PENERAPAN TEORI BILANGAN DALAM ALGORITMA ENKRIPSI PADA *PRETTY GOOD PRIVACY* (PGP)

Teguh Budi Wicaksono – NIM : 13505028

Program Studi Teknik Informatika
Sekolah Tinggi Elektro dan Informatika
Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
2007

E-mail: if15028@students.if.itb.ac.id

Abstrak

Makalah ini membahas tentang teori bilangan dalam algoritma enkripsi dan penerapannya pada *Pretty Good Privacy* (PGP). *Pretty Good Privacy* (PGP) adalah sebuah program yang menyediakan fasilitas kriptografi untuk privasi dan autentifikasi. PGP digunakan untuk keamanan dalam berkomunikasi di dalam internet baik dalam pengiriman surat ataupun dokumen elektronik.

PGP menyediakan berbagai fasilitas mulai dari autentifukasi, pengerahasiaan, pengkompresian, hingga segmentasi. PGP menggunakan berbagai metode enkripsi. Untuk autentifikasi PGP menggunakan RSA dalam *digital signature*. Sedangkan untuk enkripsinya PGP memakai bermacam-macam cara seperti CAST, IDEA, *Three-key* 3DES ataupun RSA. Di makalah ini akan dibahas mengenai PGP sendiri, algoritma RSA dan sedikit tentang DES.

RSA menggunakan manajemen kunci dalam melakukan enkripsinya. Keamanan enkripsi RSA telah diakui karena RSA mengunakan dua kunci yang saling berhubungan. Salah satu kunci adalah *public-key* dan satu lagi *private-key*. Untuk mencari kunci ini diperlukan waktu yang lama atau perhitungan yang rumit. Inilah yang menjadikan RSA belum dapat dikalahkan algoritma enkripsi lainnya.

Kata kunci: *Pretty Good Privacy*, RSA, *Data Encryption Standart*, *hash code*, enkripsi, dekripsi.

1. Pendahuluan

Internet merupakan sebuah media komunikasi dari bidang bisnis hingga hiburan. Banyak orang yang mendistribusikan aplikasi maupun sebuah pesan atau dokumen melalui internet melalui berbagai jenis vendor.

Sekarang internet berkembang hingga akhirnya memiliki peran yang sangat besar dalam berbagai bidang. Banyak orang yang menggunakan fasilitas-fasilitas di internet. Banyak orang yang bergantung pada internet untuk bisnis maupun hal lainnya. Tetapi kerena banyaknya orang yang menggunakan internet, maka ada juga orang-orang yang tidak bertanggung jawab berkrliaran di internet.

Hal ini menyebabkan sekarang ini banyak permintaan akan fasilitas kerahasiaan dan autentifikasi. Hal ini diperlukan untuk menjaga privasi seseorang dalam pendistribusian aplikasi maupun dokumen dalam dunia maya ini. Sehingga aplikasi yang dikirimkan aman dan tidak berubah hingga sampai ke tangan penerima. Untuk itu dilakukan proses enkripsi dan dekripsi terhadap data yang ingin dilirim tersebut. Memang diperlukan berbagai pengamanan dalam arus informasi, tidak hanya di dunia digital saja tentunya.

Salah saytu cara untuk menjaga isi pesan yang akan dikirim dan mengamankannya dari kebocoran adalah dengan cara mengrnkripsi pesan tersebut. Enkripsi adalah merubah isi dari pesan yang asli menjadi sebuat pesan yang tidak dimengerti isinya. Hal ini dapat menjaga kerahasiaan isi pesan yang ingin dikirim. Di pihak penerima melakukan dekripsi atas pesan rahasia tersebut sehingga dapat dibaca kembali. Enkripsi dan dekripsi biasa dilakukan dengan kunci rahasia yang telah ditentukan oleh pengirim atau penerima.

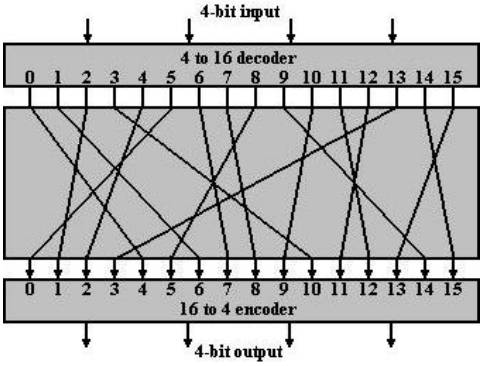
2. *Cipher* Blok

Terdapat dua jenis metode mengenkripsi yaitu:

1. *stream cipher*: metode ini mengenkripsi arus data digital satu bit atau satu byte sekali jalan

2. *Cipher* blok: metode enkripsi dimana suatu plaintext dibagi-bagi menjadi beberapa blok yang memiliki panjang yang sama.

Dalam cipher blok, suatu blok dari plaintext diangap sebagai suatu kesatuan dan digunakan untuk menghasilkan ciphertext yang memiliki panjang yang sama. Biasanya digunakan blok yang berukuran 64 atau 128 bit.



