# PERBANDINGAN ALGORITMA KRIPTOGRAFI DES DENGAN ICE

Nama: Ricky Gilbert Fernando
NIM: 13505077

*Program Studi Teknik Informatika, Institut Teknologi Bandung*
*Jl. Ganesha 10, Bandung*
E-mail: if15077@students.if.itb.ac.id

## Abstrak

Makalah ini membahas tentang studi perbandingan algoritma kriptografi antara *Data Encryption Standard* (DES) dan *Information Concealment Engine* (ICE). DES adalah sebuah *block cipher* yang dipilih oleh *National Bureau of Standards* (NBS) sebagai *Federal Information Processing Standard* (FIPS) resmi yang digunakan Amerika Serikat pada 1976 dan telah dinikmati secara internasional. Algoritma ini bekerja dengan menggunakan kunci 56 *bits*. Sayangnya algoritma ini menimbulkan banyak kecurigaan dimana terdapat sejumlah elemen dari DES yang dirahasiakan, panjang kunci yang pendek, dan adanya kemungkinan *backdoor* yang telah diketahui oleh *National Security Agency* (NSA). Kecurigaan ini mendorong banyak peneliti untuk mempelajari DES lebih mendalam hingga menemukan kriptanalisis DES.

ICE adalah sebuah *block cipher* yang dipublikasikan oleh Kwan pada 1997. Algoritma ini memiliki struktur yang mirip dengan DES, tetapi dengan tambahan permutasi bit yang tidak tergantung kunci dalam fungsi putarannya. Terdapat berbagai jenis varian ICE, yaitu Thin-ICE, standar ICE, dan ICE-*n*. Perbedaan di antara ketiganya adalah panjang kata kunci yang digunakan dan jumlah putaran. Algoritma Thin-ICE menggunakan kunci 64 *bits* dan 8 putaran. Standar ICE menggunakan kunci 64 *bits* dan 16 putaran. Algoritma ICE-*n* menggunakan kunci 64*n bits* dan 16n putaran. Penggunaan jenis algoritma dapat disesuaikan dengan kebutuhan pengguna dimana Thin-ICE memiliki tingkat keamanan terendah di antara ketiganya, sedangkan ICE-*n* yang tertinggi. Algoritma ini tidak menjadi subyek paten dan *source code* dapat digunakan dengan bebas.

Pada bagian selanjutnya akan dibahas mengenai implementasi DES dan ICE, serta pengukuran performansi. Pengukuran ini dilakukan dengan cara mencoba masing-masing algoritma untuk mengenkripsi sejumlah berkas, lalu diukur *memory usage* dan waktu yang dibutuhkan. Setelah melihat dan membandingkan *memory usage* serta waktu enkripsi yang digunakan oleh masing-masing algoritma, akan ditarik kesimpulan mengenai keunggulan, kekurangan, dan penggunaan yang cocok untuk baik DES maupun ICE.

**Kata kunci**: *Data Encryption Standard, Information Concealment Engine, block cipher*, perbandingan, *memory usage,* waktu enkripsi.

## 1. Pendahuluan

Pada jaman informasi sekarang ini, penerimaan dan penyebaran data adalah hal yang selalu terjadi. Sebagai contoh, teknologi *web* yang sedang populer saat ini. Melalui jaringan internet, setiap orang dengan mudah mengirimkan dan menerima data dari mana saja dan kapan saja. Pengguna fasilitas internet inipun berasal dari jenis pekerjaan yang bervariasi. Sejumlah pengguna hanya membutuhkan kemudahannya dalam transfer data, tetapi terdapat juga sejumlah pengguna yang mengutamakan keamanan data selama pengirimannya. Di antara pengguna tersebut, terdapat sejumlah kelompok pengusaha, pengatur ekonomi, dan pemerintahan dimana data yang mereka ingin kirim atau terima adalah data sensitif. Sayangnya, tidak seluruh teknologi internet mampu menjamin keamanan data yang mengalir di dalamnya. Terdapat sejumlah saluran komunikasi yang tidak terenkripsi sehingga memungkinkan pihak ketiga yang tidak boleh mengetahui isi data, dapat menyadap saluran tidak aman tersebut dan dapat melihat isi data tersebut. Untuk mengatasi masalah tersebut, digunakan kriptografi sebagai pengacak data yang akan dikirim sehingga hanya orang-orang yang berwenang saja yang dapat mengetahui isi dari data tersebut.

