Analisis Performa Classic Cipher (Vigenere, Playfair, dan Autokey Vigenere Cipher) pada Enkripsi Pesan

Silvester Kresna Wicaksono Prayitno Putra (18221049)
Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
silvesterkresna@gmail.com

***Abstract*—Perkembangan algoritma kriptografi sangatlah pesat untuk mengikuti perkembangan teknologi dan kebutuhan akan keamanan data. Pendahulu dari algoritma kriptografi adalah *classic cipher*. Walaupun ketinggalan jaman dan cenderung tidak aman jika dibandingkan dengan algoritma baru, *classic cipher* adalah awal mula dan landasan dari ilmu kriptografi. Pada makalah ini akan diuji performa dari segi waktu enkrpsi dan dekripsi masing-masing algoritma. Kemudian, akan diuji juga nilai kerahasiaan menggunakan *secrecy value* yang didapat dari *chi-square test* antara distribusi frekuensi *ciphertext* dengan distribusi frekuensi alfabet dalam Bahasa Inggris. Ketiga algoritma yang akan diuji yaitu *Vigenere cipher, Autokey vigenere cipher,* dan *playfair cipher*.**

Keywords—classic cipher, vigenere cipher, playfair cipher, autokey vigenere cipher, cryptography performance

I. PENDAHULUAN

Di zaman dengan teknologi yang berkembang pesat, banyak sekali algoritma dalam kriptografi. Banyaknya teknik kriptografi tersebut dikarenakan banyaknya juga tuntutan keamanan dalam menyimpan atau mengirim data.

Namun demikian, sejarah kriptografi diawali dengan *classic cipher*. *Classic cipher* adalah kriptografi yang sudah ada sejak ribuan tahun lalu sampai ditemukannya komputer digital. Pada awalnya *classic cipher* digunakan secara konvensional menggunakan kertas dan pena. Seiring berkembangnya jaman, penggunaan *classic cipher* dapat diimplementasikan pada program komputer.

Jika dibandingkan dengan algoritma kriptogradi modern, cipher klasik tentunya kalah dalam segi performa dan kerahasiaan. Namun demikian, antar klasik cipher dapat juga diteliti segi keamanan dan kecepatan pada program komputer.

Oleh karena itu, makalah ini akan membahas uji performa dan kerahasiaan tiga *classic cipher*, termasuk *Vigenere Cipher, Playfair Cipher, dan Autokey Vigenere Cipher.*

II. LANDASAN TEORI

A. Classic Cipher

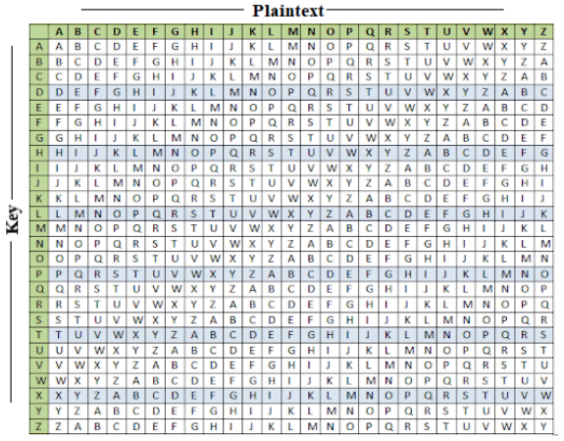
Classic cipher adalah kriptografi yang sudah ada sejak ribuan tahun lalu sampai ditemukannya komputer digital [1]. *Classic cipher* memproses pesan berbasis huruf alfabet (26 buat alfabet). Kriptografi ini dibuat menggunakan pena dan kertas.

Classic cipher terdiri dari 2 teknik utama, diantaranya:

- Teknik substitusi, mengganti huruf
- Teknik transposisi, mengubah susunan huruf

B. Vigenere Cipher

Vigenere cipher adalah *cipher* yang termasuk ke dalam *cipher* abjad-majemuk (*polyalpabetic substitution cipher*). Penggunaannya adalah menggunakan tabel Vigenere atau *Vigenere Square*.



source: [2]

Metode Enkripsi:

- Pilih kunci yang akan digunakan untuk enkripsi.
- Ulangi kunci sehingga panjangnya sama dengan panjang pesan asli.
- Untuk setiap huruf dalam pesan asli, temukan huruf di *Vigenere Table* yang berada pada baris yang sesuai dengan huruf plaintext dan kolom yang sesuai dengan huruf kunci.
- Huruf yang terletak pada pertemuan baris dan kolom tersebut menjadi huruf dalam *ciphertext*.

Metode Dekripsi:

- Gunakan kunci yang sama dengan yang digunakan untuk enkripsi.

