      return;
    }

    const fileReader = new FileReader();

    fileReader.onload = (e: ProgressEvent<FileReader>) => {
      const fileData = new Uint8Array(e.target?.result as ArrayBuffer);
      const start = performance.now();
      const encryptedData = EncryptDecryptFileInput.encrypt(fileData, key);
      setEncryptedData(encryptedData);
      const end = performance.now();

      const encryptedBlob = new Blob([new Uint8Array(encryptedData)], { type:
file.type });
      setEncryptedFile(encryptedBlob);
      setDecryptedFile(null);
      setEncryptionTime(end-start);

      const entropy = calculateEntropy(encryptedData);
      setFileEntropy(entropy);

      const secrecy = calculateSecrecy(key, encryptedData);
      setFileSecrecy(secrecy);
    };

    fileReader.readAsArrayBuffer(file);
  };
```

3. Metode Komparasi RC4 dan *Modified* RC4

Performa dihitung dengan membandingkan empat nilai ukur, yaitu:

- *Encryption time*, waktu yang lebih rendah menandakan algoritma lebih baik dan efisien
- *Decryption time*, waktu yang lebih rendah menandakan algoritma lebih baik dan efisien
- *Entropy*, entropi yang lebih besar menandakan algoritma lebih baik
- *Byte distribution*, *byte* yang lebih terdistribusi menandakan algoritma lebih baik (visual)

IV. Penelitian dan Pembahasan

1. Implementasi RC4 dan *Modified* RC4

Implementasi kode modifikasi RC4 terdiri atas dua bagian, yaitu modifikasi KSA dan PGRA. Proses modifikasi KSA dengan vigenere cipher dilakukan sebagai berikut.

- Setelah modifikasi, kunci (*key*) mengalami enkripsi menggunakan metode Vigenere cipher sebelum digunakan dalam proses pembuatan S-box
- Setelah modifikasi, sebuah modifier dihasilkan dari enkripsi Vigenere pada karakter dari kunci (key). Modifier ini kemudian ditambahkan pada perhitungan indeks variabel j selama iterasi pembuatan S-box.

Sedangkan, proses modifikasi PRGA dengan vigenere cipber dilakukan sebagai berikut.

- Setelah modifikasi, sebuah *stream modifier* dihasilkan dari enkripsi Vigenere pada karakter dari kunci. Stream modifier ini kemudian ditambahkan pada perhitungan.

Berikut adalah kode implementasi KSA dan PGRA kedua algoritma.

Tabel 3. Algoritma KSA RC4

```
function initializeSBox(key: string): number[] {
   const sBox: number[] = [];
   const keyLength = key.length;
   const s: number[] = Array.from(Array(256).keys());

   let j = 0;
   for (let i = 0; i < 256; i++) {
       j = (j + s[i] + key.charCodeAt(i % keyLength)) % 256;
       [s[i], s[j]] = [s[j], s[i]];
   }

   return s;
}
```

Tabel 4. Algoritma KSA *modified* RC4

```
function initializeSBoxModified(key: string): number[] {
   const sBox: number[] = [];
   const keyLength = key.length;
   const s: number[] = Array.from(Array(256).keys());
   key = vigenereEncrypt(key,key)
   let j = 0;
   for (let i = 0; i < 256; i++) {
       const modifier = vigenereEncrypt(key.charAt((i %
key.length)),key).charCodeAt(0);
       j = (j + s[i] + key.charCodeAt(i % keyLength) + modifier) % 256;
       [s[i], s[j]] = [s[j], s[i]];
   }

   return s;
}
```

Tabel 5. Algoritma PGRA RC4

```
function generateStream(key: string, messageLength: number): number[] {
   const sBox = initializeSBox(key);
   let i = 0;
   let j = 0;
   const stream: number[] = [];

   for (let k = 0; k < messageLength; k++) {
       i = (i + 1) % 256;
       j = (j + sBox[i]) % 256;
       [sBox[i], sBox[j]] = [sBox[j], sBox[i]];
       const t = (sBox[i] + sBox[j]) % 256;
       const pseudoRandomByte = sBox[t];
       stream.push(pseudoRandomByte);
   }

   return stream;
}
```

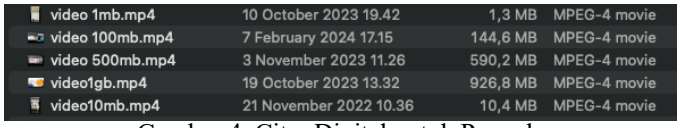
Tabel 6. Algoritma PGRA *modified* RC4