Dalam kriptografi, terdapat sejumlah algoritma yang telah ditemukan dalam mengacak data. Algoritma kriptografi tertua yang diketahui adalah *Caesar cipher* yang digunakan ketika jaman Romawi untuk mengacak isi pesan yang dikirim oleh *Caesar* kepada tujuannya. Pengacakan dilakukan dengan menggeser setiap huruf pada pesan sebanyak *n* kali, dimana *n* ditentukan oleh pengirim. Walaupun pesan yang telah terenkripsi tersebut diketahui oleh musuhnya, si musuh tidak akan mengetahui isi pesan tersebut dan mungkin menganggapnya sebagai pesan tanpa arti.

Pembentukan DES berasal pada awal 1970. Pada 1972, setelah menyelesaikan studi dalam mempelajari kebutuhan keamanan pada komputer pemerintah Amerika Serikat, badan pengatur standar Amerika Serikat, NBS, sekarang bernama *National Institute of Standards and Technology* (NIST), mengidentifikasikan sebuah kebutuhan akan standar pemerintahan untuk enkripsi informasi sensitif dan rahasia. Pada 15 Mei 1973, setelah berkonsultasi dengan NSA, NBS menerima banyak proposal untuk sebuah *cipher* yang dapat memenuhi kriteria desain yang diberikan. Tetapi tidak ada pendaftar yang dapat diterima. Permintaan kedua dikeluarkan pada 27 Agustus 1974. Kali ini, IBM mengumpulkan sebuah kandidat yang terlihat dapat diterima, sebuah *cipher* yang dikembangkan pada periode 1973-1974 berdasarkan algoritma yang dibangun berdasarkan algoritma pendahulunya, yaitu Horst Feistel's *Lucifer cipher*. Tim dari IBM yang bertanggung jawab dalam desain dan analisis *cipher* terdiri dari Feistel, Walter Tuchman, Don Coppersmith, Alan Konheim, Carl Meyer, Mike Matyas, Roy Adler, Edna Grossman, Bill Notz, Lynn Smith, and Bryant Tuckerman.Tidak seperti DES yang dibangun untuk memenuhi sebuah kebutuhan enkripsi pada seluruh kantor pemerintahan, algoritma ICE dibuat karena keinginan pengembangnya, yaitu Michael Kwan. ICE termasuk *block cipher* kunci rahasia. Algoritma lainnya yang termasuk di dalamnya adalah DES, IDEA, LOKI, dan FEAL. Bagian kunci rahasia berarti keamanannya bergantung pada kunci yang harus dijaga rahasia, tidak seperti *cipher public-key*, contohnya RSA, dimana keamanannya bergantung pada properti matematis kuncinya.

### *2. Block Cipher*

*Block cipher* adalah sebuah algoritma enkripsi kunci simetris yang mentransformasikan sebuah blok data *plaintext* dengan panjang tetap menjadi sebuah blok data *ciphertext* dengan ukuran yang sama. Transformasi ini berlangsung dengan menggunakan kunci rahasia yang disediakan *user*. Proses dekripsi dilakukan dengan menjalankan proses transformasi kebalikan kepada blok *ciphertext* dengan menggunakan kunci yang sama. Panjang tetap yang digunakan dinamakan ukuran blok, dan untuk banyak *block ciphers*, ukuran blok adalah 64 *bits*. Untuk tahun ke depan ukuran blok akan ditingkatkan menjadi 128 *bits*. Karena blok *plaintext* yang berbeda dipetakan ke blok *ciphertext* yang berbeda juga (untuk memungkinkan dekripsi yang unik), sebuah *block cipher* menyediakan sebuah fungsi permutasi (korespondensi satu-satu yang dapat dibalikkan) untuk semua pesan.

