Perbandingan Algoritma Kriptografi Kunci Simetrik BlowFish dan TwoFish

Mohamad Octamanullah – NIM 13503119

Abstrak

Makalah ini membahas perbandingan mendalam terhadap dua buah algortima kriptografi “BlowFish” dan “TwoFish”. Kedua algoritma tersebut tergolongkan algoritma kunci simetrik cipher blok. Cipher blok berarti data diolah dalam satuan blok, dan algoritma kunci simetrik berarti kunci yang dipakai untuk mengenkripsi data sama dengan kunci yang dipakai untuk mendekripsi.

Blowfish dan twofish keduanya dirancang oleh orang yang sama, yaitu Bruce Schneier. Namun keduanya berasal dari masa yang berbeda. Blowfish merupakan algoritma yang lebih tua, ia dirancang pada tahun 1993. Tujuan perancangannya adalah untuk menggantikan algoritma DES yang sudah sangat tua (sejak 1977). Algoritma DES sendiri merupakan algoritma standar kriptografi yang ditetapkan oleh NIST – sebuah lembaga yang mengatur tentang standar-standar. Antara DES dan blowfish memiliki banyak kesamaan. Kesamaan mendasar adalah keduanya memiliki panjang blok yang sama, yaitu 64 bit. Namun tentu blowfish karena jauh lebih muda, memiliki banyak kelebihan dibandingkan dengan DES. Blowfish sangat terkenal di dunia kriptografi, alasan utamanya adalah karena lisensinya yang bebas dan gratis. Bahkan komunitas open source menghargai blowfish dengan mempercayai blowfish menjadi salah satu Open Cryptography Interface (OCI) pada kernel Linux versi 2.5 keatas.

Sedangkan twofish adalah suksesor dari blowfish. Pada tahun 1997, dengan kemajuan teknologi prosesor yang sangat cepat, maka bukanlah ahal yang sulit untuk menjebol algoritma kriptografi dengan panjang kunci 64 bit. Untuk itu NIST membuka sayembara untuk umum. Semua pihak boleh mensubmit algoritmanya, namun tentu dengan syarat-syarat kualitas minimum, seperti panjang blok minimum 128 bit. Twofish merupakan salah satu peserta, dan berhasil meraih posisi 5 besar, dan bahkan secara tidak resmi mendapatkan posisi 2 besar. Lima besar algoritma yang lolos semuanya memiliki tingkat keamanan yang hampir seimbang, sehingga penilaian disampingkan menjadi performansi kecepatan. Twofish dan rijndael bersaing memperebutkan posisi teratas. Twofish unggul di nomor kunci 256 bit, namun kalah di nomor kunci 128 bit. Akhirnya rijndael yang keluar sebagai pemenang dan berubah nama menjadi AES, suksesor dari DES.

Twofish dan blowfish keduanya memiliki keunggulan tersendiri dibandingkan rivalnya (DES dan AES). Dan diantara keduanya pun masing-masing memiliki keunggulan dan kelemahan masing-masing. Untuk itu dirasakan perlu adanya sebuah studi pembandingan kedua algoritma tersebut secara mendalam. Mulai dari teknik-teknik yang dipakai oleh keduanya, hingga performansi keduanya.

Kata Kunci: Kriptografi, Blowfish, DES, Twofish, AES, NIST, Algoritma Kunci Simetrik, Block Cipher.

1. Pendahuluan

Kriptografi atau kriptologi adalah sebuah ilmu tentang kerahasiaan pesan. Pesan adalah suatu informasi yang berharga yang dipertukerkan oleh dua belah pihak, namun juga berharga bagi pihak ketiga yang tidak berhak untuk mengetahui isi pesan tersebut. Di masa modern, kriptologi merupakan salah satu cabang dari teori informasi atau informatika.

Kriptografi telah dikenal sejak jaman kekaisaran romawi dibawah kekaisaran Julius Caesar. Pada masa itu, diperlukan sebuah mekanisme penyampaian pesan rahasia yang tidak boleh diketahui pihak manapun khususnya pihak musuh. Cara menyembunyikan pesan pada saat itu adalah dengan teknik pergeseran, yaitu menggeser setiap huruf ke kanan atau ke kiri dalam urutan alfabet beberapa buah. Cara tersebut hingga saat ini dikenal dengan metoda Caesar Cipher.

Kriptografi sangat terasa dibutuhkan memang pada saat perang, selain contoh kekaisaran romawi diatas, kriptografi sangat penting pula pada saat perang dunia ke-2. Jerman memiliki mekanisme untuk menyembunyikan pesan dengan bantuan sebuah alat bernama Enigma. Jerman menyatakan bahwa tidak mungkin mengetahui isi pesan yang di enkripsi oleh mesin Enigma. Namun pihak sekutu berhasil memecahkannya, proses memecahkan sebuah algoritma kriptogrfi disebut kriptanalisis, Dan pada kasus tersebut merupakan kriptanalisis

pertama dalam sejarah. Entah mengapa algoritma Caesar Cipher yang sederhana tidak ada yang mampu memecahkannya pada waktu itu.

Sejak berakhirnya perang dunia ke-2, kriptografi mulai merasa perlu dikehidupan sehari-hari, tidak hanya pada saat perang. Khusunya untuk data-data yang bersifat rahasia dan pribadi, seperti rekening bank, PIN, tanda tangan, dll. Oleh karena itu mulai banyak bermunculan algoritma-algoritma baru sesuai perkembangan jaman.

Algoritma-algoritma yang muncul di awal-awal setelah perang dunia ke-2 adalah algoritma dengan kunci simetrik. Algoritma kunci simetrik berarti bahwa kunci untuk mengenkripsi pesan adalah sama dengan kunci untuk mendekripsi pesan. Permasalahan selanjutnya adalah, bagimana menyampaikan kunci ke pihak yang berhak, tidak mungkin kunci tersebut di enkripsi kembali dengan cara yang sama. Namun kelemahan tersebut tertutupi dengan keampuhan algoritma kunci simetrik yang sangat aman dan cepat.

Menjawab permasalahan algoritma kunci simetrik, muncullah algoritma kunci asimetrik. Sebuah algoritma yang berarti kunci untuk melakukan enkripsi berbeda dengan kunci untuk melakukan dekripsi. Tidak perlu lagi mengirim kunci ke pihak yang berhak, sebab pihak tersebut telah memiliki kuncinya sejak awal. Disamping kelebihannya tersebut, algoritma ini memeiliki kelemahan dari segi performansi, ayitu sangat lambat dalam proses enkripsi-dekripsi dibandingkan dengan algortitma kunci simetrik.

2. Algoritma kunci simetrik

Algoritma kunci simetrik adalah algoritma kriptografi yang memiliki kunci yang sama untuk proses enkripsi dan dekripsinya. Kunci tersebut merupakan satu-satunya jalan untuk proses dekripsi (kecuali mencoba membobol algoritma tersebut), sehingga kerahasiaan kunci menjadi nomor satu. Untuk mengirimkan kunci tersebut ke suatu pihak tanpa diketahui pihak yang lain merupakan masalah awal dari algoritma kunci simetrik.

Algoritma kunci simetrik terbagi menjadi dua buah bergantung pada datanya. Keduanya adalah: cipher aliran (*stream cipher*) dan cipher blok (*block cipher*). Cipher aliran memproses satu bit pesan sekali dalam satu waktu, sedangkan cipher blok memproses sekumpulan bit sekaligus sebagai satu unit. Ukuran blok yang umu dipakai adalah 64 bit.

Dari segi kecepatan komputasi, algoritma kunci simetrik lebih cepat daripada algoritma asimetrik. Kelemahan utamanya seperti yang disebutkan diatas, yaitu dalam mendistribusikan kunci ke pihak-pihak yang berkepentingan. Jika dipakai dalam suatu lingkungan yang tidak membutuhkan pendistribusian kunci (seperti penggunaan pribadi), maka algoritma ini merupakan algoritma yang terbaik.

2.1 *Reversibility* algoritma kunci simetrik

Semua fungsi kriptografi harus memiliki sifat reversibility, yaitu mampu mengembalikan cipher teks hasil enkripsi kembali ke plain teks melalui proses dekripsi. Kemampuan reversibility pada hampir semua metode pada algoritma kunci simetrik mengandalkan kemampuan *reverse operation*. Metode ini berintikan membalik semua operasi yang ada. Yaitu dengan melakukan operasi yang berlawanan. Misal operasi yang berlawanan adalah: penjumlahan & pengurangan, penggeseran ke kiri & ke kanan, dll.