Gambar 1 Substitusi blok 4-bit-4-bit

Tabel 1 tabel enkripsi dan dekripsi dari gambar 1

Plaintext	Ciphertext
0000	0100
0001	0110
0010	0001
0011	1010
0100	0010
0101	0000
0110	0111
0111	1000
1000	0101
1001	1110
1010	1001
1011	1100
1100	1011
1101	0011
1110	1111
1111	1101

Plaintext	Ciphertext
0000	0101
0001	0010
0010	0100
0011	1101
0100	0000
0101	1000
0110	0001
0111	0110
1000	0111
1001	1010
1010	0011
1011	1100
1100	1011
1101	1111
1110	1001
1111	1110

2.1. Diffusion and Confusion

Hal ini diperkenalkan pertama kali oleh Claude Shannon. Shannon berencana mencegah penyerang menyerang bedasarkan analisis statistik. Anggap penyerang mengetahui ststistik karakter dari plaintext, seperti pesan yang dapat dibaca oleh manusia dalam suatu bahasa, frekuensi penggunaan berbagai huruf, atau bahkan kalimat atau kata yang ada dalam plaintext. Jika statistik ini tercermin dalam ciphertext, maka kriptanalis bisa menduga kunci atau sebagian kunci.

Dalam *diffusion* setiap digit dalam ciphertext mempengaruhi digit dari ciphertext yang lain. Contohnya mengenkripsi sebuah pesan M = m_1, m_2, m_3, ... karakter dengan operasi untuk tiap bit ciphertext:

$$Y_n = \sum_{i=1}^{k} mn_{+i} \pmod{26}$$

Diffusion berusaha membuat rumit hubungan antara plaintext dengan ciphertext, sedangkan *confusion* berusaha membuat rumit hubungan antara ciphertext dengan nilai dari kunci yang digunakan.

3. Prinsip Public-key Cryptosystems

Konsep dari *Public-key cryptosystem* berkembang dari percobaan penyerangan ke dua masalah yang paling bermasalah pada enkripsi simetrik. Permasalahan yang pertama adalah keamanan pendistribusian kunci. Dimana untuk keamanan akan bocornya kunci diperlukan:

1. Dimana dua orang yang bersangkutan telah berbagi sebuah kunci. atau
2. Pusat pendistribusian kunci.

Permasalahan yang ke-dua adalah jika penggunaan kriptografi menyebar luas, tidak hanya di situasi militer tapi juga untuk kepentingan komersial dan privasi. Jadi surat dan dokumen elektronik perlu *signature* yang setara dalam dokumen. Dapatkah sebuah metode menjanjikan kepuasan semua pihak, dimana sebuah pesan digital telah dikirim oleh rang tertentu?

3.1 Public-key cryptosystems

Algoritma *public-key* bergantung pada sebuah kunci untuk mengenkripsi dan kunci yang berbeda tapi berhubungan untuk mendekripsi. Algoritma ini memiliki karakteristik yang penting, sebagai berikut:

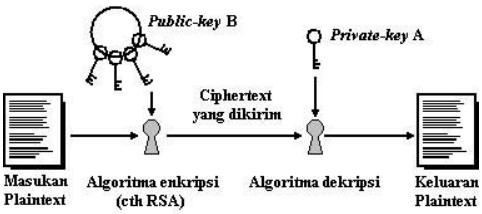
1. Secara komputasi tidak mungkin untuk menentukan kunci untuk dekripsi jika hanya diketahui algoritma enkripsi dan kunci enkripsinya.
2. Kedua kunci yang bersangkutaan dapat digunakan untuk enkripsi sedangkan kunci satu lagi digunakan untuk dekripsi.

Enkripsi dengan *public-key* memiliki enam komponen:

1. **Plaintext**: ini adalah pesan yang dapat dibaca atau dimengerti, yang

diberikan ke algoritma sebagai masukan.

2. **Algoritma enkripsi**: algoritma enkripsi yang mengandung berbagai macam transformasi.
3. ***Public* dan *private key***: pasangan kunci ini telah dipilih sehingga satu digunakan untuk enkripsi dan satu lagi untuk dekripsi.
4. **Ciphertext**: pesan yang telah teracak yang merupakan hasil keluaran algoritma. Untuk suatu pesan, dua kunci yang berbeda menghasilkan dua ciphertext yang berbeda.
5. **Algoritma dekripsi**: algoritma ini menerima ciphertext dan kunci yang cocok untuk menghasilkan pesan yang sebenarnya.



