APLIKASI KRIPTOGRAFI DALAM PGP UNTUK KERAHASIAAN EMAIL

Gia Pusfita – NIM : 13505082

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if15082@students.if.itb.ac.id

Abstrak

Makalah ini membahas tentang aplikasi kriptografi untuk kerahasiaan email melaui program PGP (*Pretty Good Privacy*). PGP menggunakan sistem pasangan *kunci privat* dan *kunci publik*. Kunci privat merupakan kunci yang dipegang oleh penggunanya dan tidak boleh diketahui orang lain, sedangkan kunci publik ditujukan untuk publik terutama orang yang akan menerima pesan enkripsi dari seseorang. Enkripsi yang digunakan dalam PGP menggunakan *algoritma* tertentu. PGP yang akan dibahas pada makalah ini adalah PGP yang menggunakan aplikasi kriptografi berupa IDEA (*International Data Encryption Algorithm*) sebagai algoritma simetri, RSA sebagai algoritma kunci publik dan algoritma MD5 untuk menghasilkan *message digest* pada dokumen elektronis (file).

RSA asimetrik *cipher* memungkinkan seseorang untuk menetapkan diri mereka sebagai satu-satunya penulis yang sah atau untuk memastikan hanya orang yang dituju yang dapat membaca pesan. Meskipun RSA dapat mengenkripsi dan mendeskripsi segala informasi yang relevan, namun ketika kunci yang panjang digunakan maka RSA akan bekerja agak lamban, sehingga untuk mendeskripsikan dan mengenkripsi teks yang panjang digunakan IDEA simetrik *cipher* yang memiliki performa sedikit lebih baik dan RSA masih tetap digunakan untuk menandai dokumen. Sehingga terdapat kombinasi antara metode yang menggunakan sepasang kunci yang ditujukan terhadap orang tunggal (RSA) dan metode yang menggunakan *session key* untuk enkripsi dan deskripsi (IDEA). Metode ketiga yang digunakan oleh PGP adalah *hashfunction* MD5 yang bertujuan untuk mengonversi teks yang sangat panjang kedalam nomor tunggal dengan 39 digit desimal yang digunakan sebagai *fingerprint* untuk merubah teks panjang dalam tanda tangan digital.

Adapun yang dimaksud kerahasiaan email yang dibahas dalam makalah ini adalah pesan yang dienkripsi dan penggunaan tanda-tangan digital. Pengirim menggunakan enkripsi PGP untuk menghasilkan tanda-tangan digital . Salah satu algoritma yang digunakan untuk digital signature algoritma MD5 yang dapat pula dikombinasikan dengan algoritma RSA untuk enkripsi *message digest* dokumen tersebut.

.

Kata kunci: enkripsi, dekripsi. *Cipher, hash, message digest, session key, message digest*

1. Pendahuluan

Sama halnya dengan dokumen konvensional di atas kertas, dokumen elektronis pun membutuhkan berbagai aspek keamanan. Salah satu aspek keamanan yang perlu dijamin dalam suatu dokumen, baik konvensional maupun digital, adalah orisinalitas. Dokumen digital sebagaimana halnya dokumen konvensional harus dapat dijamin keasliannya, yaitu sama bentuk dan isinya dengan dokumen yang dimaksud oleh pembuatnya. Aspek lain yang juga penting, adalah jaminan atas identitas pembuat dokumen. Suatu dokumen harus dapat memenuhi sifat *non-repudiation* yaitu pembuatnya tidak dapat menyangkal bahwa ia tidak pernah membuat dokumen tersebut

Dengan merebaknya email palsu yang mencatut identitas seseorang, baik yang dihasilkan oleh program seperti halnya *worm*, atau memang dilakukan oleh pihak tertentu, penggunaan teknik autentifikasi pesan menjadi lebih diperlukan. Seperti halnya surat yang pengirimannya tinggal dimasukkan ke dalam kotak pos yang banyak dijumpai di pinggir jalan, *server email* juga menerima pesan yang akan dikirimkan serupa itu. Autentifikasi umumnya hanya dilakukan terhadap alamat IP komputer pengirim, dan sepanjang alamat tadi dianggap valid, maka

siapapun dapat menulis email dari komputer tersebut.

Adalah tanggung jawab pengirim surat untuk menandai surat tersebut sehingga dapat dipercaya (*trusted*) bahwa memang pesan yang ditulis berasal darinya. Sedangkan di sisi penerima pesan, harus terdapat sebuah cara sehingga dia dapat mengetahui identitas pengirim pesan dan cukup yakin bahwa pesan tersebut memang ditulis oleh yang bersangkutan.

Cara yang digunakan di atas kertas: dituliskan tanda tangan atau stempel yang menunjukkan validitas pengirim pesan. Demikianlah tanda tangan digital (*digital signature*) juga dimaksudkan seperti itu. Tanda tersebut harus unik untuk membedakan satu pengirim dengan lainnya, sulit ditiru pihak lain, dan dapat menjaga integritas pesan yang ditandai. Tujuannya adalah menghindari pencatutan identitas dan pengubahan pesan oleh pihak ketiga di tengah jalan (*man in the middle attack*) pada saat pesan tersebut ditransmisikan. Pengubahan pesan digital lebih sulit terlihat dibanding pesan di atas surat.

Untuk keperluan yang penting ini, tersedia alat bantu yang dapat diperoleh secara cuma-cuma, yakni Pretty Good Privacy, *PGP*, dan *GNU PGP*, atau GPG. Tentu saja masih terdapat penyedia layanan tanda tangan digital lainnya, namun PGP dan GPG lebih dikenal luas dan GPG adalah produk Open Source. Untuk menggunakan PGP di luar Amerika Serikat, gunakan versi internasional, sedangkan GPG sendiri karena dikembangkan di luar wilayah hukum Amerika Serikat, maka bebas digunakan oleh siapapun. Restriksi ini berkaitan dengan aturan ekspor produk enkripsi yang berkait dengan pemakaian kunci sandi untuk pemakaian tanda tangan digital ini.