```
function generateStreamModified(key: string, messageLength: number):
number[] {
   const sBox = initializeSBoxModified(key);
   let i = 0;
   let j = 0;
   const stream: number[] = [];

   for (let k = 0; k < messageLength; k++) {
       i = (i + 1) % 256;
       j = (j + sBox[i]) % 256;
       [sBox[i], sBox[j]] = [sBox[j], sBox[i]];
       const streamModifier = vigenereEncrypt(key.charAt((i %
key.length)),key).charCodeAt(0);
       const t = (sBox[i] + sBox[j] + streamModifier) % 256;
       const pseudoRandomByte = sBox[t];
       stream.push(pseudoRandomByte);
   }
   return stream;
}
```

2. Persiapan Pengukuran

Persiapan dimulai dengan menyiapkan citra digital yang akan dijadikan bahan enkripsi dan dekripsi. Penulis menyiapkan 4 jenis video dengan ukuran data yang berbeda-beda, mulai dari 1 MB, 100 MB, 500 MB, dan 1 GB.

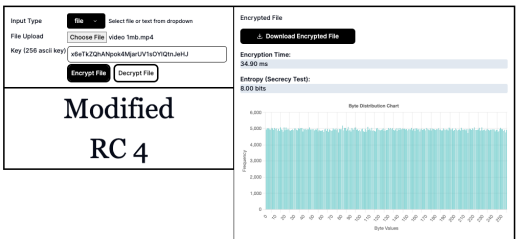


Gambar 4. Citra Digital untuk Pengukuran

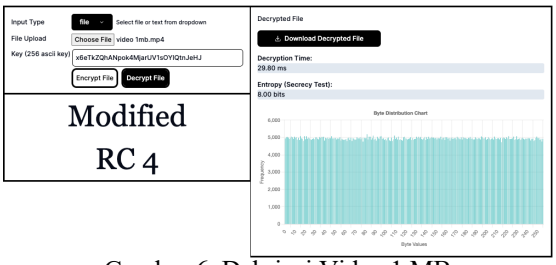
Persiapan berikutnya adalah menentukan *key*. Karena dalam RC4, *key input* bentuk *string* akan dianggap sebagai ASCII, maka 1 huruf berukuran 8 *bits*. *Key* dalam bentuk huruf akan dibangkitkan menggunakan *tools* berbasis *website*, yaitu https://acte.ltd/utils/randomkeygen. Panjang *Key* yang akan digunakan beragam, mulai dari 64 bits (8 ASCII), 128 bits (16 ASCII), 256 bits (32 ASCII), hingga 512 bits (64 ASCII).

3. Proses Pengukuran Kerahasiaan (*Secrecy*) dan Performa (*Performance*)

Pengukuran kerahasiaan dilakukan pada *website* yang dibuat oleh Penulis. Penulis membuat *website* menggunakan kerangka kerja NextJS dan dengan bahasa Typescript. Dalam halaman utama, pengguna dapat memilih algoritma enkripsi yang akan dilakukan. Pada tiap jenis algoritma, pengguna dapat mengunggah citra digital yang ingin diuji dan hasil uji performa dan kerahasiaan akan ditampilkan pada layar. Kode lengkap implementasi *website* dapat ditemukan pada lampiran. Berikut adalah dokumentasi pengujian yang telah dilakukan.



Gambar 5. Enkripsi Video 1 MB



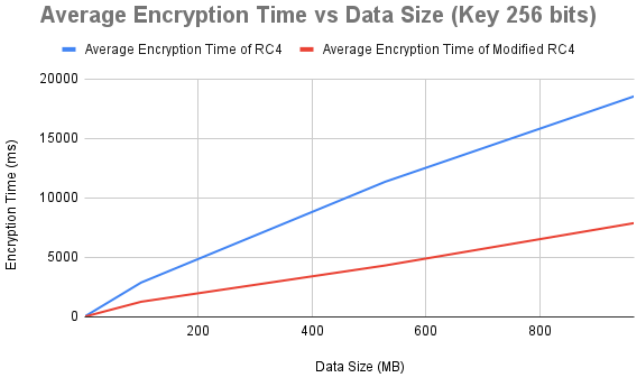
Gambar 6. Dekripsi Video 1 MB

Setelah melakukan empat pengujian terhadap tiap ukuran video yang berbeda-beda, berikut hasil perbandingan rata-rata waktu enkripsi kedua algoritma.

Tabel 7. Hasil Waktu Rata-rata Enkripsi Kedua Algoritma

Data Size (MB)	***Average Encryption Time of RC4***	***Average Encryption Time of Modified RC4***
1	28.30 ms	17.20 ms
100	2884.80 ms	1271.80 ms
528	11356.30 ms	4320.90 ms
965	18544.80 ms	7885.30 ms

Untuk memperjelas perbandingan waktu enkripsi kedua algoritma, berikut adalah grafik perbandingan kedua algoritma.



Gambar 7. Grafik Perbandingan Waktu Rata-rata Enkripsi