Gambar 2 Enkripsi Dengan *Public-key*

Langkah-langkah utamanya adalah sebagai berikut:

1. Setiap user membuat sepasang kunci yang digunakan untuk enkripsi dan dekripsi.
2. Setiap user menaruh salah satu kuncinya di tempat yang dapat diakses umum. Ini adalah *public-key*. Kunci satu lagi tetap disimpan dan dijaga kerahasiaannya
3. Jika B ingin mengirim pesan ke A, maka B akan mengenkripsi pesannya dengan menggunakan *public*-key dari A.
4. Ketika A menerima pesan ia mendekripsi pesannuya dengan *private-key*-nya, tidak ada orang lain yang bisa mendekripsi pesan tersebut, karena hanya A yang tahu *private-key* yang digunakan.

Dengan ini komunikasi menjaadi aman selama *private-key* terjaga. Seorang dapat mengganti *private-key*-nya dan mempublikasikan *public-key* yang baru, untuk menggantikan yang lama.

3.2. Aplikasi dari Public-key Cryptosystems

Tergantung dari aplikasi yang digunakan, pengirim menggunakan apakah *private-key* sang pengirim atau *public-key* penerima atau bahkan keduanya. Penggunaan Public-key Cryptosystems diklasifikasikan menjadi tiga kategori:

- Enkripsi/dekripsi: pengirim menenkrpsi pesan dengan *public-key* dari sang penerima.
- *Digital signature*: pengirim 'menandai' pesan dengan *private-key*-nya.
- Pertukaran kunci: kedua pihak berkerjasama untuk bertukan *session-key*.

Tabel 2 Aplikasi dari Public-key Cryptosystems

Algoritma	Enkripsi/ Dekripsi	Digital Signature	Pertukaran kunci
RSA	Ya	Ya	Ya
Elliptic Curve	Ya	Ya	Ya
Diffie-Hellman	Tidak	Tidak	Ya
DSS	Tidak	Ya	Tidak

4. RSA

RSA dikeluarkan oleh tiga orang kriptologi, Ron Rivest, Adi Shamir, dan Len Adleman di MIT. Nama RSA diambil dari nama tiga orang tersebut (Rivet-Shamir-Adleman). RSA dikembangkan pada tahun 1977 dan diterbitkan pada tahun 1978. sejak itu RSA mendapat kejayaan sebagai enkripsi dengan pendekatan *public-key* yang paling diterima dan banyak diimplementasikan.
RSA merupakan *block cipher*, dimana sebuah *plaintext* dan *ciphertext* merupakan integer antara 0 dan n-1. Dimana n biasanya bernilai 1024 bit atau 309 dalam desimal. Kita mulai dengan penjelasan mengenai algoritma RSA.

4.1. Algoritma RSA

Pertama-tama, *plaintext* dienkripsi menjadi blok-blok, dimana setiap blok memiliki bilangan biner kurang dari n untuk n suatu nilai. Dengan begitu jadi ukuran blok harus kurang dari atau sama dengan $\log_2(n)$. Enkripsi dan dekripsi dari suatu blok *plaintext* M dan blok *ciphertext* C:

$$C = M^e \bmod n$$
$$M = Cd \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$$

Baik pengirim maupun penerima harus tau nilai n. Pengirim selalu tahu nilai dari e, dan hanya penerima yang tahu nilai dari d. Sehinga hal ini menjadi enkripsi dengan *public-key* (KU) = {e, n} dan *private-key* (KR) = {d, n}. Untuk mendapatkan hal di atas syarat-syarat yang harus dipenuhi adalah sebagai berikut:

1. Nilai e, d, n dapat dicari, sehingga didapat $M^{ed} = M \bmod n$ untuk setiap $M < n$.
2. Relatif lebih gampang untuk menghitung M^e dan C^d untuk setiap nilai dari $M < n$.
3. Susah dalam praktek untuk mencari d dengan diberikan e dan n.

Pertama-tama fokus pada syarat pertama. Kita perlu mencari hubungan persamaan dengan bentuk

$$M^{ed} = M \bmod n$$

Sebuah persamaan dari teorema Euler (tidak dibahas di sini) cocok dengan permasalahan. Diberikan dua nilai prima, p dan q, dan dua nilai integer, n dan m, sedemikian sehingga n = pq dan 0 < m < n dan nilai integer k sembarang, persamaan menjadi:

$$m^{k\phi(n)+1} = m^{k(p-1)(q-1)+1} \equiv m \bmod n$$

Dimana $\phi(n)$ adalah fungsi totien Euler dimana nilai integer positif kurang dari n dan relatif prima terhadap n. diketahui untuk bilangan prima p,q, $\phi(pq) = (p-1)(q-1)$. Jadi kita dapatkan persamaan yang diinginkan jika

$$ed = k\phi(n) + 1$$

Persamaan ini menjadi

$$ed \equiv 1 \bmod \phi(n)$$
$$d \equiv e^{-1} \bmod \phi(n)$$

Selanjutnya kita dapat memulai masuk ke skema RSA. Hal-hal yang dibutuhkan adalah sebagai berikut:

p, q (bilangan prima)	(*private*, dipilih)
n = pq	(*public*, dihitung)
e, dengan PBB(ϕ(n),e) = 1; 1 < e < ϕ(n)	(*public*, dipilih)
d ≡ e-1 mod ϕ(n)	(*pivate*, dihitung)

Private-key adalah {d,n} dan *public-key* adalah {e.n}. Jika A mengeluarkan *public-key* dan B nerusaha mengirim pesan M ke A. Jadi B menghitung $C = M^e \pmod n$ dan mengirimkan C. Di pihak penerima *ciphertext* ini, A berusaha mendekripsi dengan menghitung $M = C^d \pmod n$.