Persiapan yang dilakukan oleh pihak yang hendak berkorespondensi tidak terlalu sulit. Kedua belah pihak menyediakan sepasang kunci, yakni kunci privat dan kunci publik (*private key, public key*). Kunci privat ini dipegang hanya oleh pemiliknya; sedangkan kunci publik diekspos kepada khalayak dan diambil oleh mereka yang akan menerima pesan. Penerima pesan tersebut menambahkan kunci publik ke dalam daftar yang dikelola oleh aplikasi, baik PGP atau GPG. Agar lebih yakin lagi bahwa kunci publik tersebut berasal dari pihak yang dimaksud, tersedia tambahan kode jejak, yakni fingerprint. Dalam kondisi ekstrim yang memerlukan validitas tingkat tinggi, fingerprint ini dipertukarkan oleh kedua pihak lewat pertemuan fisik atau media lain yang lebih dapat dipercaya.

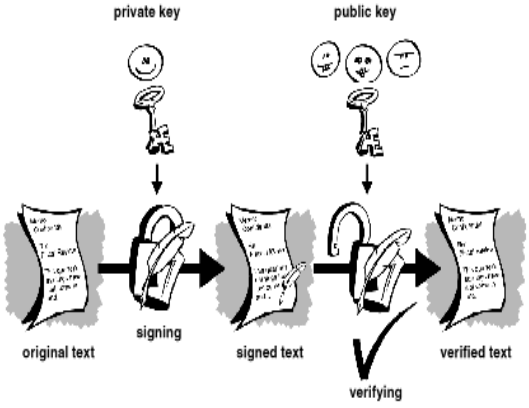
Karena tujuan pemakaian tanda tangan digital berbeda dengan enkripsi yang bersifat menyembunyikan, maka pesan tersebut tetap dapat terbaca oleh semua orang, namun di bagian bawahnya terdapat "tanda tangan" yang dapat digunakan untuk memeriksa integritas pesan dan validitas pengirimnya.

Digital Signature

Digital signature adalah teknik pemberian suatu nilai kriptografis pada suatu pesan atau arsip yang bergantung baik kepada pengirim pesan maupun pesan itu sendiri. Fungsi utama *digital signature* adalah sebagai autentikasi yang menyangkut *message integrity, user authentication,* dan *non-repudiation*. *Digital signature* dilakukan dengan dua cara, yaitu :

1. Enkripsi pesan melalui algoritma kunci simetri maupun dengan algoritma kunci publik. Pesan yang sudah terenkripsi berarti pesan tersebut sudah ditandatangani.

2. Melakukan fungsi *hash* terhadap pesan. Hasil fungsi *hash* merupakan *digital signature* atas pesan dan di-*append* di dalam pesan. Dua algoritma yang banyak digunakan dalam *digital signature* adalah RSA dan El Gamal. Pada tugas ini, *digital signature* diimplementasi dengan algoritma RSA.



Gambar 1 simple digital signatures

2. PGP

Pretty Good Privacy (PGP) dikembangkan oleh Philip Zimmerman pada akhir tahun 1980. Versi pertama PGP dirilis pada tahun 1991. PGP yang berikutnya yaitu versi 2.6.x dan 5.x (atau 3.0) sudah di implementasikan oleh seluruh sukaralawan yang bekerjasama dibawah bimbingan desain Zimmerman. PGP digunakan untuk melindungi surat elektronik (*e-mail*) dengan memberi perlindungan kerahasian (enkripsi) dan otentikasi (tanda-tangan digital).

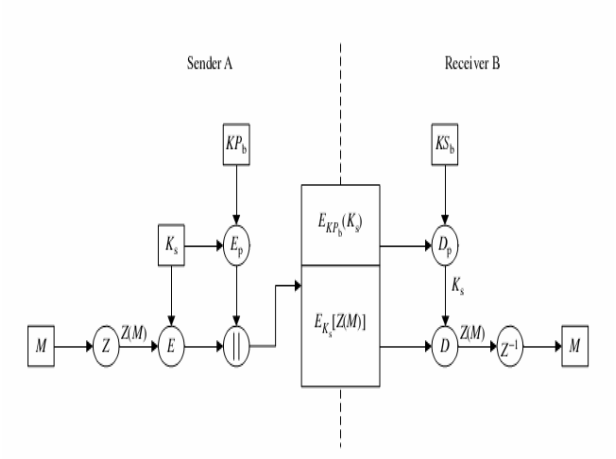
PGP dipergunakan secara meluas baik individual atau komersial yang dijalankan dalam platform berbeda-beda. PGP menggunakan kombinasi antara kriptografi simetri dan kriptografi kunci publik. Oleh karena itu, PGP mempunyai dua tingkatan kunci, kunci rahasia (simetri) yang disebut juga *session key* untuk enkripsi data, dan pasangan kunci privat - kunci publik untuk pemberian tanda-tangan dan melindungi kunci simetri.

2.1 Kerahasiaan via enkripsi

PGP menjaga kerahasiaan data dengan mengenkripsi pesan yang akan ditransmisikan atau file data yang akan dikirim menggunakan enkripsi algoritma konvensional (conventional encryption algorithm) seperti IDEA, 3-DES atau CAST-128. Di dalam PGP, setiap kunci simetrik dikenal sebagai suatu kunci sesi (*session key*), yang digunakan hanya sekali ketika.