- Ulangi kunci sehingga panjangnya sama dengan panjang *ciphertext*.
- Untuk setiap huruf dalam *ciphertext*, temukan baris dalam *Vigenere Table* yang dimulai dengan huruf kunci yang sesuai.
- Cari huruf *ciphertext* dalam baris tersebut dan temukan kolom yang sesuai.
- Huruf pada kolom tersebut adalah huruf *plaintext* yang didekripsi.

Tanpa menggunakan Vigenere Square pun enkripsi tetap dapat dihitung secara Caesar Cipher dengan menjumlahkan plainteks pj dengan kunci ki dalam modulus 26:

Enkripsi: cj = E(pj) = (pj + ki) mod 26 (1)

Dekripsi: pj = D(cj) = (cj – ki) mod 26 (2)

C. *Playfair Cipher*

Playfair cipher adalah salah satu algoritma kriptografi yang termasuk ke dalam substitusi poligrafik [3]. *Playfair* dilakukan dengan mengenkripsi pasangan huruf (bigram) dengan menggunakan *playfait matrix*.

Playfair matrix dibuat dari *key* dengan diproses terlebih dahulu. *Key* pertama dibuang huruf yang berulang kemudiah buang juga huruf J (tergantung ketentuan yang dipakai, ada yang membuang huruf I). Kemudian tambahkan huruf yang belum muncul dan masukan ke *matrix* secara terurut. Misalkan kita memiliki *key* yaitu puzzle, maka *matrix playfair* akan menjadi:

P	U	Z	L	E
A	B	C	D	F
G	H	I	K	M
N	O	Q	R	S
T	V	W	X	Y

source: [3]

Algoritma dimulai dengan menghapus huruf J pada *plaintext*, kemudian disusun menjadi bigram. Jika ada bigram yang terdiri dari 2 huruf yang sama, maka sisipkan huruf x di tengahnya. Hal yang sama terjadi ke kasus bigram yang hanya memiliki 1 huruf (sisa)

Misal:

- bigram awal: uu ai
- bigram hasil ux ua ix

Algoritma enkripsinya dan dekripsinya masing-masing secara garis besar ada 3 cara

- Jika dua huruf bigram terdapat pada baris kunci yang sama, maka tiap huruf diganti dengan huruf di kanannya (siklik)
- Jika dua huruf bigram terdapat pada kolom kunci yang sama, maka tiap huruf diganti dengan huruf di bawahnya (siklik)
- Jika dua huruf bigram tidak sebaris atau sekolom, maka huruf pertama diganti dengan perpotongan baris huruf pertama dengan kolom huruf kedua. Kemudian huruf kedua diganti dengan huruf pada titik sudut keempat yang belum dipakai ketika dibuat menjadi segi 4.

Algorita dekripsinya adalah kebalikannya.

D. *Autokey Vigenere Cipher*

Autokey Vigenere Cipher adalah salah satu variasi dari *Vigenere Cipher*. Algoritma ini memiliki perbedaan yaitu jika panjang kunci lebih kecil dari panjang *plaintext*, maka kunci akan disambung dengan *plaintext* [2]. Penggabungan dilakukan dengan menambahkan *plaintext* di akhir kunci.

Contoh:

- *Plaintext*: akukamudia
- *Key* pada Vigenere: sama
- *Key* pada *Autokey Vigenere*: samaakukam

E. *Performance Test (Encryption and Decryption Time)*

Pada kriptografi, performa identik dengan kecepatan waktu enkripsi dan dekripsi [4]. Semakin sedikit waktu yang digunakan untuk enkripsi dan dekripsi, maka semakin baik pula performa dari algoritma tersebut.

Waktu enkripsi dekripsi biasanya bergantung pada devais atau lingkungan tempat algoritma digunakan. Untuk memberikan hasil analisis yang baik, diperlukan perbandingan waktu enkripsi-dekripsi pada devais yang sama.

F. *Secrecy Test (Secrecy Value)*

Banyak cara melakukan *secrecy test*. Salah satu caranya adalah dengan menganalisis frekuensi kemunculan masing-masing alfabet (cipher klasik). Frekuensi kemunculan tersebut dapat memperlihatkan apakah hasilnya tersebar dengan baik ataukan tidak. Semakin tersebar, maka semakin baik.

Penggunaan huruf pada bahasa inggris memiliki frekuensinya masing-masing. *Secrecy value* pada makalah ini didapatkan dengan pertama mencari distribusi alfabet dari *ciphertext*, kemudian dilakukan chisquare.

Chi-square test merupakan uji nonparametrik yang digunakan untuk menguji hipotesis tidak adanya hubungan antara dua atau lebih kelompok, populasi atau kriteria dan untuk menguji kemungkinan distribusi data yang diamati sesuai dengan distribusi tersebut yang diharapkan (yaitu, untuk menguji kesesuaian) [5]. Hal ini digunakan untuk menganalisis data kategorikal (misalnya pasien pria atau wanita, perokok dan bukan perokok, dll.), ini tidak dimaksudkan untuk menganalisis parametrik atau data berkelanjutan (misalnya, tinggi badan diukur dalam sentimeter atau berat diukur dalam kg, dll.)