Ketika *block cipher* digunakan untuk mengenkripsi sebuah pesan dengan panjang tertentu, digunakan sebuah teknik yang dikenal sebagai mode operasi untuk *block cipher*. Agar dapat digunakan, sebuah mode harus setidaknya seaman dan seefisien *cipher* yang mendasarinya. Standar mode yang tersedia adalah *Electronic Code Book* (ECB), *Cipher Block Chaining* (CBC), *Cipher Feedback* (CFB), dan *Output Feedback* (OFB). Pada makalah ini, mode yang digunakan untuk enkripsi *plaintext* adalah ECB.

Dengan *block cipher*, blok *plaintext* yang sama akan dienkripsi menjadi blok *ciphertext* yang sama bila digunakan kunci yang sama pula. Ini berbeda dengan *cipher* aliran dimana bit-bit *plaintext* yang sama akan dienkripsi menjadi bit-bit *ciphertext* yang berbeda setiap kali dienkripsi.

Sebagai contoh, misalkan terdapat sebuah *plaintext x*. *Plaintext* tersebut dapat dilihat sebagai kumpulan blok:

$$x = x_1 x_2 \ldots x_n$$

dimana $x_m$, $1 \le m \le n$, merupakan satuan blok dari *ciphertext* yang berukuran 64 *bits*.

Ketika *user* mengenkripsi *x*, proses enkripsi dilakukan per blok sehingga menghasilkan *ciphertext y* yang dapat dilihat sebagai berikut:

$$y = y_1 y_2 \ldots y_n = e_K(x_1) e_K(x_2) \ldots e_K(x_n)$$

Jika difokuskan pada sebuah blok, proses enkripsi akan terlihat sebagai berikut:

$$y_n = e_K(x_n),$$

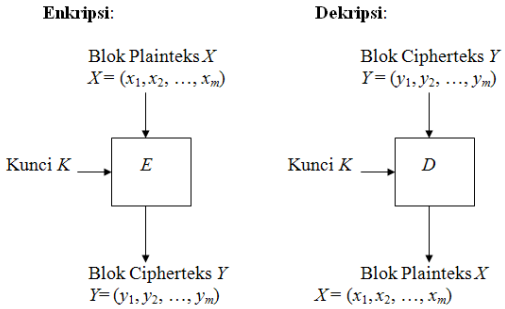
dimana *n* adalah blok ke-*n* dari *plaintext*. Sedangkan proses dekripsi block cipher akan terlihat sebagai berikut:

$$x_n = d_K(y_n),$$

Oleh karena itu, fungsi dekripsi harus merupakan fungsi kebalikan dari enkripsi.

$$d = e_{-1}$$

Gambar 2-1 Skema Enkripsi dan Dekripsi Block Cipher [MUN06] di bawah menampilkan proses enkripsi dan dekripsi *block cipher* secara umum.



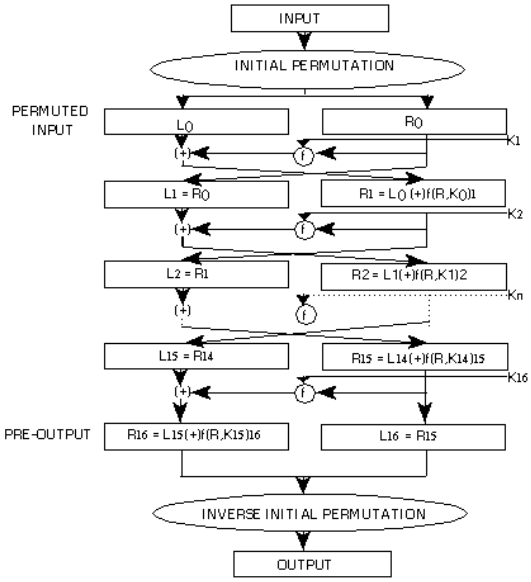

**Gambar 2-1 Skema Enkripsi dan Dekripsi Block Cipher [MUN06]**

Keterangan:

1. Diagram kiri menggambarkan proses enkripsi, sedangkan diagram kanan proses dekripsi.
2. E : proses enkripsi dengan masukan *plaintext X* dan kunci *K* yang menghasilkan *ciphertext Y*.
3. D : proses dekripsi dengan masukan *ciphertext Y* dan kunci *K* yang menghasilkan *plaintext X*.