Kunci sesi ini bekerja dengan sangat aman, dan merupakan algoritma enkripsi konvensional yang cepat untuk mengenkripsi *plaintext*. Sebuah kunci sesi baru dihasilkan sebagai 128-bit nomor acak untuk setiap pesan. Karena dipergunakan hanya sekali saja, maka kunci sesi disimpan didalam pesan dan ditransmisikan bersama *ciphertext* kepada penerima. Untuk melindungi kunci, maka kunci dienkripsi oleh kunci publik penerima. Gambar 2 mengilustrasikan urutan proses yang akan diuraikan dibawah :

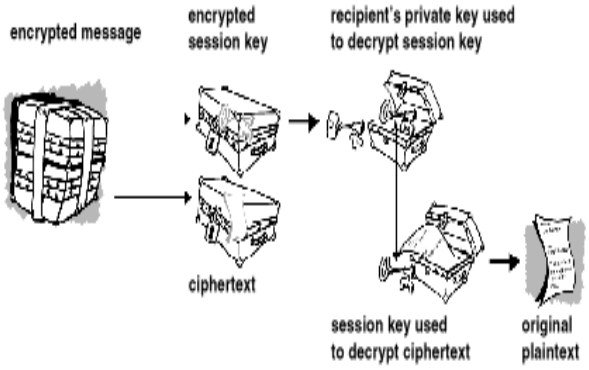
- Pengirim membuat pesan
- PGP pengirim menghasilkan 128-bit nomor acak yang digunakan sebagai sebuah kunci sesi hanya untuk pesan tersebut
- Kunci sesi dienkripsi dengan RSA, menggunakan kunci publik penerima.
- PGP pengirim mengenkripsi pesan, menggunakan IDEA, dengan kunci sesi. Pesan juga di kompresi.
- PGP penerima menggunakan RSA dengan kunci privat untuk mendeskripsi dan menutup lagi kunci sesi.
- PGP penerima mendeskripsikan pesan menggunakan kunci sesi. Jika pesan telah dikompres, maka pesan akan didekompres.



Gambar 2 skema komputasi kerahasiaan PGP dengan algoritma kompresi/ dekompresi

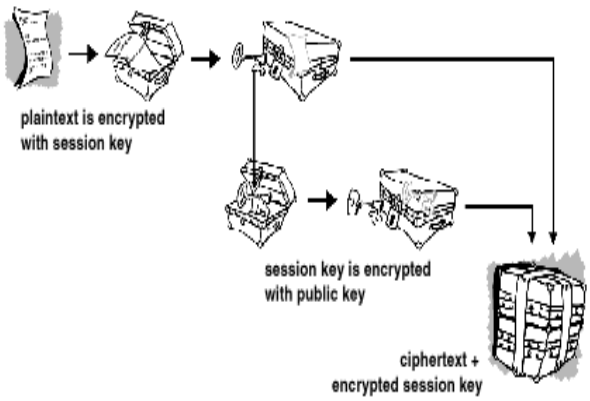
K_s = session key	H = hash function
KP_a = public key of user A	KP_b = public key of user B
KS_a = private key of user A	KS_b = private key of user B
E = conventional encryption	D = conventional decryption
E_p = public-key encryption	D_p = public-key decryption
Z = compression using zip algorithm	Z^{-1} = decompression
|| = concatenation	

Berikut adalah ilustrasi yang lebih sederhana tentang bagaimana enkripsi PGP bekerja.



Gambar 3 Ilustrasi enkripsi dalam PGP

Proses Dekripsi bekerja secara terballik. Salinan dari PGP penerima menggunakan kunci privat untuk memulihkan kunci sesi sementara, yang kemudian akan digunakan oleh PGP untuk mendeskripsikan enkripsi *chipertext* secara konvensional.



Gambar 4 Ilustrasi deskripsi dalam PGP

Kombinasi dari dua metode enkripsi yaitu kombinasi antara kemudahan dari enkripsi kunci publik dengan kecepatan dari enkripsi konvensional menghasilkan perkembangan baru dalam keamanan tanpa ada yang dikorbankan. Enkripsi konvensional mencapai 1000 kali lebih cepat dari enkripsi kunci publik. Enkripsi kunci publik menyediakan solusi untuk isu transmisi data dan distribusi.

3. Metode Kriptografi yang digunakan PGP

Pretty Good Privacy (PGP) menyediakan kerahasiaan dan otentifikasi yang dapat digunakan untuk surat elektronik. PGP menggunakan IDEA untuk konvensional enkripsi blok, bersama dengan RSA untuk enkripsi kunci publik dan MD5 untuk kode *hash*.

3.1 IDEA *cipher*

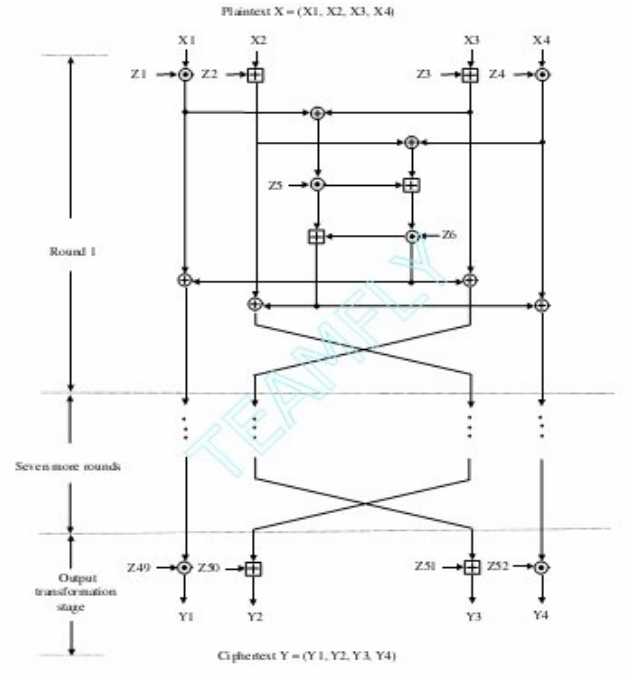
ciphertext tergantung pada *plaintext* dan kunci, yang sebagian besarnya terlibat dalam proses rumit. IDEA mencapai tujuan ini dengan mengombinasikan tiga operasi yang berbeda. Setiap operasi ditampilkan dengan masukan 16-bit untuk menghasilkan keluaran 16-bit tunggal. IDEA memiliki struktur yang dapat digunakan untuk enkripsi dan deskripsi.

