# Cipher Blok JAFT

Mario Tressa Juzar (13512016)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
mariotj.tj@gmail.com

Rama Febriyan (13511067)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
rama.febriyan25@yahoo.co.id

Ahmad (13512033)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
ashahab28@gmail.com

***Abstrak*—Dalam makalah ini kami menyajikan JAFT sebagai algoritma cipher blok yang baru dalam rangka memenuhi tugas matakuliah Kriptografi. Blok cipher JAFT menggunakan kunci 128-bit dengan plainteks juga 128-bit, terdapat shiftCells, addRoundKey, subtituteSBox, shiftRow, xorOperation, dan diakhiri dengan jaringan Feistel.**

***Kata kunci—JAFT, shiftCells, addRoundKey, subtituteSBox, shiftRow, xorOperation.***

## I. PENDAHULUAN

Cipher blok merupakan algoritma kriptografi yang cukup banyak digunakan dan cukup mudah untuk dibuat dan dikembangkan dewasa ini. Algoritma cipher blok berdasarkan pada plainteks yang dijadikan blok terlebih dahulu begitupun juga dengan kunci yang digunakan juga dijadikan blok terlebih dahulu. Blok yang digunakan bervariasi mulai dari 64-bit, 128-bit, 192-bit dan lainnya yang juga bisa lebih besar lagi.

Seiring dengan semakin berkembangnya algoritma cipher blok maka juga akan berkembang ilmu kriptanalisis untuk menemukan kelemahan dan bahkan memecahkan sebuah algoritma cipher blok.

Kita tidak bisa hanya berpegang pada sebuah algoritma cipher blok saja, karena cepat atau lambat algoritma cipher blok tersebut akan bisa dikriptanalisis/dipecahkan dengan mudah. Untuk itu kami mencoba membuat sebuah algoritma cipher blok yang merupakan modifikasi dari algoritma cipher blok yang sudah ada.

Pada makalah ini kami coba menyajikan sebuah algoritma cipher blok yang kami buat dan kembangkan yang kami beri nama JAFT. JAFT banyak mengadopsi dari cipher blok Rijndael, terutama pada teknik *round key* dan putaran yang dilakukan, meskipun jumlah putaran JAFT berbeda dengan jumlah putaran pada Rijndael. Cipher blok Rijndael sendiri merupakan algoritma cipher blok yang merupakan standar AES. Pada JAFT lebih ditekankan teknik shiftCells yakni menggeser sel pada blok. Dan pada putaran terakhir blok dilewatkan pada jaringan Feistel.

## II. DASAR TEORI

### A. *Polyalphabetic Substitution Cipher*

*Polyalphabetic Substitution Cipher* atau cipher subtitusi abjad-majemuk menggunakan prinsip setiap huruf menggunakan kunci berbeda.

Misalkan plainteks:

$P = p_1 p_2 \ldots p_m p_{m+1} \ldots p_{2m} \ldots$

Maka cipherteksnya akan menjadi seperti berikut:

$E_k(P) = f_1(p_1)\, f_2(p_2) \ldots f_m(p_m)\, f_{m+1}(p_{m+1}) \ldots f_{2m}(p_{2m}) \ldots$

### B. *Chiper Blok*

Algoritma cipher blok merupakan salah satu dari algoritma kriptografi modern. Algoritma ini beroperasi dalam mode bit. Kunci, plainteks, dan cipherteks diproses dalam rangkaian bit. Algoritma ini tetap menggunakan gagasan pada algoritma kriptografi klasik seperti subtitusi dan transposisi, tetapi lebih rumit dan sangat sulit untuk dipecahkan.

