STUDI DAN PERBANDINGAN ANTARA ALGORITMA SQUARE DAN ALGORITMA SHARK

Ray Aditya Iswara – NIM : 13504045

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl Ganesha 10, Bandung
E-mail : if14045@students.if.itb.ac.id

Abstrak

Makalah ini membahas tentang studi dan perbandingan antara algoritma Square dan algortma Shark. Kedua algoritma ini termasuk dalam algoritma simetri: *cipher* blok, *Cipher* blok sendiri merupakan algoritma kriptografi modern yang menyembunyikan data dengan cara membagi bit-bit plainteks dengan panjang yang sama, lalu dienkripsi.

Square adalah *cipher* blok yang didesain oleh Joan Daemen dan Vincent Rijmen, dan beroperasi dalam bentuk blok 128-bit. Square memiliki panjang kunci 128-bit. Desain awal dari algoritma ini terdiri dari empat transformasi dan ditekankan pada ketahanannya terhadap kriptanalisis diferensial dan linear.

Shark adalah *cipher* blok yang beroperasi dalam bentuk blok 64-bit, memiliki panjang kunci 128-bit dan dibuat oleh Daemen dan Rijmen juga. Sama dengan algoritma Square, Shark memiliki ketahanan terhadap kriptanalisis diferensial dan linear. Algoritma ini menggabungkan substitusi kotak non-linear dan kode-kode pengkoreksi yang dapat dipisah dengan jarak maksimum (kode MDS). Struktur dari Shark ini diklaim empat kali lebih cepat dari IDEA (termasuk sebuah algoritma cipher blok) dalam arsitektur 64-bit.

Untuk membandingkan kedua algoritma tersebut,, masing-masing algoritma akan diimplementasikan dalam bahasa C, kemudian ditentukan mana yang lebih baik dengan membuat perbandingan dari berbagai aspek.

Kata kunci: Square, Shark, Daemen, Rijmen, kriptografi, kriptanalisis, cipher blok, linear, diferensial, substitusi non-linear, kode MDS, transformasi

1. Pendahuluan

Masalah keamanan pada komputer adalah masalah yang sangat mudah ditemui dalam era teknologi informasi sekarang ini. Kejahatan *cyber* seperti pencurian data ataupun manipulasi data banyak terjadi di sekitar kita.

Kriptografi diciptakan sebagai salah satu solusi untuk mengatasi masalah keamanan komputer. Dalam perkembangannya, kriptografi sekarang ini sudah memasuki tahap kriptografi modern. Berbeda dengan algoritma kriptografi klasik, algoritma kriptografi modern dibuat sedemikian kompleks sehingga kriptanalisis sulit untuk dilakukan. Di antara banyak algoritma kriptografi modern, *cipher* blok merupakan salah satu dari banyak solusi yang ditawarkan. *Cipher* blok termasuk dalam algoritma kriptografi simetri yang beroperasi pada plainteks dalam bentuk blok bit.

Algoritma *Cipher* blok sendiri mempunyai banyak varian, di antaranya adalah Square dan Shark. Kedua algoritma ini muncul sebagai solusi keamanan data yang tahan terhadap kriptanaslisi linear dan diferensial.

2. Sejarah Kriptografi

Kriptografi mempunyai sejarah yang cukup panjang. Ilmu ini sudah dikenal oleh bangsa Mesir 4000 tahun yang lalu berupa hyroglyph yang tidak standar. Di Yunani, kriptografi digunakan oleh tentara Sparta pada awal tahun 400 SM dengan alat yang diberi nama *Scytale*. Lalu pada abad ke-17, kriptografi menyebabkan Queen Mary, ratu Skotlandia, dipancung karena pesan rahasianya dari balik penjara berhasil dipecahkan. Lalu pada perang dunia II, pemerintah Nazi Jerman membuat mesin enkripsi yang diberi nama *Enigma*, dan berhasil dipecahkan oleh pihak Sekutu.

Sampai dengan awal abad ke-19, kriptografi masih termasuk dalam kriptografi klasik. Algoritma yang digunakan dalam enkripsi dan dekripsi tidak sedemikian kompleks sehingga dapat dengan mudah dipecahkan.

Seiring dengan perkembangan zaman, terutama perkembangan teknologi informasi (komputer khususnya), kriptografi sekarang sudah memasuki era kriptografi modern. Kriptografi modern menawarkan tingkat keamanan yang lebih dan tingkat kompleksitas yang lebih rumit.

3. Algoritma Kriptografi Modern

Kriptografi modern menggnakan gagasan dasar yang sama seperti kriptografi klasik, tetapi penekanannya berbeda. Algoritma kriptografi modern dibuat sedemikian kompleks sehingga sulit untuk dipecahkan oleh orang lain.

Algoritma kriptografi modern pada umumnya beroperasi dalam mode bit, tidak seperti algoritma kriptografi klasik yang pada umumnya beroperasi pada mode karakter.

Pada perkembangannya, penggunaan mode berbasis bit didorong oleh penggunaan komputer digital yang merepresentasikan data dalam bentuk biner.

4. *Cipher* Blok

Cipher blok termasuk ke dalam algoritma simetri, beroperasi pada plainteks dalam bentuk blok bit, yang dalam hal ini rangkaian bit dibagi menjadi blok-blok bit yang panjangya sudah ditentukan sebelumnya, misalnya 64 bit, berarti algorima enkripsi memperlakukan 8 karakter setiap kali penyandian.

Enkripsi dilakukan terhadap blok bit plainteks menggunakan bit-bit kunci. Algoritma ini menghasilkan blok cipherteks yang berukuran sama dengan blok plainteks. Dekripsi pada *cipher* blok sama dilakukan dengan cara serupa dengan enkripsi.

Cipher blok sendiri menggunakan beberapa teknik algoritma kriptografi klasik:

1. Substitusi
   Teknik ini mengganti satu atau sekumpulan bit pada blok plainteks tanpa mengubah urutannya. Dalam prakteknya, biasanya digunakan fungsi matematis atau tabel substitusi (S-box).
2. Transposisi
   Teknik ini memindahkan posisi bit pada blok plainteks berdasarkan urutan tertentu. Secara matematis, teknik ini dapat ditulis:

   $C = PM$

   yang dalam hal ini C adalah blok cipherteks , P adalah blok plainteks, dan M adalah fungsi transposisi.
3. Ekspansi
   Teknik ini memperbanyak jumlah bit pada blok plainteks berdasarkan aturan tertentu, misalnya dari 32 bit menjadi 48 bit. Dalam prakteknya, ekspansi ditanyatakan dalam tabel.
4. Kompresi
   Teknik ini adalah kebalikan dari ekspansi, di mana jumlah bit pada blok pada bit plainteks diciutkan berdasarkan aturan tertentu. Dalam prakteknya, aturan kompresi dinyatakan dengan tabel.