Namun tidak sedikit juga algoritma kunci simetrik blok cipher memiliki sub metode yang tidak bersifat reversible jika berdiri sendiri seperti metode Expand dan Filter (Subbab 3.6). Namun metode tersebut akan bersifat reversible jika ditanamkan pada metode Jaringan Feistel (subbab 3.3). Dan metode yang digunakan tidak hanya mengandalkan reverse operation, namun melibatkan teknik-teknik lain.

2.2 Deskripsi cipher blok

Cipher blok merupakan salah satu pendekatan dalam algoritma kriptografi kunci simetrik. Pendekatan lain adalah cipher aliran. Perbedaan mendasar keduanya adalah jika cipher blok memproses dalam suatu kumpulan bit sekaligus sebagai suatu unit dan cipher aliran memproses bit per bit.

Panjang blok yang biasa diimplementasikan oleh perancang algoritma kriptografi adalah kelipatan 64 bit. Pada awal tahun 1990-an, panjang blok yang paling umum adalah 64 bit. Masayrakat merasa dengan panjang kunci 64 bit telah cukup aman dan tidak mungkin ada komputer yang mampu menyerang dengan metode *brute force*. Namun dengan berlalunya waktu, ternyata mungkin untuk menyerang algoritma dengan kunci 64 bit menggunakan metode brute force. Oleh karena itu panjang blok umum berkembang menjadi 128 bit pada awal 2000-an, atau bahkan akhir-akhir ini

mulai umum ditemukan algoritma dengan panjang blok 256 bit.

Dengan panjang blok 128 bit (hampir semua algoritma mengimplementasikan panjang kunci sama dengan panjang blok), maka secara teoritis, memerlukan $2^{128}/2^{20}$ detik (dengan asumsi satu detik dapat mencoba 1000000 kemungkinan kunci) yang merupakan lebih dari 1 triliun tahun.

Blok cipher pertama kalinya diperkenalkan oleh IBM dengan Lucifer-nya pada tahun 1970-an yang didasarkan pada karya Horst Feistel. Versi revisi dari Lucifer yang lebih dikenal dengan Data Encryption Standard (DES) diadopsi sebagai standar algoritma kriptografi oleh US National Bureau of Standard (NBS).

2.3 Modus operasi cipher blok

Dalam proses enkripsi dan dekripsi algoritma kriptografi simetrik terdapat banyak modus untuk membantu proses tersebut secaa keseluruhan dalam hal penyembunyian data/informasi. Hampir semua modus kecuali ECB juga berfungsi untuk mengacak blok masukan sebelum memasuki proses enkripsi dan dekripsi. Proses tersebut ikut membantu menyamarkan informasi yang ada sebagai bagian proses enkripsi dan dekripsi secara keseluruhan.

Modus-modus yang banyak dipakai oleh algoritma-algoritma kriptografi yang ada saat ini antara lain adalah:

- ECB (Electronic Code Book)
- CBC (Cipher Block Chaining)
- CFB (Cipher Feed-Back)
- OFB (Output Feed-Back)
- CTR (Counter)

Terdapat modus umum yang berfungsi sebagai media autentikasi yang disebut MAC (Message Authentication Codes). Modus-modus turunan dari MAC adalah modus MAC dirancang menggunakan salah satu modus tertentu seperti: CBC-MAC, OMAC, PMAC, dll.

Sedangkan modus yang dapat meng-enkripsi-dekripsi sekaligus dapat menyimpan informasi autentikasi seperti: CCM, EAX, GCM, dan OCB.

Setiap modus kecuali ECB memerlukan kotak awal (Initialization Vector). Kegunaan kotak ini adalah selain untuk mengacak setiap proses enkripsi-dekripsi, juga untuk memulai proses enkripsi-dekripsi untuk awal blok. Tidak ada kebutuhan IV haruslah rahasia, bahkan hampir selalu IV dituliskan sebagai header cipher teks bentukan proses enkripsi. Namun yang penting adalah IV yang samatidak boleh digunakan kembali untuk kunci yang sama. Untuk modus CBC dan CFB, menggunakan IV yang sama akan berdampak kebocoran blok awal plain teks. Untuk modus OFB dan CTR, menggunakan ulang IV yang sama akan berdampak menghancurkan kerahasiaan secara keseluruhan.

2.4 Hal lain berkaitan dengan cipher blok

Proses enkripsi-dekripsi cipher blok dengan menggunakan modus-modus sederhana (ECB hingga CTR) tidak memiliki proteksi terhadap integritas data dan juga tidak memiliki mekanisme pemulihan kesalahan. Kelemahan tidak memilikinya proteksi terhadap integritas data dapat dimanfaatkan oleh pihak-pihak tertentu untuk mengubah satu atau lebih bit pada cipher teks yang bila dimasukkan pada proses dekripsi dapat menghasilkan pesan yang masih dapat dimengerti namun memiliki arti yang lain.

Modus-modus operasi untuk mengatasi kelemahan diatas telah ditawarkan, yaitu dengan berbagai cara. Kebanyakan adalah dengan mengawinkan modus sederhana dengan modus Message Authentication Codes (MAC) dan dengan modus Error Correcting Codes. Modus-modus yang ditawarkan dengan memanfaatkan perkawinan tersebut antara lain adalah: IACBC, IAPM, OCB, EAX, CWC, CCM, dan GCM.

Hal lain yang perlu diperhatikan pada cipher blok adalah Padding (pengisian bit-bit sisa agar sesuai dengan panjang blok). Padding yang paling sering digunakan adalah dengan memenuhi byte-byte sisa dengan byte Null (0). Namun dalam bahasa C, sebuah string selalu diakhiri oleh byte null, dan jika oleh proses dekripsi byte null tersebut dianggap padding, maka akan mengacaukan plain teks secara keseluruhan. Oleh karena itu informasi padding harus disimpan pada header cipher teks.

3. Metoda dan algoritma cipher blok

Pada proses enkripsi dan dekripsi suatu algoritma kunci simetrik, biasanya merupakan gabungan dari berbagai metoda yang telah umum di masyarakat. Jika terdapat perbedaan, biasanya juga hanyalah dengan memodifikasi dari metoda umum tersebut.

Blowfish dan twofish keduanya juga memanfaatkan berbagai metoda umum tersebut. Untuk itu perlu pembahasan beberapa

metoda yang digunakan oleh kedua algoritma tersebut.

3.1 Pemanipulasian bit

Metoda paling sederhana dalam algoritma kriptografi cipher blok adalah pemanipulasian bit. Baik memanipulasi sekelompok bit (bahkan memanipulasian sebuah byte ke byte lain dalam algoritma kriptografi sederhana) maupun memanipulasi sebuah bit saja.

Terdapat banyak cara memanipulasi bit, diantaranya yang paling populer adalah:

- Menggeser sejumlah bit ke kanan atau ke kiri sejauh beberapa bit.
- Menerapkan operator bit terhadap suatu bit, yaitu: And, Or, Xor, dan Neg. Namun operator yang paling sering digunakan pada algoritma simetrik adalah Xor, sebab sifatnya yang *reversible* (dapat dikembalikan ke bentuk semula).
- Membalik urutan bit.
- Mensubstitusikan sebuah bit dengan bit lain, namun memiliki pola tertentu.

Jika pemanipulasian diterapkan terhadap sekumpulan bit membentuk sebuah byte, terdapat banyak sekali cara, khususnya pada algoritma kriptografi sederhana. Seperti Caesar cipher, Vigenere Cipher, dll. Metode ini tidak dijelaskan lebih jauh sebab metode ini tidak dimanfaatkan oleh blowfish maupun twofish.

3.2 Kotak permutasi / pemutihan

Tujuan dari metode ini adalah untuk mengacak urutan bit-bit pada sebuah blok. Metode ini berbeda dengan metode substitusi pada pemanipulasian bit (Subbab 3.1). Perbedaannya adalah pada metode ini digunakan acuan yang telah pasti dalam pensubstitusian bit.

Acuan tersebut tidak memiliki pola khusus, dan pada kebanyakan algoritma kriptografi, acuan tersebut telah ditetapkan oleh si perancang algoritma.

Dinamakan kotak permutasi, sebab merupakan sebuah kotak 2 dimensi yang setiap isinya memiliki informasi bit tersebut harus dipindahkan ke urutan keberapa pada blok tersebut.

