Enigma-Based Key Scheduler Block Cipher

Jesson Yo- 13519079 (*Penulis*)
Frederic Ronaldi - 13519134 (*Penulis*)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: 13519079@std.stei.ac.id, 13519134@std.stei.ac.id

***Abstract*— Makalah ini membahas desain blok cipher baru yang memanfaatkan mesin Enigma di dalam algoritma penjadwalan kunci dengan harapan meningkatkan keamanan proses enkripsi. Rancangan blok cipher yang diajukan didesain agar mirip dengan cipher lain yang sudah ada namun hanya dibedakan di bagian penjadwalan kunci dengan pendekatan yang memanfaatkan mesin Enigma. Mesin Enigma itu sendiri adalah sebuah mesin enkripsi bersejarah yang sempat digunakan oleh militer Jerman dalam Perang Dunia II. Algoritma mesin Enigma dapat dimanfaatkan untuk menghasilkan rangkaian kunci turunan yang sulit untuk diprediksi sehingga cocok untuk dicoba dalam meningkatkan keamanan blok cipher.**

Keywords—kriptografi, blok cipher, enigma, penjadwalan kunci, konfusi-difusi

I. Latar Belakang

Blok cipher merupakan komponen yang fundamental dari kriptografi modern dan digunakan di berbagai jenis aplikasi seperti enkripsi data, autentikasi, dan verifikasi integritas. Keamanan dari blok cipher ditentukan oleh banyak hal, tergantung dari implementasi dan komponen-komponen yang ada di dalamnya. Salah satunya adalah algoritma penjadwalan kunci yang berperan untuk melakukan kreasi sub-kunci dari kunci utama yang digunakan untuk setiap *round function*. Algoritma penjadwalan kunci harus didesain agar aman, efisien, dan tahan terhadap berbagai jenis serangan. Namun seiring berkembangnya kekuatan komputasional, algoritma penjadwalan kunci yang sudah ada bisa menjadi rawan terhadap serangan-serangan baru sehingga perlu dilakukan eksplorasi terus menerus mengenai pendekatan baru dalam penjadwalan kunci.

Makalah ini mengagaskan sebuah rancangan blok cipher baru yang memanfaatkan sebuah algoritma penjadwalan kunci berbasis mesin Enigma untuk meningkatkan keamanan proses enkripsi cipher. Mesin Enigma adalah sebuah mesin pengenkripsi bersejarah yang sempat digunakan oleh Jerman pada Perang Dunia II. Mesin ini dikenal sebagai mesin yang mampu melakukan enkripsi yang kompleks dan terkesan sangat acak pada zamannya sehingga sulit untuk memprediksi keluarannya. Dengan memanfaatkan mesin Enigma di dalam algoritma penjadwalan kunci maka harapannya blok cipher baru ini akan tahan terhadap serangan terutama serangan yang memanfaatkan algoritma *brute force*.

Algoritma penjadwalan kunci berbasis mesin Enigma akan diintegrasikan dengan blok cipher yang digagaskan sedemikian rupa sehingga komponen lainnya akan sangat mirip dengan cipher yang sudah ada sehingga lebih mudah menganalisa dan membandingkan efisiensi dan efektivitasnya. Di dalam makalah ini akan dideskripsikan mengenai rancangan dan implementasi cipher yang digagaskan serta hasil analisa dan pengujian. Blok cipher juga akan dikaji potensinya untuk penggunaan lain di masa depan.

II. Dasar Teori

A. Blok Cipher

Blok cipher adalah jenis algoritma yang melakukan enkripsi dengan membagi pesan menjadi beberapa blok dengan ukuran tertentu kemudian enkripsi dilakukan untuk setiap blok secara terpisah namun bisa saling berhubungan tergantung mode blok cipher yang digunakan. Hasil enkripsi blok cipher pada umumnya menghasilkan ukuran yang identik sama dengan pesan awal. Hal ini menjadi salah satu alasan mengapa blok cipher populer digunakan karena ukuran pesan sama serta komputasi lebih murah dibandingkan cipher lain misalnya cipher asimetris seperti RSA yang menambah ukuran pesan serta komputasinya relatif mahal.

1) Kriptografi Simetris

Kriptografi simetris merupakan sebuah tipe kriptografi yang menggunakan kunci yang sama dalam proses enkripsi dan dekripsi. Artinya kedua aktor, biasanya disebut penerima dan pengirim harus menyetujui sebuah nilai untuk kunci yang sifatnya rahasia sebelum mereka dapat berbagi rahasia. Walaupun aktor hanya dua namun jumlah penerima lebih dari satu misalnya dalam jenis pengiriman *broadcast*.

Dalam kriptografi simetris, pesan akan ditransformasikan sedemikian rupa menggunakan kunci rahasia sehingga menghasilkan teks cipher yang sifatnya tidak bisa dipahami apabila dibaca manusia karena tidak memiliki makna. Untuk mentransformasikan teks cipher ke pesan semula maka dilakukan proses dekripsi menggunakan kunci yang sama dengan kunci untuk enkripsi dengan fungsi transformasi yang sifatnya berkebalikan.