Prinsipnya adalah sebagai berikut:

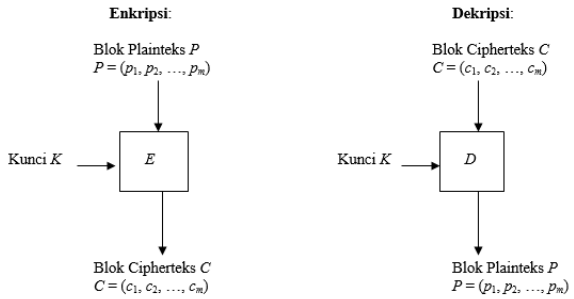
- Pesan dalam rangkaian bit dipecah menjadi beberapa blok
- *Padding bits* : merupakan bit-bit tambahan jika ukuran blok terakhir tidak mencukupi panjang blok. Padding bits mengakibatkan ukuran plainteks hasil deskripsi sedikit lebih besar dari plainteks semula.
- Pesan juga dapat dinyatakan dalam kode heksadesimal
- Bit-bit plainteks dibagi menjadi blok dengan panjang yang sama
- Panjang kunci enkripsi = panjang blok
- Enkripsi dilakukan terhadap blok bit plainteks menggunakan bit-bit kunci
- Algoritma enkripsi menghasilkan bok cipherteks yang panjangnya sama dengan panjang blok plainteks.

Blok plainteks berukuran *m* bit:

$P = (p_1, p_2, \ldots, p_m), \qquad p_i \in \{0, 1\}$

Blok cipherteks (*C*) berukuran *m* bit:

$C = (c_1, c_2, \ldots, c_m), \qquad c_i \in \{0, 1\}$




Gambar 1 : proses enkripsi dan dekripsi cipher blok

*C. Operasi XOR*

Operasi xor merupakan operasi pada level bit. Notasi yang digunakan untuk operasi xor adalah ⊕.

Operasi yang berlaku pada operasi xor:

$0 \oplus 0 = 0 \qquad 0 \oplus 1 = 1$

$1 \oplus 0 = 1 \qquad 1 \oplus 1 = 0$

Hukum-hukum yang terkait dengan operator xor:

(i) $a \oplus a = 0$

(ii) $a \oplus b = b \oplus a$

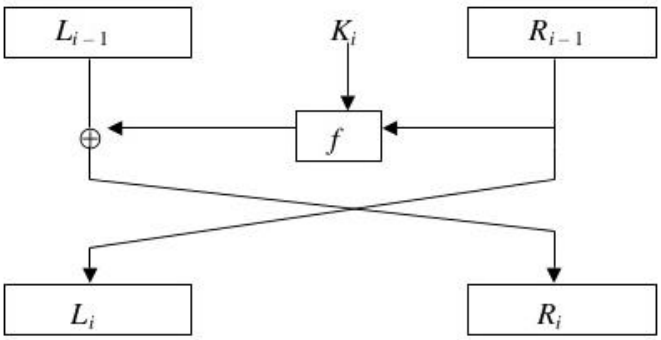
(iii) $a \oplus (b \oplus c) = (a \oplus b) \oplus c$

Operasi xor pada level bit : jika dua rangkaian dioperasikan dengan xor, maka operasinya dilakukan dengan meng-xor-kan setiap bit yang berkoresponden dari kedua rangkain bit tersebut.

Contoh: 10011 ⊕ 11001 = 01010

*D. Jaringan Feistel*

Jaringan Fiestel beroperasi terhadap panjang blok data tetap sepanjang n (genap), kemudian membagi 2 blok tersebut dengan panjang masing-masing n/2, yang dinotasikan dengan L dan R. Fiestel cipher menerapkan metode cipher berulang dengan masukan pada putaran ke-i yang didapat dari keluaran sebelumnya.




Gambar 2 : struktur jaringan feistel

Secara matematis dapat dinyatakan sebagai berikut:

$L_i = R_{i-1}$

$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$

Ki adalah kunci untuk putaran ke-I dan f adalah fungsi transformasi.