5. Algoritma Kriptografi Square

5.1. Struktur

Square adalah *cipher* blok berulang dengan panjang blok dan kunci masing-masing 128 bit. Transformasi melingkar dari Square terdiri dari 4 transformasi yang berbeda.
Blok-blok pembangun yang sederhana dari *cipher* ini adalah lima transformasi yang berbeda yang beroperasi pada array 4x4 dari *bytes*. Elemen dari sebuah keadaan *a* dalam baris *I* dan dalam kolom *j* ditulis sebagai $a_{i,j}$. *Kedua indeksnya dimulai dari 0.*

5.1.1. Transformasi Linear θ

θ adalah operasi linear yang dijalankan terpisah pada setiap 4 baris dalam sebuah keadaan. Kita mempunyai:

$\theta : b = \theta(a)$
$\Leftrightarrow b_{i,j} = c_j a_{i,0} \oplus c_{j-1} a_{i,1} \oplus c_{j-2} a_{i,2} \oplus c_{j-3} a_{i,3}$

Dimana multiplikasinya terdapat pada GF(2^8) dan indeks-indeks dari *c* harus diambil modulo 4. Baris-baris dari statu keadaan dapat dinyatakan dalam polinomial:

$a_j(x) = a_{i,0} \oplus a_{i,1}x \oplus a_{i,2}x^2 \oplus a_{i,3}x^3$

Dengan mendefinisikan $c(x) = \oplus_j c_j x^2$, dapat didefinisikan θ sebagai sebuah multiplikasi polinomial yang modular:

$b = \theta(a)$
$\Leftrightarrow b_i(x) = c(x)a_i(x) \bmod 1 \oplus x^4$ untuk $0 \le i < 4$

Invers dari θ yaitu polinomial $d(x)$ diberikan sebagai:

$d(x)c(x) = 1 \pmod{1 \oplus x^4}$

5.1.2. Transformasi Nonlinear γ

γ adalah sebuah substitusi byte nonlinear, dan identik untuk semua byte. Kita mempunyai:

$\gamma : b = \gamma(a) \Leftrightarrow b_{i,j} = S_\gamma(a_{i,j})$

Dengan S_γ adalah sebuah table substitusi 8-bit yang dapat dibolak-balik atau dikenal dengan nama *S-box*. Invers dari γ mengandung aplikasi dari subsitusi invers S_γ^{-1}, untuk semua byte dari sebuah keadaan.

5.1.3. Permutasi Byte π

π adalah perubahan dari baris-baris dan kolom-kolom dari sebuah keadaan. Kita mempunyai:

$\pi : b = \pi(a) \Leftrightarrow b_{i,j} = a_{j,i}$

π adalah sebuah involusi (kebalikan dari dirinya sendiri), karena $\pi^{-1} = \pi$

5.1.4. Penambahan Bit Kunci Putar σ
$\sigma[k^t]$ mengandung penambahan bit dari sebuah kunci putar k^t. Kita mempunyai:

$\sigma[k^t] : b = \sigma[k^t](a) \Leftrightarrow b = a \oplus k^t$

Invers dari $\sigma[k^t]$ adalah $\sigma[k^t]$ itu sendiri.

5.1.5. Evolusi Kunci Putar ψ

Kunci-kunci putar k^t diturunkan dari kunci *cipher K* dengan ketentuan sebagai berikut. k^o sama dengan kunci cipher K. Kunci putar lainnya diturunkan secara berulang dengan transformasi ψ

$\psi : k^t = \psi(k^{t-1})$

5.1.6. *Cipher* Square

Blok-blok pembangun dibangun ke dalam fungsi transformasi putar oleh $\sigma[k^t]$:

$\rho[k^t] = \sigma[k^t] \circ \pi \circ \gamma \circ \theta$

Square didefinisikan sebagai 8 putaran, didahului penambahan kunci $\sigma[k^o]$ dan θ^1 :

$$\text{SQUARE}[k] = \rho[k^8] \circ \rho[k^7] \circ \rho[k^6] \circ \rho[k^5] \circ \rho[k^4] \circ \rho[k^3] \circ \rho[k^2] \circ \rho[k^1] \circ \sigma[k^0] \circ \theta^{-1}$$

5.1.7. Invers *Cipher*

Struktur mengizinkan dirinya untuk implementasi yang efisien. Square didesain sedemiian rupa sehingga struktur dari inversnya sama dengan dirinya sendiri, dengan pengecualian dari jadwal kunci. Dari pernyataan sebelumnya, dapat disimpulkan bahwa:

$$\text{SQUARE}^{-1}[k] = \rho^{-1}[k^8] \circ \rho^{-1}[k^7] \circ \rho^{-1}[k^6] \circ \rho^{-1}[k^5] \circ \rho^{-1}[k^4] \circ \rho^{-1}[k^3] \circ \rho^{-1}[k^2] \circ \rho^{-1}[k^1] \circ \sigma^{-1}[k^0] \circ \theta$$

dengan:

$$\rho^{-1}[k^t] = \theta^{-1} \circ \gamma^{-1} \circ \pi^{-1} \circ \sigma^{-1}[k^t] = \theta^{-1} \circ \gamma^{-1} \circ \pi \circ \sigma[k^t]$$

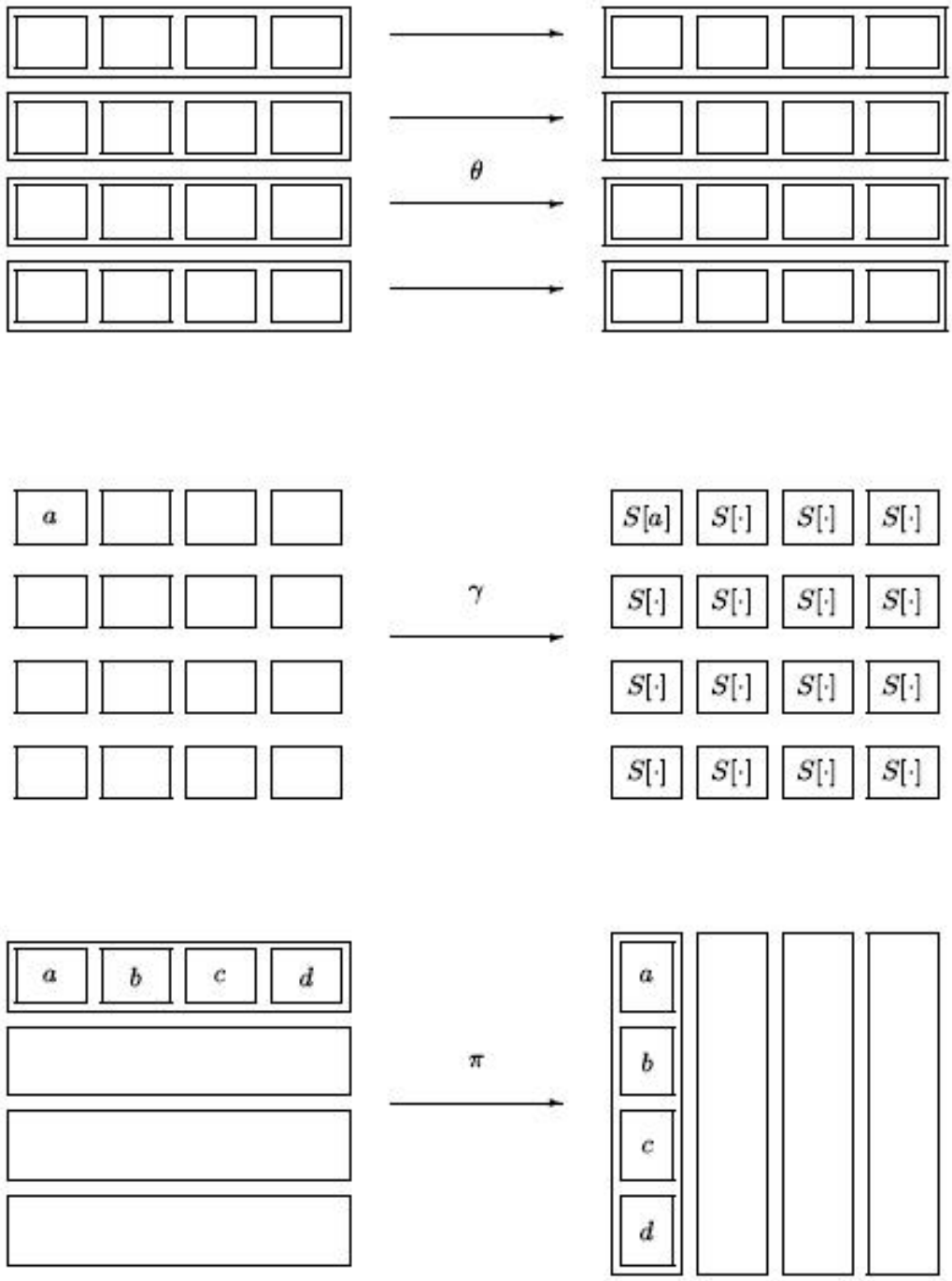
Transformasi putar dari invers *cipher* adalah:

$$\rho'[k^t] = \sigma[k^t] \circ \pi \circ \gamma^{-1} \circ \theta^{-1}$$

yang mempunyai struktur yang sama dengan ρ sendiri., kecuali γ dan θ digantikan dengan γ^{-1} dan θ^{-1}. Dengan menggunakan persamaan-persamaan di atas, dapat diturunkan:

$$\begin{aligned}
\theta \circ \sigma[k^0] \circ \rho^{-1}[k^1] &= \theta \circ \sigma[k^0] \circ \theta^{-1} \circ \gamma^{-1} \circ \pi \circ \sigma[k^1] \\
&= \theta \circ \theta^{-1} \circ \sigma[\theta(k^0)] \circ \pi \circ \gamma^{-1} \circ \sigma[k^1] \\
&= \sigma[\theta(k^0)] \circ \pi \circ \gamma^{-1} \circ \sigma[k^1] \\
&= \sigma[\theta(k^0)] \circ \pi \circ \gamma^{-1} \circ \sigma[k^1] \circ \theta^{-1} \circ \theta \\
&= \sigma[\theta(k^0)] \circ \pi \circ \gamma^{-1} \circ \theta^{-1} \circ \sigma[\theta(k^1)] \circ \theta \\
&= \rho'[\theta(k^0)] \circ \sigma[\theta(k^1)] \circ \theta
\end{aligned}$$

Persamaan ini dapat digeneralisasi secara langsung untuk mengandung lebih dari 1 putaran. Dengan $\kappa^t = \theta(k^{8-t})$, dapat dilihat:



Gambar 1. Representasi geometris dari operasi sederhana dari SQUARE. θ mengandng 4 pemetaan difusi linear paralel. γ mengandun 16 substitusi pemisah. π merupakan sebuah transposisi.

SQUARE^{-1} = $\rho'[\kappa^8] \circ \rho'[\kappa^7] \circ \rho'[\kappa^6] \circ \rho'[\kappa^5] \circ \rho'[\kappa^4] \circ \rho'[\kappa^3] \circ \rho'[\kappa^2] \circ \rho'[\kappa^1] \circ \rho'[\kappa^0] \circ \theta$

Sejak invers *cipher* sama dengan *cipher* itu sendiri dengan γ diganti oleh γ^{-1}, θ oleh θ^{-1}, dan nilai kunci putar yang berbeda-beda.

5.1.8. Putaran Pertama

θ^{-1} sebelum $\sigma[k^0]$ pada SQUARE dapat dibuat di putaran pertama. Kita mempunyai :

$$\rho[k^1] \circ \sigma[k^0] \circ \theta^{-1} = \sigma[k^1] \circ \pi \circ \gamma \circ \theta \circ \sigma[k^0] \circ \theta^{-1}$$
$$= \sigma[k^1] \circ \pi \circ \gamma \circ \sigma[\theta(k^0)]$$

Sejak inisial θ^{-1} dapat dibuang dengan menempatkan θ di putaran pertama dan mengganti k^0 dengan $\theta(k^0)$. Penyederhanaan yang sama dapat diaplikasikan pada invers *cipher*.

5.2. Kriptanalisis Linear dan Diferensial

Ketahanan terhadap kriptanalisis linear dan kriptanalisis diferensial merupakan alasan mendasar di balik kriteria, di mana substitusi S_γ dan multiplikasi polynomial $c(x)$ θ dipilih.

Propagasi yang berbeda sepanjang putaran dari sebuah *cipher* blok berulang biasanya dikenal sebagai karakteristik diferensial. Sebuah karakteristik terdiri dari sebuah kumpulan dari motif (pattern) yang berbeda-beda. Karakteristik diferensial biasa disebut *trail* diferensial. Secara umum, propagasi dari motif perbedaan input *a'* dan motif perbedaan output *b'* dinamakan sebagai sebuah diferensial.

Korelasi dari sebuah kombinasi linear dari bit-bit input dan kombinasi linear dari bit-bit output dari sebuah *cipher* blok berulang dapat dilihat sebagai suatu analogi, tapi sedikit berbeda. Trail linear terdiri dari rangkaian dari motif-motif seleksi. Untuk sebuah kunci cipher, koefisien korelasi (positif atau negatif) sehubungan dengan trail linear terdiri dari produk dari koefisien korelasi antara kombinasi-kombinasi linear dari bit-bit dari setiap pasangan tiap putaran. Korelasi antara sebuah kombinasi linear dari bit-bit input, dinyatakan dengan motif seleksi *u*, dan sebuah kombinasi linear dari bit-bit output, diyatakan dengan *v* adalah sama dengan jumlah dari koefisien-koefisisen korelasi dari semua trail linear dimulai dari *u* dan dikahiri dengan *v*.

S_γ dan $c(x)$ dipilih untuk memperkecil probabilitas maksimum dari trail diferencial dan korelasi maksimum dari trail linear di atas 4 putaran.

Strategi desain trail dimaksudkan untuk menjamin probabilitas maksimum yang rendah dari banyak putaran trail diferensial. Dalam strategi ini, transformasi putar dibentuk dari sejumlah transformai uniform, yang dipecah dalam substitusi blok nonlinear (sehubungan dengan γ) dan komposisi dari transformasi linear (sehubungan dengan $\pi \circ \theta$). Penambahan kunci putar tidak berperan dalam strategi ini. Sudah ditunjukkan bahwa probabilitas dari trail diferencial adalah produk dari probabilitas propagasi perbedaan input-output dari S-box dengan perbedaan tidak sama dengan nol. Dua mekanisme utuk menghilangkan probabilitas trail diferencial yang tinggi dan korelasi trail linear yang tinggi adalah sebagai berikut:

- Pilih S-box dimana probabilitas perbedaan propagasi maksimum dan korelasi input-output maksimum adalah sekecil mungkin
- Pilih bagian linear sedemiian sehingga tidak ada trail dengan beberapa S-box aktif.

Mekanisme pertama memberi dua kriteria yang jelas untuk seleksi dari S_γ. Mekanisme kedua memberi sebuah petunjuk bagaimana memilih multiplikasi polinomial $c(x)$.