## 3. Data Encryption Standard (DES)

Algoritma DES dirancang untuk enkripsi dan dekripsi blok data yang terdiri dari 64bits dibawah control kunci 64 bits. Proses dekripsi harus dijalankan dengan menggunakan kunci yang sama dengan kunci enkripsi, tetapi dengan penjadwalan kunci yang berbeda sehingga proses dekripsi adalaha kebalikan dari proses enkripsi. Sebuah blok yang akan dienkripsi dikenakan sebuah initial permutation (IP), lalu komputasi yang rumit dan bergantung pada kunci, dan akhirnya sebuah permutasi yang merupakan kebalikan dari IP, yaitu $IP^{-1}$. Gambar 3-1 menunjukkan skema algoritma DES.




**Gambar 3-1 Skema Algoritma DES**

### 3.1 *Initial Permutation* dan Kebalikannya

Pada tahap pertama, dilakukan IP. Prosesnya adalah sebagai berikut, masukan blok plaintext yang berukuran 64 bits, dimasukan ke dalam tabel permutasi yang ditunjukkan Gambar 3-2.

**IP**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 58 | 50 | 42 | 34 | 26 | 18 | 10 | 2 |
| 60 | 52 | 44 | 36 | 28 | 20 | 12 | 4 |
| 62 | 54 | 46 | 38 | 30 | 22 | 14 | 6 |
| 64 | 56 | 48 | 40 | 32 | 24 | 16 | 8 |
| 57 | 49 | 41 | 33 | 25 | 17 | 9 | 1 |
| 59 | 51 | 43 | 35 | 27 | 19 | 11 | 3 |
| 61 | 53 | 45 | 37 | 29 | 21 | 13 | 5 |
| 63 | 55 | 47 | 39 | 31 | 23 | 15 | 7 |

**Gambar 3-2 Initial Permutation**

Keterangan Gambar 3-2:

1. Bit ke-58 dijadikan bit ke-1.
2. Bit ke-50 dijadikan bit ke-2.
3. Bit ke-42 dijadikan bit ke-3.
4. Dan seterusnya hingga bit ke-7 dijadikan bit ke-64.

Pada tahap terakhir, keluaran komputasi rumit yang bergantung kunci yang terdapat pada Gambar 3-1, dimasukkan ke dalam kebalikan dari IP yang dinamakan sebagai $IP^{-1}$.

**$IP^{-1}$**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 40 | 8 | 48 | 16 | 56 | 24 | 64 | 32 |
| 39 | 7 | 47 | 15 | 55 | 23 | 63 | 31 |
| 38 | 6 | 46 | 14 | 54 | 22 | 62 | 30 |
| 37 | 5 | 45 | 13 | 53 | 21 | 61 | 29 |
| 36 | 4 | 44 | 12 | 52 | 20 | 60 | 28 |
| 35 | 3 | 43 | 11 | 51 | 19 | 59 | 27 |
| 34 | 2 | 42 | 10 | 50 | 18 | 58 | 26 |
| 33 | 1 | 41 | 9 | 49 | 17 | 57 | 25 |

**Gambar 3-3 Kebalikan Initial Permutation**

Pada tahap 16 komputasi, masukan yang diterima adalah blok yang telah dipermutasikan. Blok yang berukuran 64 bits ini dipandang sebagai 2 (dua) blok berukuran 32 bits, dimana blok sebelah kiri diberi simbol L dan blok sebelah kanan diberi simbol R. Terdapat K sebagai sebuah blok 48 bits yang dipilih dari kunci 64 bits. Untuk komputasi ke-1 hingga ke-16,

$$L_n = R_{n-1}$$
$$R_n = L_{n-1} (+) f(R_{n-1}, K_n)$$

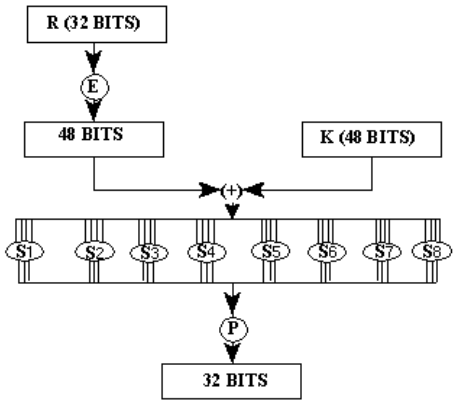
, dimana $1 \le n \le 16$ yang menyatakan komputasi ke-n dan Kn dinyatakan sebagai berikut:

