8-Series Cipher

Johanes[1], **Steve Andreas Immanuel**[2].

[1,2] Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika (STEI), Institut Teknologi Bandung (ITB), Jalan Ganesha 10, Bandung 40132
E-mail: 13517012@std.stei.itb.ac.id, 13517039@std.stei.itb.ac.id

Abstrak. Keamanan data adalah hal yang sangat penting khususnya di era digital sekarang. Pada kriptografi modern, dikenal algoritma *block cipher* yaitu algoritma enkripsi yang diterapkan pada tiap blok bit pada data. Pada makalah ini, diusulkan algoritma *block cipher* baru dinamakan 8-Series Cipher yang dapat beroperasi pada mode ECB, CBC, dan *Counter*. Gagasan yang mendasari algoritma ini adalah barisan bilangan yang memiliki pola tertentu dan bagaimana memanfaatkan barisan tersebut untuk melakukan pengacakan untuk memenuhi prinsip *confusion* dan *diffusion*. Dari hasil eksperimen, algoritma dapat menghasilkan 100% perbedaan *ciphertext* terhadap perubahan 1 bit di *plaintext* dan sekitar 99,5% perbedaan *ciphertext* terhadap perubahan 1 bit kunci.
Kata kunci: keamanan, kriptografi, *block cipher*, *confusion*, *diffusion*, barisan bilangan, *plaintext*, *ciphertext*

1. Pendahuluan

1.1 Latar Belakang

Berkembangnya teknologi digital yang pesat pada akhir ini membuat keamanan data menjadi hal yang sangat penting. Oleh karena itu, diperlukan suatu mekanisme untuk menjamin keamanan data yang dipertukarkan secara digital. Kriptografi adalah suatu teknik yang digunakan untuk melakukan komunikasi secara aman di lingkungan yang terdapat pihak ketiga atau pencuri informasi [1].

Konsep pengamanan data sudah ada sejak ribuan tahun yang lalu. Contoh kriptografi klasik yang terkenal adalah Caesar Cipher yang diciptakan oleh Julius Caesar untuk menyembunyikan informasi agar tidak diketahui oleh pihak lawannya. Kriptografi kemudian terus berkembang makin kompleks, contohnya seperti mesin Enigma yang dikembangkan Jerman pada saat perang dunia kedua. Sekarang, dengan bantuan teknologi komputer yang sudah jauh berkembang pesat, kriptografi sudah memasuki era baru yaitu kriptografi modern.

Hal kontras yang membedakan kriptografi modern dan klasik adalah pada kriptografi modern, semua operasi dilakukan pada level bit data sehingga akan membuat hasil enkripsi semakin abstrak dan sulit untuk dibongkar oleh kriptanalis. Operasi yang paling sering digunakan pada kriptografi modern adalah XOR terutama karena sifatnya yang simetris sehingga para pengembang algoritma enkripsi suka mengeksploitasi sifat tersebut untuk membuat rangkaian operasi yang kompleks.

Algoritma enkripsi pada kriptografi modern dibedakan menjadi *stream cipher* dan *block cipher*. Cara kerja *stream cipher* adalah mengenkripsi setiap bit dari *plaintext* dengan bit kunci yang didapatkan dari suatu *keystream* yang dapat di-*reproduce*. Sedangkan pada *block cipher*, bit-bit *plaintext* akan dibagi menjadi blok-blok dengan ukuran tertentu kemudian tiap blok akan dienkripsi dengan kunci tertentu.

Pada makalah ini, dirancang algoritma baru *block cipher* yang memanfaatkan konsep matematika yaitu deret bilangan dengan pola tertentu. Terdapat delapan jenis deret yang digunakan sehingga *block cipher* ini dinamakan 8-Series Cipher.

1.2 *Block Cipher* yang Sejenis

Terdapat banyak *block cipher* yang telah dikembangkan oleh kriptografer, beberapa di antaranya telah menjadi standar enkripsi yang sekarang digunakan di berbagai protokol komunikasi, pengamanan data, dan lain-lain. [3] Berikut adalah beberapa contohnya:

- *Data Encryption Standard* (DES)
  DES merupakan salah satu algoritma enkripsi kunci simetri yang dikembangkan oleh IBM pada tahun 1972. *Data Encryption Algorithm* (DEA) dibuat berdasarkan algoritma *Lucifer* milik Horst Feistel. DES menggunakan blok dan kunci eksternal berukuran 64 bit, namun pada kunci hanya 56 bit saja yang digunakan. Implementasinya menggunakan jaringan Feistel dengan 16 iterasi dan pada setiap iterasinya menggunakan kunci internal 48 bit yang berbeda. Setiap blok pesan mengalami permutasi awal dan invers permutasi di akhir enkripsi. Pada DES terdapat delapan buah matriks substitusi berukuran 4x16 yang mengubah 6 bit masukkan menjadi 4 bit keluaran.

- *Gosudarstvenny Standard* (GOST)
  Gosudarstvenny Standard yang berarti standar pemerintah dalam bahasa Rusia adalah enkripsi yang dikembangkan oleh Uni Soviet pada tahun 1970 sebagai tanggapan terhadap pemerintah Amerika yang membuat DES. GOST memiliki ukuran blok sebanyak 64 bit, ukurang kunci 256 bit, jumlah iterasi sebanyak 32 iterasi. Secara umum algoritmanya mirip dengan DES dengan menggunakan jaringan Feistel namun lebih sederhana pada proses pembangkitan kunci dan proses fungsi substitusinya. Pada GOST terdapat delapan buah kotak-S yang masing-masing sepanjang 16 yang mengubah 4 bit masukkan menjadi 4 bit keluaran.

- *Advanced Encryption Standard* (AES)
  AES merupakan standar baru yang dihasilkan dari lomba yang diadakan oleh *National Institute of Standards and Technology* (NIST) sebagai pengganti DES yang dianggap sudah tidak aman karena kuncinya dapat ditemukan dengan metode *brute force*. AES didasarkan pada algoritma *Rijndael* buatan Vincent Rijmen dan Joan Daemen dari Belgia. Algoritma *Rijndael* ditetapkan sebagai AES pada November 2001.

  Ukuran blok dan kunci pada AES dapat dipilih dari tiga mode AES, yaitu AES-128 dengan 4 *words* kunci, 4 *words* ukuran blok, dan 10 jumlah iterasi; AES-192 dengan 6 *words* kunci, 4 *words* ukuran blok, dan 12 jumlah iterasi; dan AES-256 dengan 8 *words* kunci, 4 *words* ukuran blok, dan 14 jumlah iterasi. Pada praktiknya hanya AES-128 dan AES-256 yang banyak digunakan. Algoritma AES tidak menggunakan struktur Feistel sehingga terdapat fungsi untuk enkripsi dan dekripsi. Pada AES hanya terdapat satu buah kotak-S dan inversnya dengan ukuran 16x16 yang mensubstitusi 1 byte masukkan menjadi 1 byte keluaran. Selain itu juga terdapat beberapa fungsi lain yang berbeda dengan DES dan GOST yaitu fungsi ShiftRows, MixColoumns, dan AddRoundKey.

2. Studi Pustaka

2.1 *Block Cipher*

Seperti yang telah disinggung pada Bab 1, *block cipher* adalah suatu algoritma untuk melakukan enkripsi dan dekripsi yang pemrosesannya dilakukan pada setiap blok pada *plaintext*. Masing-masing *block* memiliki panjang yang sama. Selain itu, *block cipher* juga menggunakan *symmetric key* yang berarti kunci yang sama akan digunakan untuk melakukan dekripsi dan enkripsi.

Keterkaitan antar blok dalam melakukan enkripsi dan dekripsi dapat didefinisikan dalam beberapa mode, di antaranya:

- Mode *Electronic Code Book* (ECB)

Pada mode ECB, setiap blok bersifat independen terhadap blok lainnya. Proses enkripsi dilakukan di tiap blok masing-masing dan hasilnya tidak akan mempengaruhi blok lainnya. Akibatnya, setiap blok *plaintext* yang sama akan dienkripsi menjadi *ciphertext* yang sama juga.

Kelebihan dari mode ini adalah proses enkripsi/ dekripsi dapat dilakukan secara paralel langsung dan kesalahan bit pada satu blok baik pada *plaintext* maupun *ciphertext* tidak akan berpengaruh ke blok-blok lainnya. Kelemahan dari mode ini adalah *plaintext* yang sama akan selalu menghasilkan *ciphertext* yang sama sehingga dapat dilakukan analisis frekuensi. Selain itu, dapat juga dilakukan manipulasi terhadap blok tertentu saja pada *ciphertext* sehingga jika didekripsi akan menghasilkan *plaintext* yang artinya berbeda dengan aslinya.