III. DESAIN DAN RANCANGAN SISTEM

Berikut ini adalah desain sistem dan rancangan penelitian yang akan dikembangkan dan digunakan untuk mencari performa dari masing-masing algoritma kriptografi

A. *Autokey Vigenere Cipher*

Berikut ini adalah program *autokey vigenere cipher* yang Penulis gunakan

```
// Algoritma enkripsi untuk form input plain text
const encryptAutokeyVigenereCipher = () => {
  if (!plainText || !key) {
    alert("Please input plain text and enter a key."); // Jika plain text
atau key kosong
    return;
  }
  const convertedPlainText = plainText
    .toUpperCase()
    .replace(/\s+/g, "")
    .replace(/[^A-Z]/g, "");
  const keyLen = key.replace(/\s+/g, "").length;
  const convertedKey = key.replace(/\s+/g, "").toUpperCase() +
convertedPlainText.substring(0, convertedPlainText.length - keyLen);

  let ciphertext = "";

  // Mulai waktu enkripsi
  const startTime = performance.now();

  for (let i = 0; i < convertedPlainText.length; i++) {
    const char = convertedPlainText.charAt(i);
    if (char.match(/[A-Z]/i)) {
      const plainTextCharCode = char.toUpperCase().charCodeAt(0) -
"A".charCodeAt(0);
      const keyCharCode = convertedKey[i %
convertedKey.length].toUpperCase().charCodeAt(0) - "A".charCodeAt(0);
      const encryptedCharCode = (plainTextCharCode + keyCharCode) % 26;
      const encryptedChar = String.fromCharCode(encryptedCharCode +
"A".charCodeAt(0));

      ciphertext += encryptedChar;
    }
  }

  // Selesai waktu enkripsi
  const endTime = performance.now();
  setEncryptionTime(endTime - startTime);

  // Relatif Freq
  setRelativeFrequencies(calculateRelativeFrequency(ciphertext));

  setCode("ci");
  setCipherText(ciphertext);
  setCipherTextBase64(Buffer.from(ciphertext).toString("base64"));
  setDecryptedText("");
  setDecryptedTextBase64("");
};

// Algoritma dekripsi untuk form input plain text
const decryptAutokeyVigenereCipher = () => {
  if (!plainText || !key) {
    alert("Please input plain text and enter a key."); // Jika plain text
atau key kosong
    return;
  }
  const convertedPlainText = plainText
    .toUpperCase()
    .replace(/\s+/g, "")
    .replace(/[^A-Z]/g, "");
  let convertedKey = key.replace(/\s+/g, "").toUpperCase();

  let decryptedText = "";

  // Mulai waktu dekripsi
  const startTime = performance.now();

  for (let i = 0; i < convertedPlainText.length; i++) {
    const char = convertedPlainText.charAt(i);
    if (char.match(/[A-Z]/i)) {
      const plainTextCharCode = char.toUpperCase().charCodeAt(0) -
"A".charCodeAt(0);
      const keyCharCode = convertedKey[i].toUpperCase().charCodeAt(0) -
"A".charCodeAt(0);
      const decryptedCharCode = (plainTextCharCode - keyCharCode + 26) % 26;
      const decryptedChar = String.fromCharCode(decryptedCharCode +
"A".charCodeAt(0));

      decryptedText += decryptedChar;
      convertedKey += decryptedChar;
    }
  }

  // Selesai waktu dekripsi
  const endTime = performance.now();
  setDecryptionTime(endTime - startTime);

  // Relatif Freq
  setRelativeFrequencies(calculateRelativeFrequency(decryptedText));

  setCode("deci");
  setDecryptedText(decryptedText);
  setDecryptedTextBase64(Buffer.from(decryptedText).toString("base64"));
  setCipherText("");
  setCipherTextBase64("");
};
```