$$K_n = KS(n, KEY)$$

Hal yang perlu diperhatikan adalah komputasi ke-16. Keluaran dari komputasi ke-16 yang akan menjadi masukan bagi proses kebalikan initial permutation tidak berbentuk $L_{16}R_{16}$, tetapi $R_{16}L_{16}$. Pada tahap komputasi, blok *plaintext* dikomputasikan dengan penghitungan di atas dan melalui fungsi *cipher f* yang akan dijelaskan pada bagian berikutnya.

### 3.2 Fungsi *cipher f*

Sebuah sketsa dari kalkulasi f(R,K) ditunjukkan oleh Gambar 3-4.




**Gambar 3-4 Skema Fungsi *Cipher f***

E adalah sebuah fungsi yang menerima sebuah blok 32 bits sebagai masukan dan menghasilkan sebuah blok 48 bits sebagai keluaran. Fungsi E menghasilkan keluaran 48 bits yang ditulis sebagai 8 blok yang masing-masing berukuran 6 bits, yang didapat dengan memasukkan blok 32 bits ke dalam tabel yang ditunjukkan oleh Gambar 3-5.

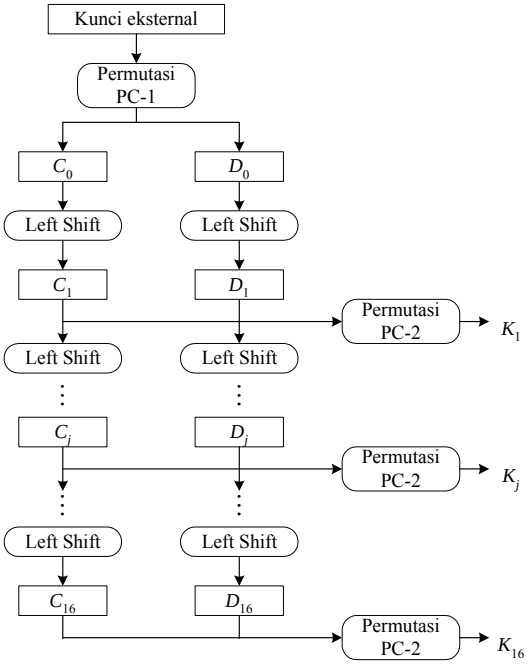
**E BIT-SELECTION TABLE**

| | | | | | |
|---|---|---|---|---|---|
| 32 | 1 | 2 | 3 | 4 | 5 |
| 4 | 5 | 6 | 7 | 8 | 9 |
| 8 | 9 | 10 | 11 | 12 | 13 |
| 12 | 13 | 14 | 15 | 16 | 17 |
| 16 | 17 | 18 | 19 | 20 | 21 |
| 20 | 21 | 22 | 23 | 24 | 25 |
| 24 | 25 | 26 | 27 | 28 | 29 |
| 28 | 29 | 30 | 31 | 32 | 1 |

**Gambar 3-5 Tabel Fungsi Seleksi Bit *E***

Lalu hasilnya akan ditambahkan dengan kunci berukuran 48 bits hingga membentuk $S_1,S_2,\ldots,S_8$.

### 3.3 Pembangkitan Kunci Internal




**Gambar 3-6 Skema Pembangkitan Kunci Internal**

Pada tahap pertama, kunci eksternal yang merupakan kunci masukan user yang berukuran 64 bits, dimasukkan ke dalam permutasi PC-1. Proses permutasi PC-1 akan mengkompresi panjang kunci menjadi 56 bits. Tabel 3-1 menampilkan tabel permutasi PC-1 yang digunakan untuk mengkompresi kunci eksternal menjadi 56 bits.