2) Mode operasi

Pada blok cipher, mode operasi merupakan spesifikasi cara algoritma enkripsi diaplikasikan terhadap

blok-blok cipher yang ada untuk skenario-skenario tertentu. Ada beberapa mode operasi yang tersedia untuk blok cipher, lima diantaranya adalah:

a. Electronic Codebook (ECB)
b. Cipher Block Chaining (CBC)
c. Cipher Feedback (CFB)
d. Output Feedback (OFB)
e. Counter (CTR)

3) Konfusi dan Difusi

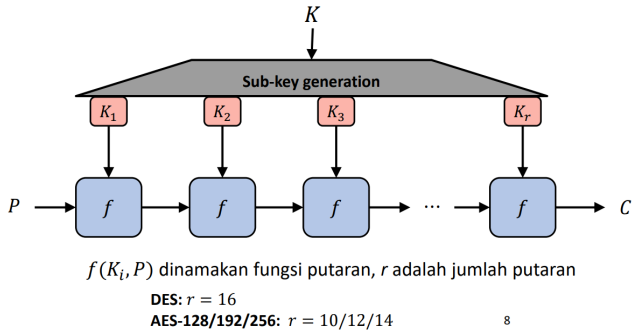
Konfusi dan difusi adalah dua konsep utama dari teorema kerahasiaan Shannon. Konfusi mengacu kepada ide bahwa hubungan antara *plaintext* dan *ciphertext* harus dibuat serumit dan sesamar mungkin. Dengan kata lain apabila *attacker* mengetahui *ciphertext* maka mereka tetap tidak bisa dengan mudah menentukan informasi mengenai *plaintext* tanpa adanya informasi kunci rahasia. Konfusi biasanya dicapai dengan penggunaan operasi matematika seperti subtitusi dan permutasi berulang sehingga hubungan antara *plaintext* dan *ciphertext* tidak dapat dilihat oleh *attacker*.

Difusi mengacu kepada ide bahwa perubahan kecil di *plaintext* harus berujung kepada perubahan yang besar dan tidak bisa diprediksi dari *ciphertext*. Ini berfungsi untuk mencegah penyerang dalam melakukan identifikasi pola atau pengulangan dalam teks cipher yang bisa digunakan untuk menginferensi informasi mengenai *plaintext*. Difusi biasa dicapai dengan penggunaan operasi XOR atau sejenis pada sebuah blok yang ingin dienkripsi dengan hasil enkripsi blok sebelumnya. Langkah ini mengakibatkan perubahan pada satu blok dapat merambat ke blok lain.

Konfusi dan difusi menjadi dasar dalam memastikan keamanan sistem kriptografi, salah satunya adalah perancangan blok cipher. Dengan memastikan hubungan antara *plaintext* dan *ciphertext* serumit mungkin dan perubahan kecil pada *plaintext* akan berujung ke perubahan *ciphertext* yang sulit diprediksi dengan skala besar maka sistem kriptografi yang demikian akan mampu memberikan tingkat konfidentialitas yang tinggi dengan mencegah serangan-serangan terhadap pesan yang terenkripsi.

4) Penjadwalan Kunci

Penjadwalan kunci merupakan komponen penting dalam blok cipher. Komponen ini digunakan untuk menciptakan kumpulan kunci putaran dari kunci enkripsi utama. Kunci putaran kemudian akan digunakan untuk setiap fungsi putaran dalam proses enkripsi maupun dekripsi pesan. Secara umum penjadwalan kunci merupakan aplikasi sekuens transfomasi terhadap kunci enkripsi utama untuk menghasilkan beberapa sub kunci. Algoritma untuk penciptaan dan penjadwalan kunci mana yang dipakai pada setiap fungsi putaran merupakan bagian dari algoritma penjadwalan kunci. Tidak ada aturan baku mengenai algoritma penjadwalan kunci tetapi umumnya merupakan gabungan operasi *bitwise*, permutasi, serta subtitusi dalam melakukan transformasi kunci utama ke sub kunci.



Gambar I. Sumber: Rinaldi Munir, "IF 4020 Kriptografi, *Block Cipher* Bagian 2", https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2022-2023/14-Prancangan-block-cipher-2023.pdf, Slide 8

Jumlah dan ukuran sub kunci yang diciptakan oleh algoritma penjadwalan kunci berbeda-beda tergantung *cipher* yang digunakan. Contohnya cipher DES menghasilkan 16 buah kunci putaran yang masing-masing berukuran 48 bit dari kunci enkripsi utama yang berukuran 64 bit sedangkan cipher AES menciptakan 10,12, atau 14 sub kunci tergantung versi AES yang digunakan yaitu AES-128 untuk panjang kunci utama 128 bit, AES-192 untuk panjang kunci utama 192 bit, serta AES-256 untuk panjang kunci utama 256 bit. Penjadwalan kunci sangat penting dalam menjamin tingkat keamanan cipher.