Blowfish dan twofish keduanya memanfaatkan kotak permutasi dalam beberapa prosesnya. Kotak permutasi ini bertujuan untuk mengacaukan urutan bit, sehingga mencegah kriptanalis yang akan menyerang algoritma tersebut menggunakan metoda seperti metoda kunci lemah.

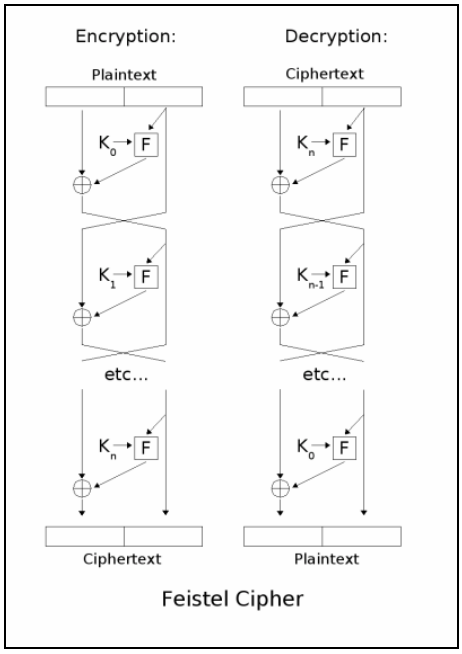
3.3 Jaringan feistel

Dinamakan jaringan feistel setelah perancangnya, Horst Feistel, berhasil menemukan metoda kriptografi yang efisien, kuat, dan mudah untuk dikembalikan ulang.

Jaringan feistel asli menerapkan beberapa langkah yang sudah terstandardisasi. Langkah-langkah tersebut dapat berupa metoda-metoda lain, seperti pergiliran kunci, ekspansi dan filter, dll. Inti dari jaringan feistel adalah sebuah blok dibagi menjadi dua buah blok sama besar. Setengah blok kanan dikopikan ke setengah blok kiri blok hasil dan juga dimasukkan ke metoda ekspansi dan filter bersamaan dengan setengah blok kiri. Hasil dari fungsi ekspansi dan filter tersebut dikopikan ke setengah blok kanan blok hasil.

Untuk mendekripsikan metoda ini cukup mudah, yaitu dengan cara mengkopikan setengah blok kiri result cipher ke setengah blok kanan blok hasil sambil dimasukkan ke fungsi ekspansi dan filter.

Kelebihan lain dari metode ini adalah kemampuannya menggunakan metoda lain yang bersifat irreversible (seperti ekspansi dan filter). Hal tersebut dapat terjadi karena dalam proses enkripsi maupun dekripsi untuk mencapai blok hasil harus melalui fungsi ekspansi dan filter dari arah “atas”, yaitu bukan dari arah “bawah”, seperti dekripsi pada umumnya.



Gambar 1 – Feistel Cipher

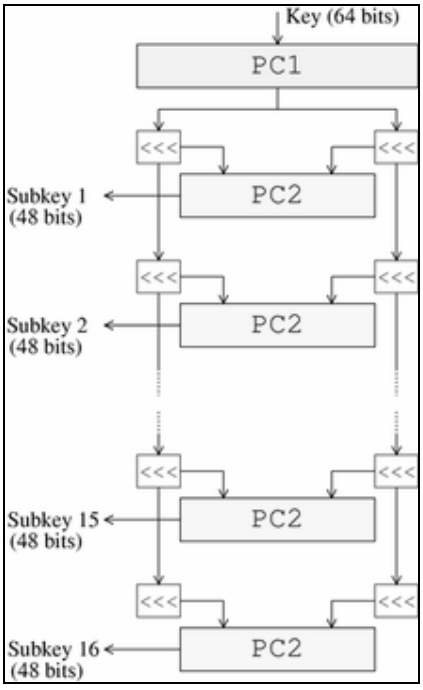
Banyak sekali modifikasi yang dapat dilakukan pada metoda ini. Modifikasi yang paling populer adalah:

- Jaringan feistel dengan tidak hanya membagi menjadi dua buah subblok saja, namun lebih.
- Jaringan feistel yang membaginya tidak rata, yaitu jumlah bit di subblok kanan tidak sama dengan jumlah bit di subblok sebelah kiri.

Blowfish dan twofish keduanya memanfaatkan metoda ini, dengan menggunakan metoda asli. Di dalam metoda ini keduanya juga mengimplementasikan pergiliran kunci dan ekspandsi dan filter.

3.4 Memutar ulang dan pergiliran kunci

Pada hampir semua algoritma modern pasti memiliki metoda memutar ulang, yaitu mengumpulkan beberapa metoda alghoritma, dan dari sekumpulan algoritma tersebut diputar ulang. Hasil blok keluaran kumpulan algoritma tersebut dimasukkan kembali kedalam kumpulan algoritma tersebut hingga beberapa kali. Bisa hanya sekali, dua kali, hingga puluhan kali.



Gambar 2 – Salah satu contoh metoda penggiliran kunci

Metoda tersebut biasanya dibarengi dengan pergiliran kunci. Kunci utama yang dimasukkan biasanya tidak digunakan terus menerus dalam semua proses. Setiap level biasanya menggunakan kunci yang berbeda.

Dalam menggilirkan kunci, kebanyakan algoritma menggunakan metode kotak permutasi dan penggeseran bit. Masukan awal adalah kunci utama, dan di setiap level, kunci tersebut digeser dan dimasukkan ke kotak permutasi menghasilkan kunci-kunci internal yang bersesuaian dengan level putarannya.

Blowfish dan twofish keduanya mengimplementasikan metode ini, dengan masing-masing memodifikasi bagaimana kunci digilirkan. Pergiliran kunci pada blowfish lebih sarat komputasi daripada pergiliran kunci pada twofish.

3.5 Transformasi Pseudo-Hadamard

Salah satu metoda lain untuk mengacaukan bit adalah transofmasi pseudo-Hadamard. Transformasi ini memanfaatkan beberapa fungsi aritmatika sederhana. Dinamakan pseudo (semi) sebab merupakan manipulasi dari transformasi Walsh-Hadamard yang cukup terkenal dalam dunia matematika dan fisika quantum. Transformasi tersebut menggunakan bantuan operasi aritmatika kompleks yang dapat dikembalikan (reversible).

Metoda asli pseudo-Hadamard yaitu dengan membagi blok menjadi dua buah subblok sama besar dan menerapkan operasi aritmatika sederhana.
Enkripsi pseudo-Hadamard:

$a' = a + b \pmod{2^n}$

$b' = a + 2b \pmod{2^n}$

Dekripsi pseudo-Hadamard:

$b = b' - a' \pmod{2^n}$

$a = 2a' - b' \pmod{2^n}$

Persamaan diatas dapat juga dituliskan sebagai matriks aljabar, dengan memandang a dan b sebagai dua elemen sebuah vektor, dan transformasinya sendiri dipandang sebagai perkalian matriks dengan bentuk:

$$H_1 = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$$

Untuk proses dekripsi dapat dengan cara menginvers matriks tersebut diatas. Kelebihan lain adalah matriks tersebut dapat digeneralkan menjadi matriks dengan dimensi yang lebih tinggi untuk mentransformasikan vektor yang lebih dari dua variabel, dengan fungsi rekursif:

$$H_n = \begin{bmatrix} 2 \times H_{n-1} & H_{n-1} \\ H_{n-1} & H_{n-1} \end{bmatrix}$$

Contoh matriks untuk mentransformasi vektor dengan 4 buah variabel (a, b, c, dan d):

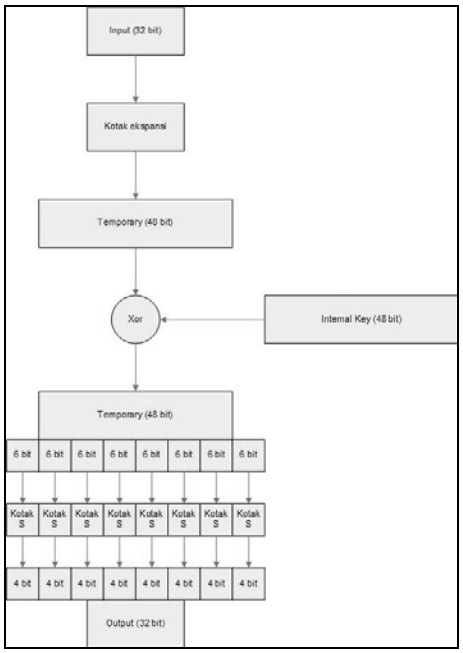
$$H_2 = \begin{bmatrix} 4 & 2 & 2 & 1 \\ 2 & 2 & 1 & 1 \\ 2 & 1 & 2 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Hanya algoritma twofish saja yang mengimplementasikan metode ini.