**Tabel 3-1 Permutasi PC-1**

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 57 | 49 | 41 | 33 | 25 | 17 | 9 | 1 | 58 | 50 | 42 | 34 | 26 | 18 |
| 10 | 2 | 59 | 51 | 43 | 35 | 27 | 19 | 11 | 3 | 60 | 52 | 44 | 36 |
| 63 | 55 | 47 | 39 | 31 | 23 | 15 | 7 | 62 | 54 | 46 | 38 | 30 | 22 |
| 14 | 6 | 61 | 53 | 45 | 37 | 29 | 21 | 13 | 5 | 28 | 20 | 12 | 4 |

Kunci hasil permutasi PC-1 dibagi menjadi 2 bagian sama besar yang berukuran 28 bits, dan diberi simbol $C_nD_n$, dimana *n* menyatakan komputasi yang sedang berlangsung. Masing-masing bagian digeser bit ke kiri sebanyak 1 atau 2 bit yang ditentukan oleh Tabel 3-2.

**Tabel 3-2 Tabel Pergeseran Bit pada Fungsi *Cipher f***

| Putaran i | Jumlah pergeseran bit |
|---|---|
| 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 |

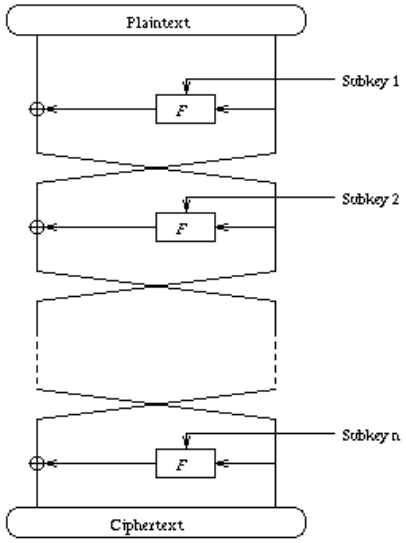
Setelah itu, hasilnya dimasukkan ke dalam tabel permutasi PC-2 yang ditunjukkan oleh Tabel 3-3 yang menjadi kunci komputasi ke-*n*.

**Tabel 3-3 Permutasi PC-2**

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 14 | 17 | 11 | 24 | 1 | 5 | 3 | 28 | 15 | 6 | 21 | 10 |
| 23 | 19 | 12 | 4 | 26 | 8 | 16 | 7 | 27 | 20 | 13 | 2 |
| 41 | 52 | 31 | 37 | 47 | 55 | 30 | 40 | 51 | 45 | 33 | 48 |
| 44 | 49 | 39 | 56 | 34 | 53 | 46 | 42 | 50 | 36 | 29 | 32 |

## 4. Information Concealment Engine (ICE)

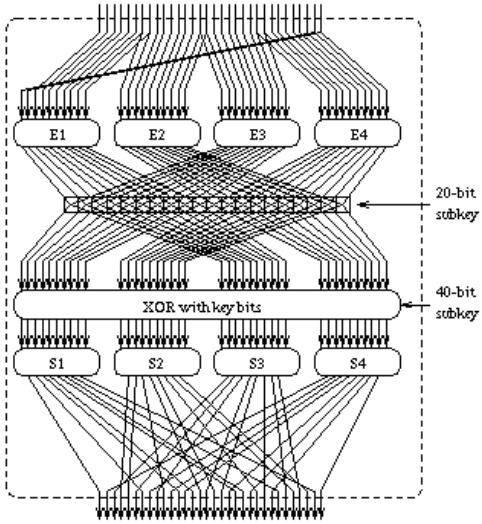
ICE adalah sebuah Feistel block cipher standar, seperti yang ditunjukkan pada Gambar 4-1.




**Gambar 4-1 Struktur Algoritma ICE**

Algoritma ini menerima plaintext 64 bits yang dibagi menjadi 2 bagian berukuran 32 bits. Dalam setiap putarannya, bagian kanan dan sebuah upa kunci 60 bits dimasukkan ke dalam fungsi *f*. Keluaran dari fungsi *f* di-XOR-kan dengan bagian kiri, lalu kedua bagian saling ditukarkan. Hal ini dilakukan terus hingga putaran terakhir, dimana pertukaran final tidak dilakukan. Jumlah putaran ditentukan oleh level varian yang digunakan. Level 0 (atau Thin-ICE menggunakan 8 putaran, sedangkan untuk level *n* menggunakan 16*n* putaran.