B. *Playfair Cipher*

Berikut ini adalah program *autokey vigenere cipher* yang Penulis gunakan

```
const encryptPlayfairCipher = () => {
  let preparePlainText = plainText
    .toUpperCase()
    .replace(/J/g, "I")
    .replace(/[^A-Z]/g, "");
  const matrix = createPlayfairMatrix(key);
  let ciphertext = "";

  // Mulai waktu enkripsi
  const startTime = performance.now();

  for (let i = 0; i < preparePlainText.length; i += 2) {
    const char1 = preparePlainText[i];
    let char2 = i + 1 < preparePlainText.length ? preparePlainText[i + 1] :
"X";

    if (char1 === char2) {
      char2 = "X";
      preparePlainText = insertCharAt(preparePlainText, char2, i + 1);
      char2 = preparePlainText[i + 1];
    }

    const [row1, col1] = findCharInMatrix(matrix, char1);
    const [row2, col2] = findCharInMatrix(matrix, char2);

    let encryptedChar1, encryptedChar2;

    if (row1 === row2) {
      encryptedChar1 = matrix[row1][(col1 + 1) % 5];
      encryptedChar2 = matrix[row2][(col2 + 1) % 5];
    } else if (col1 === col2) {
      encryptedChar1 = matrix[(row1 + 1) % 5][col1];
      encryptedChar2 = matrix[(row2 + 1) % 5][col2];
    } else {
      encryptedChar1 = matrix[row1][col2];
      encryptedChar2 = matrix[row2][col1];
    }

    ciphertext += encryptedChar1 + encryptedChar2;
  }

  const endTime = performance.now();
  setEncryptionTime(endTime - startTime);

  setCode("ci");
  setCipherText(ciphertext);
  setCipherTextBase64(Buffer.from(ciphertext).toString("base64"));
  setDecryptedText("");
  setDecryptedTextBase64("");
};

const decryptPlayfairCipher = () => {
  let ciphertext = plainText.toUpperCase().replace(/[^A-Z]/g, "");
  let plaintext = "";

  // Mulai waktu dekripsi
  const startTime = performance.now();

  // Buat matriks Playfair Cipher berdasarkan kunci
  let matrix = createPlayfairMatrix(key);
  for (let i = 0; i < ciphertext.length; i += 2) {
    const char1 = ciphertext[i];
    let char2 = i + 1 < ciphertext.length ? ciphertext[i + 1] : "X";

    if (char1 === char2) {
      char2 = "X";
      ciphertext = insertCharAt(ciphertext, char2, i + 1);
      char2 = ciphertext[i + 1];
    }

    let decryptedChar1, decryptedChar2;

    // Temukan posisi kedua huruf dalam matriks
    const [row1, col1] = findCharInMatrix(matrix, char1);
    const [row2, col2] = findCharInMatrix(matrix, char2);

    if (row1 === row2) {
      // Jika kedua huruf berada pada baris yang sama, geser ke kiri
      decryptedChar1 = matrix[row1][(col1 - 1 + 5) % 5];
      decryptedChar2 = matrix[row2][(col2 - 1 + 5) % 5];
    } else if (col1 === col2) {
      // Jika kedua huruf berada pada kolom yang sama, geser ke atas
      decryptedChar1 = matrix[(row1 - 1 + 5) % 5][col1];
      decryptedChar2 = matrix[(row2 - 1 + 5) % 5][col2];
    } else {
      // Jika kedua huruf tidak berada pada baris atau kolom yang sama
      decryptedChar1 = matrix[row1][col2];
      decryptedChar2 = matrix[row2][col1];
    }
    plaintext += decryptedChar1 + decryptedChar2;
  }

  // Selesai waktu dekripsi
  const endTime = performance.now();
  setDecryptionTime(endTime - startTime);
```

```
    setCode("deci");
    setDecryptedText(plaintext);
    setDecryptedTextBase64(Buffer.from(plaintext).toString("base64"));
    setCipherText("");
    setCipherTextBase64("");
  };
```

C. *Vigenere Cipher*

Berikut ini adalah program *autokey vigenere cipher* yang Penulis gunakan

```
  const encryptVigenereCipher = () => {
    if (!plainText || !key) {
      alert("Please input plain text and enter a key."); // Jika plain text
atau key kosong
      return;
    }
    const convertedPlainText = plainText.replace(/\s+/g, "").toUpperCase();
    const convertedKey = key.replace(/\s+/g, "").toUpperCase();

    let ciphertext = "";

    // Mulai waktu enkripsi
    const startTime = performance.now();

    for (let i = 0; i < convertedPlainText.length; i++) {
      const char = convertedPlainText.charAt(i);
      if (char.match(/[A-Z]/i)) {
        const plainTextCharCode = char.toUpperCase().charCodeAt(0) -
"A".charCodeAt(0);
        const keyCharCode = convertedKey[i %
convertedKey.length].toUpperCase().charCodeAt(0) - "A".charCodeAt(0);
        const encryptedCharCode = (plainTextCharCode + keyCharCode) % 26;
        const encryptedChar = String.fromCharCode(encryptedCharCode +
"A".charCodeAt(0));

        ciphertext += encryptedChar;
      }
    }

    // Selesai waktu enkripsi
    const endTime = performance.now();
    setEncryptionTime(endTime - startTime);

    setCode("ci");
    setCipherText(ciphertext.toUpperCase());
    setCipherTextBase64(Buffer.from(ciphertext).toString("base64"));
    setDecryptedText("");
    setDecryptedTextBase64("");
  };

  // Algoritma dekripsi untuk form input plain text
  const decryptVigenereCipher = () => {
    if (!plainText || !key) {
      alert("Please input plain text and enter a key."); // Jika plain text
atau key kosong
      return;
    }
    const convertedPlainText = plainText.replace(/\s+/g, "").toUpperCase();
    const convertedKey = key.replace(/\s+/g, "").toUpperCase();

    let decryptedText = "";

    // Mulai waktu dekripsi
    const startTime = performance.now();

    for (let i = 0; i < convertedPlainText.length; i++) {
      const char = convertedPlainText.charAt(i);
      if (char.match(/[A-Z]/i)) {
        const plainTextCharCode = char.toUpperCase().charCodeAt(0) -
"A".charCodeAt(0);
        const keyCharCode = convertedKey[i %
convertedKey.length].toUpperCase().charCodeAt(0) - "A".charCodeAt(0);
        const decryptedCharCode = (plainTextCharCode - keyCharCode + 26) % 26;
        const decryptedChar = String.fromCharCode(decryptedCharCode +
"A".charCodeAt(0));

        decryptedText += decryptedChar;
      }
    }

    // Selesai waktu dekripsi
    const endTime = performance.now();
    setDecryptionTime(endTime - startTime);

    setCode("deci");
    setDecryptedText(decryptedText.toUpperCase());
    setDecryptedTextBase64(Buffer.from(decryptedText).toString("base64"));
    setCipherText("");
    setCipherTextBase64("");
  };
```