3.6 Ekspansi dan filter

Untuk memperkuat algoritma kriptografi, beberapa perancang menggunakan metode ekspansi dan filter. Metode ini bertujuan menggelembungkan ukuran blok dan memprosesnya dengan metode-metode tertentu. Hasil keluaran proses tersebut dimasukkan ke fungsi filter yang akan mengembalikan ukuran blok menjadi seperti semula.

Metode ini bersifat irreversible (tidak dapat mengembalikan nilai awal), sehingga metode ini harus dipasangkan dengan metode lain yang memiliki kemampuan enkripsi dan dekripsi irreversible.



Gambar 3 – Salah satu contoh penggunaan ekspansi dan filter

Metode yang umumnya dipakai sebagai wadah metode ekspansi dan filter adalah jaringan feistel. Jaringan feistel memiliki sifat mampu mengenkripsi dan dekripsi dengan memanfaatkan metode irreversible seperti yang telah dijelaskan sebelumnya (subbab 3.3).

Untuk mengekspansi blok awal menjadi lebih besar diperlukan kotak ekspansi. Kotak ini bekerja seperti pada kotak permutasi, namun memiliki kemampuan membesarkan blok. Blok yang telah diekspansi tersebut biasanya lalu di lakukan operasi xor dengan kunci internal (bila dilakukan pada metode perulangan). Hasil dari xor tersebut lalu dimasukkan ke dalam kotak substitusi, sebuah kotak yang mampu mengecilkan ukuran blok.

Hanya algoritma twofish saja yang mengimplementasikan metode ini dengan sedikit perbedaan pada proses ekspansi dan filternya.

3.7 Kotak MDS

Kotak Most Distance Separable (MDS) adalah sebuah matriks yang dapat dioperasikan secara linear kepada sebuah vektor menjadi sebuah vektor hasil. Vektor hasil tersebut merupakan vektor yang terbesar yang dapat dihasilkan namun dengan jarak minimum dari vektor awal.

Jika ada vektor dengan “a” elemen, dan ada vektor lain dengan “b” elemen, akan menghasilkan vektor baru dengan “a+b” elemen. Dengan mengandung jumlah minimum elemen bukan nol pada setiap bukan nol vektor adalah “b+1”. Sehingga “jarak” antara dua vektor yang berbeda yang dihasilkan dengan kotak MDS adalah sekurang-kurangnya “b+1”.

Hanya algoritma twofish saja yang mengimplementasikan metode ini.

4. Blowfish

4.1 Deskripsi dan pengenalan

Blowfish merupakan sebuah algoritma kunci simetrik cipher blok yang dirancang pada tahun 1993 oleh Bruce Schneier untuk menggantikan DES. Pada saat itu banyak sekali rancangan algoritma yang ditawarkan, namun hampir semua terhalang oleh paten atau kerahasiaan pemerintah Amerika. Schneier menyatakan bahwa blowfish bebas paten dan akan berada pada domain publik. Dengan pernyataan Schneier tersebut blowfish telah mendapatkan tempat di dunia kriptografi, khususnya bagi masyarakat yang membutuhkan algoritma kriptografi yang cepat, kuat, dan tidak terhalang oleh lisensi.

Keberhasilan blowfish dalam menembus pasar telah terbukti dengan diadopsinya blowfish sebagai Open Cryptography Interface (OCI) pada kernel Linux versi 2.5 keatas. Dengan diadopsinya blowfish, maka telah menyatakan bahwa dunia open source menganggap

blowfish adalah salah satu algoritma yang terbaik.

Kesuksesan blowfish mulai memudar setelah kehadiran algoritma-algoritma dengan ukuran blok yang lebih besar, seperti AES. AES sendiri memang dirancang untuk menggantikan DES. Sehingga secara keseluruhan AES lebih unggul dari DES dan juga blowfish.

4.2 Algoritma

Blowfish adalah algoritma kriptografi kunci simetrik cipher blok dengan panjang blok tetap sepanjang 64 bit. Blowfish menerapkan teknik kunci yang berukuran sembarang. Ukuran kunci yang dapat diterima oleh blowfish adalah antara 32 bit hingga 448 bit, dengan ukuran default sebesar 128 bit. Blowfish memanfaatkan teknik pemanipulasian bit (subbab 3.1), kotak permutasi (subbab 3.2), jaringan feistel (subbab 3.3), dan teknik pemutaran ulang dan pergiliran kunci (subbab 3.4) yang dilakukan sebanyak 16 kali.

Algoritma utama terbagi menjadi dua subalgoritma utama, yaitu bagian ekspansi kunci dan bagian enkripsi-dekripsi data. Pengekspansian kunci dilakukan pada saat awal dengan masukan sebuah kunci dengan panjang 32 bit hingga 448 bit, dan keluaran adalah sebuah array subkunci dengan total 4168 byte.

Bagian enkripsi-dekripsi data terjadi dengan memanfaatkan perulangan 16 kali terhadap jaringan feistel. Setiap perulangan terdiri dari permutasi dengan masukan adalah kunci, dan substitusi data. Semua operasi dilakukan dengan memanfaatkan operator Xor dan penambahan. Operator penambahan dilakukan terhadap empat array lookup yang dilakukan setiap putarannya.

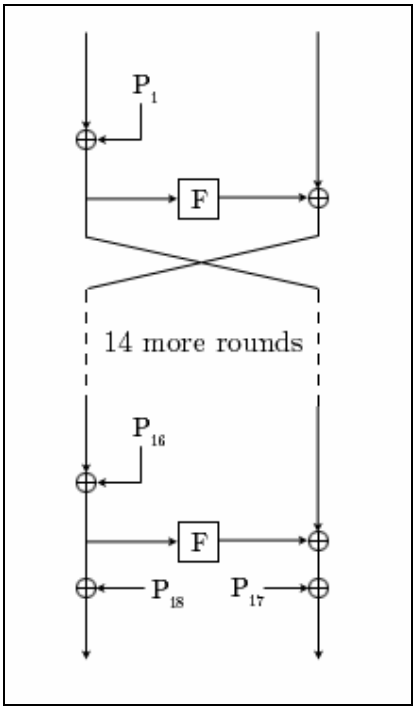
Pembangkitan kunci-kunci

Blowfish menggunakan subkunci berukuran besar. Kunci-kunsi tersebut harus dikomputasikan pada saat awal, sebelum pengkomputasian enkripsi dan dekripsi data. Langkah-langkahnya adalah sebagai berikut:

1. Terdapat kotak permutasi (P-box) yang terdiri dari 18 buah 32 bit subkunci: P1, P2, P3, ... P18. P-box ini telah ditetapkan sejak awal, 4 buah P-box awal adalah sebagai berikut:
   P1 = 0x243f6a88
   P2 = 0x85a308d3
   P3 = 0x13198a2e
   P4 = 0x03707344
2. Xorkan P1 dengan 32 bit awal kunci, xorkan P2 dengan 32 bit berikutnya dari kunci, dan teruskan hingga seluruh panjang kunci telah terxorkan (kemungkinan sampai P14, 14x32 = 448, panjang maksimal kunci).
3. Terdapat 64 bit dengan isi kosong, bit-bit tersebut dimasukkan ke langkah 2.
4. Gantikan P1 dan P2 dengan keluaran dari langkah 3.
5. Enkripsikan keluaran langkah 3 dengan langkah 2 kembali, namun kali ini dengan subkunci yang berbeda (sebab langkah 2 menghasilkan subkunci baru).
6. Gantikan P3 dan P4 dengan keluaran dari langkah 5
7. Lakukan seterusnya hingga seluruh P-box teracak sempurna.
8. Total keseluruhan, terdapat 521 iterasi untuk menghasilkan subkunci-subkucni yang dibutuhkan. Aplikasi hendaknya menyimpannya daripada menghasilkan ulang subkunci-subkunci tersebut.

