Bahan Kuliah ke-15

IF5054 Kriptografi

Algoritma RSA dan ElGamal

Disusun oleh:

Ir. Rinaldi Munir, M.T.

Departemen Teknik Informatika
Institut Teknologi Bandung
2004

15. Algoritma RSA dan ElGamal

15.1 Pendahuluan

- Dari sekian banyak algoritma kriptografi kunci-publik yang pernah dibuat, algoritma yang paling populer adalah algoritma *RSA*.

- Algoritma *RSA* dibuat oleh 3 orang peneliti dari *MIT* (*Massachussets Institute of Technology*) pada tahun 1976, yaitu: Ron (R)ivest, Adi (S)hamir, dan Leonard (A)dleman.

- Keamanan algoritma RSA terletak pada sulitnya memfaktorkan bilangan yang besar menjadi faktor-faktor prima. Pemfaktoran dilakukan untuk memperoleh kunci privat. Selama pemfaktoran bilangan besar menjadi faktor-faktor prima belum ditemukan algoritma yang mangkus, maka selama itu pula keamanan algoritma RSA tetap terjamin.

15.2 Properti Algoritma *RSA*

- Besaran-besaran yang digunakan pada algoritma RSA:
  1. p dan q bilangan prima (rahasia)
  2. $n = p \cdot q$ (tidak rahasia)
  3. $\phi(n) = (p - 1)(q - 1)$ (rahasia)
  4. e (kunci enkripsi) (tidak rahasia)
  5. d (kunci dekripsi) (rahasia)
  6. m (plainteks) (rahasia)
  7. c (cipherteks) (tidak rahasia)

15.2 Perumusan Algoritma *RSA*

- Algoritma *RSA* didasarkan pada teorema Euler (lihat bahan kuliah Teori Bilangan) yang menyatakan bahwa

$$a^{f(n)} \equiv 1 \pmod{n} \qquad (15.1)$$

yang dalam hal ini,

1. a harus relatif prima terhadap n
2. $f(n) = n(1 - 1/p_1)(1 - 1/p_2) \ldots (1 - 1/p_r)$, yang dalam hal ini $p_1, p_2, \ldots, p_r$ adalah faktor prima dari n.

- $f(n)$ adalah fungsi yang menentukan berapa banyak dari bilangan-bilangan 1, 2, 3, …, n yang relatif prima terhadap n.

- Berdasarkan sifat $a^k \equiv b^k \pmod{n}$ untuk k bilangan bulat ≥ 1, maka persamaan (15.1) dapat ditulis menjadi

$$a^{k\phi(n)} \equiv 1^k \pmod{n}$$

atau

$$a^{k\phi(n)} \equiv 1 \pmod{n} \qquad (15.2)$$

- Bila a diganti dengan m, maka persamaan (15.2) menjadi

$$m^{k\phi(n)} \equiv 1 \pmod{n} \qquad (15.3)$$

- Berdasarkan sifat $ac \equiv bc \pmod{n}$, maka bila persamaan (15.3) dikali dengan m menjadi:

$$m^{k\phi(n)+1} \equiv m \pmod{n} \qquad (15.4)$$

yang dalam hal ini m relatif prima terhadap n.

- Misalkan e dan d dipilih sedemikian sehingga

$$e \cdot d \equiv 1 \pmod{\phi(n)} \qquad (15.5)$$

atau

$$e \cdot d = k\phi(n) + 1 \qquad (15.6)$$

- Sulihkan (15.6) ke dalam persamaan (15.4) menjadi:

$$m^{e \times d} \equiv m \pmod{n} \qquad (15.7)$$

- Persamaan (15.7) dapat ditulis kembali menjadi

$$(m^{e})^{d} \equiv m \pmod{n} \qquad (15.8)$$

yang artinya, perpangkatan m dengan e diikuti dengan perpangkatan dengan d menghasilkan kembali m semula.