3.1.1 Proses enkripsi IDEA

Skema keseluruhan enkripsi IDEA diilustrasikan pada gambar 5. Sama seperti keseluruhan blok *cipher*, di sana terdapat dua masukan untuk fungsi enkripsi, misalnya blok *plaintext* dan kunci enkripsi. Di dalam IDEA, plaintext memiliki panjang 64 bit dan panjang ukuran kunci 128 bit. Algoritma IDEA didasarkan pada kombinasi tiga operasi yang berbeda. Operasi ini adalah :

$\oplus$ Bit-by-bit XOR of 16-bit sub-blocks
$\boxplus$ Addition of 16-bit integers modulo 2^{16}
$\odot$ Multiplication of 16-bit integers modulo $2^{16} + 1$

Di dalam gambar 5, algoritma IDEA memiliki delapan putaran yang diikuti oleh transformasi keluaran akhir. Blok masukan 64 bit terbagi menjadi empat sub-blok 16 bit, yang dilabelkan dengan X_1, X_2, X_3, X_4. Keempat sub-blok ini menjadi input untuk putaran pertama algoritma IDEA. Subkunci generator menghasilkan total 52 blok subkunci yang semuanya dihasilkan dari 128-bit kunci enkripsi asli. Setiap blok subkunci terdiri atas 16 bit.



Gambar 5 skema enkripsi IDEA

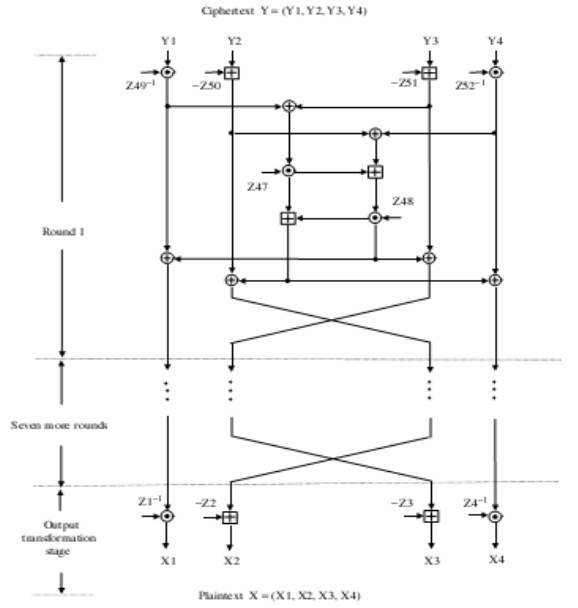
Putaran pertama menghasilkan 16 bit subkunci (Z_1, Z_2,...,Z_6), sedangkan keluaran transformasi ahir menggunakan empat subkunci 16 bit (Z_{49},..,Z_{52}). Tahap akhir transformasi juga

menghasilkan empat blok 16 bit, yang disambungkan dengan *ciphertext* 64 bit. Dalam setiap putaran , empat sub-blok 16 bit mengalami operasi XOR, penjumlahan dan perkalian dengan lainnya dan dengan enam sub-blok 16 bit. Antara setiap putaran, sub-blok kedua dan ketiga mengalami perubahan. Penukaran ini meningkatan pencampuran bits yang sedang diproses dan membuat algoritma IDEA lebih resisten terhadap kriptoanalis berbeda. Pada setiap putaran, operasi beruntun yang terjadi adalah :

(1) $X_1 \odot Z_1$
(2) $X_2 \boxplus Z_2$
(3) $X_3 \boxplus Z_3$
(4) $X_4 \odot Z_4$
(5) $(X_1 \odot Z_1) \oplus (X_3 \boxplus Z_3) = (1) \oplus (3)$
(6) $(X_2 \boxplus Z_2) \oplus (X_4 \odot Z_4) = (2) \oplus (4)$
(7) $(X_1 \odot Z_1) \oplus (X_3 \boxplus Z_3) \odot Z_5 = ((1) \oplus (3)) \odot Z_5$
(8) $(((X_2 \boxplus Z_2) \oplus (X_4 \odot Z_4)) \boxplus (((X_1 \odot Z_1) \oplus (X_3 \boxplus Z_3)) \odot Z_5)) = ((2) \oplus (4)) \boxplus (((1) \oplus (3)) \odot Z_5)$
(9) $(8) \odot Z_6$
(10) $(7) \boxplus (9) = (((1) \oplus (3)) \odot Z_5) \boxplus ((8) \odot Z_6)$
(11) $(X_1 \odot Z_1) \oplus ((8) \odot Z_6) = (1) \oplus (9)$
(12) $(X_3 \boxplus Z_3) \oplus (9) = (3) \oplus (9)$
(13) $(X_2 \boxplus Z_2) \oplus (10) = (2) \oplus (10)$
(14) $(X_4 \odot Z_4) \oplus (10) = (4) \oplus (10)$

3.1.2 Proses deskripsi IDEA

Proses deskripsi IDEA sama dengan proses enkripsi, kecuali bahwa kunci sub-blok dibalik dan penggunaan pemilihan sub-kunci yang berbeda. Kunci deskripsi pertama dan keempat sama dengan hasil perkalian invers modulo $(2^{16} + 1)$. untuk putaran kedua dan kedelapan, subkunci deskripsi kedua dan ketiga sama dengan penjumlahan inverse modulo 2^{16}. untuk putaran satu dan sembilan, subkunci deskripsi kedua dan ketiga sama dengan penjumlahan inverse modulo 2^{16} dari hubungan antara subkunci kedua dan ketiga.