- Mode *Cipher Block Chaining* (CBC)

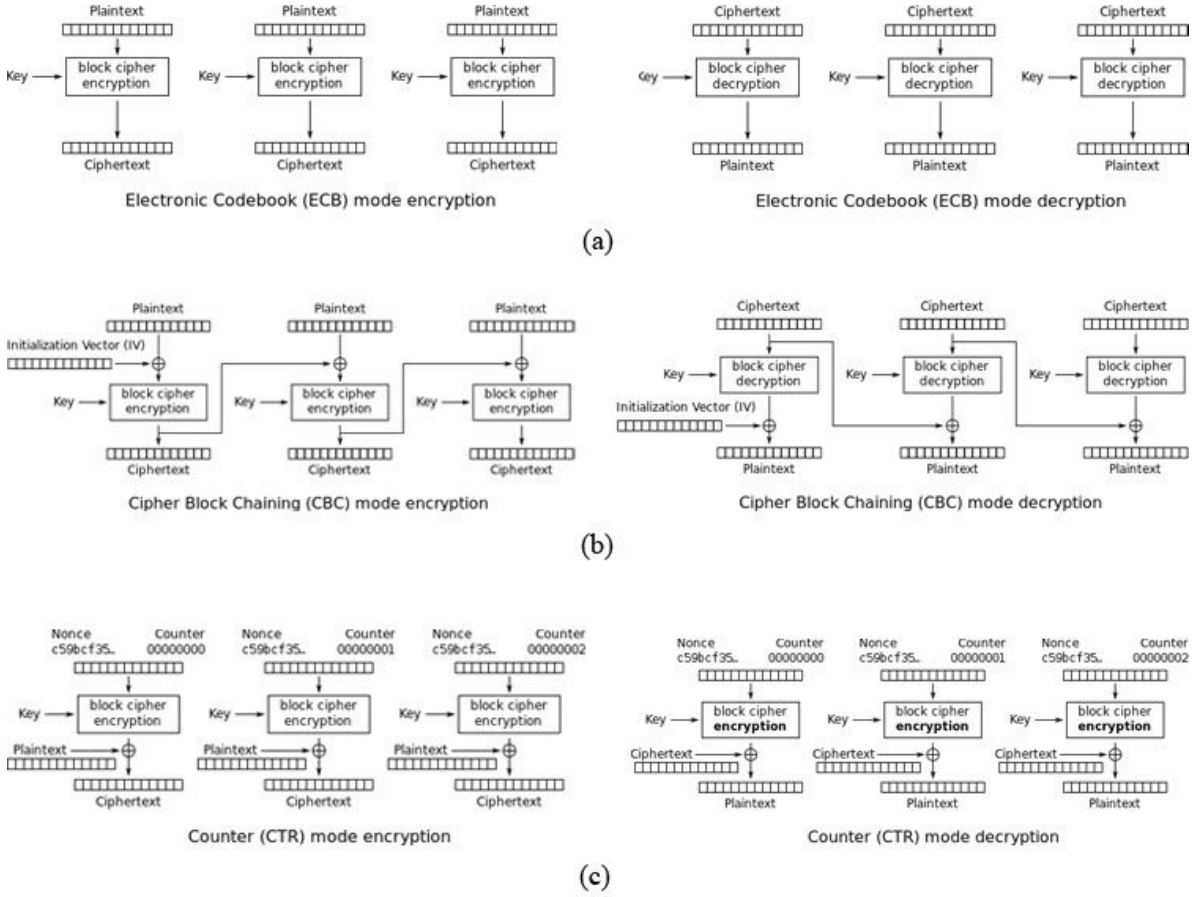
  Berbeda dengan mode ECB, pada mode CBC setiap blok memiliki ketergantungan satu dengan yang lain. Setiap blok *plaintext* akan dienkripsi dan hasilnya akan digunakan pada proses enkripsi blok berikutnya seperti yang tampak pada Gambar 1 (b). Pada blok pertama diperlukan suatu umpan untuk melakukan enkripsi yang dinamakan sebagai *initialization vector*.

  Kelebihan dari mode ini adalah blok-blok *plaintext* yang sama tidak akan selalu menghasilkan *ciphertext* yang sama dikarenakan ketergantungan antar blok sehingga akan sangat sulit untuk dilakukan analisis frekuensi. Kelemahan mode ini adalah pada saat enkripsi jika terjadi kesalahan pada blok *plaintext* maka kesalahan tersebut akan terus menjalar ke blok-blok sesudahnya hingga habis. Namun pada proses dekripsi, jika terjadi kesalahan di blok *ciphertext*, maka kesalahan tersebut hanya akan menjalar pada blok *plaintext* hasil dekripsi blok tersebut dan 1 blok *plaintext* sesudahnya saja.

- Mode *Counter*

  Pada mode *counter*, tidak terdapat dependensi antar blok seperti pada CBC. Sesuai namanya, terdapat suatu *counter* yang akan terus bertambah setiap melakukan enkripsi pada suatu blok. Setiap *counter* akan dienkripsi terlebih dahulu kemudian hasilnya akan dilakukan operasi XOR dengan blok *plaintext* dan langsung menghasilkan blok *ciphertext*.

  Mode ini menggabungkan kelebihan dari mode ECB dan CBC, dengan setiap blok *plaintext* yang sama belum tentu menghasilkan *ciphertext* yang sama (karena adanya *counter*), namun setiap blok tetap independen antara satu dengan yang lain. Hal penting dari mode ini adalah proses dekripsi dilakukan dengan cara memasukkan *counter* ke fungsi enkripsi (bukan dekripsi) lagi, kemudian hasilnya di-XOR dengan *ciphertext* dan akan langsung didapatkan *plaintext*. Namun kelemahannya adalah karena *ciphertext* didapatkan hanya dengan melakukan operasi XOR antara *plaintext* terhadap *counter* yang sudah dienkripsi, akibatnya perubahan 1 bit pada *plaintext* hanya akan mempengaruhi tepat 1 bit di *ciphertext* dan sebaliknya.



Gambar 1. Mode *block cipher* (a) ECB, (b) CBC, (c) *Counter*
Sumber: https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

2.2 Prinsip *Block Cipher*

Dalam membuat *block cipher*, terdapat beberapa prinsip yang perlu ditaati agar algoritma yang dihasilkan dapat kokoh dan aman. Berikut adalah prinsip tersebut:

- *Confusion* dan *Diffusion*
  Kedua prinsip ini diperkenalkan oleh Claude Shannon pada tahun 1945 di [2]. Keduanya saling berkaitan.

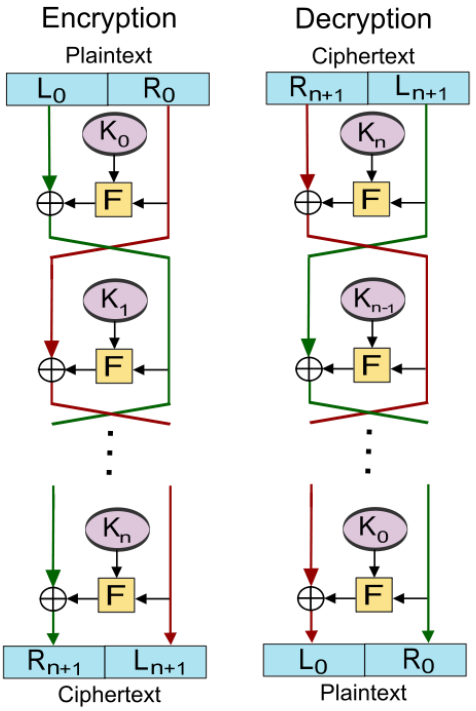
  Prinsip *confusion* mengatakan bahwa setiap bit pada *ciphertext* harus bergantung pada beberapa bagian dari *key* yang digunakan sehingga akan menyembunyikan hubungan antar keduanya. Dengan demikian jika salah satu bit pada *key* berubah, perubahan itu akan mengubah hampir atau semua bit pada *ciphertext*.

  Prinsip *diffusion* mengatakan bahwa pengaruh dari perubahan 1 bit pada *plaintext* harus terdifusi ke sebanyak mungkin bit pada *ciphertext*. Akibatnya, tidak mungkin dapat dilakukan analisis frekuensi terhadap *ciphertext* yang dihasilkan.

- *Iterated Cipher*
  Terdapat fungsi transformasi yang bersifat *invertible* yang dapat diterapkan pada blok berulang-ulang untuk mengubah *plaintext* menjadi *ciphertext*. Pada setiap iterasi akan digunakan suatu kunci putaran.

- Jaringan Feistel
  Jaringan Feistel adalah suatu struktur yang berbentuk simetris yang sering digunakan dalam membangun *block cipher*. Struktur yang simetris ini menyebabkan sifat *reversible* sehingga tidak diperlukannya algoritma yang berbeda untuk melakukan enkripsi dan dekripsi.

  Seperti yang terlihat pada Gambar 2, terdapat fungsi F yang sama untuk melakukan enkripsi dan dekripsi. Fungsi ini bersifat independen terhadap jumlah iterasi sehingga dapat dibuat sekompleks mungkin.



Gambar 2. Jaringan Feistel
Sumber: https://en.wikipedia.org/wiki/Feistel_cipher

- Kotak-S
  Kotak-S adalah suatu matriks yang digunakan untuk melakukan substitusi berupa pemetaan suatu m-bit masukan menjadi n-bit keluaran. Substitusi ini bersifat *non-linear* dan bertujuan untuk membuang hubungan antara *key* dan *ciphertext*.

2.3 Barisan dan Deret

Dalam matematika, barisan bilangan adalah suatu koleksi angka yang nilai angka tersebut akan terikat dengan aturan tertentu berdasarkan urutannya. Aturan pada suatu barisan dapat dipandang sebagai suatu pola berulang yang ada. Sebagai contoh, pada barisan bilangan kuadrat setiap angkanya merupakan kuadrat dari urutan kemunculan angka tersebut di barisan. Deret adalah penjumlahan dari setiap angka pada suatu barisan. Angka pada barisan biasa disebut dengan istilah suku. Biasanya, suku ke-n pada barisan dapat diformulasikan menggunakan rumus tertentu menggunakan n.