- Berdasarkan persamaan (15.8), maka enkripsi dan dekripsi dirumuskan sebagai berikut:

$$E_e(m) = c \equiv m^e \bmod n \qquad (15.9)$$

$$D_d(c) = m \equiv c^d \bmod n \qquad (15.10)$$

- Karena $e \cdot d = d \cdot e$, maka enkripsi diikuti dengan dekripsi ekivalen dengan dekripsi diikuti enkripsi:

$$D_d(E_e(m)) = E_e(D_d(m)) \equiv m^d \bmod n \qquad (15.11)$$

- Oleh karena $m^d \bmod n \equiv (m + jn)^d \bmod n$ untuk sembarang bilangan bulat j, maka tiap plainteks m, $m + n$, $m + 2n$, …, menghasilkan cipherteks yang sama. Dengan kata lain, transformasinya dari banyak ke satu.

  Agar transformasinya satu-ke-satu, maka m harus dibatasi dalam himpunan $\{0, 1, 2, \ldots, n - 1\}$ sehingga enkripsi dan dekripsi tetap benar seperti pada persamaan (15.8) dan (15.9).

15.5 Algoritma Membangkitkan Pasangan Kunci

1. Pilih dua buah bilangan prima sembarang, p dan q.
2. Hitung $n = p \cdot q$ (sebaiknya $p \neq q$, sebab jika $p = q$ maka $n = p^2$ sehingga p dapat diperoleh dengan menarik akar pangkat dua dari n).
3. Hitung $\phi(n) = (p - 1)(q - 1)$.
4. Pilih kunci publik, e, yang relatif prima terhadap $\phi(n)$.
5. Bangkitkan kunci privat dengan menggunakan persamaan (15.5), yaitu $e \cdot d \equiv 1 \pmod{\phi(n)}$.

   Perhatikan bahwa $e \cdot d \equiv 1 \pmod{\phi(n)}$ ekivalen dengan $e \cdot d = 1 + k\phi(n)$, sehingga d dapat dihitung dengan

   $$d = \frac{1 + kf(n)}{e} \qquad (15.12)$$

   Akan terdapat bilangan bulat k yang memberikan bilangan bulat d.

Hasil dari algoritma di atas:

- Kunci publik adalah pasangan (e, n)
- Kunci privat adalah pasangan (d, n)

Catatan: n tidak bersifat rahasia, namun ia diperlukan pada perhitungan enkripsi/dekripsi.

Contoh 15.1. Misalkan *A* akan membangkitkan kunci publik dan kunci privat miliknya. *A* memilih $p = 47$ dan $q = 71$ (keduanya prima).
Selanjutnya, *A* menghitung

$$n = p \cdot q = 3337$$

dan

$$f(n) = (p - 1)(q - 1) = 3220.$$

A memilih kunci publik $e = 79$, karena 79 relatif prima dengan 3220. *A* mengumumkan nilai *e* dan *n*.

Selanjutnya *A* menghitung kunci dekripsi *d* seperti yang dituliskan pada langkah instruksi 5 dengan menggunakan persamaan (15.12),

$$d = \frac{1 + (k \times 3220)}{79}$$

Dengan mencoba nilai-nilai $k = 1, 2, 3, \ldots$, diperoleh nilai *d* yang bulat adalah 1019. Ini adalah kunci privat untuk mendekripsi pesan. Kunci ini harus dirahasiakan oleh *A*.

Kunci publik: ($e = 79$, $n = 3337$)
Kunci privat: ($d = 1019$, $n = 3337$)

15.6 Algoritma Enkripsi/Dekripsi

Enkripsi

1. Ambil kunci publik penerima pesan, *e*, dan modulus *n*.
2. Nyatakan plainteks m menjadi blok-blok m_1, m_2, …, sedemikian sehingga setiap blok merepresentasikan nilai di dalam selang [0, n – 1].
3. Setiap blok m_i dienkripsi menjadi blok c_i dengan rumus
   $c_i = m_i^{\,e} \bmod n$