Gambar 6 skema deskripsi IDEA

3.2 Kriptosistem Kunci Publik RSA

Algortima Enkripsi RSA merupakan algoritma kuci-publik yang paling terkenal dan paling banyak aplikasinya. RSA ditemukan oleh tiga peneliti dari *MIT,* yaitu Ron Rivest, Adi Shamir, dan Len Adleman, pada tahun 1976.

Kriptosistem RSA menyerupai sistem pertukaran kunci D-H dalam penggunaan eksponensial terhadap aritmetika modula untuk enkripsi dan deskripsi, namun dalam mengooperasikan aritmetika RSA menggunakan nomor komposit. Keamanan RSA tergantung pada kerumitan dalam memfaktorkan angka besar.

3.2.1 Algoritma Enkripsi RSA

misalkan e adalah kunci public dan n modulus, kunci privat d untuk deskripsi dapat diketemukan dengan memfaktorkan n. Pilih 2 angka prima yang besar, p dan q, dan hitung modulus n yang merupakan hasil kali dua angka prima :

$$n = pq$$

pilih kunci enkripsi e sedemikian sehingga e dan Ø(n) koprima. Misalnya gcd(e,Ø(n)) =1, dimana Ø(n) = (p-1)(q-1) disebut sebagai fungsi totient Euler.

Dengan menggunakan algoritma euclidean, kunci privat d untuk deskripsi dapat ditemukan dengan mengalikan inverse e :

$$d = e^{-1} \pmod{Ø(n)}$$
atau $ed = 1 \pmod{Ø(n)}$

kunci deskripsi d dan modulus n adalah relatif prima. Jumlah e dan n disebut sebagai kunci publik, dan jumlah d disebut sebagai kunci privat.

Untuk mengenkripsi pesan m, hubungan *chipertext* c dengan blok pesan dapat ditemukan dengan formula enkripsi :

$$c = m^e \pmod{n}$$

utnuk mendeskripsikan *chipertext* c, c dipangkatkan oleh d untuk merecover pesan m :

$$m = c^d \pmod{n}$$

ini membuktikan bahwa

$$c^d = (m^e)^d = m^{ed} = m \pmod{n}$$

sepadan pada hasil bahwa $ed = 1 \pmod{Ø(n)}$
karena formula Euler adalah $m^{Ø(n)} = 1$, pesan m adalah relatif prima terhadap n sehingga gcd(m,n) = 1. karena $m^{\lambda Ø(n)} = 1 \pmod{n}$ untuk beberapa integer λ, hal ini dapat dituliskan menjadi $m^{\lambda Ø(n)+1} = m \pmod{n}$, sebab $m^{\lambda Ø(n)+1} = mm^{\lambda Ø(n)} = m \pmod{n}$.
Gambar 5 dan tabel 1 mengilustrasikan Algoritma RSA untuk deskripsi dan enkripsi.

kunci publik e:

n (hasil kali p dan q(bilangan rahasia))
e (kunci enkripsi, relatif prima terhadap Ø(n) = (p-1)(q-1))

kunci privat d:

d (kunci deskripsi, $d = e^{-1} \pmod{Ø(n)}$)
$ed = 1 \pmod{Ø(n)}$)

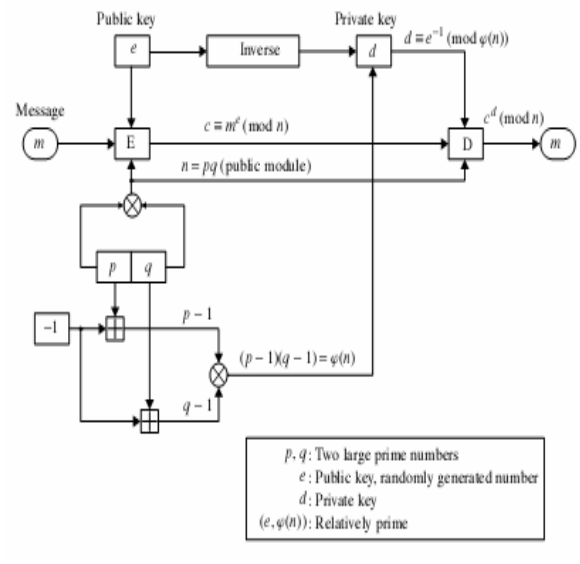
enkripsi:

$c = m^e \pmod{n}$, m adalah *plaintext*

deskripsi:

$m = c^d \pmod{n}$, c adalah *chipertext*

Tabel 1 Algoritma enkripsi RSA



Gambar 7 Kunci publik RSA kriptosistem untuk enkripsi/dekripsi

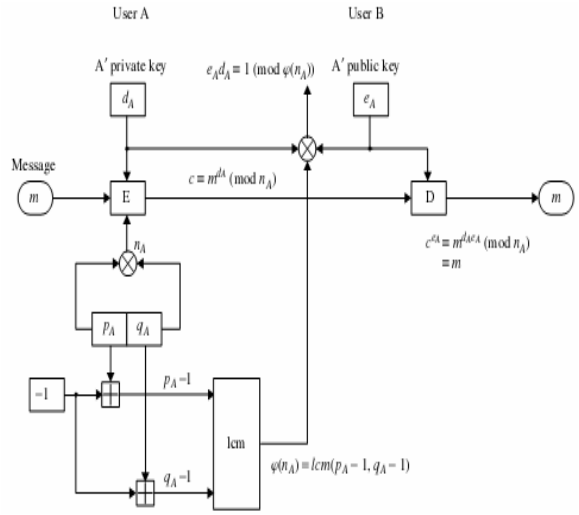
3.2.2 Skema Signature RSA

Kunci publik kriptosistem RSA dapat digunakan untuk mengenkripsi dan signature. Setiap user mempunyai tiga buah integer yaitu e,d dan n, n = pq, dengan pq adlah angka yang besar. Untuk pasangan kunci (e,d), persamaan $ed = 1 \pmod{Ø(n)}$ harus terpenuhi. Jika pengirim A ingin mengirimkan pesan yang ditandai (c) yang dihubungkan dengan pesan m kepada penerima B, maka A menandainya dengan menggunakan kunci privat milik A, yang dapat dicari dari formula $c = m^{dA} \pmod{n_A}$. Pertama-tama A menghitung