Blok plainteks adalah gabungan L dan R awal atau secara formal plainteks dinyatakan dengan ($L_0$, $R_0$). Sedangkan blok cipherteks didapatkan dari L dan R hasil putaran terakhir setelah terlebih dahulu dipertukarkan atau secara formal cipherteks dinyatakan dengan ($R_r$, $L_r$).

Jaringan Feistel banyak dipakai pada algoritma kriptografi DES, LOKI, GOST, FEAL, Lucifer, Blowfish, dan lain-lain karena model ini bersifat reversible untuk proses enkripsi dan dekripsi.Sifat reversible ini membuat kita tidak perlu membuat algoritma baru untuk mendekripsi cipherteks menjadi plainteks.

Contoh: $L_{i-1} \oplus f(R_{i-1}, K_i) \oplus f(R_{i-1}, K_i) = L_{i-1}$

Sifat reversible tidak bergantung pada fungsi f sehingga fungsi f dapat dibuat serumit mungkin.

## III. CIPHER BLOK JAFT

JAFT merupakan bagian dari cipher blok. JAFT menggunakan kunci dengan 128-bit. Kunci ini dalam format heksadesimal. Untuk kuncinya akan berbeda karena ada putaran pada proses enkripsinya sehingga kunci akan dibangkitkan sesuai dengan proses putaran.

Blok plainteks pada JAFT juga berukuran 128-bit dalam format heksadesimal. Ukuran bloknya yakni 4x4 dengan di masing-masing sel terdapat satu heksadesimal.

Secara garis besar cipher blok JAFT menggunakan prinsip-prinsip berikut:

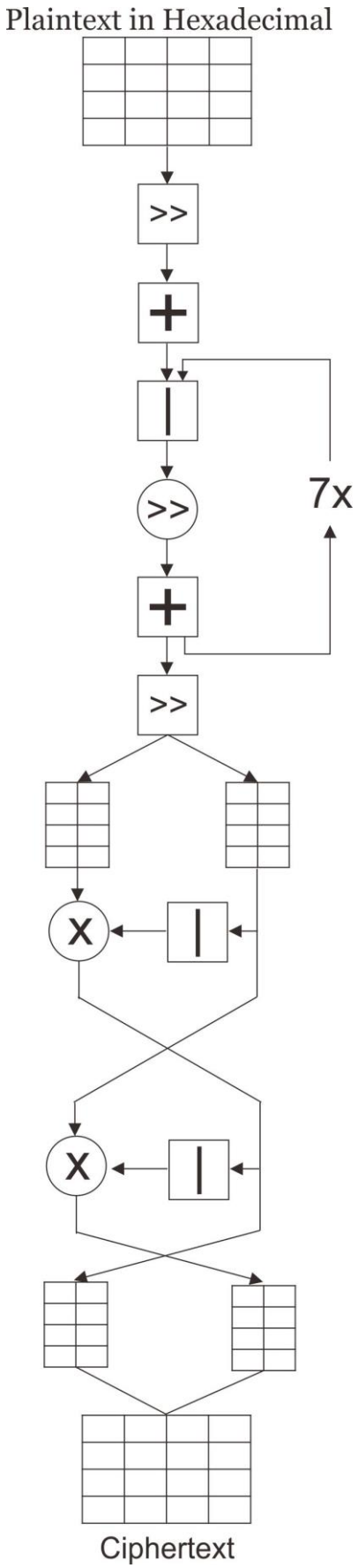
1. *shiftCells*. Pada blok, masing masing sel digeser ke kanan sebanyak kunci tertentu yang diambil dari key 128-bit, yang diambil hanya satu sel heksadesimal di bagian pojok kiri atas.
2. *addRoundKey*. Pada proses ini, kunci dibangkitkan sesuai putaran, kemudian kunci tersebut dilakukan operasi xor dengan blok plainteks.
3. *subtituteSBox*. Pada blok plainteks isi dari masing-masing sel diganti dengan heksadesimal yang ada pada S-Box yang telah ditentukan terlebih dahulu.
4. *shiftRow*. Pada blok plainteks masing-masing baris dilakukan pergeseran. Pada baris pertama tidak dilakukan pergeseran, pada baris kedua digeser sebanyak 1 ke kanan, pada baris ketiga digeser sebanyak 2 ke kanan, dan pada baris keempat digeser sebanyak 3 ke kanan.
5. *xorOperation*. Merupakan operasi xor biasa.