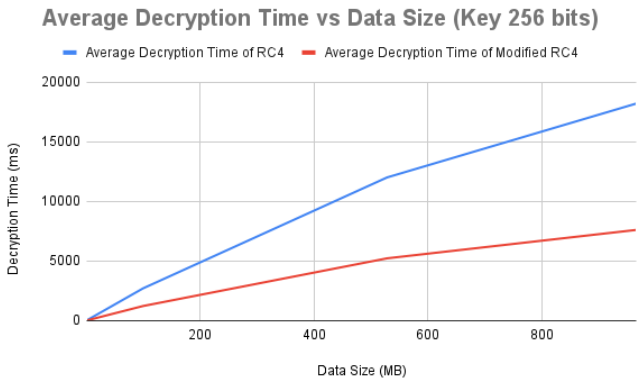
Dengan kunci yang sama, berikut adalah hasil pengujian dan perbandingan waktu dekripsi kedua algoritma.

Tabel 8. Hasil Waktu Rata-rata Dekripsi Kedua Algoritma

Data Size (MB)	***Average Decryption Time of RC4***	***Average Decryption Time of Modified RC4***
1	25.20 ms	18.30 ms
100	2713.90 ms	1232.00 ms

528	12015.30 ms	5231.00 ms
965	18209.70 ms	7612.40 ms

Untuk memperjelas perbandingan waktu enkripsi kedua algoritma, berikut adalah grafik perbandingan kedua algoritma.



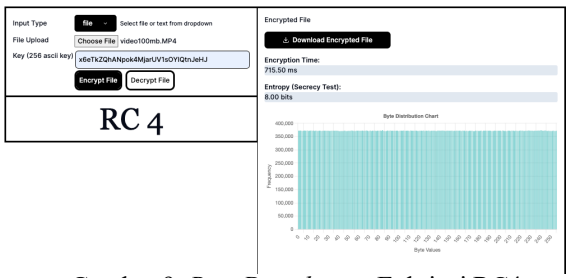
Gambar 8. Perbandingan Waktu Rata-rata Dekripsi

Selain waktu, Penulis juga membandingkan rata-rata entropi dengan perubahan panjang kunci dengan ukuran data tetap sebesar 100 MB. Berikut adalah hasil percobaan yang ditemukan.

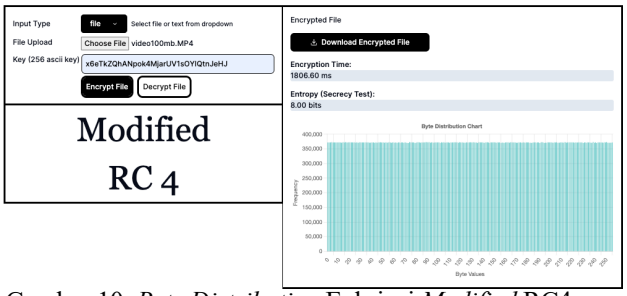
Tabel 8. Perbandingan Rata-rata Entropi

Key Length (bits)	***Average Entropy of RC4***	***Average Entropy of Modified RC4***
8	8 bits	8 bits
16	8 bits	8 bits
32	8 bits	8 bits
64	8 bits	8 bits

Terakhir, Penulis juga membandingkan distribusi *byte* dari kedua algoritma dengan masukan video 100 MB.



Gambar 9. *Byte Distribution* Enkripsi RC4



Gambar 10. *Byte Distribution* Enkripsi *Modified* RC4

4. Analisis Hasil

Dari bagian hasil pengujian sebelumnya, terdapat beberapa hal yang dapat dianalisis. Dari Gambar 7, dapat dilihat bahwa modified RC4 memiliki performa yang lebih rendah dalam melakukan enkripsi. Waktu enkripsi dari *modified* RC4 selalu lebih tinggi (lama) jika dibandingkan dengan RC4. Selisih dari perbedaan kecepatan enkripsi antara keduanya berbanding lurus dengan ukuran file. Semakin besar ukuran *file*, semakin besar pula perbedaan waktu enkripsi antara kedua algoritma tersebut. Waktu enkripsi *modified* RC4 memiliki rata-rata 122,34% lebih besar dari RC4 dengan perbandingan waktu enkripsi *modified* RC4 dengan RC4 yaitu 222 : 100.

Gambar 8 juga menunjukan bahwa hal yang serupa terjadi pada performa dekripsi modified RC4 dan RC4. Waktu dekripsi modified RC4 lebih lama dibandingkan RC4. Hal tersebut dapat terjadi karena RC4 merupakan *cipher* simetris yang tidak hanya menggunakan kunci enkripsi dan dekripsi yang sama, tetapi juga memiliki algoritma enkripsi dan dekripsi yang sama dengan alur yang berkebalikan.

Tabel 7, mengenai hasil pengujian *entropy*, menunjukkan bahwa seluruh percobaan memiliki nilai yang sama, yaitu 8 bits. Hal tersebut terjadi karena video yang digunakan memiliki jumlah byte yang sangatlah banyak (100MB = 100.000.000 Bytes). Banyaknya *byte* tersebut mengakibatkan seluruh kemungkinan bit (256) mengalami kemunculan setidaknya satu kali. Oleh karena itu, *entropy* yang dihasilkan adalah 8 bits.