$$Ø(n_A) = \text{lcm}(p_A - 1, q_A - 1)$$

Lcm adalah singkatan dari *least common multiple.* Pengirim A memilih sendiri pasangan kunci (e_A, d_A) seperti :

$$eA.dA = 1 \pmod{Ø(n_A)}$$



Gambar 8 Skema RSA signature

Di dalam hardware, RSA memiliki kecepatan 1000 kali lebih lambat daripada DES. RSA juga diimplementasikan didalam *smartcards,* tetapi implementasi ini lambat. Bagaimanapun RSA tidak akan menyaingi kecepatan dari algoritma simetrik *cipher*.

Sudah disebutkan sebelumnya bahwa keamanan RSA tergantung dari kerumitan dalam memfaktorkan bilangan nonprima menjadi faktor primanya dalam jumlah besar, yang dalam hal ini n = a x b. Untuk menemukan kunci privat dari kunci publik e dan modulus n, maka kita harus memfaktorkan n. Dan n haruslah lebih besar dari 129 digit desimal modulus. Cara mudah untuk memecahkan RSA belum ditemukan. Cara penyerangan *brute-force* sangatlah tidak efisian dari pada mencoba memfaktorkan n. Enkripsi RSA dan signature dapat menjadi lebih cepat jika kita menggunkan nilai rendah e, tetapi tidak aman.

3.3 Fungsi *Hash* dan Algortima MD-5

3.3.1 Fungsi *Hash*

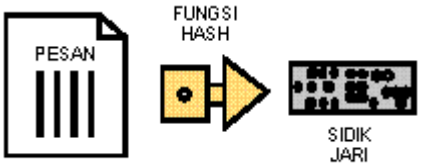
Fungsi *hash* adalah fungsi yang menerima masukan string yang panjangnya sembarang dan mengkonversinya menjadi string keluaran yang panjangnya tetap (fixed) (umumnya berukuran jauh lebih kecil daripada ukuran string semula).
Fungsi *hash* dapat menerima masukan string apa saja. Jika string menyatakan pesan (message), maka sembarang pesan M berukuran bebas dikompresi oleh fungsi *hash* H melalui persamaan

$$h = H(M)$$

Keluaran fungsi *hash* disebut juga nilai *hash* (*hash*-value) atau pesan-ringkas (*message digest*). h adalah nilai *hash* atau *message digest* dari fungsi H untuk masukan M.Aplikasi fungsi *hash* misalnya untuk memverifikasi kesamaan salinan suatu arsip di dengan arsip aslinya yang tersimpan di dalam sebuah basis data terpusat.

Fungsi *hash* satu-arah adalah fungsi *hash* yang bekerja dalam satu arah: pesan yang sudah diubah menjadi message digest tidak dapat dikembalikan lagi menjadi pesan semula. *Hash* satu arah (*one-way hash function*), terkadang disebut sidik jari (*fingerprint*), *hash*, *message integrity check (MIC),* atau *manipulation detection code.*
Cirinya antara lain:

- Dari sebuah pesan M, relatif mudah untuk membuat/mengkalkulasi *hash*nya H(M). Jadi tidak CPU intentsive seperti Public Key kriptografi
- Jika diketahui H(M), M tidak bisa diketahui
- Secara komputasi, sulit dari pesan-pesan yang berbeda M_1, M_2, … M_n membuat sebuah *hash* $H(M_n)$ yang sama. Jadi kalau pesannya berbeda sedikit saja, *hash*nya hampir pasti berbeda.



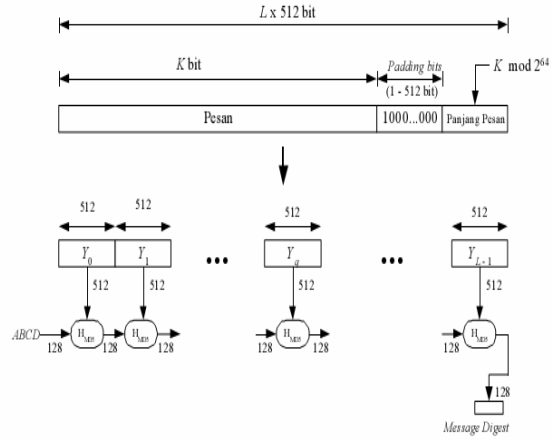
Gambar 9 Fungsi *hash* satu arah

Contoh algoritma fungsi *hash* satu arah adalah MD-5 dan SHA (Simple *Hash* Algorithm).

3.3.2 Algoritma MD-5

MD5 adalah fungsi *hash* satu-arah yang dibuat oleh Ron Rivest. MD5 merupakan perbaikan dari MD4 setelah MD4 berhasil diserang oleh kriptanalis. Algoritma MD5 menerima masukan berupa pesan dengan ukuran sembarang dan menghasilkan message digest yang panjangnya 128 bit. Gambaran pembuatan *message digest*

dengan algoritma MD5 diperlihatkan pada Gambar 10.



Gambar 10 Pembuatan *message digest*

Langkah-langkah pembuatan *message digest* secara garis besar adalah sebagai berikut:
1. Penambahan bit-bit pengganjal (padding bits).
2. Penambahan nilai panjang pesan semula.
3. Inisialisasi penyangga (buffer) MD.
4. Pengolahan pesan dalam blok berukuran 512 bit.