5) Fungsi putaran

Di dalam sebuah algoritma enkripsi blok cipher, fungsi putaran merupakan salah satu komponen yang fundamental. Idenya adalah untuk memperkuat hasil cipher dengan melakukan *enciphering* berkali-kali. Hal ini dicapai dengan mengaplikasikan sebuah fungsi transformasi berulang ke sebuah blok dengan fungsi transformasi yang identik namun menggunakan kunci berbeda (kunci putaran/sub kunci) yang diturunkan dari kunci enkripsi utama. Fungsi putaran dapat dinyatakan sebagai berikut.

$$C_i = f(C_{i-1}, K_i)$$

Huruf i melambangkan putaran ke-i sedangkan C, f, dan K melambangkan teks cipher, fungsi transformasi, dan sub kunci secara berurutan dengan C_0 merupakan kasus khusus karena melambangkan pesan awal (*plaintext)*. Sebuah fungsi putaran biasa berisi beberapa sub fungsi lagi yang biasanya merupakan fungsi subtitusi (bisa dibantu kotak S), permutasi (bisa dibantu kotak P), dan pencampuran seperti XOR atau penjumlahan dalam modulus tertentu. Pilihan jumlah putaran yang dilakukan terhadap blok harus dikonsiderasi dan dianalisa. Semakin banyak jumlah putaran akan selalu menambah waktu komputasi namun tidak selalu menambah kekuatan cipher (*diminishing return*).

6) Kotak S

Dalam kriptografi, kotak S atau kotak subtitusi adalah komponen yang digunakan untuk menambah *confusion* dan non linearitas ke proses enkripsi. Kotak S pada

dasarnya adalah sebuah *lookup table* yang memetakan sejumlah bit masukan ke sejumlah bit keluaran dengan pemetaan yang sifatnya satu ke satu.

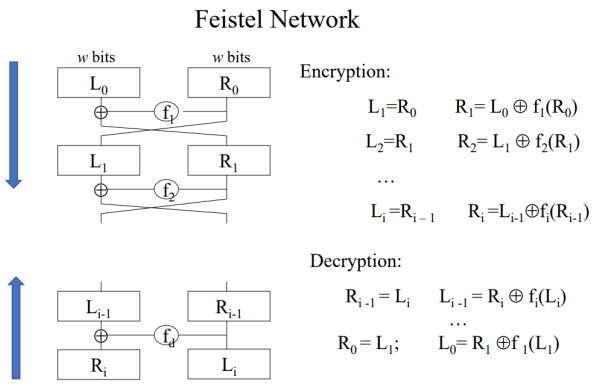
Penggunaan kotak S akan mempersulit penyerang dalam mencari hubungan antara teks cipher dan plainteks bahkan jika penyerang mengetahui algoritma enkripsi dan teks cipher. Dengan subtitusi non linear dengan suatu pola yang sifatnya konstan maka kotak S membantu menyamarkan struktur hubungan dalam algoritma enkripsi dan membuatnya tahan terhadap serangan kriptanalisis.

7) Kotak P

Kotak P mirip dengan kotak S namun alih-alih dipakai untuk melakukan operasi subtitusi, kotak P dimanfaatkan dalam melakukan permutasi dengan pola tertentu untuk mengatur ulang urutan bit pada blok. Manfaat dari kotak P adalah merambatkan pengaruh perubahan pada salah satu bit masukan ke seluruh blok untk memberikan efek difusi pada proses enkripsi.

8) Jaringan Feistel

Jaringan feistel dinamakan dari Horst Feistel yaitu seorang ilmuwan yang mengajukan struktur ini di tahun 1973 dengan tujuan untuk menciptakan sebuah algoritma enkripsi yang kuat dan efisien. Jaringan feistel adalah sebuah struktur yang sering digunakan dalam desain algoritma enkripsi simetris. Intinya, jaringan feistel adalah sebuah sekuens berulang dari fungsi putaran yang melibatkan pemecahan masukan menjadi dua bagian yaitu kiri dan kanan kemudian mengaplikasikan serangkaian operasi transformasi ke kedua bagian dengan cara yang *reversible*.



Gambar II. Sumber: Rinaldi Munir, "IF 4020 Kriptografi, *Block Cipher* Bagian 2", https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2022-2023/14-Prancangan-block-cipher-2023.pdf , Slide 23

B. Mesin Enigma

Mesin Enigma memiliki beberapa jenis model yang diketahui telah diproduksi. Model yang berbeda dapat memiliki perbedaan di komponen-komponen kunci seperti jumlah rotor, *wiring* rotor, keberadaan *plugboard*, keberadaan *reflector*, serta fitur-fitur lain. Perbedaan model akan berujung kepada variasi kompleksitas dan keamanan dengan model yang lebih modern biasanya memiliki fitur tambahan dan mekanisme yang lebih kompleks untuk meningkatkan proses enkripsi. Namun demikian seluruh model dari mesin Enigma memiliki prinsip dasar operasi kriptografi yang sama. Secara sederhana mesin Enigma umumnya memiliki 4 komponen dasar yaitu rotor, *wiring* dari rotor, *plugboard,* serta *reflector*. Setiap huruf akan dienkripsi secara independen dan akan melewati keempat komponen dengan alur subtitusi umum sebagai berikut: *plugboard,* subtitusi maju melalui seluruh rotor, dipantulkan dengan reflektor, subtitusi mundur kembali ke seluruh rotor, dan terakhir subtitusi kembali menggunakan *plugboard*.