5.3. Multiplikasi Polinomial c(x)

Transformasi θ memperlakukan baris-baris yang berbeda dari sebuah keadaan secara terpisah dan dengan cara yang sama . Kita akan mempelajari perbedaan propagasi dan korelasi dari teta, konsentrasi pada baris tunggal. Anggap perbedaan input dispesifikasikan dengan a'(x) = a(x) $\oplus$ a*(x) . Perbedaan output akan diberikan dengan :

$$b'(x) = c(x)a(x) \oplus c(x)a^*(x) \bmod 1 \oplus x^4 = c(x)a'(x) \bmod 1 \oplus x^4.$$

Di sisi lain, kombinasi linear dari bit output, dispesifikasikan dengan seleksi polinomial :

$$v(x) = c(x^{-1})u(x) \bmod 1 + x^4.$$

Secara intuitif , baik linear maupun diferensial akan menguntungkan dari sebuah multiplikasi polynomial yang akan membatasi jumlah objek bukan nol pada perbedaan polynomial input dan output. Hal ini pasti ingin kita hindari dengan

memilih polinomial dengan kekuatan difusi yang kuat, yang ditunjukkan dengan jumlah cabang.
Jika $W_h(a)$ menyatakan bobot Hamming pada sebuah vektor, sebagai contoh komponen jumlah objek bukan nol pada vektor. Berdasarkan keadaan a, perbedaan motif *a'* atau motif seleksi *u*, koresponden ini untuk jumlah dari byte yang tidak nol. Jumlah cabang B dari sebuah pemetaan linear yang dapat dibalik didefinisikan sebagai :

$$\mathcal{B}(\theta) = \min_{a \neq 0}(w_h(a) + w_h(\theta(a))).$$

Hal ini berdasarkan pada jumlah sepasang bobot Hamming pada perbedaan motif input dan output (atau motif pilihan) pada θ yang pada akhirnya adalah B .Hal ini dapat dengan mudah ditunjukkan bahwa B adalah batas bawah dari jumlah S-box yang aktif pada 2 putaran linear atau diferensial yang berurutan. Sejak teta dioperasikan pada masing-masing baris secara terpisah, pada umunya kita akan mendapatkan B = 5.
Sudah ditunjukkan bagaimana pemetaan linear pada $GF(2^m)$ dengan B yang optimal (B=n+1) yang dapat dikonstuksikan dari jarak maksimum kode yang dapat dipisahkan. Kode MDS yang digunakan adalah Reed Solomon kode pada $GF(2^m)$: bila G_e = [$I_{nxn}B_{nxn}$] adalah bentuk echelon dari pembentukan matriks dan (2n,n,n+1) Rs code, kemudian θ : X→Y = B –X mendefinisikan pemetaan linear dengan jumlah cabang yang optimal.
Multiplikasi polinomial dengan c(x) mengkoresponden untuk bagian special dari kode MDS, mempunyai properti tambahan dimana B adalah matriks sirkulan. Matriks sirkulan adalah matriks dimana setiap setiap baris terdiri dari elemen yang sama, menempati satu posisi atau $b_{i,j\ -I\ mod\ n}$. Properti ini dieksploitasi untuk memproduksi implementasi memori yang efisien pada cipher.

5.4. Substitusi non linear

Seperti dijelaskan di atas, kriteria yang relevan dengan S-box yang tertinggi menjadikan korelasi antara beberapa pasang kombinasi linear dari bit output (dinyatakan dengan gamma)
Dan yang tertinggi menjadikan probabilitas korespondensi pada beberapa pasang dari perbedaan motif input dan output. Koresponden ini pada angka tertinggi biasa disebut table eksor pada gamma S-box, didefinisikan sebagai :

$$E_{ij} = \#\{x | S(x) \oplus S(x \oplus i) = j\}.$$

Didefinisikan $\delta = \max_{i,j}\{E_{ij}\} \cdot 2^{-8}$.
Selain itu ada 3 pilihan alternatif pada S-box: transformasi aljabar non linear yang dikonstruksi secara eksplisit, sedikit versi modifikasi, dan pemilihan secara acak dari pemetaan yang dapat dibalik.

Karena nilai optimal dari λ dan δ, diputuskan untuk memilih S_γ, sebuah S-box yang dikonstruksikan dengan mengambil pemetaan $x \rightarrow x^{-1}$ dan menerapkan sebuah transformasi affine pada bit output. Transformasi affine mempunyai properti yang memiliki deskripsi yang rumit pada $GF(2^8)$ pada pencegahan serangan interpolasi. Pilihan ini memaksa semua trail diferensial 4 putaran untuk mempunyai kemungkinan asosiasi yang tidak lebih dari 2^{-150}, jauh dibawah nilai kritis 2^{-127} Secara ekivalen, linear 4 putaran mempunyai korelasi asosiasi tidak lebih dari 2^{-75}, jauh di bawah nilai kritis 2^{-64}. Bagaimanapun, struktur blok spesifik dari cipher dapat membuat serangan diferensial semakin efisien.

5.5. Jumlah putaran

Berdasarkan pada serangan- serangan, jumlah putaran harus ditingkatkan kurang lebih tujuh. Tetapi untuk batas amannya, ditetapkan jumlah putarannya sampai dengan delapan.

Para pemakai konservatif bebas untuk meningkatkan jumlah putaran. Hal ini dapat dilakukan dengan cara langsung dan tidak diperlukan adaptasi dari penjadwalan kunci.

5.6. Evolusi kunci

Penjadwalan kunci menspesifikasikan turunan dari putaran kunci pada kunci *cipher*. Hal ini berfungsi untuk memberikan ketahanan melawan berbagai tipe serangan, diantaranya :

- serangan yang masing-masing bagian dari kunci Cipher diketahui untuk kriptanalisi, misalnya jika Cipher digunakan dengan kunci yang lebih pendek dari 128 bits.
- Serangan dimana kunci memasuki Cipher yang diketahui atau dipilih, misalnya bila Cipher digunakan sebagai fungsi kompresi dari algoritma hash.
- Serangan kunci yang yang berelasi.

Ketahanan melawan tipe serangan yang pertama dapat ditingkatkan dengan penjadwalan kunci yang putaran kuncinya mengalami transformasi dengan difusi tinggi. Untuk mendapatkan skema yang baik, pengetahuan dari jumlah bit tertentu dari satu putaran kunci menentukan beberapa bits pada putaran kunci yang lain. Dua tipe serangan yang lain menunjukkan regularity pada struktur dari penjadwalan kunci dengan kompensasi secara local perbedaan putaran kunci.

Penjadwalan kunci juga memerankan peranan penting pada eliminasi dari simetri :

- Simetri dari putaran transformasi : Putaran transformasi memperlakukan semua byte pada suatu keadaan yang sama. Simetri ini dapat dihilangkan dengan mendapatkan putaran konstan pada penjadwalan kunci.
- Simetri antara putaran : putaran transformasi adalah sama untuk semua putaran. Penjumlahan ini dapat dihilangkan dengan mendapatkan putaran konstan yang bergantung pada penjadwalan kunci.

Penjadwalan kunci ini dijelaskan pada baris kunci. Kita dapat mendefinisikan operasi rotasi byte kiri pada baris sebagai :

$$\text{rotl}[a_{i,0}a_{i,1}a_{i,2}a_{i,3}] = [a_{i,1}a_{i,2}a_{i,3}a_{i,0}]$$

Dan pada rotasi byte sebelah kanan sebagai inversnya.