Konsep angka-angka yang berurutan dengan pola tertentu inilah yang menjadi ide dasar dari 8-Series Cipher ini. Terdapat delapan jenis barisan yang digunakan dalam pembangunan algoritma, yaitu:

- Barisan Fibonacci
  Definisi: Setiap sukunya adalah hasil penjumlahan dari dua suku sebelumnya.
  Pola: 1, 1, 2, 3, 5, 8, dst.
  Rumus: $U(n) = U(n-1) + U(n-2)$

- Barisan bilangan kuadrat
  Definisi: Setiap sukunya adalah kuadrat dari urutan kemunculannya.
  Pola: 1, 4, 9, 16, 25, 36, dst.
  Rumus: $U(n) = n^2$

- Barisan Lazy Caterer
  Definisi: Setiap sukunya adalah jumlah potongan maksimal yang mungkin dihasilkan dari lingkaran jika terdapat sejumlah garis lurus.
  Pola: 1, 2, 4, 7, 11, 16, 22, dst.
  Rumus: $U(n) = \frac{n^2+n+2}{2}$

- Barisan bilangan segitiga
  Definisi: Setiap sukunya adalah total titik yang ada dari segitiga dengan sisi sepanjang titik tertentu.
  Pola: 1, 3, 6, 10, 15, 21, dst.
  Rumus: $U(n) = \frac{n\times(n+1)}{2}$

- Barisan bilangan prima
  Definisi: Setiap sukunya adalah bilangan prima.
  Pola: 2, 3, 5, 7, 11, 13, dst.
  Rumus: -

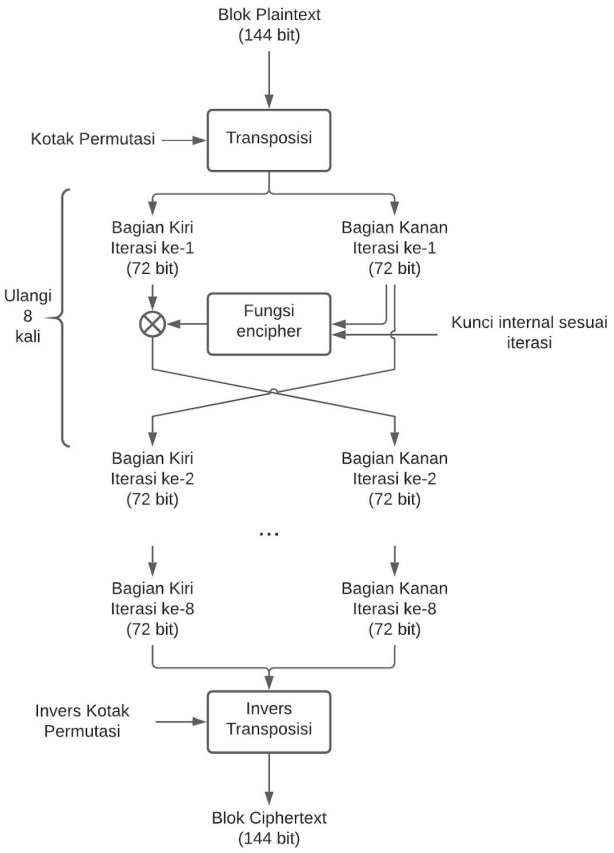
- Barisan bilangan segi lima
  Definisi: Setiap sukunya adalah total titik yang ada dari segi lima dengan sisi sepanjang titik tertentu.
  Pola: 1, 2, 5, 7, 12, 15, dst.
  Rumus: $U(n) = \frac{3n^2-n}{2}$

- Barisan bilangan segi enam
  Definisi: Setiap sukunya adalah total titik yang ada dari segi enam dengan sisi sepanjang titik tertentu.
  Pola: 1, 6, 15, 28, 45, 66, dst.
  Rumus: $U(n) = 2n^2 - n$

- Barisan Magic Square
  Definisi: Magic square adalah suatu kotak yang nilai penjumlahan tiap baris, kolom, dan diagonalnya sama. Pada barisan ini, tiap sukunya adalah nilai penjumlahan jika terdapat kotak berukuran tertentu.
  Pola: 15, 34, 65, 111, 175, 260, dst.
  Rumus: $U(n) = \frac{n\times(n^2+1)}{2}$

3. *Block Cipher* yang Dirancang

Pengembangan dari *block cipher* baru yang dibuat penulis terinspirasi dari barisan bilangan yang memiliki pola tertentu. Pada algoritma yang dirancang, digunakan delapan jenis barisan dengan pola yang berbeda untuk membangkitkan kotak permutasi, kotak-s, dan kunci internal. Ukuran blok yang digunakan adalah sebesar 144 bit dan panjang kunci yang digunakan dapat bervariasi dengan minimal panjang 144 bit. Alasan dipilihnya angka 144 ini akan dibahas di sub bab 3.3.



Gambar 3. Skema utama algoritma 8-Series Cipher

Terdapat dua ide dasarnya yang digunakan dalam pengembangan algoritma ini yaitu:

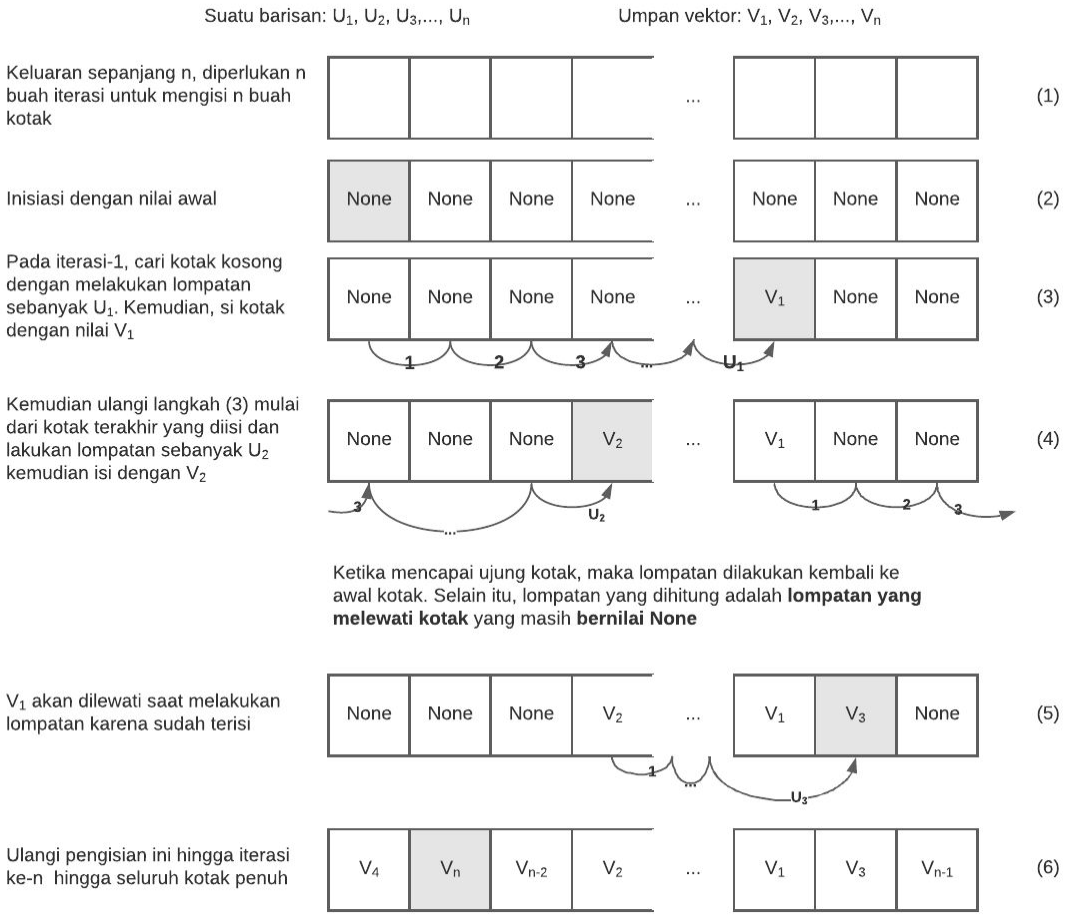
1. Bagaimana cara memanfaatkan suatu barisan bilangan untuk melakukan pengacakan suatu umpan vektor dengan panjang n
2. Bagaimana cara memanfaatkan suatu barisan bilangan untuk memetakan suatu vektor dengan panjang m ke suatu keluaran dengan panjang n.

Untuk menyelesaikan kedua persoalan di atas, penulis mengusulkan suatu algoritma baru yang diberi nama *mindful jumping* dan *careless jumping*. *Mindful jumping* digunakan untuk menyelesaikan persoalan pertama, sedangkan *careless jumping* digunakan untuk menyelesaikan persoalan kedua.

3.1 *Mindful Jumping*

Pada *mindful jumping*, dibutuhkan dua masukan yaitu suatu barisan dengan pola tertentu dengan n suku dan umpan vektor berukuran n. Keluaran dari proses ini adalah vektor baru berukuran n.

Proses ini terdiri atas n iterasi. Pertama, yang dilakukan adalah membuat vektor keluaran kosong berukuran n. Posisi kotak awal dimulai dari indeks pertama vektor keluaran. Pada suatu iterasi ke-n, dilakukan lompatan ke kanan dari posisi kotak sekarang sebanyak nilai suku ke-n dari barisan yang dipilih. Kemudian, tempat berhenti setelah melakukan lompatan tersebut pada vektor keluaran akan diisi dengan nilai dari elemen ke-n dari umpan vektor yang digunakan.