Gambar 9 dan 10 menunjukan bahwa *byte distribution* dari kedua algoritma memiliki tingkat persebaran yang baik. Hal ini berarti data teracak dengan baik. Perbandingan antara kedua algoritma juga tidak menunjukkan perbedaan yang signifikan.

V. KESIMPULAN

Dari hasil penelitian, dapat disimpulkan bahwa dari segi performa waktu enkripsi dan dekripsi, modified RC4 menggunakan vigenere cipher tidaklah efektif. RC4 tanpa modifikasi memiliki performa waktu yang jauh lebih baik. Kemudian, dari segi kerahasiaan berdasarkan *entropy* dan *byte distribution*, keduanya tidak memiliki perbedaan yang signifikan. Namun, teknik penggabungan terbukti dapat mempengaruhi performa dan kerahasiaan sebuah algoritma. Namun, pemilihan algoritma yang akan digabungkan harus memiliki performa yang jauh lebih cepat agar dapat memberikan dampak positif setelah penggabungan. Dari hasil penelitian, dapat disimpulkan bahwa modifikasi RC4 dengan vigenere cipher tidak efektif dari segi performa (waktu enkripsi dan dekripsi), tetapi cukup efektif dalam segi kerahasiaan. Skenario

penggunaan modified RC4 dengan vigenere cipher yang paling cocok adalah ketika mencari algoritma *stream cipher* yang hasil enkripsinya mirip dengan RC4, tetapi teknik dekripsinya tidak diketahui banyak orang (cenderung unik).

VI. Daftar Pustaka

[1] D. R. Munir, "Informatika STEI ITB," 2024. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/2023-2024/06-Stream-cipher-2024.pdf. [Accessed 06 2024].

[2] V. S. Arintamy, "ANALISIS ALGORITMA RC4 SEBAGAI METODE ENKRIPSI WPA-PSK PADA SISTEM KEAMANAN JARINGAN WIRELESS LAN," *ANALISIS ALGORITMA RC4 SEBAGAI METODE ENKRIPSI WPA-PSK PADA SISTEM KEAMANAN JARINGAN WIRELESS LAN,* vol. 1, 2014.

[3] D. R. Munir, "Informatika STEI ITB," 2024. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/2023-2024/03-Ragam-Cipher-Klasik-Bagian2-(2024).pdf. [Accessed 06 2024].

[4] R. Muriliasari, "ANALISIS PERBANDINGAN METODE LI DAN CHAN-VESE PADA PROSES SEGMENTASI CITRA DIGITAL," *ANALISIS PERBANDINGAN METODE LI DAN CHAN-VESE PADA PROSES SEGMENTASI CITRA DIGITAL,* vol. 1, 2013.

[5] P. N. Andono, Pengolahan Citra Digital, Yogyakarta: ANDI, 2017.

[6] D. R. Munir, "Informatika STEI ITB," 2006. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2006-2007/Landasan%20Matematika%20Untuk%20Kriptografi.ppt. [Accessed 06 2024].

[7] M. Chromiak, "Stream security particularities in Java," *Stream security particularities in Java,* 2008.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 12 Juni 2024

Josua Adriel Sinabutar

LAMPIRAN

Source Code:
https://github.com/JosuaAdriel/Tugas-Akhir-Kriptografi

Video:
https://youtu.be/Iupb5wIhL0M