Transformasi pengulangan penjadwalan kunci $k^{t+1} = \psi(k^t)$ dan inversnya didefinisikan sebagai berikut :

$$\begin{aligned} k_0^{t+1} &= k_0^t \oplus \text{rotl}(k_3^t) \oplus C_t & \kappa_3^{t+1} &= \kappa_3^t \oplus \kappa_2^t \\ k_1^{t+1} &= k_1^t \oplus k_0^{t+1} & \kappa_2^{t+1} &= \kappa_2^t \oplus \kappa_1^t \\ k_2^{t+1} &= k_2^t \oplus k_1^{t+1} & \kappa_1^{t+1} &= \kappa_1^t \oplus \kappa_0^t \\ k_3^{t+1} &= k_3^t \oplus k_2^{t+1} & \kappa_0^{t+1} &= \kappa_0^t \oplus \text{rotr}(\kappa_3^t) \oplus C_t' \end{aligned}$$

SHARK terdiri dari *R* putaran dengan penambahan sebuah kunci, substitusi nonlinear, dan sebuah lapisan difusi, yang merupakan invers dari lapisan difusi putaran.

6.1.1. Lapisan Difusi

Kesederhanaan dari invers penjadwalan kunci dikarenakan oleh θ dan ψ

.

Putaran konstan C_t juga didefinisikan berulang. Kita dapatkan $C_0 = 1_x$ dan $C_t=2_x .C_{t-1.}$

Pilihan ini dapat memberikan difusi tinggi dan menghilangkan kereguleran secara efisien.

5.7. Performa

Implementasi algoritma ini dalam bahasa C berjalan pada 2.63 MB/s pada komputer berprosesor Pentium 100 MHz, dengan sistem operasi Windows95. Invers cipher dapat dimplementasikan dengan cara yang persis sama sebagai cipher itu sendiri dan mempunyai performa yang sama. Perbedaaanya terletak pada tabel-tabel dan prekalkulasi dari kunci-kunci putar.

6. Algoritma Kriptografi Shark

6.1. Struktur dan Desain Shark

Shark memiliki fungsi transformasi:

Y = F(K,X)

dimana *F(K,X)* adalah fungsi yang sama jika dibalik. Struktur ini mirip dengan permutasi substtusi jaringan dan juga digunakan dalam MMB, SAFER, dan 3-WAY. Tiap putaran merubah seluruh putaran input.

Tranformasi putar yang digunakan terdiri dari tiga blok pembangun yang berbeda:

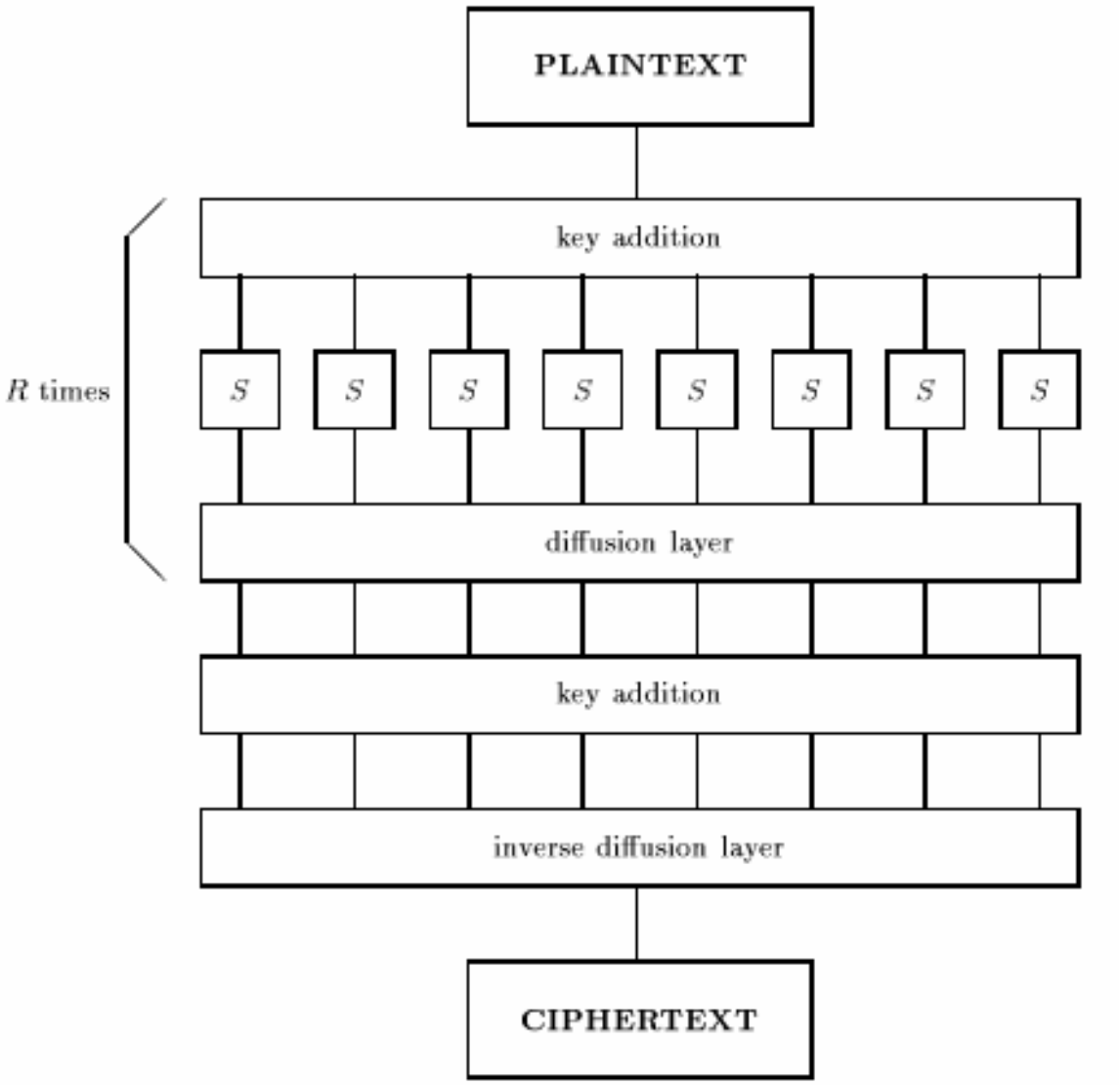
- sebuah lapisan (layer) nonlinear
- sebuah lapisan difusi
- sebuah penjadwalan kunci untuk menghasilkan kunci-kunci putar dari kunci

Dengan mempertimbangkan blok pembangun secara terpisah, kita mendapatkan sebuah *cipher* yang kuat.

S-box pada Shark ini adalah permutasi *m*-bit. Jumlah S-box parallel dinyatakan dengan *n*, dan jumlah putaran dnyatakan dengan *R*

Lapisan difusi ini terdiri dari *n* nilai *m*-bit sebagai input, dan memberi *n* m-bit output.

Penggunaan lapisan difusi ini adalah untuk memberi efek yang besar, baik dalam konteks perbedaan dan aproksimasi linear. Dalam konteks linear, hal ini berarti bahwa tidak ada korelasi antara kombinasi linear dari sekumpulan kecil (m-bit) input dan kombinasi linear dari



Gambar 2. Struktur dari SHARK

sekumpulan kecil (m-bit) output. Dalam konteks diferensial, hal ini berarti perubahan kecil pada input dapat menyebabkan perubahan besar pada output, dan kebalikannya, untuk membuat perubahan output yang kecil, diperlukan perubahan input yang besar. Untuk sebuah linear mapping θ, efek ini dapat dihitung dari jumlah cabang β.