Penambahan bit-bit pengganjal

Pesan ditambah dengan sejumlah bit pengganjal sedemikian sehingga panjang pesan (dalam satuan bit) kongruen dengan 448 modulo 512. Ini berarti panjang pesan setelah ditambahi bit-bit pengganjal adalah 64 bit kurang dari kelipatan 512. Angka 512 ini muncul karena MD5 memperoses pesan dalam blok-blok yang berukuran 512.

Pesan dengan panjang 448 bit pun tetap ditambah dengan bit-bit pengganjal. Jika panjang pesan 448 bit, maka pesan tersebut ditambah dengan 512 bit menjadi 960 bit. Jadi, panjang bit-bit pengganjal adalah antara 1 sampai 512. Bit-bit pengganjal terdiri dari sebuah bit 1 diikuti dengan sisanya bit 0.

Penambahan nilai panjang bit semula

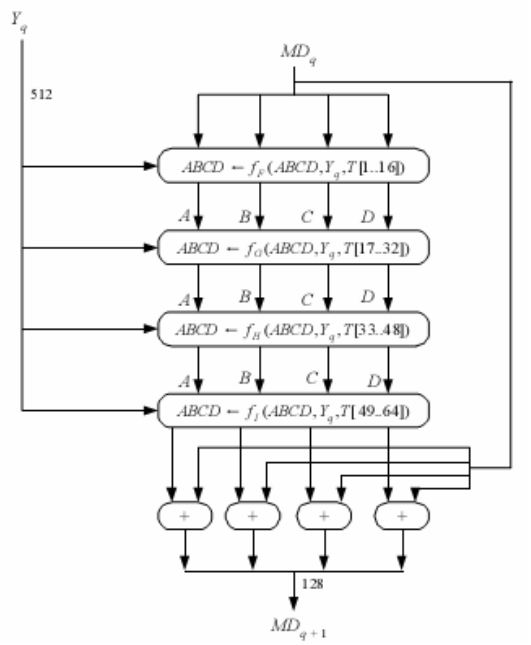
Pesan yang telah diberi bit-bit pengganjal selanjutnya ditambah lagi dengan 64 bit yang menyatakan panjang pesan semula. Jika panjang pesan > 2^{64} maka yang diambil adalah panjangnya dalam modulo 2^{64}. Dengan kata lain, jika panjang pesan semula adalah K bit, maka 64 bit yang ditambahkan 6 menyatakan K modulo 2^{64}. Setelah ditambah dengan 64 bit, panjang pesan sekarang menjadi 512 bit.

Inisialisasi Penyangga MD

MD5 membutuhkan 4 buah penyangga (buffer) yang masing-masing panjangnya 32 bit. Total panjang penyangga adalah 4 × 32 = 128 bit. Keempat penyangga ini menampung hasil antara dan hasil akhir.
Keempat penyangga ini diberi nama A, B, C, dan D. Setiap penyangga diinisialisasi dengan nilai-nilai (dalam notasi HEX) sebagai berikut:

A = 01234567
B = 89ABCDEF
C = FEDCBA98
D = 76543210

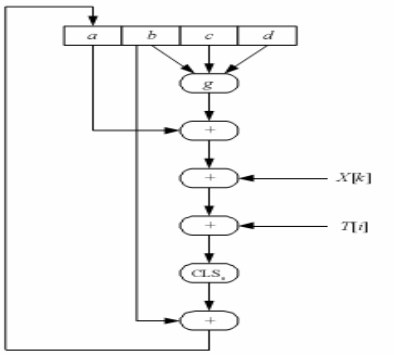
Pesan dibagi menjadi L buah blok yang masing-masing panjangnya 512 bit (Y0 sampai YL – 1). Setiap blok 512-bit diproses bersama dengan penyangga MD menjadi keluaran 128-bit, dan ini disebut proses HMD5. Gambaran proses HMD5 diperlihatkan pada Gambar 11.



Gambar 11 Pengolahan blok 512 bit (H_{MD5})

Proses HMD5 terdiri dari 4 buah putaran, dan masing-masing putaran melakukan operasi dasar MD5 sebanyak 16 kali dan setiap operasi dasar memakai sebuah elemen T. Jadi setiap putaran memakai 16 elemen Tabel T. Pada Gambar 11, Yq menyatakan blok 512-bit ke-q dari pesan

yang telah ditambah bit-bit pengganjal dan tambahan 64 bit nilai panjang pesan semula. MDq adalah nilai *message digest* 128-bit dari proses HMD5 ke-q. Pada awal proses, MDq berisi nilai inisialisasi penyangga MD. Fungsi-fungsi fF, fG, fH, dan fI masing-masing berisi 16 kali operasi dasar terhadap masukan, setiap operasi dasar menggunakan elemen Tabel T. Operasi dasar MD5 diperlihatkan pada Gambar 12.



Gambar 12 Operasi dasar *MD5*

Operasi dasar MD5 yang diperlihatkan pada Gambar 12 dapat ditulis dengan sebuah persamaan sebagai berikut:

$a \leftarrow b + CLSs(a + g(b, c, d) + X[k] + T[i])$

yang dalam hal ini,
a, b, c, d = empat buah peubah penyangga 32-bit (berisi nilai penyangga A, B, C, D)
g = salah satu fungsi F, G, H, I
CLSs = circular left shift sebanyak s bit
X[k] = kelompok 32-bit ke-k dari blok 512 bit message ke-q. Nilai k = 0 sampai 15.
T[i] = elemen Tabel T ke-i (32 bit)
\+ = operasi penjumlahan modulo 2^{32}

4. Pengujian dan Analisis Hasil

Pengubahan karakter di dalam tanda-tangan digital

Pengubahan karakter dalam tanda tangan digital mengakibatkan tidak terautentikasinya dokumen. Tanda tangan digital merupakan hasil dari fungsi *hash* dokumen awal yang dienkripsi. Dengan demikian, apabila tandatangan tersebut mengalami perubahan maka ketika didekripsi kembali hasilnya tidak akan sama dengan hasil fungsi *hash* pada dokumen awal. Apabila proses autentikasi dijalankan yaitu membandingkan fungsi *hash* hasil dari dekripsi tanda tangan digital dan hasil fungsi *hash* dokumen yang diproses, maka akan terjadi ketidaksamaan antara kedua hasil fungsi *hash* sehingga autentikasi dokumen akan gagal.