Ringkasan dari algoritma RSA adalah sebagai berikut:

Key Generator

Pilih p,q	p dan q prima, $p \neq q$
Hitung	$n = p \times q$
Hitung	$\phi(n) = (p-1)(q-1)$
Pilih integer e	PBB (ϕ(n),e) = 1; 1 < e < ϕ(n)
Hitung d	$d \equiv e^{-1} \bmod \phi(n)$
Public-key	KU = {e,n}
Private-key	KR = {d,n}

Enkripsi

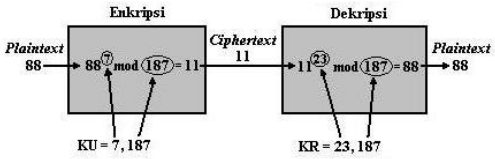
Plaintext	M < n
Ciphertext	$C = M^e \pmod n$

Dekripsi

Ciphertext	C
Plaintext	$M = C^d \pmod n$

Mari kita ambil contoh yang sederhana

1. Pilih dua bilangan prima, p = 17 dan q = 11. (Dalam kenyataan bilangan prima yang dipilih > ±1000000 untuk keamanan).
2. Hitung n = pq =17 × 11 =187.
3. Hitung $\phi(n) = (p-1)(q-1) = 16 \times 10 = 160$.
4. Pilih e sehingga e relatif prima dengan $\phi(n) = 160$dan kurang dari $\phi(n)$; kita pilih e = 7.
5. Tentukan d sehingga de ≡ 1 mod 160 dan d < 160. Nilai yang benar adalah d = 23, karena 23 × 7 = 161 = 10 × 160 + 1;



Gambar 3 Contoh Algoritma RSA

Hasilnya adalah *public-key* KU = {7, 187} dan *private-key* KR = {23, 187}. Kita coba kunci ini denga input M = 88. Untuk enkripsi kita hitung $C = 88^7 \bmod 187$. dari sifat modulus kita dapat:

$88^7 \bmod 187 = [(88^4 \bmod 187) \times (88^2 \bmod 187) \times (88^1 \bmod 187)] \bmod 187$

$88^1 \bmod 187 = 88$
$88^2 \bmod 187 = 7744 \bmod 187 = 77$
$88^4 \bmod 187 = 59.969.536 \bmod 187 = 132$
$88^7 \bmod 187 = (88 \times 77 \times 132) \bmod 187 = 894.432 \bmod 187 = 11$
C = 11

Untuk dekripsi, kita hitung $M = 11^{23} \bmod 187$:

$11^{23} \bmod 187 = [(11^1 \bmod 187) \times (11^2 \bmod 187)) \times (11^4 \bmod 187)) \times (11^8 \bmod 187)) \times (11^8 \bmod 187)] \bmod 187$

$11^1 \bmod 187 = 11$
$11^2 \bmod 187 = 121$
$11^4 \bmod 187 = 14.641 \bmod 187 = 55$
$11^8 \bmod 187 = 21.358.881 \bmod 187 = 33$

$11^{23} \bmod 187 = (11 \times 121 \times 55 \times 33 \times 33) \bmod 187 = 7.720.245 \bmod 187 = 88$

$M = 88$

4.2. Keamanan RSA

Ada tiga cara untuk menyerang algoritma RSA, yaitu:

1. ***Brute Force***: Mencoba semua kemungkinan kunci.
2. **Seangan secara matematis**: ada beberapa pendekatan, semuanya mencari cara dengan memfaktorkan hasil dari dua bilangan prima.
3. ***Timing attack***: hal ini bergantung pada waktu berjalannya algoritma dekripsi.

Untuk bertahan dari serangan *brute force*, gunakan kunci yang berukuran besar. Semakin besar ukuran dari e dan d semakn baik. Tapi karena adanya perhitungan dalam algoritma, maka semakin besar kunci, semakin lambat sistem berkerja. Dengan memperbesar ukuran kunci juga dapat memperbesar keamanan dari serangan matematis. Untuk *timing attack* ada beberapa cara simpel untuk menanganinya:

- **Waktu eksponensiasi tetap**: Menjamin bahwa tiap eksponensiasi memerlukan waktu yang sama. Ini merupakan perbaikan yang simple, tetapi menurunkan performa.
- **Penundaan acak**: Menambahkan penundaan secara acak dalam algoritma eksponensial untuk membingungkan *timing attack*.
- ***Blinding***: Menggandakan ciphertext dengan sebuah angka acak sebelum melakukan eksponensiasi.