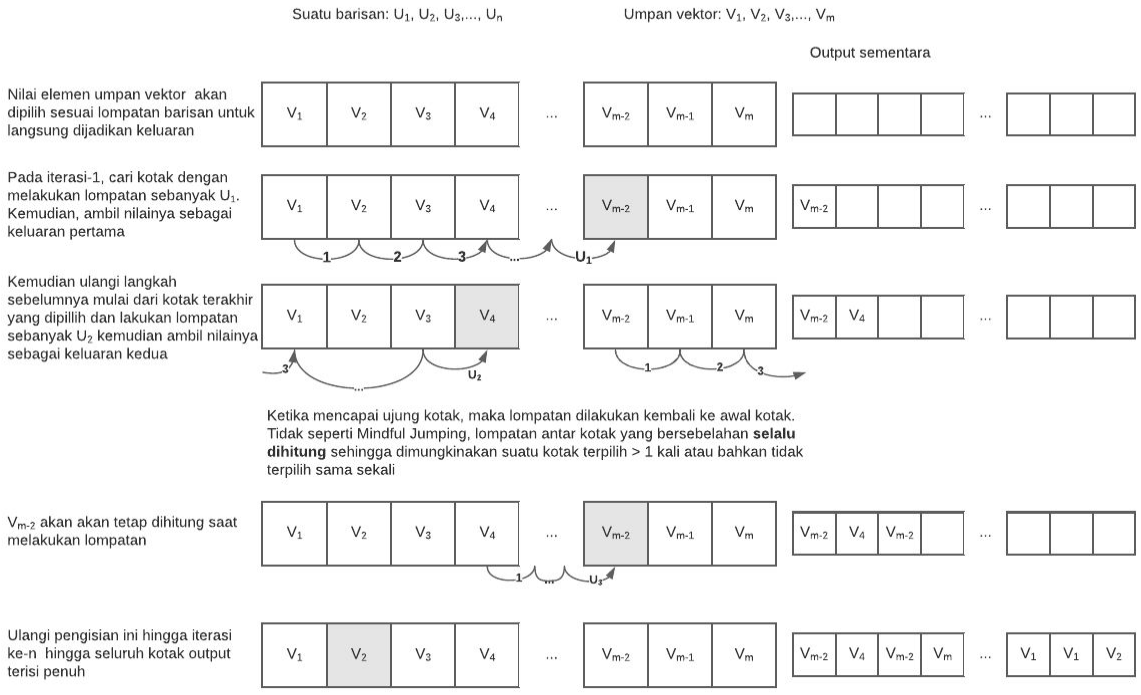


Gambar 4. Algoritma *mindful jumping*

Perlu diperhatikan bahwa lompatan yang dihitung adalah lompatan yang melewati kotak yang masih kosong seperti yang diperlihatkan pada Gambar 4 proses (5). Hal inilah alasan kenapa proses ini dinamakan *mindful jumping*. Selain itu, jika lompatan sudah mencapai ujung paling kanan dari kotak pada vektor keluaran, maka lompatan akan dilanjutkan dari kotak awal. Seperti pada Gambar 4 proses (6) proses pengisian ini akan diulangi hingga vektor keluaran penuh dan seluruh nilai dari umpan vektor telah diisikan ke dalam vektor keluaran. Sehingga keluaran yang dihasilkan adalah umpan vektor yang telah diacak menggunakan pola dari barisan yang digunakan.

3.2 *Careless Jumping*

Secara umum algoritma ini memilih nilai keluaran dari umpan vektor sebanyak jumlah suku dari suatu barisan. Untuk prosesnya dapat dilihat pada Gambar 5.



Gambar 5. Algoritma *careless jumping*

Pertama dibutuhkan sebuah umpan vektor yang akan diambil nilainya dan sebuah barisan yang akan digunakan sebagai nilai lompatan. Posisi awal akan dimulai dari nilai pertama dari umpan vektor. Kemudian dilakukan n buah iterasi untuk memilih nilai dari umpan vektor yang berukuran m nilai sehingga akan dihasilkan keluaran berukuran n nilai.

Pada setiap iterasi, akan dilakukan lompatan ke kanan sebanyak nilai suku ke-n dari barisan. Perlu diperhatikan, terdapat sedikit perbedaan dengan *mindful jumping*, pada *careless jumping* setiap kotak akan selalu dihitung sebagai lompatan meskipun sudah pernah terpilih sebelumnya. Oleh sebab itu algoritma ini disebut *careless jumping* karena memungkinkan suatu nilai menjadi tidak pernah terpilih ataupun menjadi sering terpilih. Nilai yang terpilih tersebut akan ditambahkan ke dalam vektor keluaran sebagai suku ke-n. Untuk memulai iterasi selanjutnya, posisi lompatan akan dimulai dari posisi suku terakhir yang diambil nilainya.

Proses iterasi ini akan diulang hingga didapatkan n buah nilai keluaran sesuai dengan jumlah suku pada barisan. Pada vektor keluaran dimungkinan terdapat pengulangan nilai dari umpan vektor dan dimungkinan suatu nilai dari umpan vektor tidak terpilih, mengingat ukuran umpan vektor adalah m yang berbeda dengan ukuran keluaran yang n.

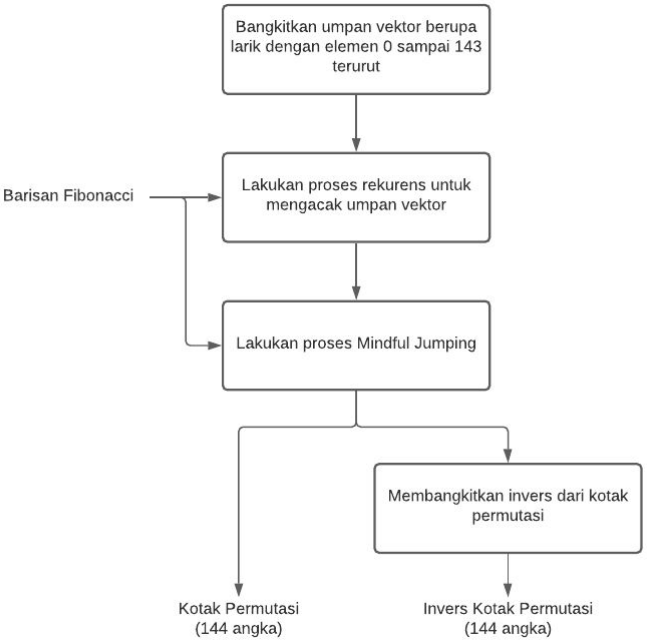
3.3 Pembangkitan Kotak Permutasi

Kotak permutasi dibuat untuk melakukan transposisi pada setiap blok *plaintext*. Kotak permutasi direpresentasikan sebagai vektor dengan panjang 144 yang masing-masing elemennya menyatakan indeks untuk melakukan transposisi terhadap tiap bit di blok *plaintext*. Untuk membangkitkan kotak permutasi, digunakan barisan Fibonacci.

Pertama perlu dibuat umpan vektor. Umpan vektor ini awalnya dibentuk dari larik dengan elemen 0 sampai 143 secara terurut sehingga memiliki panjang 144. 144 adalah suku ke-12 dari barisan fibonacci (jika dimulai dari angka 1). Memanfaatkan keunikan barisan Fibonacci, maka setiap suku di barisannya selalu bisa dinyatakan sebagai penjumlahan dari dua suku sebelumnya. Sebagai contoh:

$$144 = 89 + 55 \mid 55 = 34 + 21$$
$$89 = 55 + 34 \mid 34 = 21 + 13$$

Hal tersebut dapat dimanfaatkan untuk mengacak umpan vektor yang masih terurut tadi. Proses ini dilakukan secara rekursif. Pertama-tama umpan vektor akan dibagi dalam ukuran 89 dan 55 kemudian masing-masing bagian tersebut diurutkan secara terbalik. Masing-masing bagian ini kemudian akan dibagi lagi dan dilakukan proses yang sama. Bagian dengan panjang 89 akan dibagi menjadi 55 dan 34, kemudian bagian 55 akan dibagi menjadi 34 dan 21. Proses tersebut diulang hingga bagian vektor telah dibagi menjadi 2 dan 1. Maka akan didapatkan umpan vektor sepanjang 144 yang teracak namun masih memiliki sedikit pola.



Gambar 6. Skema pembangkitan kotak permutasi

Untuk lebih mengacak lagi umpan vektor yang sudah didapatkan tadi, maka dilakukan proses *mindful jumping* menggunakan barisan Fibonacci. Keluaran yang didapatkan adalah sebuah vektor dengan panjang 144 yang dipakai sebagai kotak permutasi. Selain itu, perlu dibangkitkan pula invers dari kotak permutasi untuk melakukan pembalikan transposisi yang dilakukan.

Proses transposisi dilakukan dengan menyusun ulang urutan bit pada *plaintext* sesuai dengan nilai indeks yang ada di kotak permutasi. Untuk melakukan invers, hanya perlu melakukan cara yang sama namun menggunakan invers kotak permutasi.