D. *Secrecy Value and Frequency Histogram*

Berikut ini adalah program *autokey vigenere cipher* yang Penulis gunakan

```
const calculateAlphabetDistribution = (text) => {
  const frequency = Array(26).fill(0);

  const cleanText = text.toUpperCase().replace(/[^A-Z]/g, "");

  cleanText.split("").forEach((char) => {
    const index = char.charCodeAt(0) - 65;
    if (index >= 0 && index < 26) {
      frequency[index]++;
    }
  });

  const total = cleanText.length;

  const relativeFrequency = frequency.map((count) => (count /
total) * 100);

  return relativeFrequency;
};

const calculateChiSquared = (observed, expected) => {
  return observed.reduce((chiSq, obsFreq, i) => {
    const expFreq = expected[i];
    return chiSq + Math.pow(obsFreq - expFreq, 2) / expFreq;
  }, 0);
};

const calculateSecrecyValue = (distribution) => {
  const englishFrequencies = [8.2, 1.5, 2.8, 4.3, 12.7, 2.2, 2.0,
6.1, 7.0, 0.2, 0.8, 4.0, 2.4, 6.7, 7.5, 1.9, 0.1, 6.0, 6.3, 9.1,
2.8, 1.0, 2.4, 0.2, 2.0, 0.1];

  const chiSquared = calculateChiSquared(distribution,
englishFrequencies);

  const secrecyValue = chiSquared / distribution.length;
  return secrecyValue.toFixed(2);
};
```

E. *Persiapan Pengukuran*

Sebelum melakukan penelitian, Penulis mempersiapkan *plain text* dan *key* yang akan digunakan dalam penelitian. Kedua aspek tersebut akan menjadi variabel bebas dalam penelitian ini. Oleh karena itu, masing-masing dipersiapkan empat .txt dengan panjang yang bervariasi.

Name	Date modified	Type	Size
500 Characters Text	11/06/2024 22:12	Text Document	1 KB
2500 Characters Text	11/06/2024 22:20	Text Document	3 KB
9000 Characters Text	11/06/2024 22:24	Text Document	9 KB
20000 Characters Text	11/06/2024 22:33	Text Document	21 KB
Key 64 Characters	12/06/2024 11:41	Text Document	1 KB
Key 128 Characters	12/06/2024 11:39	Text Document	1 KB
Key 256 Characters	12/06/2024 11:40	Text Document	1 KB
Key 512 Characters	12/06/2024 11:43	Text Document	1 KB

Seluruh .txt yang digunakan pada penelitian ini ada pada *folder* /public/Text File pada GitHub yang sudah dicantumkan *link*-nya.

F. Metode Pengukuran Perbandingan

Performa dihitung dengan membandingkan 3 nilai ukur berikut:

- *Encryption time*, waktu yang lebih rendah menandakan algoritma lebih baik dan efisien
- *Decryption time*, waktu yang lebih rendah menandakan algoritma lebih baik dan efisien
- *Secrecy value*, nilai yang lebih besar menandakan algoritma lebih baik

Performa diukur dengan melakukan percobaan sebanyak lima kali kemudian diukur rata-ratanya. Pengukuran dilakukan menggunakan laptop penulis yang memiliki spesifikasi sebagai berikut:

- CPU: Ryzen 7 4800H
- Jumlah Core: 8
- Jumlah Thread: 16
- Base Clock: 2.9 GHz
- Max Boost Clock: 4.2 GHz
- Kapasitas Memori: 16 GB (2 x 8 GB)
- Sistem Operasi: Windows 11

G. Implementasi Sistem

IV. HASIL PENELITIAN

Peneltiian dilakukan pada *website* yang dibangun oleh Penulis. *Website* dibuat menggunakan *framework* NextJS dan mayoritas bahasa pemrograman yang digunakan adalah Typescript. *Website* dapat diakses melalui *link* https://classic-cipher-performance.vercel.app/autokey-vigenere-cipher

Penelitian dilakukan dengan mengganti variabel secara bergantian. Variabel bebas yang dimiliki adalah panjang *plain text* dan panjang *key*.