Dekripsi

1. Setiap blok cipherteks c_i didekripsi kembali menjadi blok m_i dengan rumus
   $m_i = c_i^{\,d} \bmod n$

Contoh 15.2. Misalkan *B* mengirim pesan kepada *A*. Pesan (plainteks) yang akan dikirim oleh *A* adalah

m = HARI INI

atau dalam sistem desimal (pengkodean ASCII) adalah

7265827332737873

B memecah *m* menjadi blok yang lebih kecil, misalnya *m* dipecah menjadi enam blok yang berukuran 3 digit:

$m_1 = 726$ $m_4 = 273$
$m_2 = 582$ $m_5 = 787$
$m_3 = 733$ $m_6 = 003$

Nilai-nilai m_i ini masih terletak di dalam selang [0, 3337 – 1] agar transformasi menjadi satu-ke-satu.

B mengetahui kunci publik *A* adalah $e = 79$ dan $n = 3337$. *A* dapat mengenkripsisikan setiap blok plainteks sebagai berikut:

$c_1 = 726^{79} \bmod 3337 = 215$; $c_2 = 582^{79} \bmod 3337 = 776$;
$c_3 = 733^{79} \bmod 3337 = 1743$; $c_4 = 273^{79} \bmod 3337 = 933$;
$c_5 = 787^{79} \bmod 3337 = 1731$; $c_6 = 003^{79} \bmod 3337 = 158$

Jadi, cipherteks yang dihasilkan adalah

c = 215 776 1743 933 1731 158.

Dekripsi dilakukan dengan menggunakan kunci privat

$d = 1019$

Blok-blok cipherteks didekripsikan sebagai berikut:

$m_1 = 215^{1019} \bmod 3337 = 726$
$m_2 = 776^{1019} \bmod 3337 = 582$
$m_3 = 1743^{1019} \bmod 3337 = 733$
…

Blok plainteks yang lain dikembalikan dengan cara yang serupa. Akhirnya kita memperoleh kembali plainteks semula

$m = 7265827332737873$

yang dalam sistem pengkodean ASCII adalah

m = HARI INI.

15.7 Keamanan RSA

- Kamanan algoritma *RSA* didasarkan pada sulitnya memfaktorkan bilangan besar menjadi fakto-faktor primanya.

  Masalah pemfaktoran: Faktorkan *n*, yang dalam hal ini *n* adalah hasil kali dari dua atau lebih bilangan prima.

  Pada *RSA*, masalah pemfaktoran berbunyi: Faktorkan *n* menjadi dua faktor primanya, *p* dan *q*, sedemikian sehingga $n = p \cdot q$.

- Sekali *n* berhasil difaktorkan menjadi *p* dan *q*, maka $\phi(n) = (p - 1)(q - 1)$ dapat dihitung. Selanjutnya, karena kunci enkrispi *e* diumumkan (tidak rahasia), maka kunci dekripsi *d* dapat dihitung dari persamaan $e \cdot d \equiv 1 \pmod{\phi(n)}$.

- Penemu algoritma *RSA* menyarankan nilai *p* dan *q* panjangnya lebih dari 100 digit. Dengan demikian hasil kali $n = p \times q$ akan berukuran lebih dari 200 digit.

  Menurut Rivest dan kawan-kawan, usaha untuk mencari faktor prima dari bilangan 200 digit membutuhkan waktu komputasi selama 4 milyar tahun, sedangkan untuk bilangan 500 digit membutuhkan waktu 10^{25} tahun!
  (dengan asumsi bahwa algoritma pemfaktoran yang digunakan adalah algoritma yang tercepat saat ini dan komputer yang dipakai mempunyai kecepatan 1 milidetik).

- Untunglah algoritma yang paling mangkus untuk memfaktorkan bilangan yang besar belum ditemukan. Selama 300 tahun para matematikawan mencoba mencari faktor bilangan yang besar namun tidak banyak membuahkan hasil. Semua bukti yang diketahui menunjukkan bahwa upaya pemfaktoran itu lura biasa sulit.