Bobot hamming dari a dinyatakan dalam $w_h(a)$, contohnya jumlah bilangan bukan nol dari a. Komponen-komponen ini dapat berupa bit-bit, atau elemen dari $GF(2^m)$ di sini. Lalu

$$\mathcal{B}(\theta) = \min_{a \neq 0}(\mathrm{w_h}(a) + \mathrm{w_h}(\theta(a))) .$$

β memberikan pengukuran untuk kasus difusi terburuk: merupakan batas bawah untuk jumlah dari S-box aktif dalam dua putaran konsekutif dari sebuah trail linear atau dari sebuah karakteristik diferensial. Karena kriptanalisis selalu menunjukkan kasus terburuk, hal ini merupakan pengukuran yang baik untuk properti difusi.

Sebuah kode linear dapat diprepresentasikan dengan membuat matrix $G_{k \times n}$. Matriks ini mempunyai dimensi $k \times n$. C dibentuk dari

subruang dari dimensi *k* yang disediakan dari baris-baris pada *G*.

$$C = \{G \cdot X | X \in GF(2^m)^k\}.$$

Pembuatan matriks dari sebuah kode tidaklah unik. Jika *G* adalah pembuatan matriks dari *C*, maka setiap matriks $G_1 = T_{k \times k}$. Matriks $G_e = T.G = [I_{k \times k}\ B_{k \times (n-k)}]$ biasa disebut sebagai bentuk *echelon* dari *G*.

Kode-kode dengan $d = n - k + 1$ disebut sebagai kode *Maximal Distance Separable* (MDS code). Kode MDS yang banyak diketahui adalah kode Reed-Solomon. Kode RS ini dapat mempunyai panjang sampai dengan $q+1$, dimana q adalah jumlah dari elemen dari field berhingga dimana kode tersebut didefinisikan (disini $q = 2^m$).

Proposition 1 *Let C be a $(2n, n, n+1)$-code over the Galois field $GF(2^m)$. Let G_e be the generator matrix of C in echelon form:*

$$G_e = [I_{n \times n} B_{n \times n}].$$

Then C defines an optimal invertible linear mapping γ:

$$\gamma : GF(2^m)^n \to GF(2^m)^n : X \mapsto Y = B \cdot X.$$

Proof: First we show that $\mathcal{B} = n + 1$. The definition of $\mathcal{B}$ gives:

$$\mathcal{B}(\gamma) = \min_{X \neq 0}(w_h(X) + w_h(\gamma(X))) = \min_{X \neq 0}(w_h(X, \gamma(X)).$$

Hal ini mengikuti definisi dari γ bahwa semua tuple 2n adalah *codeword* dari kode *C*. Nilai minimum dari bobot Hamming dari *codeword* bukan nol setara dengan $d = n + 1$.
Dibuktikan dengan kontradiksi bahwa γ dapat dibalik. Misalkan γ tidak dapat dibalik, hal ini berarti ada 2 vektor *a,b* sehingga $a \neq b$ dan $\gamma(a) = \gamma(b)$. Karena γ adalah linear, $\gamma(a-b) = \gamma(a) - \gamma(b) = 0$, maka:

$$\mathcal{B} = \min_{c \neq 0}(w_h(c) + w_h(\gamma(c))) \leq w_h(a-b) + w_h(\gamma(a-b)) \leq n + 0 < n + 1.$$

Hal ini merupakan kontradiksi dari $\beta = n + 1$.

Jumlah branch dari sebuah lapisan difusi sangat penting.

6.1.2. Box Substitusi

Box substitusi (S-box) nonlinear memberikan ketahanan terhadap kriptanalisis linear dan kriptanalisis diferensial.

Tabel exor *E* dari sebuah pemetaan γ didefinisikan sebagai berikut:

$$E_{ij} = \#\{x | \gamma(x) \oplus \gamma(i \oplus x) = j\}.$$

Entri-entri tinggi pada tabel exor dapet menuju karakteristik diferensial dengan probabilitas yang tinggi. Hal ini menyebabkan cipher ini mampu ditembus oleh serangan diferensial.

Shark menggunakan S-box berdasarkan pemetaan $F(x) = x^{-1}$ dari $GF(2^m)$. Kelas S-box ini mempunyai properti sebagai berikut:

- 4-uniform secara diferensial. Hal ini berarti nilai tertinggi dari tabel exor sama dengan empat. Faktanya, setiap baris pada tabel exor mengandung tepat 1 buah empat, nilai lain yang mungkin adalah 2 dan 0.
- Jarak minimal ke fungsi *affine* adalah $2^{m/2}$.
- Urutan non-linear dari setiap kombinasi linear dari bit output adalah *m-1*.

Kelemahan dari box ini adalah mereka mempuyai deskripsi yang sederhana dalam $GF(2^m)$, yang juga merupakan field dimana lapisan difusi adalah linear.

6.1.3. Penjadwalan Kunci

Penjadwalan kunci mengekspansi kunci *K* menjadi kunci putar K_i. Sebuah penjadwalan kunci yang baik menghasilkan kunci putar dengan entropi maksimum.

Pertama-tama, ada dua alternatif untuk menghasilkan kunci putar dalam fungsi putar. Yang pertama adalah Exor, dan yang kedua adalah transformasi *affine*.

Exor. Bit input *nm* dari sebuah putaran di-exorkan dengan bit kunci *nm*. Metode ini cepat dan seragam: tidak ada kunci yang lebih kuat atau lebih lemah selama difusi dan lapisan nonlinear mempunyai properti yang sama untuk semua kunci. Kelemahan dari metode sederhana ini adalah entropi dari kunci putar kebanyakan adalah *nm*.

Transformasi Affine. Jika κ_I adalah kunci dependen yang dapat dibalik dari matriks nxn, pada GF(2^m). Operasi kunci tersebut didefinisikan sebagai:

$$Y = \kappa_i \cdot X \oplus K_i.$$

Operasi ini tetap linear dan oleh karena itu, tidak terdapat kunci lemah. Setiap putaran sekarang menghasilkan lebih banyak material kunci, menaikkan entropi dari kunci-kunci putar ke $O(mn^2)$.

Pembentukan sub-kunci. Banyak serangan pada cipher berulang menemukan bagian dari kunci putar. Pengetahuan ini banyak digunakan untuk mengetahui kunci putar lainnya atau kunci itu sendiri. Untuk membuat serangan ini tidak efisien, dapat dihasilkan kunci-kunci putar dengan meng-*hash* kunci tersebut dengan fungsi ketahanan, seperti pada algoritma Blowfish atau CAST.