A. Encryption Time vs Plain Text Length (Key 256 chars)

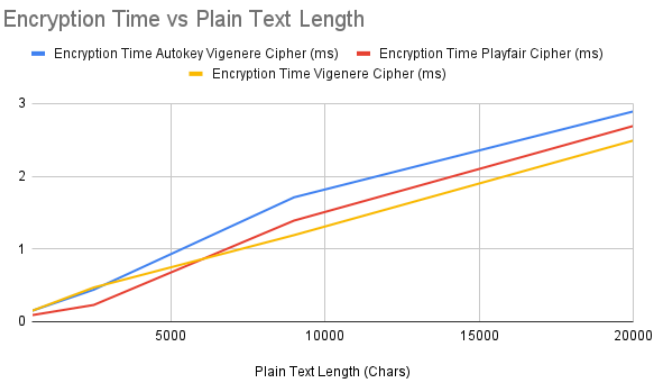
Berikut ini adalah tabel *Encryption Time* vs *Plain Text Length* dengan panjang *key* adalah 256 karakter.

TABLE 1. Encryption Time vs Plain Text Length

Plain Text Length (Chars)	Encryption Time Autokey Vigenere Cipher (ms)	Encryption Time Playfair Cipher (ms)	Encryption Time Vigenere Cipher (ms)
500	0,15	0,09	0,15
2500	0,44	0,23	0,47
9000	1,71	1,39	1,19
20000	2,89	2,69	2,49

Berikut ini adalah diagram garis perbandingan dari tabel diatas.

FIGURE 1. Encryption Time vs Plain Text Length



B. Decryption Time vs Plain Text Length (Key 256 chars)

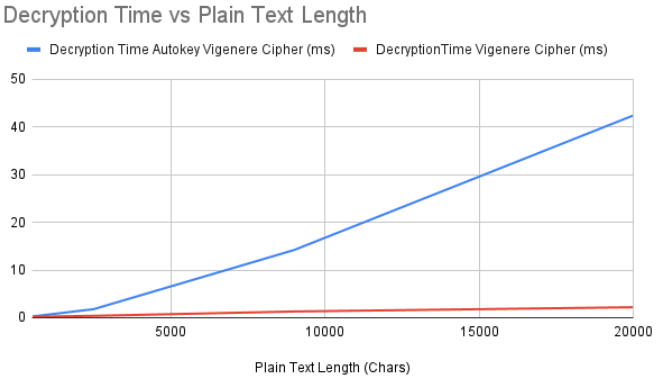
Berikut ini adalah tabel *Decryption Time* vs *Plain Text Length* dengan panjang *key* adalah 256 karakter.

TABLE 2. Decryption Time vs Plain Text Length

Plain Text Length (Chars)	Decryption Time Autokey Vigenere Cipher (ms)	Decryption Time Playfair Cipher (ms)	DecryptionTime Vigenere Cipher (ms)
500	0,25	-	0.20
2500	1,79	-	0,39
9000	14,19	-	1,30
20000	42,39	-	2,19

Berikut ini adalah diagram garis perbandingan dari tabel diatas.

FIGURE 2. Decryption Time vs Plain Text Length



C. Encryption Time vs Key Length (Plain Text 20000 chars)

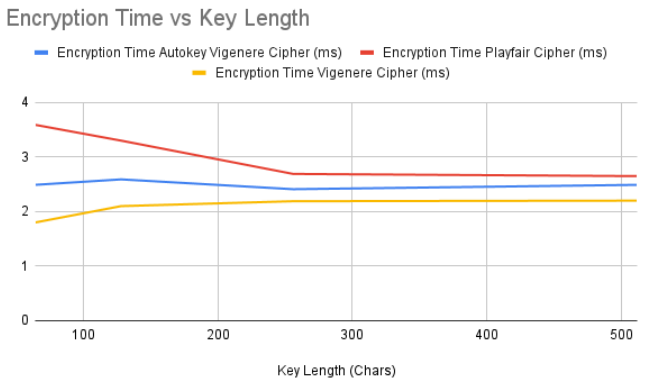
Berikut ini adalah tabel *Encryption Time* vs *Key Length* dengan panjang *plain text* adalah 20000 karakter.