Enkripsi-dekripsi data

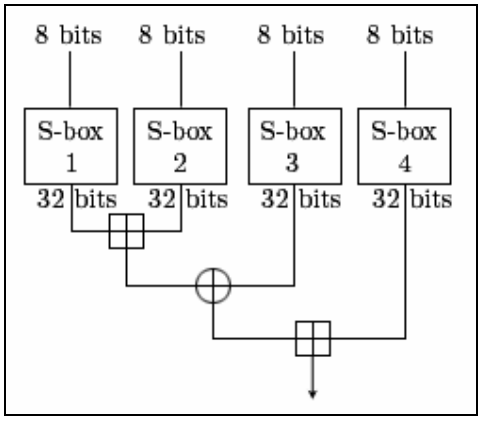
Proses enkripsi-dekripsi data pada algoritma blowfish adalah sebagai berikut:



Gambar 4 – Proses enkripsi Blowfish

1. Masukan dari proses ini adalah 64 bit data yang diinisialkan “x”.

2. Bagi x menjadi 2 buah bagian sama besar, xL (x kiri) sepanjang 32 bit, dan xR (x kanan) sepanjang 32 bit.
3. Lakukan iterasi sebanyak i=1 hingga i=16:
   xL = xL XOR P[i];
   xR = F(xL) XOR xR;
   Swap(xL, xR);
4. Fungsi F adalah sebagai berikut:
   bagi xL menjadi 4 buah 8 bit a, b, c, dan d.
   $F(xL) = ((S[1,a] + S2[2,b] \bmod 2^{32}) \text{ XOR } S[3,c]) + S[4,d] \bmod 2^{32}$.
5. Langkah terakhir adalah:
   Swap(xL, xR);
   xR = xR XOR P[17];
   xL = xL XOR P[18];
   gabungkan xL dan xR menjadi 64 bit
   return hasil gabungan
6. Pada proses dekripsi langkah-langkahnya sama persis dengan proses enkripsi, namun hanya saja P-box digunakan dengan urutan yang terbalik.



Gambar 5 – Proses ekspansi dan filter (fungsi F) pada Blowfish

4.3 Mini Blowfish

Walaupun blowfish dengan ukuran bloknya sepanjang 64 bit terasa sudah cukup kecil, namun terdapat kebutuhan untuk sebuah algoritma kriptografi yang lebih sederhana. Kebutuhan ini dirasakan oleh masyarakat yang hanya membutuhkan kriptografi untuk kepentingan tidak serius, hanya untuk merahasiakan sesuatu yang tidak cukup penting. Untuk itulah dirancang sebuah Mini Blowfish. Mini blowfish mempunyai blok sepanjang 32 bit. Sedangkan algoritma utamanya sama persis dengan blowfish.

Mini blowfish selain menjawab kebutuhan algoritma yang sederhana, juga untuk kelinci percobaan kriptanalisis. Untuk mencoba menjebol blowfish tentu cukup sukar, lebih mudah mencoba menjebol mini blowfish yang hanya memiliki panjang blok 32 bit. Untuk itu perancang blowfish membuat mini blowfish. Dengan harapan, dikarenakan keduanya memiliki algoritma dasar yang sama, maka teknik-teknik penjebolan terhadap mini blowfish dapat juga diterapkan kepada blowfish.

5. Twofish

4.1 Deskripsi dan pengenalan

Pada tahun 1972 dan 1974, US the National Bureau of Standards (sekarang bernama the National Institute of Standards and Technology, atau NIST) mengeluarkan publikasi pertama untuk sebuah standar enkripsi, yang menghasilkan algoritma data Encryption Standard (DES) [NBS77], yang tidak dapat disangkal sebagai algoritma kriptografi yang sangat terkenal dan sangat berhasil.

Disamping kepopulerannya, DES sering diserang dengan berbagai kontroversi. Beberapa orang kriptografer mempertanyakan desain algoritma DES yang mengimplementasikan kunci dan ukuran blok yang terlalu kecil. Dengan berkembangnya pengetahuan tentang kunci terdistribusi akhir-akhir ini, tidak ada keraguan bahwa DES memang memiliki ukuran blok dan kunci yang tidak sesuai dengan kebutuhan masyarakat saat ini.

Triple-DES ditawarkan untuk menjawab pertanyaan diatas, dan memang merupakan solusi sementara dari permaslahan diatas. TDES telah diadopsi oleh banyak sekali aplikasi yang membutuhkan tingkat keamanan tinggi seperti perbankan namun algoritma TDES dirasakan sangat lamban, apalagi dengan hardware pada saat itu.

Merespon kebutuhan masyarakat untuk pengganti DES, NIST mengeluarkan semacam sayembara bernamakan program Advanced Encryption Standard (AES) pada tahun 1997 [NIS97a]. Sebuah sayembara untuk para kriptografer dunia untuk merancang algoritma kriptografi baru sebagai calon pengganti DES.

NIST mengumumkan syarat-syarat minimum yang harus dimiliki algoritma baru tersebut, semua kebutuhan tersebut tertuangkan pada subbab 4.2.

Cara penilaian untuk menentukan algoritma terbaik yaitu dengan cara algoritma yang masuk dipublikasikan ke umum, dan dengan mendengar tanggapan masyarakat akan ditemukan algoritma yang terbaik.

Twofish merupakan salah satu peserta sayembara tersebut, twofish berhasil mencapai peringkat lima besar algoritma terbaik, namun hasil akhir memutuskan bahwa twofish belum mampu menjadi algoritma standar.

4.2 Kriteria NIST

Untuk menggantikan DES, NIST membuka sayembara kepada masyarakat untuk merancang algoritma kriptografi yang sesuai dengan kebutuhan NIST [NIS97b]. Adapun kriteria-kriteria yang disyaratkan NIST adalah sebagai berikut:

- Merupakan algoritma kriptografi simetris cipher blok, dengan panjang blok adalah 128 bit.
- Memiliki tiga opsi panjang kunci, yaitu: 128 bit, 192 bit, dan 256 bit.
- Tidak memiliki kunci lemah (*weak key*).
- Memiliki efisiensi yang tinggi, baik pada software diatas prosesor kelas pentium, maupun ditanamkan pada chip hardware.
- Memiliki rancangan yang fleksibel, yaitu dapat dimodifikasi dengan mudah, misal memperpanjang kunci, dapat diaplikasikan pada cipher aliran, memiliki fungsi hash yang dapat diubah, dll.
- Walaupun menuntut banyak kemampuan, namun desain algoritma harus sesederhana mungkin.
- Terdapat beberapa kriteria yang menyangkut hardware yang spesifik, tidak ditampilkan disini.

Setelah twofish diterima sebagai kandidat pengganti DES, maka NIST mensyaratkan kriteria tambahan khusus untuk twofish, yaitu:

- Twofish 16 putaran tidak boleh memiliki *chosen-palintext attack* yang memerlukan kurang dari 2^{80} chosen plaintext dan menghabiskan waktu 2^n dengan n adalah panjang kunci.
- Twofish 12 putaran tidak boleh memiliki *related-key attack* yang memerlukan kurang dari 2^{64} chosen plaintext dan menghabiskan waktu 2^n dengan n adalah panjang kunci.

Dari semua kriteria yang disyaratkan NIST, twofish berhasil memenuhinya. Denagn terpenuhinya kriteria-kriteria tersebut, twofish maju dalam 5 besar memperebutkan tahta pengganti DES.

4.3 Algoritma

Twofish merupakan algoritma kriptografi kunci simetrik cipher blok dengan panjang setiap blok adalah tetap 128 bit. Sedangkan kunci yang dapat diterima adalah: 128, 192, atau 256 bit. Twofish memanfaatkan teknik pemanipulasian bit (subbab 3.1), kotak permutasi / pemutihan (subbab 3.2), jaringan feistel (subbab 3.3), pemutaran ulang dengan pergiliran kunci dengan jumlah perputaran dan pergiliran kunci sebanyak 16 kali (subbab 3.4), tranformasi pseudo-Hadamard (subbab 3.5), ekspansi dan filter (subbab 3.6), dan kotak MDS (Most Distance Separable) (subbab 3.7).

Seperti halnya blowfish, twofish juga memiliki dua tahapan utama, yaitu tahap pembangkitan kunci dan tahap algoritma utama.

Pembangkitan kunci-kunci

Jumlah kunci internal yang harus dibangkitkan adalah sejumlah 40 kunci masing-masing 32 bit (K_0 hingga K_{39}). Dan juga dibutuhkan pembangkitan 4 buah kotak substitusi dari yang bergantung pada kunci. Twofish dapat menerima kunci sepanjang 128, 192, dan 256 bit (N).