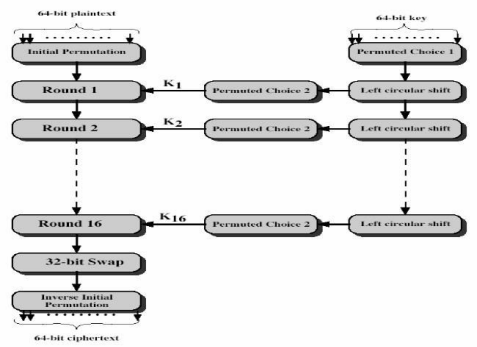
5. Data Encryption Standard (DES)

DES merupakan *block cipher* yang banyak digunakan di dunia. DES menengkrip data dalam blok berukuran 64-bit. Dengan ukuran kunci sebesar 56 bit. Biasanya berukuran 64 bit dengan delapan pariti untuk pengecekan dan diabaikan). Menghasilkan output *cipher text* berukuran 64 bit. DES menggunakan algoritma yang simetrik. Algoritma dan kunci yang sama digunakan dalam mengenkripsi dan mendekripsi.

5.1. Algoritma DES

Gambaran besar algoritma dari DES adalah sebagai berikut:

- Dimulai dengan blok dari input text sebesar 64 bit. Dipermutasi oleh *initial permutation*
- Blok tersebut dipecah menjadi setengah bagian kanan dan setengah bagian kiri, dengan ukuran masing-masing 32 bit.
- Lalu melalui 16 *round* (putaran) operasi yang identik, disebut *Function* f, dimana data digabungkan dengan kunci.
- Setelah putaran ke 16 belahan kiri dan kanan digabungkan kembali dan melakukan permutasi terakhir sebagai penutup algoritma.



Gambar 4 16 Putaran Dalam Algoritma DES

Agar lebih jelas secara lebih detail akan dijelaskan langkah perlangkahnya. Pertama blok dari text akan di permutasi oleh *initial permutation* (IP). P memiliki permutasi inversnya yaitu *invers initial permutation* (IP^{-1}) dimana jika M adalah bilangan biner 64 bit. Maka,

$$X = IP(M) \quad \text{dan}$$
$$M = IP^{-1}(X) = IP^{-1}(IP(M))$$

Kemudian hasil IP dibelah menjadi dua denga ukurran masing-masing 32 bit. Disetiap putaran kedua belahan diproses seperti persamaan berikut:

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

Dimana K_i adalah kunci yang telah di permutasi oleh PC-2.

Selanjutnya belahan kanan mulai memasuki fungsi f, dimana dimulai dengan pelebaran belahan kanan menjadi 48 bit dengan *expansion permutation*. Kemudian digabungkan dengan 48 bit kunci yang telah di-shift dan di permutasi, dengan XOR.

Tabel 3 Table permutasi untuk DES

(a) Initial Permutation (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

(b) Invers Initial Permutation (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

(c) Expansion Permutation (E)

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

(d) Permutation Function (P)

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

Kemudian 48 bit tersebut dikirim menuju *S-boxes* yang terdiri dari 8 box dan menghasilkan 32 bit baru. Dimana tiap box menerima masukan 6 bit dan menghasilkan keluaran 4 bit. Dalam tiap box yang memiliki isi tersendiri (lhat table 2), masukan 6 bit diolah dengan cara melihat 2 bit pertama sebagai nomer baris dan 4 bit selanjutnya sebagai no kolom. Contohnya bila input *S-box* 1 adalah 110110, maka 11 (2 bit awal) menunjukkan baris ke 3 dan 0110 (4 bit selanjutnya) menunjukkan no kolom ke 6. dari tabel didapat angka 1, sehingga *S-box* 1 menghasilkan 0001.

Kemudian 32 bit hasil tersebut dipermutasikan kembali. Ini adalah operasi penutup dari fungsi f. Output dari fungsi f kemudian digabungkan dengan belahan kiri dengan XOR lagi. Hasilnya menjadi belahan kanan yang baru, sedangkan belahan kanan yang lama menjadi belahan kiri yang baru. Operasi ini diulang sebanyak 16 kali, menjadi 16 putaran di DES. 16 putaran tersebut dapat dilihat pada gambar 4.

Tabel 4 *S-Boxes*

S_1	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Tabel 5 Pengaturan Kunci

(a) *Permuted Choice One* (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

(b) *Permuted Choice Two* (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

(c) Penjadwalan *shift* kiri

Round number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Dalam tiap putaran keadaan kunci selalu berubah-ubah. Sebelum memasuki putaran pertama kunci di permutasi dengan PC-1. Kunci pada bit ke 8 dan kelipatannya diabaikan dalam permutasian dengan PC-1, sehingga kunci menjadi berukuran 56 bit. Ketika masuk tiap putaran kunci dibelah menjadi dua, masing-masing 28 bit. Kemudian kedua belahan di-*shift* ke kiri sebanyak 1 bit atau 2 bit sesuai yang telah ditentukan oleh tabel 3.c secara terpisah. Setelah itu keduanya digabungkan dalam PC-2 untuk di gabungkan dalam fungsi f. Kedua belahan tersebut (sebelum di permutasi dengan PC-2) menjadi input untuk putaran selanjutnya. Begitu seetrusnya hingga putaran ke-16.