*A. Rancangan JAFT*

Rancangan algoritma JAFT yakni sebagai berikut:

1. Pertama, plainteks diubah kedalam format heksadesimal 128-bit. Untuk kunci juga memiliki panjang 128-bit, diubah ke format heksadesimal.
2. Blok plainteks kemudian diterapkan *shiftCells*.
3. Hasilnya di xor dengan *key* pada proses *roundkey*. *Key* kemudian di subtitusi dengan S-Box, kita dapatkan *roundkey* 1.
4. Masuk dalam *looping*, pertama blok plainteks di subtitusi dengan S-Box.
5. Selanjutnya diterapkan *shitRow*, yakni menggeser barisnya.
6. Kemudian diterapkan *addroundkey* blok dengan *round key* 1.
7. *Key* 1, kolom 1 dan kolom 4 di xor terlebih dahulu untuk mendapatkan kolom 5. Kolom 1 dihapus, lalu kolom 2 sampai kolom 5 merupakan *round key* 2. Begitu seterusnya.
8. Ulangi dari langkah 3 sampai 7 kali melewati putaran.
9. Hasil setelah keluar dari *looping* kemudian diterapkan kembali *shiftCells*.
10. Kemudian blok dilewatkan pada jaringan Feistel. Fungsi yang ada dalam jaringan Feistel yakni subtitusi dan operasi xor.
11. Setelah melewati jaringan Feistel maka didapatkanlah hasil cipherteksnya. Inilah cipherteks hasil dari cipher blok JAFT.

Rancangan cipher blok JAFT dapat dilihat pada gambar 3.




Gambar 3 : enkripsi JAFT

Keterangan:


Untuk proses dekripsi tidak sulit karena bisa dengan seperti membalikkan proses enkripsi (memulai dari bawah untuk kembali mendapatkan plainteks).

*B. Pengaturan shiftCells*

*shiftCells* merupakan fungsi untuk menggeser setiap sel pada blok sebesar n kearah kanan. Angka n didapat dari kunci, diambil satu buah sel dan mengambil isi heksadesimalnya untuk mendapatkan nilai n tersebut, setelah didapat nilai n kemudian digeser sebesar n ke kanan.

Untuk *shiftCells* yang pertama, nilai n didapat dengan formula

n = ((int) key1 mod 15) + 1

Untuk *shiftCells* yang kedua, nilai n didapat dengan formula

n = ((int) key7 mod 15) + 1

*key1* berarti *key* pertama sebelum putaran dilakukan, diambil yang pojok kiri atas. *key7* berarti *key* pada putaran ke-7, untuk heksadesimal yang diambil merupakan heksadesimal di posisi pojok kanan atas.

Formula di atas menjamin akan adanya pergeseran pada blok plainteks, minimal terjadi satu pergeseran dan maksimal terjadi 15 pergeseran.

*C. Penjadwalan kunci*

Kunci pada putaran didapat dari turunan kunci yang ada sebelumnya. Pada pembuatan kunci baru untuk setiap putaran dilakukan subtitusi terlebih dahulu terhadap blok kunci, kemudian blok pada kolom n dilakukan operasi xor dengan kolom n-3 untuk mendapatkan kolom baru n+1, dengan n>=4.

Pada JAFT pembangkitan kunci tidak tergantung pada blok plainteks yang sedang diproses, pembangkitan kunci bisa dilakukan secara independen tanpa terkait dengan blok plainteks. Kunci hanya terkait dengan S-Box yakni pada saat subtitusi heksadesimal.