Tabel 1. Kotak Permutasi yang digunakan pada 8-Series Cipher

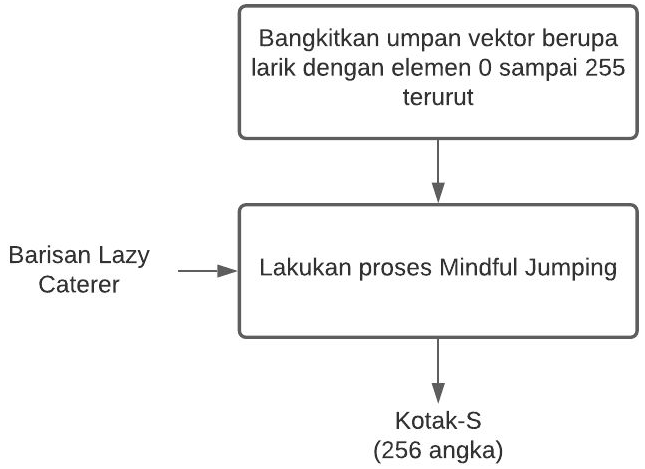
Kotak Permutasi
[64, 63, 104, 67, 96, 78, 65, 11, 128, 3, 123, 66, 81, 120, 71, 125, 85, 45, 90, 57, 46, 42, 51, 127, 137, 112, 122, 121, 68, 87, 129, 20, 55, 16, 119, 43, 110, 101, 95, 49, 89, 91, 94, 126, 99, 53, 111, 102, 54, 108, 40, 50, 48, 56, 80, 79, 83, 70, 75, 31, 26, 36, 114, 32, 8, 115, 124, 24, 44, 13, 69, 118, 7, 59, 74, 133, 100, 88, 23, 98, 38, 97, 138, 107, 29, 106, 17, 61, 92, 134, 109, 93, 10, 12, 0, 142, 58, 130, 37, 41, 30, 143, 84, 82, 34, 2, 76, 60, 103, 105, 5, 35, 15, 132, 136, 140, 9, 28, 39, 86, 141, 131, 139, 27, 73, 1, 22, 21, 47, 113, 18, 117, 77, 52, 19, 25, 14, 72, 4, 135, 116, 33, 62, 6]

Tabel 2. Kotak Invers Permutasi yang digunakan pada 8-Series Cipher

Kotak Invers Permutasi
[94, 125, 105, 9, 138, 110, 143, 72, 64, 116, 92, 7, 93, 69, 136, 112, 33, 86, 130, 134, 31, 127, 126, 78, 67, 135, 60, 123, 117, 84, 100, 59, 63, 141, 104, 111, 61, 98, 80, 118, 50, 99, 21, 35, 68, 17, 20, 128, 52, 39, 51, 22, 133, 45, 48, 32, 53, 19, 96, 73, 107, 87, 142, 1, 0, 6, 11, 3, 28, 70, 57, 14, 137, 124, 74, 58, 106, 132, 5, 55, 54, 12, 103, 56, 102, 16, 119, 29, 77, 40, 18, 41, 88, 91, 42, 38, 4, 81, 79, 44, 76, 37, 47, 108, 2, 109, 85, 83, 49, 90, 36, 46, 25, 129, 62, 65, 140, 131, 71, 34, 13, 27, 26, 10, 66, 15, 43, 23, 8, 30, 97, 121, 113, 75, 89, 139, 114, 24, 82, 122, 115, 120, 95, 101]

3.4 Pembangkitan Kotak-S

Cara pembangkitan kotak-S memanfaatkan algoritma *mindful jumping* untuk menghasilkan kotaknya. Hasil kotak-S yang digunakan merupakan larik satu dimensi berisi 256 nilai. Proses pembangkitan ini menerima vektor umpan berupa larik dengan elemen 0 sampai 255 secara terurut sehingga memiliki panjang 256. Kemudian, pada umpan vektor ini dilakukan algoritma *mindful jumping* dengan menggunakan barisan Lazy Caterer. Keluarannya merupakan sebuah larik satu dimensi dengan nilai dari 0 hingga 255 yang urutannya sudah diacak.



Gambar 7. Skema pembangkitan kotak-S

Cara penggunaan kotak-S ini adalah, setiap 8 bit dari pesan akan diinterpretasi sebagai sebuah bilangan. Bilangan tersebut digunakan sebagai indeks dari larik kotak-S ini. Sehingga setiap 8 bit akan disubstitusi dengan 8 bit baru dari kotak-S pada indeks bilangan tersebut. Pada algoritma 8-Series Cipher hanya terdapat satu kotak-S.

Tabel 3. Kotak-S yang digunakan pada 8-Series Cipher

Kotak-S
[0, 41, 1, 127, 190, 17, 2, 248, 159, 151, 214, 51, 186, 3, 202, 229, 66, 83, 115, 139, 132, 118, 188, 96, 4, 78, 93, 189, 34, 67, 251, 210, 108, 60, 130, 191, 219, 211, 89, 57, 5, 245, 181, 88, 195, 140, 64, 11, 233, 136, 54, 39, 37, 85, 168, 86, 91, 160, 117, 42, 119, 179, 6, 198, 182, 45, 99, 141, 153, 165, 154, 29, 223, 121, 122, 92, 166, 237, 131, 184, 142, 14, 242, 200, 81, 124, 30, 238, 44, 63, 107, 7, 28, 80, 176, 169, 212, 49, 147, 16, 170, 221, 25, 171, 199, 46, 213, 206, 143, 21, 33, 65, 216, 20, 156, 201, 125, 227, 148, 222, 48, 253, 215, 161, 209, 172, 128, 220, 8, 12, 174,

```
116, 22, 197, 69, 112, 129, 133, 239, 106, 135, 19, 146, 31, 246, 113, 111,
120, 208, 50, 27, 235, 100, 180, 68, 123, 254, 38, 247, 185, 244, 137, 144,
175, 192, 94, 249, 163, 134, 243, 98, 231, 43, 110, 9, 207, 71, 228, 23,
157, 252, 105, 224, 114, 164, 75, 73, 150, 203, 18, 177, 236, 250, 193, 77,
230, 187, 173, 59, 217, 84, 58, 52, 47, 90, 240, 15, 232, 101, 152, 61,
204, 87, 167, 56, 53, 183, 158, 149, 55, 104, 70, 13, 194, 109, 225, 82,
138, 62, 205, 10, 32, 255, 162, 196, 155, 26, 226, 102, 79, 234, 103, 241,
145, 35, 218, 97, 72, 95, 126, 74, 24, 36, 178, 40, 76]
```

Untuk contoh, misalkan bit pesannya adalah 0000 0010 yang dalam desimal merupakan angka 2. Pada kotak-S nilai pada indeks 2 adalah 1. Maka bit pesan tersebut akan diubah menjadi 0000 0001.

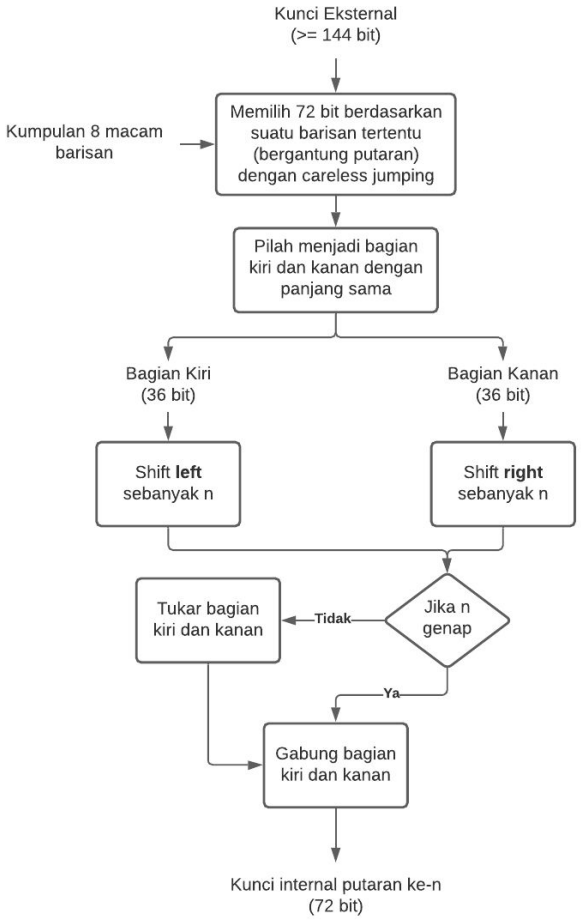
3.5 Pembangkitan Kunci Putaran

Pada 8-Series Cipher, dilakukan delapan kali putaran saat melakukan enkripsi sehingga dibutuhkan delapan kunci internal. Kunci internal dibangkitkan dengan memanfaatkan delapan jenis barisan berbeda di setiap putarannya.

Tabel 4. Jenis barisan yang digunakan di tiap putaran pembangkitan kunci internal

Putaran	Jenis Barisan
1	Fibonacci
2	Lazy Caterer
3	Bilangan kuadrat
4	Bilangan prima
5	Bilangan segitiga
6	Magic Square
7	Bilangan segi enam
8	Bilangan segi lima

Hal yang perlu diperhatikan adalah setiap kunci putaran dibangkitkan dari kunci eksternal dan bukan dari kunci putaran sebelumnya. Proses pembangkitan ini memanfaatkan algoritma *careless jumping* dengan menggunakan delapan barisan berbeda seperti yang tampak pada Tabel 4. Pada setiap akhir iterasi, posisi nilai terakhir yang digunakan akan menjadi posisi awal pada iterasi selanjutnya untuk melakukan *careless jumping*. Skema lebih jelasnya dapat dilihat pada Gambar 8. Di akhir proses akan didapatkan delapan buah kunci internal yang masing-masing memiliki panjang 72-bit.