Setelah keluaran dari putaran ke-16 dilakukan *swap*. Lalu dipermutasikan dengan IP^{-1} yang merupakan invers dari IP, sehingga dihasilkan ciphertext berukuran 64-bit

5.2 Three-key Triple DES

Sebagai alternatf banyak para peneliti yang menawarkan metode enkripsi lipat tiga dengan menggunakan tiga kunci. *Three-key* 3DES memiliki panjang kunci efektif 168 bit. Fungsinya adalah melakukan tahapan enkirpsi-dekripsi-enkripsi (EDE)

$$C = E_{k3} [D_{k2} [E_{k1} [P]]]$$

Dengan 3DES ini serangan *brute force* untuk mencari kunci meningkat 10^{52} dibandingkan dengan DES. Sehingga 3DES menjadi populer untuk digunakan. Seperti PGP dan S/MIME.

6. Pretty Good Privacy

Pretty good privacy (PGP) adalah fenomena yang luar biasa. Program ini di temukan oleh Philip Zimmermann. PGP menyediakan fasilitas pengecekan dan autentifikasi yang dapat digunakan dalam surat elektronik dan aplikasi penyimpanan file. Hal-hal yang dilakukan oleh Zimmermann adalah sebagai berikut:

1. Memilih algoritma-algoritma encripsi terbaik
2. Menggabungkan algoritma-algoritma tersebut menjadi suatu aplikasi yang independen dari sistem operasi dan prosesor.
3. Membuat *package* dan dokumentasinya, termasuk *source code*, tersedia via internet.
4. Membuat perjanjian dengam perussahaan (Viacrypt) untuk mendukung versi comersial yang murah dan *fully compatible*.

PGP sekarang telah banyak digunakan dan berkembang pesat karena beberapa alasan, yaitu sebagai berikut:

- PGP tersedia secara gratis di seluruhdunia dalam berbagai versi *platform*, termasuk Windows, UNIX, Macintosh, dan masih banyak lagi. Sebagai tambahan, versi komersial dapat memuaskan user yang menginginkan produng yang didukung oleh sebuah vendor.
- PGP menggunakan algoritma yang telah bertahan dari berbagai review dan dianggap sangat aman. Secara spesifik, paket yang memakai RSA, DSS, dam Diffie-Hellman untuk enkripsi dengan *public-key*; CAST-128. IDEA dan 3DES untuk enkripsi simetrik; dan SHA-1 untuk mengkode hashnya.
- Penggunaan PGP sangat luas. Dari perusahaan yagn ingin memilih dan membuat skema standar untuk mengenkripsi file dan surat sampai individu yang ingin berkomunikasi secara aman dengan orang lain di di dunia melalui internet ataupun jaringan lainnya
- PGP tidak dikembangkan maupun dikendalikan oleh pemerintahan atau oraganisasi tertentu. Hal ini menjadi sangat menarik bagi pihak-pihak tertentu.
- PGP sekarang berada di jalur standar internet.

Digital signature dalam PGP menggunakan DSS atau RSA untuk mengenkripsi pesan dengan *private key* pengirim dan *hash code* pesan dibuat dengan SHA-1. Untuk enkripsi pesan menggunakann CAST-123 atau IDEA atau *Three-key Triple DES* ataupun dengan menggunakan RSA juga. Pesan dapat terlebih dahulu dikompresi untuk penimpann atau peniriman dengan menggunakan ZIP. Hasil enkripsi ditampilkan dalam string ASCII dengan menggunakan pengkonversian radix 64. Untuk mengakomodasikan batas ukuran maksimum pesan, PGP melakukan segmentasi dan penyusunan ulang.

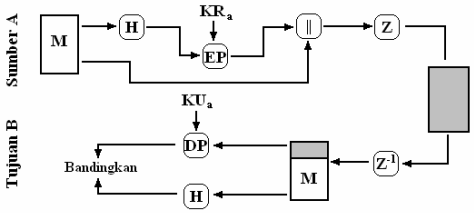
Seperti yang disebutkan di atas. PGP menggunakan sistem managemen kunci. Dalam dokumentasi, PGP sering menggunakan kunci rahasia, yaitu pasangan kunci *public-key* dan *private-key* dalam enkripsinya. PGP dalam operasinya menyangkut lima proses:

autentifikasi, pengerahasiaan, kompresi, *e-mail compatibility*, dan segmentasi.