*D. S-Box*

S-Box yang digunakan berukuran 16x16 karena mengikuti format heksadesimal yang telah digunakan. S-Box yang digunakan adalah S-Box yang sama dengan S-Box pada blok cipher Rijndael.

*E. Pengaturan shiftRow*

*ShiftRow* merupakan operasi untuk menggeser setiap sel pada baris tertentu di dalam blok. Untuk pengaturannya, pada baris pertama tidak dilakukan pergeseran sel, pada baris kedua dilakukan pergeseran sel sebanyak 1 ke kanan, pada baris ketiga dilakukan pergeseran sebanyak 2 ke kanan, dan pada baris keempat dilakukan pergeseran sebanyak 3 ke kanan.

## IV. EKSPERIMEN DAN PEMBAHASAN HASIL

Untuk menguji algoritma yang telah dibuat, dilakukan uji coba enkripsi pesan. Pengujian dilakukan kepada dua buah pesan teks dengan kunci yang berbeda.

Pengujian pertama:

pesan teks : RAMAFEBRIYANRAMA

kunci : 2b 28 ab 09 7e ae f7 cf 15 d2 15 4f 16 a6 88 3c

hasil enkripsi:

```
run:
j»L ,]þuª 6É
Û
{BUILD SUCCESSFUL
```

Gambar 4 : hasil enkripsi percobaan pertama

Kunci pesan pada pengujian pertama diambil dari serangkaian bilangan heksadesimal dari salah satu referensi. Setiap karakter pada kunci merupakan karakter ASCII yang tidak dapat dibaca.

Pengujian kedua:

pesan teks: cipher blok jaft

kunci : tubeskriptografi

hasil enkripsi:

```
run:
3¥É†ÓÔ÷JËZ " ã CBUILD SUCCESSFUL
```

Gambar 5 : hasil enkripsi percobaan kedua

Setelah dilakukan enkripsi, tidak terlihat adanya hubungan antara cipherteks dengan plainteks pada kedua hasil pengujian. Pada pengujian pertama bahkan muncul karakter new line yang heksadesimalnya 0x0A. Kemunculan karakter ini sama sekali tidak dapat diduga. Pasalnya, pada plainteks sama sekali tidak ada karakter yang dapat memicu munculnya karakter new line.

Berdasarkan pengujian enkripsi yang telah dilakukan, karakter yang muncul menjadi tidak beraturan bahkan untuk huruf yang sama memiliki cipherteks yang berbeda secara keseluruhan. Ini tentunya akan meingkatkan keamanan dari pesan karena tidak terlihat adanya suatu pola pada cipherteks.

Untuk proses dekripsi pesan, plainteks yang didapatkan sebagai hasil dekripsi sama dengan plainteks awal saat proses enkripsi. Ini membuktikan bahwa proses dekripsi dapat dilakukan dengan cara yang berlawanan dengan enkripsi, sehingga tidak dibutuhkan suatu algoritma khusus untuk proses dekripsi.

## V. ANALISIS KEAMANAN

Aspek keamanan merupakan aspek yang selalu diutamakan dalam pembuatan algoritma kriptografi. Keamanan mencakup aspek kerahasiaan, aspek integritas, serta aspek otentik. Cakupan ini merupakan tujuan utama dari ilmu kriptografi modern.

Untuk menganalisis keamanan dari cipher blok JAFT, kami membahas dari dua sudut pandang. Analisis pertama merupakan analisis *key space*, serta analisis kedua merupakan analisis *known plain-text attack*.

### *A. Analisis Key Space*

*Key space* merupakan nilai yang menyatakan berapa banyaknya jumlah bit yang digunakan untuk mengenkripsi suatu *plain text*. Ukuran dari kunci haruslah cukup besar agar dapat menghindari serangan *brute force search attack*. Misalkan panjang dari kunci yang digunakan adalah sepanjang N bits, maka akan ada kombinasi sebanyak $2^N$ kombinasi nilai yang mungkin untuk kunci tersebut. Apabila nilai N sangat besar, maka akan sangat sulit bagi penyerang untuk mencari semua kemungkinan dari kunci tersebut. Adapun untuk kasus terburuk, jumlah minimum pencarian yang harus dieksekusi oleh penyerang untuk mendapatkan kunci yang sesuai seharusnya senilai dengan $2^{N/2}$.