TABLE 3. Encryption Time vs Key Length

Key Length (Chars)	Encryption Time Autokey Vigenere Cipher (ms)	Encryption Time Playfair Cipher (ms)	Encryption Time Vigenere Cipher (ms)
64	2,49	3,59	1,80
128	2,59	3,30	2,10
256	2,41	2,69	2,19
512	2,49	2,65	2,20

Berikut ini adalah diagram garis perbandingan dari tabel diatas.

FIGURE 3. Encryption Time vs Key Length



D. Decryption Time vs Key Length (Plain Text 20000 chars)

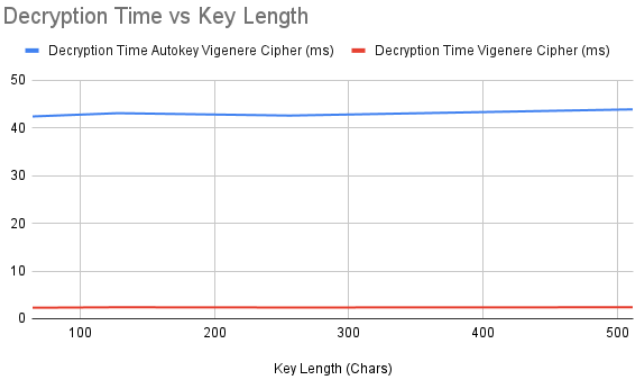
Berikut ini adalah tabel *Decryption Time* vs *Key Length* dengan panjang *plain text* adalah 256 karakter.

TABLE 4. Decryption Time vs Key Length

Key Length (Chars)	Decryption Time Autokey Vigenere Cipher (ms)	Decryption Time Playfair Cipher (ms)	Decryption Time Vigenere Cipher (ms)
64	42,40	-	2,30
128	43,09	-	2,40
256	42,59	-	2,35
512	43,89	-	2,40

Berikut ini adalah diagram garis perbandingan dari tabel diatas.

FIGURE 4. Decryption Time vs Key Length



E. Secrecy Value vs Plain Text Length (Key 256 characters)

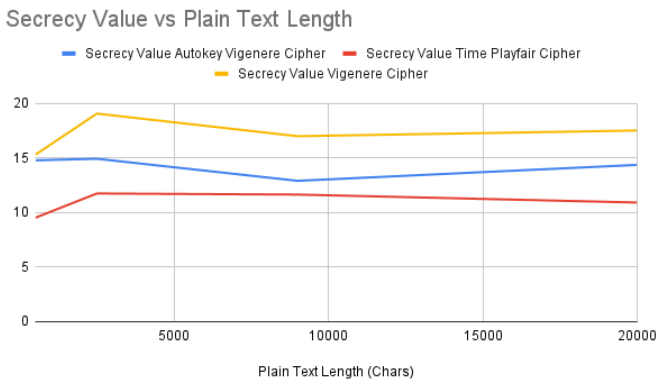
Berikut ini adalah tabel *Secrecy Value* vs *Plain Text Length* dengan panjang *key* adalah 256 karakter.

TABLE 5. Secrecy Value vs Plain Text Length

Plain Text Length (Chars)	Secrecy Value Autokey Vigenere Cipher	Secrecy Value Time Playfair Cipher	Secrecy Value Vigenere Cipher
500	14,79	9,52	15,29
2500	14,93	11,75	19,07
9000	12,91	11,65	17,00
20000	14,37	10,92	17,52

Berikut ini adalah diagram garis perbandingan dari tabel diatas.

FIGURE 5. Secrecy Value vs Plain Text Length



F. Secrecy Value vs Key Length (Plain Text 20000 chars)

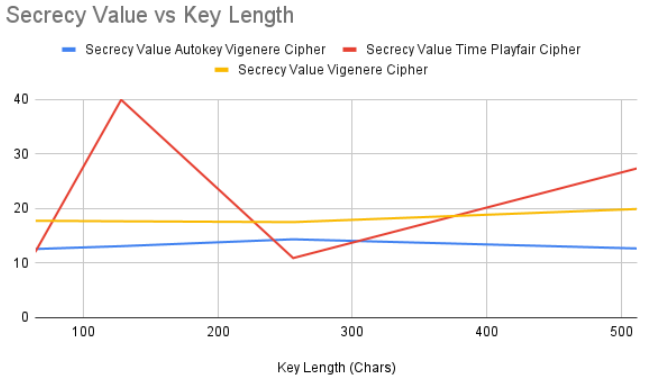
Berikut ini adalah tabel *Secrecy Value* vs *Key Length* dengan panjang *plain text* adalah 20000 karakter.