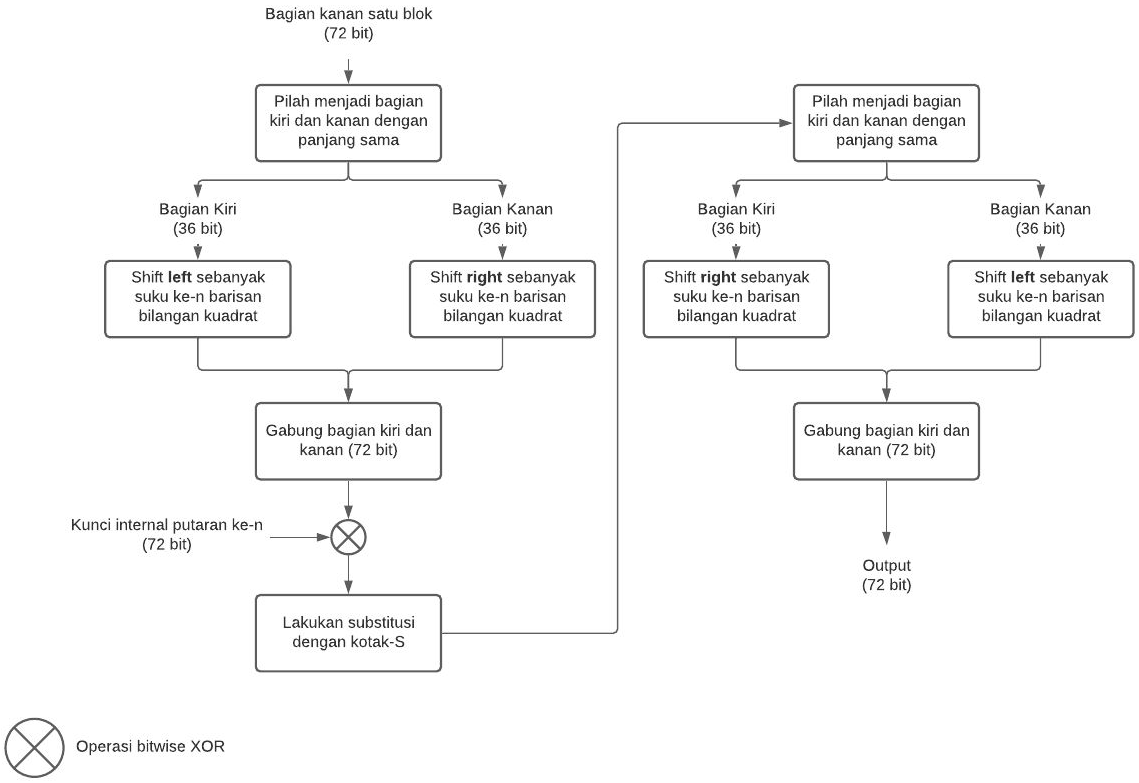


Gambar 8. Skema pembangkitan kunci putaran pada iterasi ke-n

3.6 Fungsi *Encipher*

Fungsi *encipher* akan melakukan enkripsi terhadap setengah blok pesan. Masukkan berupa blok dengan ukuran 72 bit (setengah dari 144 bit). Pada proses enkripsi dibutuhkan kunci internal setiap iterasi, blok pesan, dan kotak-S. Kotak-S yang digunakan selalu sama pada setiap iterasi sedangkan kunci internalnya berbeda untuk setiap iterasi.

Pertama blok pesan akan dibagi dua menjadi blok kiri dan kanan yang masing-masing berukuran 36 bit. Kemudian kedua blok ini akan dilakukan *bitwise shift* menjauhi tengah sebanyak nilai iterasi, blok kiri akan di-*shift* ke kiri sedangkan blok kanan akan di-*shift* ke kanan. Untuk contoh, pada iterasi pertama akan dilakukan *shift* sebesar 1 bit sedangkan pada iterasi ke tujuh akan dilakukan *shift* sebanyak 7 bit. Setelah itu kedua blok tersebut digabungkan untuk dilakukan operasi XOR terhadap kunci internal (72 bit) pada iterasi tersebut. Hasil XOR tersebut akan dilakukan substitusi menggunakan kotak-S dengan algoritma seperti yang dijelaskan pada bagian 3.4. Setelah dilakukan substitusi, blok pesan akan dibagi kembali menjadi dua bagian, kiri dan kanan, untuk dilakukan *bitwise shift* ke tengah sebanyak nilai iterasi. Blok kiri akan di-*shift* ke kanan sedangkan blok kanan akan di-*shift* ke kiri. Hasil *shift* tersebut akan digabungkan dan akan menjadi keluaran (72 bit) dari fungsi enkripsi.



Gambar 9. Skema proses *encipher* pada iterasi ke-n

4. Eksperimen dan Analisis

8-Series Cipher ini diimplementasikan menggunakan bahasa pemrograman Python. Mode operasi yang didukung adalah mode ECB, CBC, dan Counter.

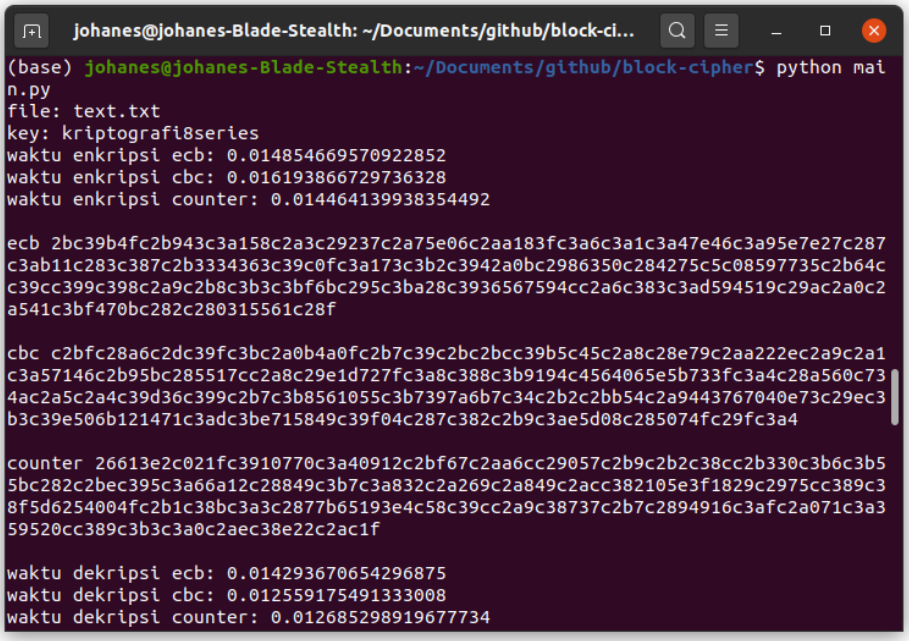
4.1 Eksperimen

Berikut ini adalah contoh hasil enkripsi dan dekripsi menggunakan tiga mode operasi tersebut.

Tabel 5. Hasil enkripsi dan dekripsi menggunakan tiga mode dan waktu eksekusinya. Kunci yang digunakan untuk enkripsi adalah '**kriptografi8series**'. Untuk tampilan aplikasinya dapat dilihat pada Gambar 10.

	Plain Text	ini adalah contoh teks sederhana yang akan digunakan untuk enkripsi dan dekripsi
ECB	Hasil Enkripsi (HEX) dalam 14.85ms	2bc39b4fc2b943c3a158c2a3c29237c2a75e06c2aa183fc3a6c3a1c3a47e46c3a95e7e27c287c3ab11c283c387c2b3334363c39c0fc3a173c3b2c3942a0bc2986350c284275c5c08597735c2b64cc39cc399c398c2a9c2b8c3b3c3bf6bc295c3ba28c3936567594cc2a6c383c3ad594519c29ac2a0c2a541c3bf470bc282c280315561c28f
	Hasil Dekripsi dalam 14.29ms	ini adalah contoh teks sederhana yang akan digunakan untuk enkripsi dan dekripsi

CBC	Hasil Enkripsi (HEX) dalam 16.19ms	694ac2a5c29ac39e59c291c29d2739c3a049c38b1c31c3963111c38a1a0c3b60c3a273c2a4c29a4521c2bac2a1c3b9c298c3aec2963ac29fc28e54c3a8c3b160c2afc2aac28fc38815c3a872c3bd1664095dc397453c1c76572861130106c2a2c2a9c3afc38f09043cc2a4c284406dc28cc29dc383c2ab22c3965fc3ab2bc2a1c38ec287c3a9383f2f6534c2bec39d1dc2bac381c288c290c2a7c2afc2bbc3ac2271c29d
	Hasil Dekripsi dalam 12.56ms	ini adalah contoh teks sederhana yang akan digunakan untuk enkripsi dan dekripsi
Counter	Hasil Enkripsi (HEX) dalam 14.46ms	26613e2c021fc3910770c3a40912c2bf67c2aa6cc29057c2b9c2b2c38cc2b330c3b6c3b55bc282c2bec395c3a66a12c28849c3b7c3a832c2a269c2a849c2acc382105e3f1829c2975cc389c38f5d6254004fc2b1c38bc3a3c2877b65193e4c58c39cc2a9c38737c2b7c2894916c3afc2a071c3a359520cc389c3b3c3a0c2aec38e22c2ac1f
	Hasil Dekripsi dalam 12.69ms	ini adalah contoh teks sederhana yang akan digunakan untuk enkripsi dan dekripsi



Gambar 10. Tampilan program CLI aplikasi yang menerima nama *file* dan kunci kemudian melakukan enkripsi dan dekripsi isi *file* tersebut dengan menggunakan ketiga mode enkripsi.