Kemudian terdefinisi k=N/64. Kunci M terdiri dari 8k byte, m_0, ..., m_{8k-1}. Byte-byte tersebut pertama-tama diubah menjadi 2k buah yang masing-masing terdiri dari 32 bit.

$$M_i = \sum_{j=0}^{3} m_{(4i+j)} \cdot 2^{8j} \qquad i = 0, \ldots, 2k-1$$

Hasil fungsi diatas kemudian digolongkan menjadi dua buah, ganjil dan genap.

$M_e = (M_0, M_2, ..., M_{2k-2})$

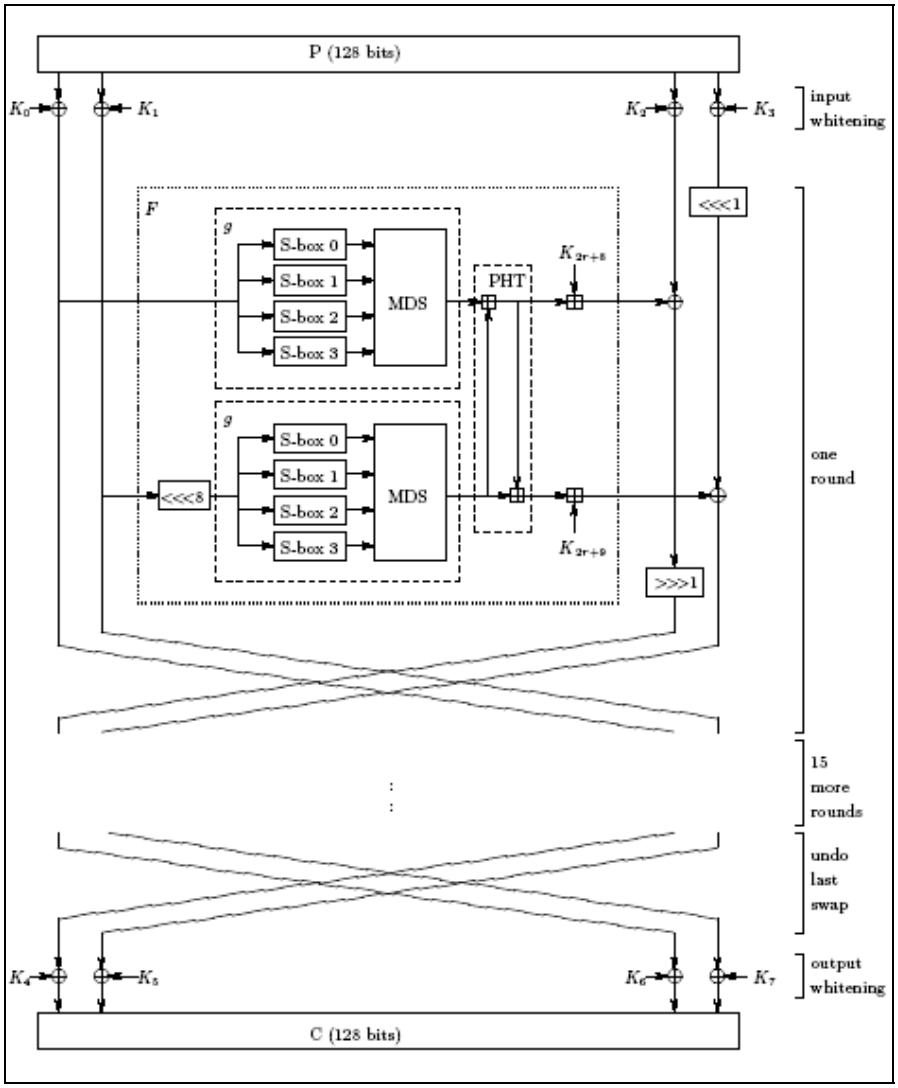
$M_o = (M_1, M_3, ..., M_{2k-1})$

Selanjutnya adalah kotak S. Langkah pertama adalah dengan mengelompokkan kunci menjadi masing-masing 8. Kemudian kelompok kunci tersebut dikalikan dengan matriks 4x8 yang diturunkan dari RS. Setiap hasil sepanjang 4 byte duartikan sebagai satu buah 32 bit, menghasilkan kotak S.

$$\mathrm{RS} = \begin{pmatrix} 01 & A4 & 55 & 87 & 5A & 58 & DB & 9E \\ A4 & 56 & 82 & F3 & 1E & C6 & 68 & E5 \\ 02 & A1 & FC & C1 & 47 & AE & 3D & 19 \\ A4 & 55 & 87 & 5A & 58 & DB & 9E & 03 \end{pmatrix}$$

Hasil keluaran tahap ini adalah 2 buah matriks, matriks M genap dan matriks M ganjil, dan sebuah matriks kotak substitusi.

Algoritma utama



Gambar 6 – Algoritma twofish

Langkah-langkah algoritma twofish adalah sebagai berikut:

1. Masukan satu blok plain teks adalah 128 bit. Satu blok tersebut dibagi menjadi 4 buah subblok yang masing-masing sepanjang 32 bit (A, B, C, dan D).
2. Masing-masing subblok tersebut diputihkan dengan mengxorkan dengan kunci K0, K1, K2, dan K3.

Langkah-langkah 1 putaran adalah sebagai berikut:

1. 2 buah 32 bit yang kiri (A dan B) merupakan input dari fungsi g (yang merupakan bagian dari fungsi f), yang salah satunya (B) di geser ke kiri sejauh 8 bit dahulu.
2. Fungsi g memiliki 4 buah kotak substitusi yang dibangkitkan oleh kunci.
3. Keluaran fungsi kotak substitusi dilakukan percampuran linear menggunakan kotak Most Distance Separable (subbab 3.7).
4. Keluaran fungsi g dimasukkan ke fungsi transformasi pseudo-Hadamard, kemudian ditambahkan dengan 2 buah 32 bit dari kunci.
5. Dua buah 32 bit hasil kemudian di xorkan dengan C dan D. Hasil xor dengan C digeser ke kanan sejauh 1 bit. Dan untuk D sebelum dixorkan digeser ke kiri sejauh 1 bit.
6. 2 buah 32 bit kiri dan kanan dipertukarkan (A dan B dipertukarkan dengan C dan D).

Langkah diatas dilakukan hingga 16 kali putaran. Kemudian langkah-langkah selanjutnya:

1. Hasil keluaran setelah diputar 16 kali, ditukar lagi (A dan B diperukarkan dengan C dan D).
2. Hasil dari pertukaran tersebut di xorkan dengan empat buah 32 bit dari kunci menghasilkan cipher teks.

Fungsi F

Fungsi F adalah permutasi yang bergantung pada kunci dengan nilai 64 bit. Fungsi ini menerima 3 argumen, dua buah 32 bit R0 dan R1, dan nomor putaran untuk menentukan subkunci mana yang dipakai. R0 akan diserahkan ke fungsi g yang akan mengembalikan T0. R1 akan digeser sejauh 8 bit yang kemudian di berikan juga ke fungsi g yang akan mengembalikan T1. Hasil T0 dan T1 kemudian dikombinasikan ulang menggunakan transformasi pseudo-Hadamard, yang kemudian ditambahkan dengan dua buah 32 bit dari kunci.

T0 = g(R0);

T1 = g(shiftLeft(R1,8));

$F0 = (T0+T1+K_{2r+8}) \bmod 2^{32}$;

$F1 = (T0+2T1+K_{2r+9}) \bmod 2^{32}$;

F0 dan F1 adalah hasil dari F, yang masing-masing sepanjang 32 bit. Hasil keluaran ini nantinya akan dipertukarkan dan dimasukkan kembali ke putaran selanjutnya.

Fungsi G

Fungsi g merupakan jantung dari keseluruhan algoritma twofish. 32 bit masukan X dari fungsi F dipecah menajdi 4 buah yang masing-masing sepanjang 8 bit. Setiap 8 bit kemudian diproses dengan kotak S yang bersesuaian. Setiap kotak S bersifat bijektif, yaitu menerima 8 bit dan mengeluarkan 8 bit pula. 4 buah 8 bit hasil keluaran kemudian dikalikan dengan matriks Most Distance Separable (MDS) 4x4. Hasil pengalian kemudian diartikan sebagai 32 bit, yang merupakan keluaran dari fungsi g, yang kemudian akan dikembalikan kembali ke fungsi F.