Pengubahan karakter di dalam teks diubah

Pengubahan suatu karakter dalam dokumen yang telah diberi *digital signature* akan mengakibatkan dokumen tersebut tidak terautentikasi. Kegagalan autentikasi dokumen tersebut terjadi karena pada saat dokumen tersebut kembali dicari hasil fungsi *hash*-nya, hasilnya tidak akan sama dengan hasil fungsi *hash* dokumen sebelum terjadinya perubahan. Karena *digital signature* yang telah didekrip merepresentasikan hasil fungsi *hash* dari dokumen sebelum perubahan, maka dalam pembandingan antara *hash* dokumen awal dan *hash* dokumen yang telah diubah akan terjadi ketidaksamaan. Ketidaksamaan tersebut menandakan bahwa dokumen telah mengalami perubahan dan dengan demikian autentikasi dokumen tersebut gagal.

Penggunaan kunci privat tidak berpadanan kunci publik

Kunci privat dan kunci publik dihasilkan pada proses algoritma RSA secara berpasangan. Untuk suatu kunci publik tertentu terdapat satu kunci privat pasangannya. Kedua kunci ini dimanfaatkan dalam hal enkripsi dan dekripsi hasil fungsi *hash* suatu dokumen. Suatu fungsi *hash* dienkrip dengan algoritma RSA menggunakan suatu kunci privat akan menghasilkan suatu *cipher text* yang hanya dapat dikembalikan ke bentuk awalnya dengan kunci publik pasangan kunci privat yang digunakan. Apabila kunci yang digunakan tidak berpasangan, maka proses dekripsi suatu tanda tangan digital tidak akan menghasilkan fungsi *hash* dokumen awal. Dalam proses autentikasi, fungsi *hash* hasil dekripsi tersebut akan mengalami ketidaksamaan dengan fungsi *hash* dari dokumen yang diproses. Dengan demikian apabila menggunakan kunci privat dan kunci publik yang tidak berpadanan, proses autentikasi akan mengalami kegagalan pula.

5. Kesimpulan

Algoritma RSA dapat digunakan untuk mengenkripsi dokumen secara aman. Dengan

menggunakan enkripsi asimetri, algoritma RSA dapat digunakan sebagai jaminan identitas pembuat dokumen (*non-repudiation*). Suatu dokumen yang berhasil didekrip dengan suatu kunci publik dapat memastikan bahwa pembuatnya adalah pemilik kunci privat pasangan kunci publik tersebut.

Algoritma MD5 dapat digunakan untuk memastikan orisinalitas suatu dokumen dengan membangkitkan suatu *message digest*. Suatu dokumen dapat dijamin orisinalitasnya dengan menjalankan kembali MD5 pada dokumen salinan dan membandingkan hasilnya dengan *message digest* yang diikutsertakan dalam dokumen salinan.

Tanda tangan digital dapat dilakukan dengan menggunakan kombinasi dari algoritma MD5 dan algoritma RSA. Algoritma MD5 menjamin orisinalitas suatu dokumen yang berarti bahwa suatu salinan dokumen adalah sama dengan dokumen aslinya. Algoritma RSA menjamin identitas pembuat dokumen yaitu autentikasi terhadap digital signature yang berhasil memastikan bahwa pemilik kunci privat adalah pembuat dokumen yang diproses.

DAFTAR PUSTAKA

[1] Fung T, Kwok (2005). Network Security Technologies. Auerbach Publication.

[2] Rhee Y, Man (2003). Internet Security Cryptograhic principles, Algorithms and Protocol 2nd. John Wiley & Sons

[3] Munir, Rinaldi. (2004). Bahan Kuliah IF5054 Kriptografi. Departemen Teknik Informatika, Institut Teknologi Bandung

[4] Rahardjo, Budi(2002). Keamanan Sistem Informasi Berbasis Internet. PT Insan Infonesia-Bandung & PT Indocisc-Jakarta

[5] Menezer, A, Van Oorschot. (1996). Handbook of Applied Cryptography. CRC Press, Inc.

[6] Encryption. *http://www.wikipedia.org*. tanggal akses 27 Desember 2006 pukul 10.23.

[7] How PGP works. *http://www.pgpi.org* tanggal akses 27 Desember 2006 pukul 10.26

[8] Digital signatures. *http://www.wikipedia.org* tanggal akses 29 desember 2006 pukul 9.59

[9] RSA. *http://www.wikipedia.org*. tanggal akses 29 Desember 2006 pukul 11.00

[10] Public-key criptography. *http://www.wikipedia.org*. tanggal akses 29 Desember 2006 pukul 10.07

[11] Digital signatures. *http://www.SearchSecurity.com*. tanggal akses 29 Desember 2006 pukul 10.35

[12] Tanda Tangan Digital pada Dokumen dengan menggunakan Algoritma RSA. *http://students.if.itb.ac.id/~if11080/DiSign/* tanggal akses 2 Januari 2007 pukul 10.04

[13] GNU Privacy Guard. http://yulian.firdaus.or.id/gnupg.php tanggal akses 3 Januari 2007 pukul 6.56