### 4.1 Fungsi *f*




**Gambar 4-2 Fungsi *f* dalam ICE**

Fungsi ini yang ditunjukkan pada Gambar 4-2 mirip dengan fungsi yang digunakan dalam DES, kecuali penggunaan permutasi kunci, yang diatur oleh upa kunci 20 bits dalam setiap putarannya. Pada dasarnya, 20 bits ini menentukan jalan yang akan diambil oleh bits yang meninggalkan E1, E2, E3, dan E4. Jika sebuah bit dalam upa kunci diset, bit yang bersesuaian dari E1 atau E2 akan ditukarkan dengan sebuah bit dari E3 dan E4. Jika upa kunci tidak diset, bits tersebut akan berlanjut tidak diubah-ubah. Permutasi kunci ini melakukan hal yang sama dengan nilai *salt* dalam perintah Unix crypt, kecuali *salt* berukuran 20 bits, bukan 12 bits, dan diturunkan dari kunci rahasia dalam setiap putaran daripada dibuat tetap dan diketahui umum.

S-box menerima masukan 10 bits dan menghasilkan keluaran 8 bits. Bagian kiri dan kanan dari masukan digabung untuk menghasilkan R, sebuah selektor baris 2 bits. Delapan bits di tengah membentuk C, selektor

kolom. Untuk setiap baris R, terdapat sebuah XOR *offset O* dan sebuah ruang Galois prima *P*. Keluaran 8 bits dari S-box diberikan oleh

$$(C \text{ xor } O)^7 \bmod P$$

dalam ruang Galois matematis 8 bits. Keluaran dari S-box dipermutasikan menjadi sebuah nilai berukuran 32 bits melalui sebuah P-box. P-box didesain untuk memaksimalkan penyebaran dari setiap S-box dan untuk memastikan bahwa bits yang dipisahkan oleh 16 tempat tidak berasal dari S-box yang sama atau dari S-box yang berbeda oleh 2 tempat, contoh $S_1$ dan $S_3$.

### 4.2 Penjadwalan Kunci

Penjelasan singkat mengenai penjadwalan kunci untuk berbagai macam varian ICE dijelaskan sebagai berikut:

1. Level 0 (Thin-ICE) menggunakan 8 putaran pertama dari penjadwalan ICE standar (level 1).
2. Level 1 menggunakan penjadwalan 16 putaran yang diturunkan dari kunci 64 bits. Setiap putaran menggunakan 60 bits, sehingga setiap kunci digunakan 15 kali. Di antara putaran, bits dipermutasikan dan setelah setiap kali sebuah bit kunci digunakan, bit tersebut dibalikkan.
3. Level yang lebih tinggi dari ICE memperluas penjadwalan kunci dari level di bawahnya. Hal dilakukan dengan mengggunakan penjadwalan kunci, memecahnya ada poin pertengahan, dan memasukkan 16 putaran tambahan di tengah-tengah. Enam belas putaran baru ini menggunakan penjadwalan ICE standar yang diturunkan dari 64 bits selanjutnya dari kunci.

## 5. Perbandingan Performansi

Pada bagian ini akan dibahas mengenai perbandingan performansi dari kedua algoritma. Cara yang digunakan dalam perbandingan adalah

1. Kedua algoritma telah dibentuk menjadi program yang mampu mengenkripsi berkas, baik berkas dokumen, gambar, lagu, atau film.
2. Disediakan 3 (tiga) buah berkas, yaitu berkas dokumen yang berukuran 4MB, berkas lagu berukuran 20MB, dan berkas acak berukuran 45MB.
3. Masing-masing program akan mengenkripsi masing-masing berkas dan dihitung waktu dan memort usage yang digunakan untuk enkripsi.
4. Dilakukan pengecekan performa melalui data yang didapat.