1) Rotor

Rotor digunakan untuk melakukan operasi subtitusi monoalfabetik berulang dengan jumlah perulangan paling sederhananya adalah berjumlah tiga kali. Hasil subtitusi rotor pertama akan menjadi masukan untuk rotor kedua dan seterusnya. Hal ini direalisasikan dengan 3 rotor yang berputar secara independen satu sama lain. Untuk setiap huruf yang diproses maka rotor pertama akan maju sebanyak 1/n putaran dengan n adalah jumlah takik dan apabila rotor pertama menyentuh *turnover notch* yaitu suatu posisi tertentu yang akan menyebabkan rotor disebelahnya ikut berputar maka putaran akan merambat ke rotor disebelahnya. Hal memiliki implikasi yaitu seluruh rotor tidak peduli jumlahnya akan memiliki kesempatan untuk berputar asalkan jumlah huruf yang ingin dienkripsi cukup banyak.

Putaran pada rotor mengakibatkan subtitusi monoalfabetik berubah sehingga huruf yang sama tidak akan dipetakan menjadi huruf lain yang sama. Tujuannya adalah untuk menghilangkan hubungan frekuensi pada teks cipher. Keberadaan lebih dari satu rotor yang saling bergerak independen juga akan semakin mengaburkan hasil teks cipher karena akan membuatnya semakin sulit diprediksi.

Sebuah rotor pada mesin Enigma umumnya memiliki 26 takik yang setiap takiknya merepresentasikan sebuah alfabet romawi (A-Z). Subtitusi juga dilakukan dari alfabet romawi ke alfabet romawi. Enigma juga memiliki parameter penentu kondisi awal yang dapat dikonfigurasi yaitu posisi rotor dan *rings*. Parameter posisi rotor akan menjadi posisi awal (jumlah putaran dari posisi 0) untuk setiap rotor. Parameter kedua adalah *rings* yang merupakan pergeseran posisi alfabet relatif terhadap *wiring* pada rotor.

Jenis rotor juga bisa dikonfigurasi pada sebuah mesin Enigma. Model mesin Enigma yang berbeda pada umumnya memiliki beberapa pilihan rotor yang eksklusif untuk model tersebut. Jenis rotor yang berbeda memiliki *turnover notch* dan *rotor wiring* yang berbeda. Efeknya subtitusi monoalfabetik menjadi berbeda serta posisi ketika rotor menyebabkan perambatan putaran ke rotor setelahnya juga akan berubah. Secara matematis ini akan berdampak ke permutasi huruf yang jauh berbeda antar model mesin dan jenis rotor. Contohnya mesin Enigma I memiliki tiga konfigurasi rotor yaitu I, II, dan III. *Wiring* pada masing-masing rotor secara berurutan adalah:

a. EKMFLGDQVZNTOWYHXUSPAIBRCJ

b. AJDKSIRUXBLHWTMCQGZNPYFVOE
c. BDFHJLCPRTXVZNYEIWGAKMUSQO

Cara membacanya untuk contoh (a.) adalah A akan dipetakan menjadi E, B akan dipetakan menjadi K, dan seterusnya untuk posisi dan *rings* dasar. Apabila posisi sudah berubah atau nilai *rings* tidak nol maka ada *offset* yang perlu dikonsiderasi pada proses subtitusi monoalfabetik. Subtitusi seperti demikian akan dilakukan pada proses subtitusi maju mesin Enigma. Pada subtitusi mundur, *wiring* yang sama dapat digunakan untuk menurunkan nilai invers dari *wiring* dan subtitusi akan memanfaatkan invers dari *wiring* dengan formula yang sama pada subtitusi maju. Contoh invers *wiring* untuk masing-masing rotor secara berurutan adalah:

a. UWYGADFPVZBECKMTHXSLRINQOJ
b. AJPCZWRLFBDKOTYUQGENHXMIVS
c. TAGBPCSDQEUFVNZHYIXJWLRKOM

2) Plugboard

Plugboard memberikan lapisan tambahan pada keamanan mesin Enigma. Komponen ini melakukan subtitusi bolak-balik antara 2 huruf misalnya EO pada untuk setiap huruf E akan menjadi O dan setaip huruf O akan menjadi E. Sifat dari *plugboard* adalah opsional sehingga tidak semua huruf wajib dilakukan subtitusi bolak-balik atau bisa juga dianggap sebagai subtitusi dengan diri sendiri (A menjadi A, B menjadi B, dst). Penambahan *plugboard* terbukti lebih efektif menambah keamanan mesin Enigma dibandingkan penambahan satu buah rotor karena jumlah enumerasi kemungkinan pengaturan yang diberikan jauh lebih tinggi dibandingkan penambahan sebuah rotor.

3) Reflector

Tujuan dari reflektor adalah untuk menghilangkan perlunya fungsi enkripsi dan dekripsi yang berbeda pada mesin Enigma serta memastikan tidak ada huruf yang dipetakan ke huruf yang sama. Dengan reflektor, dua mesin Enigma berbeda yang memiliki konfigurasi sama dapat dilakukan untuk melakukan enkripsi dan dekripsi dengan prosedur yang sama persis. Secara mekanik, reflektor juga bertujuan untuk mengembalikan arus listrik dari reflektor kembali ke rotor. Dari sudut pandang alur subtitusi mesin Enigma, hal ini tampak untuk membuat proses subtitusi mundur dapat dilakukan setelah proses subtitusi maju. Substitusi yang terjadi pada *reflector* sama dengan substitusi yang terjadi pada *plugboard* namun subtitusi ini terjadi untuk tahap yang berbeda pada alur subtitusi mesin Enigma. Ditambah lagi dengan fakta bahwa *reflector* bukanlah komponen yang opsional sehingga untuk setiap huruf diperlukan pemetaan yang sifatnya bolak-balik antara sepasang huruf dan tidak boleh dilakukan pemetaan dengan dirinya sendiri karena akan mengabaikan tujuan awal dari diciptakannya reflektor.