Tabel 6. Waktu enkripsi dan dekripsi dengan beberapa panjang kunci dan ukuran *plaintext*. Waktu didapatkan dari rata-rata tiga percobaan untuk masing-masing konfigurasi.

Panjang Kunci (terurut A-Z)	Ukuran *Plaintext* (Lorem ipsum/ byte)	ECB		CBC		Counter	
		Waktu Enkripsi (ms)	Waktu Dekripsi (ms)	Waktu Enkripsi (ms)	Waktu Dekripsi (ms)	Waktu Enkripsi (ms)	Waktu Dekripsi (ms)
18	10/ 82	7.43	4.61	6.99	5.03	5.74	5.68
18	50/ 369	35.97	16.97	19.33	15.76	17.00	16.89
18	500/ 3669	172.72	144.20	138.81	125.09	141.07	171.17
18	10.000/ 76.091	3,076.15	2,669.21	2,666.07	3,776.07	2,761.40	3,936.97
26	10/ 82	7.32	4.44	6.14	4.65	5.67	4.69
26	50/ 369	46.04	15.64	22.97	21.53	16.64	15.85
26	500/ 3669	230.37	165.85	138.48	157.81	170.21	142.47
26	10.000/ 76.091	2,651.06	2,643.34	2,694.45	3,308.74	2,615.07	2,803.39

Berdasarkan Tabel 6, dapat dilihat bahwa implementasi saat ini hanya cepat untuk ukuran pesan yang kecil, semakin besar ukuran pesan maka semakin lama pula waktu yang dibutuhkan. Meskipun lebih lama namun pertambahan waktunya tidak linear terhadap ukurang pesan. Selain itu dapat dilihat bahwa panjang kunci tidak menjadi faktor penentu waktu eksekusi melainkan hanya bergantung pada panjang pesan. Algoritma yang saat ini digunakan masih kurang efisien karena masih terjadi konversi antara *list of int* dan *list of bits* dan operasi aritmatik yang belum efisien.

4.2 Analisis Keamanan

4.2.1 Analisis *Diffusion* dan *Confusion*

Pada 8-Series Cipher prinsip *diffusion* diterapkan dengan melakukan permutasi saat awal dan akhir eksekusi dan juga pada proses pembangkitan urutan kunci internal. Sedangkan untuk penerapan prinsip *confusion* dilakukan substitusi dengan kotak-S sesuai dengan Tabel 3. Berikut ini adalah hasil eksperimen terhadap prinsip *diffusion* dan *confusion*. Eksperimen dilakukan terhadap data teks 76.091 *byte* berisi teks Lorem ipsum 10.000 dengan menggunakan kunci **kriptografi8series**.

Tabel 7. Persentase perbedaan jika dilakukan perubahan 1 bit atau 1 byte pada kunci

Perbedaan Kunci	Persentase Perbedaan		
	ECB	CBC	Counter
kri**q**tografi8series	99.30%	99.58%	99.23%
kriptografi**9**series	99.53%	99.59%	99.38%
kriptogr**=**fi8series	99.57%	99.59%	99.64%
kriptografi**!**series	99.58%	99.59%	99.61%
rata-rata	99.50%	99.59%	99.47%

Tabel 8. Persentase perbedaan jika dilakukan perubahan 1 bit atau 1 byte pada *plaintext*

Perbedaan Kunci	Persentase Perbedaan (dalam blok)		
	ECB	CBC	Counter
Lorem i**q**sum dolor	100.00%	100.00%	0.06%
Morem ipsum dolor	100.00%	100.00%	0.06%
Aorem ipsum dolor	100.00%	100.00%	0.06%
Lorem **!**psum dolor	100.00%	100.00%	0.06%
rata-rata	100.00%	100.00%	0.06%

Berdasarkan dua tabel di atas dapat dilihat bahwa perubahan 1 bit maupun 1 *byte* pada *plaintext* maupun pada kunci sangat berpengaruh terhadap hasil *ciphertext*-nya. Pada tabel di atas, dua baris pertama merupakan percobaan merubah 1 bit sedangkan dua baris terakhir merupakan percobaan perubahan *byte*. Berdasarkan tabel tersebut, persentase perubahannya >= 99.47%. Pada Tabel 8 eksperimen yang dilakukan hanya pada level satu blok saja karena untuk ECB dan Counter, perubahan hanya bersifat lokal sehingga jika dihitung untuk seluruh *plaintext*, maka nilai perubahannya akan kecil. Nilai persentase perubahan yang kecil pada mode *Counter* disebabkan oleh cara kerja mode *Counter* itu sendiri. Karena yang dilakukan enkripsi adalah sebuah nilai counter yang sama antara data *plaintext* yang diubah maupun tidak, maka hasil enkripsinya akan sama. Kemudian hasil enkripsi tersebut di XOR dengan *plaintext* yang hanya berubah 1 bit, maka persentase perubahannya akan sangat kecil. Dengan demikian, prinsip *diffusion* dan *confusion* telah dipenuhi.

4.2.2 Analisis Serangan *Brute Force*

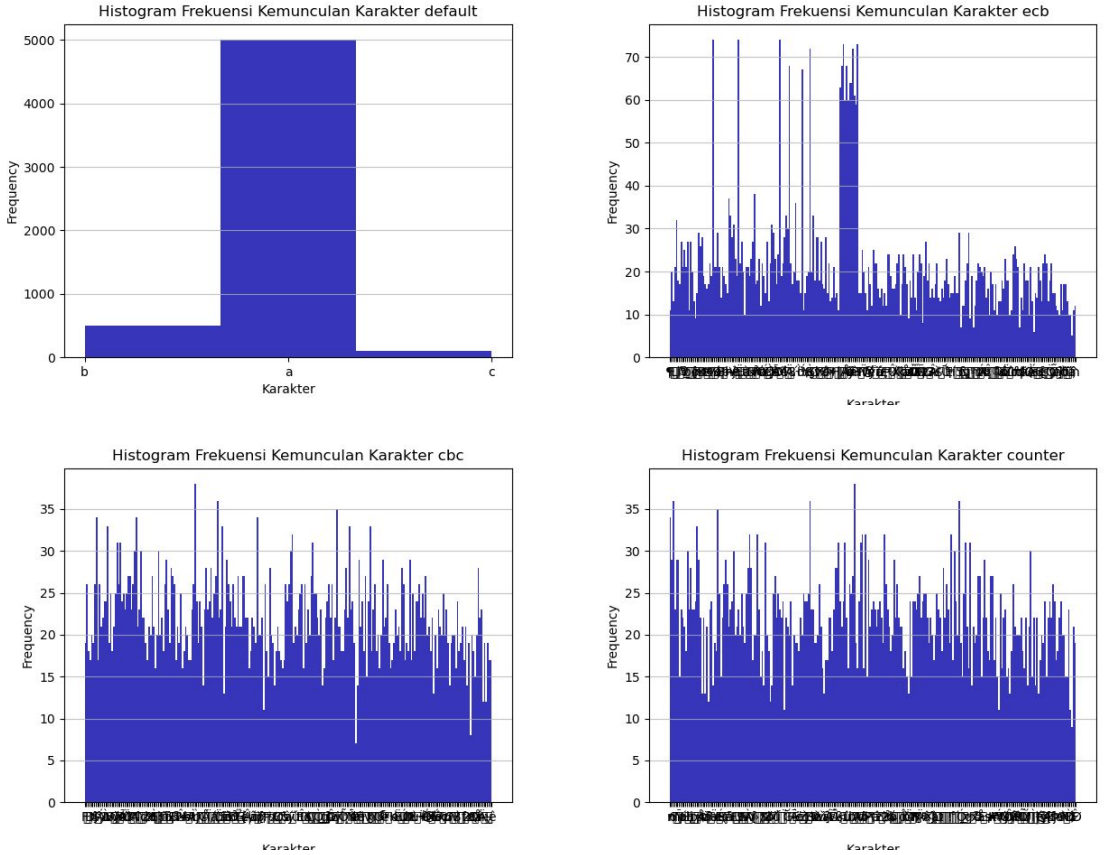
Pada 8-Series Cipher digunakan panjang kunci yang bervariasi dengan minimal sepanjang 144 bit atau 18 *byte*. Untuk penyerangan dengan *brute force* maka kombinasi kunci yang mungkin adalah sebanyak

$$2^{144} \approx 2.230075 \times 10^{43}$$

Misalkan dilakukan 1 juta operasi per detik, maka dibutuhkan $2.230075 \times 10^{37}\, detik = 7.066144 \times 10^{29}\, tahun$. Atau jika terdapat 1 juta operasi per milidetik, maka dibutuhkan $7.066144 \times 10^{26}\, tahun$. Karena kunci bisa lebih panjang dari 144 bit maka lama waktu akan bertambah secara eksponensial hanya dengan ditambahnya 1 bit. Dengan demikian, 8-Series Cipher sudah cukup aman dari serangan *brute force*.