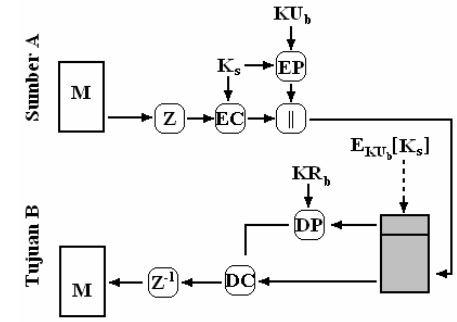
6.1 Autentifikasi

Gambar x.x mengilustrasikan proses autentifikasi dan penyelubungan. proses autentifikasi dapat dilihat pada gambar x.xa. Langkah-langkah dalam autentifikasi tersebuat adalah sebagai berikut:

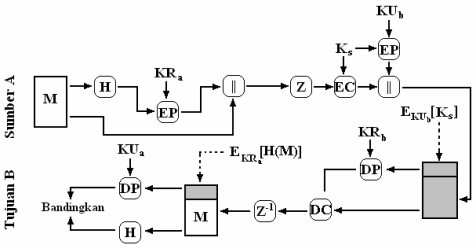
1. Pengirim membuat pesan.
2. SHA-1 digunakan dalam membuat 160-bit *hash code* pesan tersebut.
3. *Hash code* tersebut dienkripsi dengam menggunakan RSA dengan *private-key* sang pengirim yang kemudian dikirim sebagai pesan tersebut.
4. Penerima menggunakan RSA dengan *public-key* dari pengirim untuk mendekripsi dan mengembalikan *hash code*-nya.
5. Penerima membuat ulang *hash code* dari pesan itu dan membandingkannya dengan *hash code* yang telah didekripsi tersebut. jika keduanya cocok maka pesan tersebut dipercaya benar.



(a) Autentifikasi



(b)Pengerahasiaan



(c) pengerahasiaan dan autentifikasi

Gambar 5 Fungsi kriptografi PGP

Keterangan:

K_s = *session-key* yang digunakan dalam enkripsi simetrik
KR_a = *private-key* user A, digunakan dalam enkripsi dengan *plublic-key*
KU_a = *public-key* user A, digunakan dalam enkripsi dengan *plublic-key*
EP = enkripsi *public-key*
DP = dekripsi *public-key*
EC = enkripsi simetrik
DC = dekripsi simetrik
H = fungsi hash
|| = konkatenasi
Z = kompresi dengan algoritma zip
R64 = konversi ke radix 64 dalam format ASCII

Kombinasi dari SHA-1 dengan RSA menyediakan skema digital signature yang efektif. Karena kekuatan dari RSA, penerima yakin hanya prosesor dari yang memiliki *private-key* yang cocok, yang dapat membuat *signature* tersebut. Karena kekuatan dari SHA-1, penerima pesan yakin tidak ada orang lain yang dapat membuat pesan baru yang cocok dengan kode hash tersebut dan *signature* dari pesan asal.

Sebagai alternatif *signature* dapat di buat denga DSS/SHA-1.

Walaupun biasanya *signature* biasa ditemukan menempel pada pesan ataupun file. Tapi tidak selalu seperti itu. Dapat juga *signature* tidak ditempelkan pada pesan atau file yang bersangkutan. *Signature* yang terpisah bisa disimpan atau dikirim terpisah dari pesannya.

6.2 Pengerahasiaan

Untuk merahasiakan isi pesan, PGP mengenkripsi isi pesan yang akan dikirimkan. Algoritma yang digunakan adalah CAST-128, IDEA atau 3DES. Mode 64 bit *cipher feedback* (CFB) digunakan.

Dalam PGP sebuah kunci simetrik hanya digunakan sekali. Lalu sebuah kunci baru dibuat dari sebuah nilai acak dari 128 bit angka. Ini yang disebut sebagai *session-key*, ini merupakan kunci sekali pakai. Karena hanya sekali pakai maka kunci terikat pada pesan dan ikut dikirimkan bersama pesan tersebut. Untuk keamanan kunci tersebut dienkripsi dengan *public-key* sang penerima pesan. Hal ini di gambarkan pada gambar 5.a dan lebih jelasnya adalah sebagai berikut:

1. Pengirim membut pesan dan kunci dari 128 bit angka secara acak dan hanya digunakan untuk pesan ini saja.

2. Pesan dienkripsi dengan CAST-128, IDEA atau 3DES dengan kunci tersebut.
3. Lalu kunci tersebut dienkripsi denga RSA, dengan menggunakan *public-key* sang penerima pesan dan ikut dikirimkan bersama pesan.
4. Penerima menggunakan RSA dan – *rivate-key* untuk mendekripsi *session-key*.
5. Kemudian *session-key* digunakan untuk mendekripsi pesan tersebut.

Sebagai alternatif RSA digunakan untuk menenkripsi pesan tersebut. Lalu *session-key* untuk mendeskripsinya di enkripsi dengan *public-key* dari sang penerima pesan.