III. Implementasi

Enigma-Based Key Scheduler Block Cipher adalah algoritma *block cipher* yang bekerja dengan memanfaatkan jaringan feistel dan mesin enigma. Mesin engima digunakan untuk membangkitkan kunci putaran yang akan digunakan dalam jaringan feistel. Dalam perancangannya, *Enigma-Based Key Scheduler Block Cipher* menerapkan prinsip-prinsip perancangan algoritma *block cipher* yaitu *confusion* dan *diffusion* dengan menggunakan fungsi putaran dalam jaringan feistel.

Implementasi dari *Enigma-Based Key Scheduler Block Cipher* akan ditulis dalam bahasa Python. Secara sederhana, plainteks akan dibagi ke blok-blok yang berukuran 16 *bytes* / 128 *bits*. *Enigma-Based Key Scheduler Block Cipher* dapat menerima kunci eksternal berukuran 128 *bits* dan memiliki jaringan feistel dengan fungsi putaran berjumlah 16 putaran.

Dalam proses enkripsi, hal yang pertama dilakukan adalah mengecek apabila plainteks berukuran kelipatan 16 *bytes*/128 *bits*. Apabila plainteks tidak berkelipatan 128 *bits*, maka akan ditambahkan *padding* agar plainteks dapat dibagi menjadi blok-blok yang berukuran 128 *bits*. Setelah itu, setiap blok akan dibagi menjadi 2 bagian (kiri dan kanan) yang masing-masing berukuran 64 *bit*. Agar mendukung mode operasi CBC, maka akan dilakukan XOR antara plainteks dengan *initialization vector* untuk blok pertama, dan blok selanjutnya akan menggunakan hasil cipherteks sebelumnya.

Selanjutnya merupakan tahap jaringan feistel dan fungsi putaran yang dilakukan 16 kali. Tahap pertama adalah pembangkitan kunci putaran menggunakan mesin Enigma. Mesin enigma yang digunakan adalah mesin Enigma dengan versi 3 rotor yang dimodifikasi agar bisa memproses *byte* alih-alih karakter alfabet dengan *wiring* dan *reflector* yang baru. Kunci putaran pertama akan dibangkitkan dari kunci eksternal, sedangkan kunci putaran selanjutnya akan dibangkitkan dari kunci putaran sebelumnya. Posisi dan *rings* awal sebagai parameter Enigma dapat dihitung sebagai berikut:

1. Membagi kunci putaran sebelumnya/kunci eksternal untuk tahapan pertama menjadi 2 berukuran 64 *bit*. (C dan D)
2. *Left shift* sebesar 1 atau 2 yang bergantung pada tahapannya. Tahapan dan jumlah *shift* yang dilakukan adalah sebagai berikut:

Tahapan	Jumlah *shift*
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2

9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

3. Mengambil 3 *byte* pertama dalam C dan D yang dapat merepresentasikan 3 angka untuk masing-masing posisi dan *rings*
4. Membangkitkan kunci putaran dengan cara mengguna mesin engima dengan parameter tersebut dengan masukan kunci putaran sebelumnya/kunci eksternal secara penuh

Pembangkitan kunci putaran dilakukan sebanyak fungsi putaran dilakukan yaitu 16 putaran. Masing-masing putaran akan memliki posisi dan *rings* yang berbeda-beda. Setiap tahapan akan menyimpan *state* untuk posisi dan *rings*-nya yang akan dibawa ke blok selanjutnya sehingga blok yang serupa akan menghasilkan cipherteks yang berbeda.

Pada masing-masing putaran juga akan dilakukan proses-proses seperti jaringan feistel pada umumnya. Bagian kiri yang baru adalah bagian kanan yang lama, sedangkan bagian kanan yang baru adalah hasil antara XOR bagian kiri yang lama dan hasil fungsi putaran yang melibatkan bagian kanan yang lama dan kunci putaran yang didapatkan sebelumnya pada tahapan tersebut. Berikut adalah tahapan-tahapan yang dilakukan pada fungsi putaran:

1. Karena bagian kanan berukuran 64 *bit*, dan kunci putaran berukuran 128 *bit*, maka akan dilakukan ekspansi agar bagian kanan berukuran 128 *bit* juga. Ekspansi dilakukan dengan teknik permutasi dengan P-Box sebagai berikut:

```
pbox = [
    37, 26, 26, 8, 4, 7, 3, 1, 19, 34,
    29, 57, 30, 48, 12, 63, 32, 45, 11, 49,
    27, 33, 41, 46, 15, 28, 60, 31, 20, 5,
    61, 35, 54, 45, 39, 53, 41, 10, 2, 25,
    1, 53, 35, 22, 12, 11, 50, 40, 36, 48,
    56, 49, 56, 50, 31, 59, 13, 42, 24, 39,
    20, 44, 28, 29, 16, 46, 37, 62, 52, 3,
    51, 51, 59, 14, 27, 40, 17, 14, 52, 38,
    55, 44, 58, 18, 9, 2, 61, 34, 33, 25,
    47, 8, 55, 21, 0, 5, 10, 24, 13, 6,
    54, 17, 30, 9, 6, 42, 62, 36, 38, 15,
    43, 43, 7, 60, 23, 22, 63, 23, 18, 4,
    21, 0, 16, 19, 57, 47, 58, 32
]
```