Pada SHARK, pembentukan kunci putar adalah sebagai berikut. Nilai *R+1 mn*-bit diinisialisasi dari entri *R+1* pertama dari tabel substitusi T_0. Matriks κ_1 diinisialisasi dari matriks unit *I.* Kunci yang dipilih user kemudian digaung denan dirinya sendiri sampai mempunyai panjang $2(R+1)mn$ bit. Ini digunakan sebagai input dari SHARK dalam mode 64-bit CFB. $2(R+1)mn$ bit output digunakan sebagai kunci-kunci putar sebenarnya untuk enkripsi dari plainteks: $(R+1)$ yang pertama adalah nilai-nilai dari K_i, bit-bit selanjutnya diinpretasikan sebagai $(R+1)n$ elemen dari $GF(2^m)$ dan membentuk elemen-elemen diagonal dari κ_i. Jika salah satu dari elemen-elemen ini adalah nol, elemen tersebut dibuang. Secara berurutan, nilai-nilai selanjutnya digeser ke bawah satu tempat dan sebuah enkripsi tambahan dari semua string nol ditambahkan di bagian akhir.
Walaupun mekanisme ini membuat penggunaan kunci dari $2(R+1)mn$ bit menjadi mungkin, tetapi lebih dianjurkan menggunakan panjang kunci yang tidak melebihi 128-bit.

6.2. Ketahanan Terhadap Kriptanalisis Diferensial dan Linear

Pada sebuah karakteristik diferencial, S-box yang mempunyai sebuah input exor bukan nol disebut sebagai S-box aktif; S-box ini menghasilkan exor output yang diminta dengan sejumlah kemungkinan. S-box dari SHARK dipilih sedemikian sehingga probabilitas ini paling tinggi 2^{2-m}. S-box yang tidak aktif mempnyai sebuah input exor nol dan secara konsekuen mereka selalu mempunyai sebuah output exor nol. Lapisan difusi menekankan bahwa dua putaran konsekutif mempunyai total paling sedikit $B = n + 1$ S-box yang aktif.

Dalam sebuah serangan linear, kriptanalis berupaya untuk mencari korelasi antara kombnasi-kombinasi linear dari bit-bit input dengan kombinasi linear dari bit-bit output. S-box di mana beberapa input bit dan beberapa output bit terlibat disebut sebagai S-box aktif. S-box di mana tidak ada input bit dan output bit yang terlibat dalam kombinasi linear disebut sebagai S-box tidak aktif. Dengan menganggap bahwa input-input ke S-box yang berbeda adalah independen, kita dapat menghitung korelasinya dengan mengalikan korelasi S-box yang aktif. Korelasi-korelasi dalam S-box SHARK paling besar $2^{1-m/2}$, yang berarti bahwa setiap S-box aktif menaikkan jumlah teks yang dibutuhkan dengan faktor paling sedikit 2^{m-2}.

Dimensi dari S-box *m*, dan jumlah dari S-box paralel *n* berjumlah 8. Hal ini menunjukkan bahwa cipher ini bekerja pada blok teks 64 bit. Chiper ini dibuat dengan 6 putaran. Untuk aplikasi yang membutuhkan keamanan hanya 40 bit, 4 putaran mungkin sudah cukup.

Tabel di halaman selanjutnya menunjukkan sejumlah nilai untuk kemungkinan terbaik dari karakteristik diferensial dan korelasi yang dikotakkan untuk aproksimasi linear terbaik ketia sebuah fungsi dari jumlah putaran *R,* dibandingkan dengan nilai-nilai pada DES. Jumlah putaran harus terlebih dahulu digandakan untuk memperoleh perbandingan yang adil.

Perhatikan bahwa seorang kriptanalis yang menyerang sebuah skema *R-putaran* tidak memerlukan sebuah aproksimasi *R-putaran* ataupun karakterisitiknya. Dapat diasumsikan bahwa untuk SHARK, sebuah karakteristik atau aproksimasi *(R-2)-putaran* dapet digunakan. Juga, kemungkinan dari diferensial terbaik dapet beberapa kali lebh tinggi daripada kemungkinan dari karakteristik terbaik. Secara ekivalen, korelasi antara bit input dengan bit output dari cipher ini hanya diperkirakan dari produk dari korelasi dalam setiap putaran. Jelas sekali bahwa untuk sebuah cipher 64-bit dengan kunci yang pasti, kemungkinan dari sebuah diferensial lebih dari 2^{-63}, atau sama dengan nol. Juga korelasi-koelasi antara bit input dan bit output adalah beberapa dari 2^{-63}. Lebih spesifik lagi, dalam sebuah *cipher* blok 64-bit, nilai yang diperkirakan untuk kemungknan diferensial dibatasi dengan 128.2^{-64}. Kemungkinan-kemungkinan dan korelasi-korelasi pada tabel dihitung dengan mengasumsikan kunci-kunci

putar yang independen dan variabel. Nilai-nilai ini hanya memberikan indikasi dari batas aman terhadap serangan linear dan diferensial. Ketika kemungkinan dari sebuah karakteristik atau korelasi dari sebuah perkiraan linear di bawah 2^{-63}, dapat dianggap bahwa hal tersebut tidak relevan.

Untuk aplikasi-aplikasi di mana batas keamanan konservatif lebih penting daripada kecepatan enkripsi, dapat digunakan putaran yang lebih banyak. Jika menggunakan kemungkinan dan korelasi dari tabel sebagai pengukuran, 8 putaran SHARK memberi tingkat keamanan yang ekivalen dengan triple-DES. Orang-orang konservatif lainnya dapat menggunakan SHARK dengan delapan atau sepuluh putaran, dan dengan kunci 128-bit.

SHARK			DES		
R	p (dc)	c^2 (lc)	R	p (dc)	c^2 (lc)
2	2^{-54}	2^{-54}	4	$2^{-9.6}$	2^{-6}
4	2^{-108}	2^{-108}	8	$2^{-30.5}$	$2^{-19.5}$
6	2^{-162}	2^{-162}	12	$2^{-46.2}$	$2^{-31.5}$
			48	2^{-128}	2^{-148}

Tabel 1. Kemungkinan dari karakteristik diferensial terbaik dan aproksimasi linear sebagai sebuah fungsi dari jumlah putaran. Diberikan nilai untuk karakteristik/aproksimasi *R-putaran.*

6.3. Pertimbangan Implementasi

Anggap X_1, ... , X_n menyatakan input dari sebuah putaran, setelah penambahan kunci, dan anggap Y_1, ... , Y_n menyatakan output. Kita mempunyai:

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \dots \\ Y_n \end{bmatrix} = A \cdot \begin{bmatrix} S_1[X_1] \\ S_2[X_2] \\ \dots \\ S_n[X_n] \end{bmatrix}$$

$$= \begin{bmatrix} a_{11} \\ a_{21} \\ \dots \\ a_{n1} \end{bmatrix} \cdot S_1[X_1] \oplus \begin{bmatrix} a_{12} \\ a_{22} \\ \dots \\ a_{n2} \end{bmatrix} \cdot S_2[X_2] \oplus \dots \oplus \begin{bmatrix} a_{1n} \\ a_{2n} \\ \dots \\ a_{nn} \end{bmatrix} \cdot S_n[X_n] .$$

Disini, S_i adalah tabel substitusi m x m, "⊕" dan "·" menyatakan penambahan dan multiplikasi dalam GF(2^n), dan A adalah matriks yang mendefinisikan lapisan difusi. Kita dapat juga menuliskan sebagai berikut:

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \dots \\ Y_n \end{bmatrix} = \begin{bmatrix} a_{11} \cdot S_1[X_1] \\ a_{21} \cdot S_1[X_1] \\ \dots \\ a_{n1} \cdot S_1[X_1] \end{bmatrix} \oplus \begin{bmatrix} a_{12} \cdot S_2[X_2] \\ a_{22} \cdot S_2[X_2] \\ \dots \\ a_{n2} \cdot S_2[X_2] \end{bmatrix} \oplus \dots \oplus \begin{bmatrix} a_{1n} \cdot S_n[X_n] \\ a_{2n} \cdot S_n[X_n] \\ \dots \\ a_{nn} \cdot S_n[X_n] \end{bmatrix}$$

dengan tabel substitusi *m x mn* T_i yang diekspansi:

$$T_i[X] = \begin{bmatrix} a_{1i} \cdot S_i[X_i] \\ a_{2i} \cdot S_i[X_i] \\ \dots \\ a_{ni} \cdot S_i[X_i] \end{bmatrix}$$

Operasi gabungannya menjadi :

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \dots \\ Y_n \end{bmatrix} = T_1[X_1] \oplus T_2[X_2] \oplus \dots \oplus T_n[X_n] .$$

Operasi ini hanya memerlukan tabel pencarian *n* dan *n-1* penambahan bit dan penggeseran (dari nilai bit *nm*).