### 5.1 Spesifikasi Komputer Pengujian

Spesifikasi perangkat keras komputer yang digunakan untuk pengujian pada makalah kali ini, adalah:

1. Intel Quad Core Q6600 @ 2.4GHz
2. RAM 2GB

Spesifikasi perangkat lunak komputer, yaitu:

1. Operating System Microsoft Windows XP SP3.

### 5.2 Pengujian Algoritma

Pada bagian pertama, algoritma yang akan diuji adalah DES. Berikut adalah hasil pengujiannya:

**Tabel 5-1 Hasil Uji DES**

| Ukuran berkas | Waktu Enkripsi | Memory Usage |
|---|---|---|
| 4MB | 0.82 | 215KB |
| 20MB | 3.24 | 212KB |
| 45MB | 6.37 | 228KB |

Bagian selanjutnya, algoritma yang akan diuji adalah ICE. Berikut adalah hasil pengujiannya:

**Tabel 5-2 Hasil Uji ICE**

| Ukuran berkas | Waktu Enkripsi | Memory Usage |
|---|---|---|
| 4MB | 0.79 | 240KB |
| 20MB | 2.40 | 252KB |
| 45MB | 3.96 | 250KB |

### 5.3 Evaluasi Hasil Uji

Dalam tingkat keamanan yang sama, ICE lebih cepat daripada DES dalam setiap pengujian. Hal tersebut dikarenakan ICE tingkat melewati tahapan permutasi inisial dan final yang justru memakan banyak waktu. Melihat memory usage kedua algoritma, penggunaannya tidak jauh berbeda dikarenakan adanya kemiripan struktur algoritma DES dengan ICE. Lambatnya perubahan kunci dikarenakan penjadwalan kunci ICE berlangsung hanya dengan menggunakan satu kunci setiap saat, sedangkan implementasi DES mengoperasikan blok 28 bit. Hal ini dapat dijadikan sebuah kemajuan atau kemunduran bergantung pada cara pandangnya, sebab di satu pihak, proses ini memperlambat pencarian kunci secara brute force, tetapi hal ini dapat memperlambat penggunaan pada umumnya.

Tabel 5-3 menampilkan hasil tes uji yang dilakukan oleh Michael Kwan antara algoritma Thin-ICE, ICE, ICE-2, dan DES. Spesifikasi komputer yang digunakan adalah 100MHz 486PC dengan operating system Linux.

**Tabel 5-3 Hasil Tes Uji Michael Kwan**

| Operation (x 100000) | Time (seconds) |
|---|---|
| DES encryption | 2.37 |
| DES decryption | 2.40 |
| DES key change | 4.98 |
| ICE encryption | 1.63 |
| ICE decryption | 1.59 |
| ICE key change | 44.79 |
| Thin-ICE encryption | 0.88 |
| Thin-ICE decryption | 0.87 |
| Thin-ICE key change | 22.45 |
| ICE-2 encryption | 3.12 |
| ICE-2 decryption | 3.04 |
| ICE-2 key change | 89.63 |

Seperti yang dapat dilihat, waktu kerja yang dibutuhkan ICE-2 berkisar 2 kali waktu yang dibutuhkan oleh ICE standar. ICE-2 menyediakan tingkat keamanan yang setara dengan Triple-DES, tetapi tanpa tiga faktor perlambatan.

## 6. Kesimpulan

Kesimpulan yang dapat diambil dari makalah ini adalah:

1. ICE lebih cepat melakukan enkripsi dibandingkan DES dalam menyediakan tingkat keamanan yang sama.
2. Melihat Tabel 5-1 dan Tabel 5-2, untuk berkas berukuran 4MB ke bawah, penggunaan DES dan ICE tidak berbeda jauh, sehingga tidak perlu dipermasalahkan penggunaannya.
3. Kedua algoritma dapat berjalan pada komputer dengan RAM kecil.
4. Kedua algoritma tergolong cepat dalam mengenkripsi berkas.
5. Walaupun berkas yang akan dienkripsi berukuran besar, memory usage relatif tidak berubah.

## DAFTAR PUSTAKA


[1] http://www.itl.nist.gov/fipspubs/fip46-2.htm, waktu akses: 31 Maret 2009 21:00, durasi: 1 jam.

[2] http://www.rsa.com/rsalabs/node.asp?id=2168, waktu akses: 31 Maret 2009 21:00, durasi: 1 jam.

[3] http://www.darkside.com.au/ice/, waktu akses: 1 April 2009 19:00, durasi: 1 jam.