2. Melakukan XOR terhadap hasil ekspansi dan kunci putaran pada tahapan tersebut
3. Melakukan substitusi dengan cara membagi hasil XOR yang berukuran 128 *bits*/16 *bytes* menjadi 8 bagian yang berukuran masing-masing 2 *bytes*. Setiap blok berukuran 2 *bytes* tersebut akan disubtitusi dengan 2 *bytes* kembali yang bergantung pada kotak S yang sudah didefinisikan sebelumnya. Nomor baris dapat dihitung dengan mengambil *bit* pertama dan *bit* terakhir dalam 2 *bytes* tersebut, menghasilkan nomor baris 0 atau 1. Nomor kolom dapat dihitung dari 4 *bit* terkecil dalam *byte* pertama di blok tersebut, menghasilkan nomor kolom antara 0 hingga 15. Hal ini menunjukkan bahwa kotak-S berukuran 2 x 16 yang dapat diisi oleh data berukuran 2 *bytes*. Berikut adalah kotak-S yang digunakan:

```
sboxes = {
    0: [
        [0x7E, 0x16, 0x75, 0x14, 0x18, 0x09, 0x78, 0x3F, 0x24, 0x6D, 0x1B, 0x52, 0x2A, 0x3D, 0x67, 0x45],
        [0x5C, 0x5D, 0x5E, 0x5F, 0x60, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x67, 0x68, 0x69, 0x6A, 0x6B],
    ],
    1: [
        [0x3F, 0x3E, 0x3D, 0x3C, 0x3B, 0x3A, 0x39, 0x38, 0x37, 0x36, 0x35, 0x34, 0x33, 0x32, 0x31, 0x30],
        [0x6B, 0x6A, 0x69, 0x68, 0x67, 0x66, 0x65, 0x64, 0x63, 0x62, 0x61, 0x60, 0x5F, 0x5E, 0x5D, 0x5C],
    ],
    2: [
        [0x45, 0x67, 0x3D, 0x2A, 0x1B, 0x24, 0x52, 0x6D, 0x3F, 0x78, 0x09, 0x18, 0x14, 0x75, 0x16, 0x7E],
        [0x1E, 0x1F, 0x20, 0x21, 0x22, 0x23, 0x24, 0x25, 0x26, 0x27, 0x28, 0x29, 0x2A, 0x2B, 0x2C, 0x2D],
    ],
    3: [
        [0x7E, 0x16, 0x75, 0x14, 0x18, 0x09, 0x78, 0x3F, 0x24, 0x6D, 0x1B, 0x52, 0x2A, 0x3D, 0x67, 0x45],
        [0x5C, 0x5D, 0x5E, 0x5F, 0x60, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x67, 0x68, 0x69, 0x6A, 0x6B],
    ],
    4: [
        [0x3F, 0x3E, 0x3D, 0x3C, 0x3B, 0x3A, 0x39, 0x38, 0x37, 0x36, 0x35, 0x34, 0x33, 0x32, 0x31, 0x30],
        [0x6B, 0x6A, 0x69, 0x68, 0x67, 0x66, 0x65, 0x64, 0x63, 0x62, 0x61, 0x60, 0x5F, 0x5E, 0x5D, 0x5C],
    ],
    5: [
        [0x45, 0x67, 0x3D, 0x2A, 0x1B, 0x24, 0x52, 0x6D, 0x3F, 0x78, 0x09, 0x18, 0x14, 0x75, 0x16, 0x7E],
        [0x1E, 0x1F, 0x20, 0x21, 0x22, 0x23, 0x24, 0x25, 0x26, 0x27, 0x28, 0x29, 0x2A, 0x2B, 0x2C, 0x2D],
    ],
    6: [
        [0x1E, 0x1F, 0x20, 0x21, 0x22, 0x23, 0x24, 0x25, 0x26, 0x27, 0x28, 0x29, 0x2A, 0x2B, 0x2C, 0x2D],
        [0x3F, 0x3E, 0x3D, 0x3C, 0x3B, 0x3A, 0x39, 0x38, 0x37, 0x36, 0x35, 0x34, 0x33, 0x32, 0x31, 0x30],
    ],
    7: [
        [0x5C, 0x5D, 0x5E, 0x5F, 0x60, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x67, 0x68, 0x69, 0x6A, 0x6B],
        [0x45, 0x67, 0x3D, 0x2A, 0x1B, 0x24, 0x52, 0x6D, 0x3F, 0x78, 0x09, 0x18, 0x14, 0x75, 0x16, 0x7E],
    ]
}
```

Setelah satu fungsi putaran selesai dilakukan, maka akan dilakukan putaran selanjutnya hingga 16 kali. Setelah itu, maka bagian kiri dan kanan yang baru akan digabungkan dan menjadi cipherteks untuk blok tersebut. Hal ini akan dilakukan untuk setiap blok hingga didapatkan cipherteks yang utuh.