Penambahan kunci dapat digabungkan ke dalam S-box. Penambahan dengan sebuah kunci yang tetap sebelum substitusi tabel adlaah ekivalen dengan sebuah pengaturan sederhana dari baris-baris tabel. Jika *K* adalah sebuah diagonal matriks, operasi ini sekali lagi ekivalen dengan pengaturan baris dari tabel subsitusi.

Mode enkripsi dan dekripsi dari blok cipher sangat mirip dengan struktur Feistel, hanya urutan dari kunci putar harus dibalik. Untuk SHARK, konversi dari mode enkripsi ke mode dekripsi lebih terlibat.

Anggap ada sebuah versi SHARK dua putaran. Operasi linear dinyatakan dengan *l*, substitusi non linear dinyatakan dengan *s,* dan penambahan kunci k_i dinyatakan dengan a_k. Fungsi enkripsi diperoleh sebagai:

$$Y = (l^{-1} \circ a_{k_3} \circ \underbrace{l \circ s \circ a_{k_2} \circ l \circ s}_{r} \circ a_{k_1})(X) ,$$

Dimana *r* menyatakan operasi difusi S-box gabungan. Karena penambahan kunci dan

lapisan difusi merupakan operasi linear, kita dapat mengganti urutannya:

$$(l \circ a_k)(X) = B \cdot (\kappa \cdot X \oplus K) = (B \cdot \kappa \cdot X) \oplus (B \cdot K)$$
$$= ((B \cdot \kappa \cdot B^{-1}) \cdot (B \cdot X)) \oplus (B \cdot K)$$
$$= (a_{k'} \circ l)(X),$$

Dengan

$$a_{k'}(X) = \kappa' \cdot X \oplus K'$$
$$= (B \cdot \kappa \cdot B^{-1}) \cdot X \oplus (B \cdot K)$$

Maka enkripsi tesebut menjadi :

$$Y = (a_{k_3'} \circ l^{-1} \circ l \circ s \circ a_{k_2} \circ r \circ a_{k_1})(X)$$
$$= (a_{k_3'} \circ s \circ a_{k_2} \circ r \circ a_{k_1})(X).$$

Persamaan terakhir diimplementasikan. Persamaan tersebut mengandung *R* tabel pencarian dan *R+1* penambahan kunci. Dengan membalikkan persamaaan awal, kita mendapatkan operasi dekripsi:

$$X = (a_{k_1^{-1}} \circ s^{-1} \circ l^{-1} \circ a_{k_2^{-1}} \circ s^{-1} \circ l^{-1} \circ a_{k_3^{-1}} \circ l)(Y).$$

Dan dengan mengubah penambahan kunci dan difusi, kita memperoleh:

$$X = (a_{k_1^{-1}} \circ s^{-1} \circ a_{k_2^{-1}'} \circ \underbrace{l^{-1} \circ s^{-1}}_{r'} \circ a_{k_3^{-1}'})(Y).$$

Persamaan ini mempunyai struktur yang sama dengan operasi enkripsi.

6.4. Performa

Karena SHARK bekerja pada kata-kata 64-bit, akan lebih menguntungkan jika dijalankan pada arsitektur 64-bit. Tabel 2 membandingkan performa dari implementasi C dari SHARK, SHARK* (SHARK dengan S-box yang bergantung pada kunci), SAFER, IDEA, dan MD5 pada 266 MHz DEC-ALPHA dan pada sebuah kompute Pentium 90 MHz. Pada Pentium, SHARK bekerja pada kecepatan yang sama dengan SAFER. Ekspermen dengan S-box yang lebih kecil menunjukkan bahwa penurunan performa dikarenakan besar cache pada chip yang terbatas; menggandakan cache ini akan memungkinkan peningkatan performa dari 0.8 MB/s menjadi 2.3 MB/s

	ALPHA	Pentium
SHARK	6.30 Mbyte/s	0.800 Mbyte/s
SHARK*	5.10 Mbyte/s	
SAFER	1.03 Mbyte/s	0.725 Mbyte/s
IDEA	1.53 Mbyte/s	
MD5	16.00 Mbyte/s	7.500 Mbyte/s

Tabel 2. Performa dari SHARK, SAFER, IDEA, dan MD5 pada sebuah workstation 64-bit dan Pentium.

7. Perbandingan Antara SQUARE dan SHARK

ASPEK	SQUARE	SHARK
Putaran (Langkah)	8	6
Blok (bit)	128	64
Kata (bit)	32	64
Endianness	Netral	Netral
Tabel (byte)	4K	16K
Performa (Mbits/s)	47.2	9.85
Performa (cycle/blok)	244	585
Dipatenkan	Belum	Belum

Tabel 3. Performa dalam clock cycle per block dari ouput dan Mbit per detik pada sebuah 90-MHz Pentium. Semua implementasinya ditulis dalam bahasa assembly.

Tabel di atas menunjukkan perbandingan antara algoritma SQUARE dan SHARK. SHARK membutuhkan memori tabel jauh lebih banyak daripada SQUARE. Jika diasumsikan bahwa semua data ditampung dalam cache, maka performa dari SHARK dapat meningkat 4 kali lipat menjadi 43.2 Mbit/s. Kedua algoritma tersebut belum dipatenkan oleh siapapun, sehinga algoritma ini bebas dipakai oleh siapapun.

Secara umum, algoritma SQUARE lebih unggul dalam performa jika dibandingkan dengan algoritma SHARK, maka dapat disimpulkan bahwa algoritma SQUARE lebih baik daripada algoritma SHARK.

DAFTAR PUSTAKA

[1] Munir, Rinaldi. "Diktat Kuliah IF5054 : Kriptografi". 2006. Bandung : Institut Teknologi Bandung

[2] http://www.comms.scitech.susx.ac.uk/fft/crypto/square.pdf

[3] http://www.ddj.com/184410756

[4] http://citeseer.ist.psu.edu/29479.html

[5] http://citeseer.ist.psu.edu/daemen97/block.html

[6] http://homes.esat.kuleuven.be/~rijmen/square/

[7] http://www.kremlinencrypt.com/algorithms.htm

[8] http://search.cpan.org/~jcduque/Crypt-Shark-1.0.1/Shark.pm

[9] http://search.cpan.org/~jcduque/Crypt-Square-1.0.0/Square.pm

[10] http://en.wikipedia.org/wiki/Square_%28cipher%29

[11] http://en.wikipedia.org/wiki/Shark_(cipher)