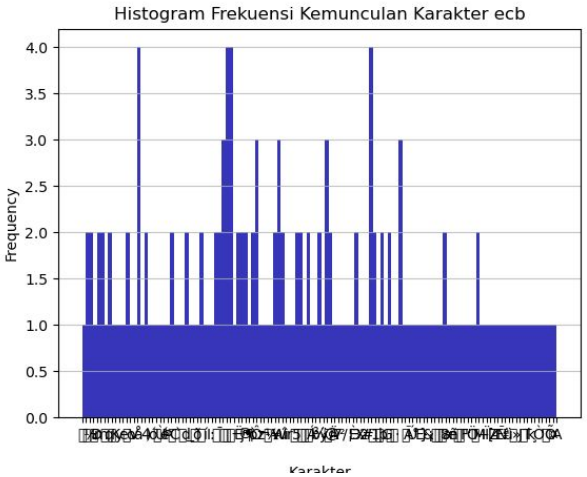
4.2.3 Analisis Statistik

Untuk melakukan analisis statistik ini, dilakukan enkripsi terhadap data yang distribusi karakternya tidak seimbang. *Plaintext* yang digunakan adalah suatu teks dengan huruf a, b, dan c yang terdistribusi tidak merata sebagai berikut.

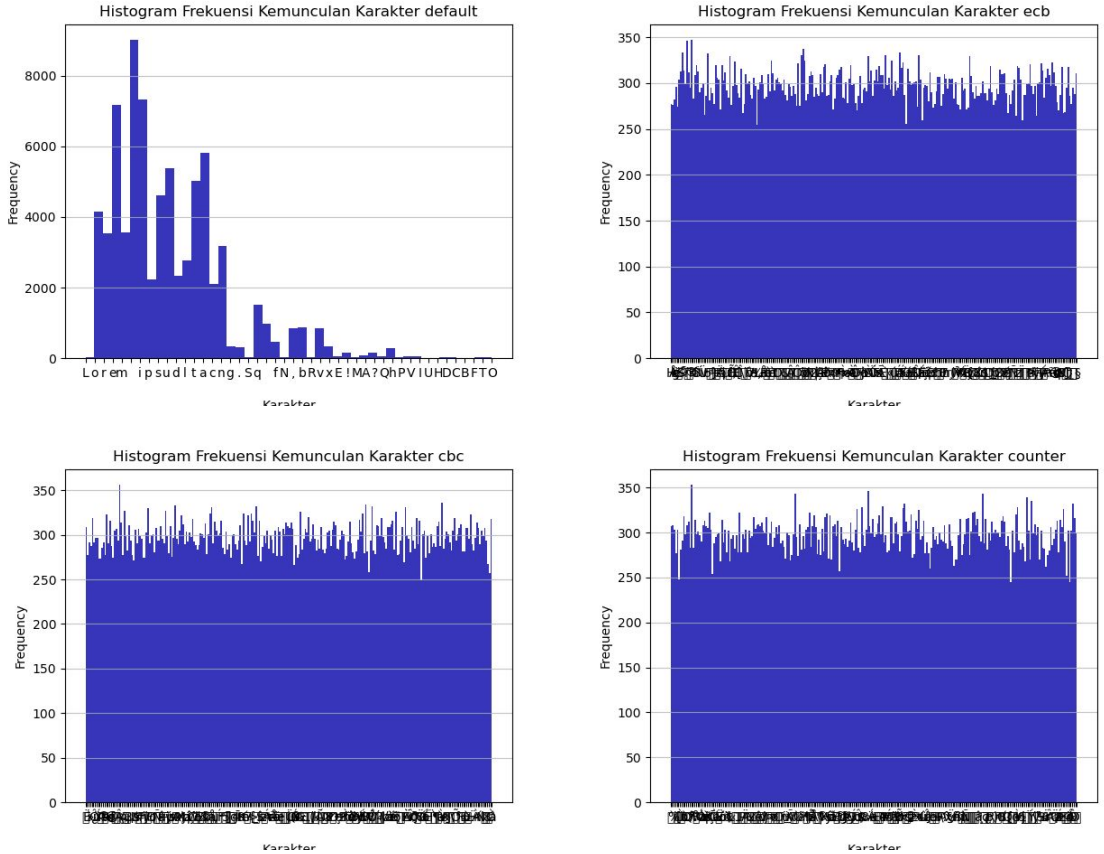


Gambar 11. Hasil histogram dari *plaintext* a, b, c yang terdistribusi tidak merata. Dari pojok kiri atas ke kanan bawah, (1) histogram *plaintext*, (2) histogram ECB, (3) histogram CBC, dan (4) histogram Counter

Berdasarkan grafik histogram tersebut dapat disimpulkan, distribusi karakter pada *ciphertext* sudah baik, hal ini dapat dilihat dari histogram hasil enkripsi yang mengandung 256 karakter ASCII yang hanya berasal dari tiga karakter saja. Kemudian untuk mode CBC dan *Counter*, hasil histogramnya terlihat sudah tersebar merata. Sedangkan untuk mode ECB, hasilnya terlihat berfokus pada beberapa karakter saja, hal ini dikarenakan ECB mengenkripsi secara blok dan karena jumlah karakter a sangat banyak pada *plaintext*, kemungkinan besar banyak blok-blok yang hanya berisi karakter a dan selalu dienkripsi menjadi *ciphertext* yang sama antar blok. Meskipun CBC mengenkripsi secara blok, namun ada *chaining* dengan blok sebelumnya yang membuatnya menjadi lebih acak. Terakhir, pada mode *Counter* meskipun sama dengan ECB namun nilai *counter* yang dimasukkan ke dalam fungsi enkripsi selalu berbeda setiap blok, sehingga hasilnya akan menjadi lebih acak.



Gambar 12. Grafik histogram untuk 10 blok pertama pada mode ECB



Gambar 13. Hasil histogram dari *plaintext* Lorem ipsum 10.000. Dari pojok kiri atas ke kanan bawah, (1) histogram *plaintext*, (2) histogram ECB, (3) histogram CBC, dan (4) histogram Counter

Pada contoh analisis frekuensi untuk data nyata seperti pada *plaintext* Lorem ipsum sepanjang 10.000 kata, frekuensi karakter pada *plaintext* tidak merata seperti pada Gambar 13 bagian (1) namun hasil enkripsinya sudah tersebar merata ke semua karakter pada ASCII. Dengan demikian 8-Series Cipher tidak mudah dipecahkan dengan melakukan analisis frekuensi.

5. Kesimpulan dan Saran

5.1 Kesimpulan

Algoritma 8-Series Cipher dapat digunakan sebagai *block cipher* untuk melakukan enkripsi dan dekripsi. Selain itu algoritma ini sudah menerapkan prinsip-prinsip *block cipher* yang baik yaitu, prinsip *diffusion* dan *confusion*, menggunakan jaringan Feistel, melakukan iterasi atau pengulangan, dan melakukan substitusi kotak-S. Algoritma ini sudah aman jika dilihat dari analisis *diffusion* dan *confusion*, analisis *brute force attack*, dan analisis frekuensi. Kelebihan dari algoritma ini adalah kunci eksternal yang digunakan bersifat fleksibel sehingga dapat diisikan dengan panjang berapa pun (bahkan kurang dari 144 bit). Semakin panjang kunci eksternal maka akan semakin aman hasil enkripsinya. Selain itu, jumlah putaran yang diterapkan pun sebenarnya dapat dilakukan lebih dari delapan kali jika diinginkan hasil yang lebih aman lagi.

5.2 Saran

Pada proses pembangkitan kunci internal, masih ada beberapa bit kunci yang pengaruhnya sangat kecil terhadap bit enkripsi, hal ini dikarenakan lompatan pada saat pemilihan bit tidak memilih bit-bit kunci yang belum terpilih melainkan seluruh bit kunci sehingga ada bit yang tidak terpilih atau jarang terpilih. Selain itu algoritma ini masih kurang efisien dalam proses eksekusinya, dengan banyaknya konversi representasi bit, menyebabkan proses eksekusi menjadi lambat. Untuk pengembangan selanjutnya algoritma ini dapat diimplementasi dalam bahasa lain yang lebih cepat misalnya C++ dan melakukan penanganan representasi bit dengan lebih baik. Terakhir, salah satu metrik yang belum dianalisis di sini adalah masalah memory dan CPU, pada pengembangan selanjutnya dapat dilakukan analisis dan optimasi pada metrik tersebut.

6. Referensi

[1] Rivest, Ronald L. (1990). "Cryptography". In J. Van Leeuwen (ed.). Handbook of Theoretical Computer Science. 1. Elsevier.

[2] D. R. Anderson, “Information Theory and Entropy,” *Model Based Inference Life Sci. A Prim. Evid.*, no. 1994, pp. 51–82, 2008, doi: 10.1007/978-0-387-74075-1_3.

[3] Munir, Rinaldi. (2019). “Kriptografi”. Bandung: Informatika.

7. *Acknowledgments*

Puji syukur ke hadirat Tuhan yang Maha Esa karena telah memberikan kesempatan kepada penulis untuk menyelesaikan makalah dengan judul 8-Series Cipher ini. Penulis juga berterima kasih kepada dosen pengampu mata pelajaran IF4020 Kriptografi, Dr. Ir. Rinaldi Munir, MT., yang telah memberikan pengetahuan mengenai kriptografi dan *block cipher* sehingga 8-Series Cipher ini dapat dibuat. Selain itu penulis juga berterima kasih kepada semua pihak yang turut ikut serta membantu menyelesaikan makalah ini. Penulis berharap agar makalah ini dapat membantu menambah wawasan bagi pembaca mengenai *block cipher*.