Dalam proses dekripsi, proses yang dilakukan serupa dengan proses enkripsi dengan beberapa perbedaan. Pertama, akan dibagi menjadi blok-blok yang berukuran 128 *bit*. Setelah itu, dibangkitkan seluruh 16 kunci putaran secara penuh dari kunci eksternal yang sama untuk digunakan dalam

proses dekripsi. Selain itu, akan disimpan pula hasil dekripsi sebelumnya yang akan digunakan untuk XOR hasil dekripsi sebelumnya dengan *individual vector* atau hasil dekripsi yang baru. Setelah itu, proses dekripsi dilanjutkan dengan jaringan feistel yang dipanggil dari bagian kiri dan kanan untuk setiap blok cipherteks, dengan kunci putaran yang dibalik. Tahapan pertama dalam jaringan feistel akan menggunakan kunci putaran yang terakhir, tahapan kedua akan menggunakan kunci putaran ke-15, dst. Setelah proses dekripsi selesai untuk setiap blok, maka hasil akan digabungkan dan menjadi plainteks yang utuh.

IV. Eksperimen dan Pembahasan Hasil

Dalam membahas hasil dan pengetesan implementasi Enigma-Based Key Scheduler Block Cipher, penulis menggunakan kunci berikut:

b'Sixteen byte key'

Pengetesan implementasi pertama menghasilkan hasil seperti berikut:

Plainteks	Cipherteks
b'Makalah ini membahas desain blok cipher baru yang memanfaatkan mesin Enigma di dalam algoritma penjadwalan kunci dengan harapan meningkatkan keamanan proses enkripsi. Rancangan blok cipher yang diajukan didesain agar mirip dengan cipher lain yang sudah ada namun hanya dibedakan di bagian penjadwalan kunci dengan pendekatan yang memanfaatkan mesin Enigma. Mesin Enigma itu sendiri adalah sebuah mesin enkripsi bersejarah yang sempat digunakan oleh militer Jerman dalam Perang Dunia II. Algoritma mesin Enigma dapat dimanfaatkan untuk menghasilkan rangkaian kunci turunan yang sulit untuk diprediksi sehingga cocok untuk dicoba dalam meningkatkan keamanan blok cipher.'	b'\x00:3{]1E1u7\x1a\x07_\x01~JW\x0fY+vv#\tz7\x12Y[{\x08V-fT7*@~z\x15\tqEJc{\\DK]@t>1v4aCU"8V+a\tA\x107VD\x0c\x02\x0c\x7fF,$o#1g7zI9:jU.RRHEx\tGiA\x1b\x1aCM0\x0c&9\x10o\x0c\x11,e=\|oJu\x1e1F&u\\v#inSb0D*"E\x1f~{\x0cN[,\x02e0@r\|}5\x01\n\x03U"%"nw\x063NtLq1~I\x0ff\x04\x1b3\x06nZ\x01u7(\x1b\\%\nmj\x1d\x0c\x1cm\r\x03w\\\x19&n\x12I\x07Q&qd.!\x12;s;qrPix\x01bW7\t6\x07\x1fI\'d\x1e\x1dl\x13(mZuFM\x1dj\x1cG\x0e\x10{x\t2q-\x06\x11{\n/\x11.,X\nZa3"\x14.:*+".d\x0fB<yN6\\\x04]OB+LN^:h5L!?TR\|61J\x1dn>\x0fh\x1b#\|ZJj\x12\x0f`\x01\x18\x18Q\|\x0e]IT):AHT(\x04q1;kTzFd:#)HJq\\=>\x11\x1c%#\x056[82X6e\x189\x07\x03\x12~}`lU\x13%h\x11OO\tMWV!\x1b\x7f#<Z$O?<\'\\\x17^(0KLNLI^BJ4]R9P[u\x08Jk!w"P=\x124;+T\\\x0cKgz}wN2D\x13\x12\x08O_7(W8WD_\x17%7u;1S\x065B\x16C\x0bR\x11\x19pD^\x19J|:Eu+D\x1e;_bt/CHG9_/8L"\x0e)c.^9YzTy1]M\x14}`[\x15\x11A4k;#4.-}\x7fk-C},<\x18F+H"5Q+\x1aXa;*h\x120<)o<:1;\x0fSr\x19CGbE%3~Z[B\tX=:T\x14;N,G"\n{\x13`\x19\x15DuQS*.Ym\x12\n\x1d~yx\x1f-4mj[omS\x026vU\n1v\x1b\x0ce\x02,yz3\x12jP\x08QnPN$TEk\x1c\x0c\x08+lKo2f\x05B)C\x1c*m`:kP\x05E\x04Y\x0f@pI(n<AiO\x06"4!)`k^$'

Untuk pengujian waktu eksekusi, akan dilakukan dengan ukuran plainteks sebagai berikut:

Ukuran plainteks	Ukuran Plainteks	Waktu enkripsi / dekripsi
Small	10 bytes	0.001997709274291992 detik / 0.001999378204345703 detik
Medium	1,000 bytes	0.10810446739196777 detik / 0.10672330856323242 detik
Large	10,000 bytes	1.0355820655822754 detik / 1.0504446029663086 detik
Very Large	1,000,000 bytes	10.793660163879395 detik / 10.673699378967285 detik

Selanjutnya akan dilakukan pengujian efek longsoran (*avalanche effect*) yaitu pengujian terhadap perubahan bit untuk melihat perubahan terhadap cipherteksnya. Plainteks akan diubah menjadi yang lebih pendek agar perubahan cipherteks dapat lebih terlihat.