Keamanan dari suatu algoritma kriptografi ditentukan oleh kemampuannya untuk bertahan dari serangan *brute force*. Untuk algoritma JAFT ini dengan ukuran dari kunci yang maksimal yaitu N = 128, maka akan dibutuhkan waktu sekitar 5.61e+36 tahun (dengan asumsi processor membutuhkan waktu satu detik untuk satu juta proses) untuk melakukan pencarian *brute force*. Hal ini tidak sebanding dengan waktu untuk pengiriman pesan dibandingkan dengan waktu untuk pencarian kata kunci yang sesuai. Sehingga menurut analisis *key space* algoritma JAFT merupakan algoritma yang aman dari serangan *brute force*.

### *B. Analisis Known Plain-Text Attack*

Untuk menguji segi keamanan lainnya dari algoritma JAFT, kami mencoba melihat bagaimana pengaruh ketahanannya ketika diberikan serangan *known plain-text attack*. Untuk kasus mode operasi ECB pada *plain-text*, analisis untuk mengetahui bagaimana pola dari tiap blok akan samar-samar karena pada algoritma JAFT setiap bit posisinya akan diubah berkala dan diterapkan *substitueSBox* sehingga tidak dapat dianalisa bagaimana pola dari kunci yang ingin ditebak, hal ini menyebabkan JAFT tidak dapat diserang secara statistik.

Algoritma JAFT juga menerapkan mode operasi CBC, sehingga dibuat adanya ketergantungan antar blok. Setiap blok cipherteks bergantung tidak hanya pada blok plainteksnya saja tetapi bergantung pada blok plainteks sebelumnya. Hal ini menyebabkan cipherteks makin sulit untuk dipecahkan oleh kriptanalis.

## VI. KESIMPULAN DAN SARAN

JAFT merupakan algoritma cipher blok yang menerapkan blok pesan 128-bit dan panjang kunci juga 128-bit.JAFT memiliki keunikan di shiftCells dengan menggeser sel pada blok. Putaran pada proses enkripsi dilakukan sebanyak 7 kali. Kemudian fase terakhir dengan melewatkan pada jaringan Feistel.

Dari analisis keamanan, *key space* dari JAFT sangat besar dengan menggunakan 128-bit, butuh waktu yang sangat lama untuk bisa memecahkannya, pada blok juga memiliki ketergantungan dengan blok yang lain, ini sudah meyakinkan bahwasanya algoritma cipher blok JAFT aman dari serangan kriptanalisis.

## REFERENSI


[1] Joan Daemen and Vincent Rijmen, “The Block Cipher Rijndael”.

[2] Paulo S.L.M. Barreto and Vincent Rijmen, “The KHAZAD Legacy-Level Block Cipher”.

[3] Alex Biryukov, “Analysis of Involutional Ciphers: Khazad and Anubis”.

[4] Zheng Gong, Svetla Nikova, and Yee Wei Law, “KLEIN: A New Family of Lightweight Block Ciphers”.

[5] Ronald L. Rivest, M.J.B. Robshaw, R. Sidney, and Y.L. Yin, “The RC6 Block Cipher”.

[6] Deukjo Hong, Jaechul Sung, Seokhie Hong, Jongin Lim, Sangjin Lee, Bon-Seok Koo, Changhoon Lee, Donghoon Chang, Jesang Lee, Kitae Jeong, Hyun Kim, Jongsung Kim, and Seongtaek Chee, “HIGHT: A New Block Cipher Suitable for Low-Resource Device”.