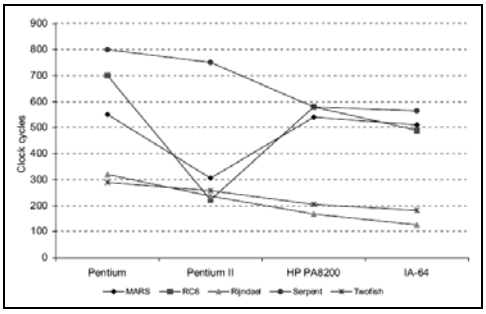
Matriks MDS yang setiap elemennya ditampilkan sebagai heksadesimal adalah sebagai berikut:

$$\mathrm{MDS} = \begin{pmatrix} 01 & EF & 5B & 5B \\ 5B & EF & EF & 01 \\ EF & 5B & 01 & EF \\ EF & 01 & EF & 5B \end{pmatrix}$$

4.4 Perbandingan dengan lima besar AES

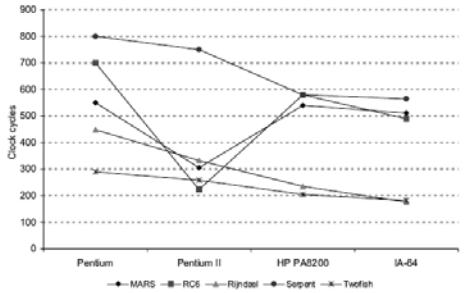
Hasil penseleksian algoritma yang berhak menggantikan DES menjadi standar baru membuahkan 5 buah algoritma. Algoritma-algoritma tersebut telah memenuhi syarat-syarat minimum yang telah disyaratkan oleh NIST (subbab 4.2). Selain telah memenuhi persyaratan-persyaratan tersebut, kelima algoritma tersebut juga telah dipublikasikan ke khalayak umum untuk mendapatkan tanggapan. Selain masukan dari publik, tentu penilaian juga berdasarkan kualitas dari algoritma itu sendiri. Yaitu keamanan dan performansinya. Penilaian keamanan dan performansi dilakukan secara terbuka oleh NIST.

Kelima algoritma yang berhasil meraih tingkat lima besar adalah: MARS, RC6, Rijndael, Serpent dan Twofish. Dari kelima algoritma tersebut kesemuanya memiliki tingkat keamanan yang dapat dikatakan hampir seimbang. Oleh karena itu penilaian terbesar adalah dari segi performansi kecepatan. Dari segi performansi kecepatan komputasi, Rijndael dan twofish meninggalkan ketiga peserta lainnya. Rc6 dan mars sangat cepat pada prosesor-prosesor tertentu, khususnya keluarga intel pentium, namun lambat pada keluarga prosesor lainnya. Sedangkan serpent selalu pada urutan terakhir pada pengujian performansi apapun pada keluarga prosesur apapun.



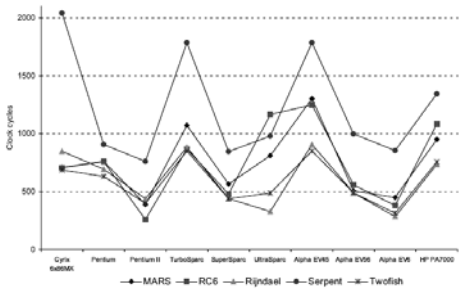
Gambar 7 – Grafik performansi lima besar AES untuk kunci 128 bit

Untuk panjang kunci 128 bit, performansi twofish hampir sama dengan rijndael. Twofish menang pada prosesor pentium, namun kalah pada prosesor lainnya, dengan selisih yang tidak terpaut jauh. Sedangkan algoritma lain, mars hampir membutuhkan waktu dua kali lipat dari pada rijndael/twofish. Dan algoritma rc6 dan serpent keduanya hampir menghabiskan waktu tiga kali lipat daripada rijndael/twofish.



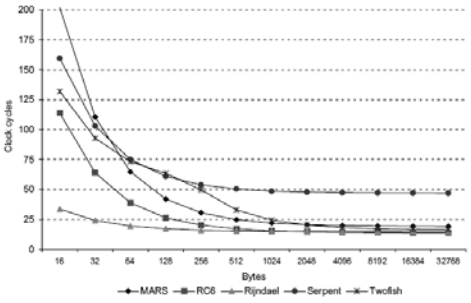
Gambar 8 – Grafik performansi lima besar AES untuk kunci 256 bit

Untuk panjang kunci 256 bit, performansi twofish jauh meninggalkan semua algoritma lainnya. Terurut dari performansi terbaik hingga terburuk adalah: twofish, rijndael, mars, rc6, dan serpent. Namun standar yang akan digunakan oleh NIST adalah kunci dengan panjang 128 bit, sehingga kemenanagn twofish di kunci 256 bit kurang mendukung penilaian secara keseluruhan.



Gambar 9 – Grafik performansi lima besar AES dalam bahasa pemrograman C

Setelah kelima besar AES diimplementasikan menggunakan bahasa C, ternyata performansi kecepatan keseluruhan kecuali algoritma serpent hampir sama. Hal ini menandakan bahwa kekuatan twofish dan rijndael yang menang mutlak pada bahasa assembly adalah kekuatan permainan bit, yang memang akan lebih cepat pada bahasa assembly.



Gambar 10 – Grafik performansi lima besar AES dalam tahap inisialisasi kunci-kunci internal

Dalam grafik diatas terlihat bahwa twofish menempati urutan ketiga, terkalahkan oleh rijndael di urutan pertama, rc6 di urutan kedua. Hal tersebut terjadi sebab, memang ciri khas twofish (dan juga blowfish) adalah banyaknya kombinasi kunci internal yang harus dibangkitkan. Twofish dan blowfish harus membangkitkan lebih dari 500 bit kunci-kunci internal sebelum memasuki proses enkripsi-dekripsi.

Setelah melalui proses seleksi yang ketat, akhirnya terpilihlah Rijndael sebagai pemenang, sebab memang dari segi performansi kecepatan komputasi, rijndael hampir selalu menang di semua pengujian. Dengan demikian DES telah mendapatkan penggantinya, rijndael, yang sejak saat itu lebih dikenal dengan AES (Advanced Encryption Standard).

6. Perbandingan blowfish dan twofish

6.1 Perbandingan umum

Bidang	Blowfish	Twofish
Tahun perancangan	1993	1997
Algoritma pembanding	DES	AES
Perancang	Bruce Schneier	
Panjang blok	64 bit	128 bit
Panjang kunci	32 – 448 bit, kelipatan 8 bit	128, 192, atau 256 bit
Manipulasi bit	Ya (subbab 3.1)	
kotak permutasi	Ya (subbab 3.2)	
Jaringan Feistel	Ya (subbab 3.3)	
Putaran & pergiliran kunci	Ya (subbab 3.4)	
Transformasi Pseudo-Hadamard	Tidak	Ya (subbab 3.5)
Ekspansi dan Filter	Tidak	Ya (subbab 3.6)
Kotak MDS	Tidak	Ya (subbab 3.7)
Jumlah putaran	16 putaran	
Performansi kecepatan	18 CU / 1 byte enkripsi	

Lisensi	Tidak berlisensi
Biaya	Gratis

Tabel 1 – Perbandingan umum antara blowfish dengan twofish

6.2 Performansi kecepatan

Algoritma	#CU/ putaran	# putaran	#CU/1 byte enkripsi
Blowfish	9	16	18
Twofish	9	16	18
Khufu	5	32	20
RC5	12	16	23
DES	18	16	45
3DES	50	8	50
IDEA	18	48	108

Tabel 2 – Perbandingan performansi kecepatan algoritma blowfish, twofish, dan beberapa algoritma terkenal lainnya

Keterangan:
CU (CPU Unit): 1 CPU Unit secara teoritis berarti 1 Hz, CPU dengan clock speed 1 GHz mampu memproses 1 Giga CU/detik. Namun CPU tidak hanya dipakai untuk memproses enkripsi/dekripsi tersebut saja, namun juga dipakai untuk sistem operasi, program-program lainnya.

Dari tabel diatas terlihat bahwa performansi kecepatan antara blowfish dan twofish sama persis, yaitu sebesar 18 CPU Unit / byte. Secara teoritis, untuk memproses data sebesar 1 MB, memerlukan 18 Mega CPU Unit, dengan prosesor 1800 MHz, maka hanya memerlukan waktu 0,01 detik, jika seluruh prosesor hanya untuk menjalankan proses tersebut (yang tentu saja tidak demikian).