6.3. Kompresi

Secara *default* PGP mengkompress pesan setelah memberikan *signature*, tapi sebelum melakukan enkripsi. Hal ini untuk menghemat ukuran file yang dikirim maupun yang disimpan. Algoritma kompresi yang digunakan adalah ZIP.

6.4. E-mail compatibility

Ketika PGP digunakan, palingtidak sebagian dari blok yang akan dikirimkan terenkripsi (dengan *private-key* sang pengirim). Jika fasilitas pengerahasiaan digunakan, pesan dan *signature* (jika ada) akan terenkripsi (dengan kunci simetrik sekali pakai). Sehingga, sebagian atau seluruh hasil blok terdiri dari kumpulan bilangan oktet 8-bit yang acak. Walaupun begitu banyak sistem surat elektronik yang hanya memperbolehkan penggunaan blok yang terdiri dari text ASCII. Untuk mengatasi masalah ini, PGP menyediakan fasilitas untuk merubahnya ke karakter ASCII yang dapat di print.

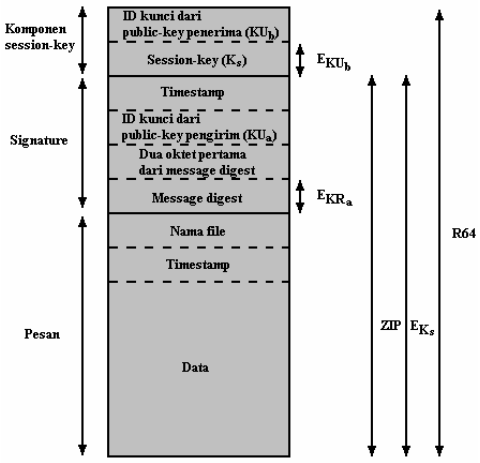
Skema yang digunakan untuk masalah ini adalah radix-64. Setiap kelompok yang terdiri dari tiga oktet data biner, dipetakan ke empat karakter ASCII.

6.5. Segmentasi

Fasilitas e-mail sering dibatasi dengan panjang pesan maksimum. Sebagai contoh banyak fasilitas yang dapat diakses di internet meminta panjang maksimum 50.000 oktet. Lebih dari itu harus dipecah menjadi segmen-segmen yang lebih kecil, yang setiap segmen akan dikirim secara terpisah.

Untuk mengatasi masalah ini, PGP secara automatis membagi pesan yang terlalu besar menjadi segmen-segmen yang cukup kecil untuk dikirim dengan e-mail. Proses segmentasi dilekukan setelah melakukan semua proses yang lain, termasuk pengonversian dengan radix-64.

6.6 gambaran umum PGP



Gambar 6 Format PGP (dari A ke B)

Keterangan:

- ***TimeStamp***: Waktu kapan *signature* dibuat.
- ***Message Digest***: 160-bit SHA-1 *digest*, terenkripsi dengan *private-key signature* pengirim.
- **Dua oktet pertama dari *message digest***: Membuat penerima dapat menentukan apakah *public-key* yang benar telah digunakan untuk mendekripsi pesan untuk autentifikasi, dengan membandingkan kopian plaintext ini dari dua oktet pertama dua oktet pertama dari *digest* yang terenkripsi.
- **ID kunci dari *public-key* pengirim**: pengidentifikasi *public-key* yang seharusnya digunakan untuk mendekripsi *message digest*.

7. Kesimpulan

Kesimpulan yang dapat diambil dari makalah ini adalah:

1. PGP merupakan pelayanan pengamanan untuk surat dan dokumen elektronik yang populer di gunakan.
2. PGP melakukan berbagai pelayanan di dalamnya seperti *digital signature* dengan RSA, pengenkripsian dengan CAST atau IDEA atau *three-key* 3DES atau RSA, dan sebagainya.
3. Keamanan PGP didukung oleh beberapa algoritma enkripsi, seperti

RSA dan 3DES. Sehingga keamanan dari PGP terjamin.

4. RSA merupakan algoritma yang menggunakan menejemen kunci dalam melakukan enkripsinya. Dimana RSA membut dua buah kunci yang dapat digunakan untuk enkripsi dan satunya lagi untuk dekripsi.
5. RSA memiliki tingkat keamanan yang tinggi. Dimana untuk membuka paksa sebuah kunci diperlukan waktu bertahun-tahun dan semakin besar kunci yang digunakan semakin aman berkali-kali lipat.
6. DES adalah algoritma pengenkripsi yang sempat digemari, tapi karena keamanannya kurang, mulai ditinggalkan. Kemudian dibuat pengembangan dari DES yaitu 3DES yang lebih aman.

DAFTAR PUSTAKA

[1] Wikipedia. (2007). http://www.wikipedia.org. Tanggal akses: 2 Januari 2006 pukul 09:30.

[2] Schneier, Bruce. (1996). Applied Cryptography *second edition*. New York John Wiley & Sons

[3] Stakkings, William. (2003). Cryptography and Network Security. Pearson Education International