TABLE 6. Secrecy Value vs Key Length

Key Length (Chars)	Secrecy Value Autokey Vigenere Cipher	Secrecy Value Time Playfair Cipher	Secrecy Value Vigenere Cipher
64	12,59	12,02	17,77
128	13,11	39,94	17,67
256	14,37	10,93	17,52
512	12,69	27,37	19,92

Berikut ini adalah diagram garis perbandingan dari tabel diatas.

FIGURE 6. Secrecy Value vs Key Length



G. Hasl Analisis

Fig 1. menunjukan bahwa waktu enkripsi ketiga jenis algoritma berbanding lurus dengan panjang *plaintext*. Ketiganya memiliki waktu yang hampir miril. Algoritma Vigenere Cipher memiliki *overall* waktu yang paling rendah (efisien). Menurut penelitian, diantara ketiganya, metode Autokey yang paling lambat (tidak efektif)

Fig 2. menunjukan bahwa waktu dekripsi vigenere cipher jauh lebih lambat jika dibandingkan dengan vigenere cipher.

Fig 3. menunjukan bahwa waktu enkripsi dari ketiga algoritma tidak terlalu terpengaruh oleh panjangnya *key*. Waktu enkripsi cenderung konstan di angka tertentu dengan algoritma *vigenere cipher* yang paling cepat, diikuti dengan *autokey* dan *playfair*.

Fig 4. menunjukan bahwa waktu dekripsi keduanya cenderung konstan dengan waktu dekripsi *autokey vigenere* cipher jauh lebih lama jika dibandingkan dengan *vigenere cipher*.

Fig 5. menunjukan bahwa dari segi kerahasiaan, algoritma *vigenere cipher* memiliki *secrecy value* yang lebih tinggi dibanding dua lainnya. Nilai *secrecy* dari ketiga algoritma cenderung konstan dan tidak berbanding dengan panjang *plaintext*.

Fig 6. menunjukan bahwa algoritma *vigenere* dan *autokey vigenere* cenderung konstan dan tidak berbanding dengan panjang *key*. Di samping itu, nilai *secrecy Playfait cipher* sangat terpengaruhi oleh panjang *key* dan isi *key*.

V. KESIMPULAN

Dari penelitian, dapat disimpulkan bahwa dari segi performa, ketiga algoritma tidak memiliki algoritma yang menang di segala aspek. Namun demikian, dari waktu enkripsi dan dekripsi, *vigenere cipher* masing paling efisien karena kesederhanannya tersebut.

Di lain sisi, menurut kerahasiaan, *playfair cipher* memiliki kerahasiaan (dari *secrecy value*) yang lebih unggul dibanding dua lainnya. *Playfair cipher* sangat terpengaruhi oleh panjang dan nilai *key*. Untuk menjaga kerahasiaan dan keamanan, *playfair cipher* paling cocok untuk digunakan jika dibandingkan dengan dua algoritma lainnya.

Source Code Link at Github

Berikut ini adalah link dari *source code* makalah pada GitHub saya:

https://github.com/silvester-kw/classic-cipher-performance

Video Link at Youtube

Berikut ini adalah link dari video penjelasan makalah yang sudah saa unggah ke YouTube :

https://youtu.be/L0Qh6U4BpqY

Link Lainnya

Berikut ini adalah link dari *deployed website* yang saya buat:
https://classic-cipher-performance.vercel.app/autokey-vigenere-cipher

Berikut ini adalah link dari *spreadsheet* yang saya gunakan pada makalah ini:
https://docs.google.com/spreadsheets/d/1LCx8UqCrgjtnnHUdxH4Y9ArmkWWUag7CUXjcWrBBKXs/edit?usp=sharing

References

[1] R. Munir, "Ragam Cipher Klasik (Bagian 1),"," 2024. Accessed: Jun. 11, 2024. [Online]. Available:https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/2023-2024/02-Ragam-Cipher-Klasik-Bagian1-(2024).pdf

[2] R. Munir. "03 -Ragam Cipher Klasik (Bagian 2)," 2024. Accessed: Jun. 11, 2024. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/2023-2024/03-Ragam-Cipher-Klasik-Bagian2-(2024).pdf

[3] R. Deepthi. "A Survey Paper on Playfair Cipher and its Variants ". International Research Journal of Engineering and Technology (IRJET), 2024.

[4] Hossain, Md. Alam & Hossain, Md & Uddin, Md & Imtiaz, Shariar Md. (2016). Performance Analysis of Different Cryptography Algorithms. International Journal of Advanced Research in Computer Science and Software Engineering. 6.

[5] Singhal, Richa & Rana, Rakesh. (2015). Chi-square test and its application in hypothesis testing. Journal of the Practice of Cardiovascular Sciences. 1. 10.4103/2395-5414.157577.

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.

Jakarta, 12 Juni 2024

Silvester Kresna Wicaksono Prayitno Putra (18221049)