Perbandingannya dengan algoritma Triple DES yang juga sangat populer, khususnya di dunia perbankan. Triple DES memerlukan jumlah CPU Unit hampir tiga kali lipat dari pada CPU Unit yang dibutuhkan oleh blowfish dan twofish.

6.3 Performansi bahasa dan prosesor

Prosesor	Bahasa	Waktu untuk mengenkripsi
Pentium II	Assembly	285
	MS C	600
	Borland C	640
Pentium	Assembly	290
	MS C	630
	Borland C	870
UltraSPARC	C	750
PowerPC 750	C	590
68040	C	3500

Tabel 3 – Perbandingan prosesor dan bahasa pemrograman terhadap performansi kecepatan enkripsi blowfish dan twofish

Dari tabel diatas terlihat bahwa disamping prosesor, juga terdapat bahasa pemrograman yang mempengaruhi performansi kecepatan. Dapat dilihat diatas bahwa untuk memaksimalkan performansi, yaitu dengan mengimplementasikan algoritma blowfish dan twofish menggunakan bahasa pemrograman assembly.

6.4 Area aplikasi

Blowfish dan twofish keduanya sangat terkenal di dunia kriptografi, hal ini ditandai dengan banyaknya aplikasi yang mengimplementasi-kan blowfish, twofish, maupun keduanya. Aplikasi yang paling menonjol adalah kernel Linux versi 2.5 keatas, dimana blowfish menjadi Open Cryptography Interface. Kelebihan lain yang ditawarkan oleh keduanya adalah bebas lisensi dan paten, sehingga semua aplikasi dapat memanfaatkan algoritma blowfish yang hebat tanpa perlu membeli paten apapun.

Kedua algoritma tersebut dapat diterapkan disetiap aplikasi apapun, namun untuk lebih spesifik, aplikasi-aplikasi sederhana yang tidak memerlukan keamanan super, setidaknya dengan algoritma blowfish cukup, walaupun dari segi performansi, kedua algoritma tersebut hampir sama (subbab 6.2). Kelebihan aplikasi yang mengimplementasikan blowfish adalah blowfish telah menjadi Open Cryptography Interface pada beberapa sistem operasi, sehingga lebih memudahkan pengembang aplikasi dalam mengakses API kriptografi blowfish.

Namun untuk aplikasi-aplikasi yang menginginkan keamanan ekstra, hendaknya memilih twofish, sebab dari segi teknologi twofish lebih baik dengan memanfaatkan teknik transformasi pseudo-Hadamard (subbab 3.5), teknik ekspansi dan filter (subbab 3.6), dan teknik kotak MDS (subbab 3.7), yang mana tidak terdapat pada blowfish.

Dari segi kesulitan pengkodean, tentu twofish lebih sulit untuk dikode daripada blowfish, mengingat langkah-langkah twofish lebih

banyak daripada blowfish. Namun untuk bahasa-bahasa standar, seperti C, C++, Java, dll. Telah ada beberapa versi yang tersedia secara open source maupun gratis yang dapat diterapkan pada program yang membutuhkan.

7. Kesimpulan

NIST, sebuah lembaga penstadardisasi, pada tahun 1976 telah menetapkan DES menjadi standar algoritma kriptografi kunci simetrik cipher blok. DES adalah manipulasi dari algoritma Lucifer yang dikembangkan oleh IBM.

Pada saat itu, DES telah cukup aman dengan panjang kunci sebesar 56 bit dan panjang blok sebesar 64 bit. Namun berlalunya waktu, dengan panjang kunci itu tidak mampu menahan serangan-serangan terhadap algoritma, bahkan oleh serangan brute-force sekalipun.

Pada tahun 1993, Bruce Schneier merancang algoritma kunci simetrik cipher blok bernama Blowfish. Blowfish dirancang untuk menggantikan DES. Blowfish memiliki berbagai macam keunggulan dibandingkan DES. Yaitu memiliki teknik-teknik baru di bidang kriptografi pada saat itu. Kehadiran blowfish sangat diterima oleh masyarakat, alasan utama adalah karena algoritma tersebut bebas lisensi dan gratis. Selain memiliki kelebihan tersebut, dari segi kualitas keamanan dan performansi juga sangat baik. Kehebatan algoritma tersebut diakui oleh komunitas open source dengan diimplementasikannya blowfish dalam kernel Linux versi 2.5 keatas.

Namun blowfish juga akhirnya harus menyerah dengan waktu, pada tahun 1997, dengan perkembangan prosesor yang sangat cepat, algoritma-algoritma kriptografi dengan kunci 64 bit kebawah sudah dirasakan tidak tepat. Sebab dengan brute force tidak lama dapat menjebol algoritma tersebut. Untuk itulah NIST merasakan perlu adanya standar baru pengganti DES. Akhirnya NIST membuka sayembara untuk umum. Siapapun boleh mensubmit algoritma kriptografi, namun dengan syarat minimum yang telah dipublikasikan sebelumnya, yang diantaranya adalah minimum panjang blok dan kunci adalah 128 bit. Dari sekian banyak algoritma yang disubmit, semuanya dipublikasikan ke khalayak umum untuk dinilai.

Setelah melalui proses seleksi, tampillah 5 buah algoritma yang menjadi nominasi pengganti DES, kelimanya adalah: MARS, RC6, Rijndael, Twofish, dan Serpent. Dari kelima algoritma tersebut dapat dikatakan dua besarnya adalah Rijndael dan Twofish. Keduanya sangat aman, dan performansinya sangat baik. Namun dalam hal performansi kecepatan rijndael unggul dibandingkan twofish hampir pada setiap prosesor, kecuali pentium. Pada tahun 2001, akhirnya rijndael terpilih sebagai standar baru algoritma kriptografi kunci simetrik menggantikan DES. Algoritma rijndael pun telah berganti nama menjadi algoritma AES.

Daftar Pustaka

[ALB94] Albrecht, Beutelspacher. (1994). The Future Has Already Started or Public Key Cryptography.

[COP94] Coppersmith, (1994). The Data encryption Standard (DES) and its strength against attacks. IBM journal of research and Development 38.

[KAH79] Kahn, David. (1979). Cryptography Goes Public.

[LIS02] Liskov, M. (2002). Tweakable Block Ciphers.

[LUC00] Lucks, Stefan. (2000). The Saturation Attack – a Bait for Twofish. Theoretische Informatik, University of Mannheim Germany.

[MUN06] Munir, Rinaldi. (2006). Bahan kuliah IF5054 Kriptografi. Departemen Teknik Informatika, Institut Teknologi Bandung Indonesia.

[NBS77] National Bureau of Standards, NBS. (1977). FIPS PUB 46, “Data Encryption Standard,". National Bureau of Standards, U.S. Department of Commerce, Jan 1977.

[NIS97a] National Institute of Standards and Technology. (1997). Announcing Development of a Federal Information Standard for Advanced Encryption Standard. Federal Register, v. 62, n. 1, 2 Jan 1997, pp. 93-94.

[NIS97b] National Institute of Standards and Technology. (1997). Announcing Request for Candidate Algorithm Nominations for the Advanced Encryption Standard (AES). Federal Register, v. 62, n. 117, 12 Sep 1997, pp. 48051-48058.

[NIS01] National Institute of Standards and Technology. (2001). Announcing the Advanced Encryption Standard

(AES). Federal Register, 26 Nov 2001.

[RIJ97]. Rijmen, Vincent. (1997). Cryptanalysis and Design of Iterated Block Ciphers.

[RSA06] RSA Laboratories. (2006). RSA Laboratories' Frequently Asked Question About Today's Cryptography. http://www.rsasecurity.com/rsalabs/node.asp?id=2152. (Akses: 10 Oktober 2006)

[SCH93] Schneier, Bruce. (1993). Description of a New Variable-Length Key, 64-bit Block Cipher (Blowfish). Fast Software Encryption.

[SCH95]. Schneier, Bruce. (1995). The Blowfish Encryption Algorithm – One Year Later.

[SCH96] Schneier, Bruce. (1996). Applied Cryptography, Protocols, Algorithms, and Source Code in C, Second Edition. John wiley & Sons.

[SCH98] Schneier, Bruce. (1998). Twofish: A 128-Bit Block Cipher. Counterpane Systems USA.

[SCH00] Schneier, Bruce. (2000). A Performance Comparison of the Five Finalists. Counterpane Internet Security USA.

[VAU96] Vaudenay, Serge. (1996). On the Weak Keys of Blowfish.