Plainteks	Cipherteks
b'Stop overthinking, calm	b'Rx\x1c\x7f-^I\x7f\x1e(iJl\

down, and relax'	x16s6RX>}w70\x04\r*BRA /\x04LsAfY$LY4\x16|sQGZ #y'
b'Ptop overthinking, calm down, and relax'	b"F,'\x13\x1a{n]W!.ic)dc`:H -#J7`t\x1c(t&e1?I\x7f\x18` H}oT\x02A\x1f1s\x07*H"
b'Stop overthinking, palm down, and relax'	b"Rx\x1c\x7f-^I\x7f\x1e(iJl\ x16s6}]\x1ef\x18d x\x18\x13=s!g$f\x12~n"\x1 b``CJz\\OV\x0eF"
b'Stop overthinking, calm down, and relas'	b'Rx\x1c\x7f-^I\x7f\x1e(iJl\ x16s6RX>}w70\x04\r*BRA /\x04L\x0fp\x16iw}N0ewgq H5\x17\x02'

Terlihat bahwa perubahan pada suatu *bit*, merubah cipherteks pada blok tersebut dan setelahnya. Pada contoh kedua, perubahan pada *bit* pertama merubah hasil secara sepenuhnya. Perubahan pada *bit* di tengah merubah cipherteks pada blok tersebut dan seterusnya, namun tidak merubah cipherteks blok-blok awal. Sedangkan perubahan pada *bit* terakhir merubah blok cipherteks di akhir cipherteks.

Karena panjang kunci ditetapkan maka untuk melakukan serangan *brute force* maka dibutuhkan pengecekan sebanyak 2^128. Jika diasumsikan bahwa sebuah mesin dapat melakukan pengecekan sebanyak 10^8 setiap detiknya maka dibutuhkan waktu 1,0790 x 10^23 tahun. Selain itu, parameter rotor yang digunakan oleh mesin Enigma juga tidak diketahui karena berdasarkan dari kunci eksternal yang diberikan pengguna sehingga serangan *brute force* harus dilakukan pada rotor, posisi, dan juga *ring*. Enigma biasa yang hanya mendukung karakter alfabet (26 karakter) saja sulit ditembus, sedangkan Enigma yang digunakan pada *Enigma-Based Key Scheduler Block Cipher* mendukung 256 *bytes*. Selain itu, serangan seperti analisis frekuensi juga tidak dapat dilakukan karena frekuensi kemunculan *byte* sangat berbeda pada cipherteks yang dihasilkan.

V. Kesimpulan dan Saran

Dapat disimpulkan dari percobaan pada hasil implementasi di atas, *Enigma-Based Key Scheduler Block Cipher* merupakan algoritma yang kuat untuk melawan teknik-teknik kriptanalisis. Efek longsoran berjalan dengan baik dengan terjadinya perubahan bit pada cipherteks yang banyak pada setiap perubahan *bit* di plainteks. Hal tersebut dikarenakan *Enigma-Based Key Scheduler Block Cipher* memenuhi prinsip-prinsip perancangan *block cipher* dan memiliki teknik pembangkitan kunci putaran yang kuat karena menggunakan mesin Enigma yang juga bergantung pada kunci putaran sebelumnya. Walaupun fungsi putaran pada jaringan feistel tidak terlalu kompleks, namun terlihat bahwa hasil enkripsi dan dekripsi bekerja dengan sangat baik dengan tingginya keamaan yang didapatkan. Tingginya keamanan diketahui dari waktu yang dibutuhkan untuk mendapatkan kunci maupun parameter mesin Enigma yang digunakan untuk mendeskripsi cipherteks menjadi plainteks. Enigma biasa yang hanya mendukung karakter alfabet (26 karakter) saja sulit ditembus, sedangkan Enigma yang digunakan pada *Enigma-Based Key Scheduler Block Cipher* mendukung 256 *bytes*.

Salah satu kelemahan Enigma yang diimplementasi *Enigma-Based Key Scheduler Block Cipher* adalah tidak dimanfaatkannya plugboard yang dapat memberikan keamanan yang lebih tinggi dari penambahan satu buah rotor. Implementasi berikutnya dapat menurunkan *plugboard* dari kunci eksternal yang diberikan oleh pengguna sehingga tingkat keamanan dapat meningkat dan lebih sulit untuk dikriptanalisis.

VI. Daftar Referensi

[1] R. Munir, Kriptografi Modern, Jakarta, Indonesia: PT. Integra Komputindo, 2020.

[2] Munir, Rinaldi. 2020. Slide Kuliah IF4020 Kriptografi: Kriptografi Modern (Block Cipher, Bagian 2)

[3] National Institute of Standards and Technology, Data Encryption Standard (DES). (U.S.:U.S.Department of Commerce)

[4] National Institute of Standards and Technology, Advanced Encryption Standard (AES). (U.S.:U.S.Department of Commerce)

[5] Claude E. Shannon, "Communication Theory of Secrecy Systems", Bell System Technical Journal, vol. 28-4, pages 656–715, 1949.