- Fakta inilah yang membuat algoritma *RSA* tetap dipakai hingga saat ini. Selagi belum ditemukan algoritma yang mangkus untuk memfaktorkan bilangan bulat menjadi faktor primanya, maka algoritma *RSA* tetap direkomendasikan untuk mengenkripsi pesan.

- Untuk informasi *RSA* lebih lanjut, kunjungi situs *web* `www.rsasecurity.com`

15.7 Algoritma ElGamal

- Algoritma ElGamal juga adalah algoritma kriptografi kunci-publik. Algoritma ini pada mulanya digunakan untuk *digital signature*, namun kemudian dimodifikasi sehingga juga bisa digunakan untuk enkripsi dan dekripsi.

- Keamanan algoritma ini terletak pada sulitnya menghitung logaritma diskrit.

  Masalah logaritma diskrit: Jika p adalah bilangan prima dan g dan y adalah sembarang bilangan bulat. Carilah x sedemikian sehingga $g^x \equiv y \pmod{p}$

- Besaran-besaran yang digunakan di dalam algoritma ElGamal adalah:

  1. Bilangan prima, p (tidak rahasia)
  2. Bilangan acak, g ($g < p$) (tidak rahasia)
  3. Bilangan acak, x ($x < p$) (rahasia, kunci privat)
  4. $y = g^x \bmod p$ (tidak rahasia, k. publik)
  5. m (plainteks) (rahasia)
  6. a dan b (cipherteks) (tidak rahasia)

Algoritma Membangkitkan Pasangan Kunci

1. Pilih sembarang bilangan prima p (p dapat di-*share* di antara anggota kelompok)
2. Pilih dua buah bilangan acak, g dan x, dengan syarat $g < p$ dan $1 \leq x \leq p - 2$
3. Hitung $y = g^x \bmod p$.

Hasil dari algoritma ini:

Kunci publik: tripel (y, g, p)
Kunci privat: pasangan (x, p)

Catatan: g dan p tidak rahasia, namun diperlukan pada perhitungan enkripsi/dekripsi (lihat algoritma enkripsi/dekripsi di bawah).

Algoritma Enkripsi/Dekripsi

Enkripsi:

1. Susun plainteks menjadi blok-blok m_1, m_2, …, sedemikian sehingga setiap blok merepresentasikan nilai di dalam selang $[0, p - 1]$.
2. Pilih bilangan acak k, yang dalam hal ini $1 \leq k \leq p - 2$.
3. Setiap blok m dienkripsi dengan rumus
   $a = g^k \bmod p$
   $b = y^k m \bmod p$
   Pasangan a dan b adalah cipherteks untuk blok pesan m. Jadi, ukuran cipherteks dua kali ukuran plainteksnya.

Dekripsi

1. Gunakan kunci privat x untuk mendekripsi a dan b menjadi plainteks m dengan persamaan:

$$m = b/a^x \bmod p$$

(Catatan: $(a^x)^{-1} = a^{p-1-x} \bmod p$)

(Catatlah bahwa karena $a^x \equiv g^{kx} \pmod p$ maka

$$\begin{aligned} b/a^x &\equiv y^k m/a^x \\ &\equiv g^{xk} m/g^{xk} \\ &\equiv m \pmod p, \end{aligned}$$

yang berarti bahwa plainteks dapat ditemukan kembali dari pasangan cipherteks a dan b)

Contoh 15.3. Misalkan A ingin membangkitkan pasangan kuncinya. A memilih $p = 2357$, $g = 2$, dan $x = 1751$.
A menghitung

$$y = g^x \bmod p = 2^{1751} \bmod 2357 = 1185$$

Jadi, kunci publik A adalah ($y = 1185$, $g = 2$, $p = 2357$)
dan kunci privatnya adalah ($x = 1751$, $p = 2357$).

Enkripsi: Misalkan B ingin mengirim plainteks $m = 2035$ kepada A (nilai m masih berada di dalam selang [0, 2357 – 1]).

B memilih bilangan acak $k = 1520$ (nilai k masih berada di dalam selang [0, 2357 – 1]).
B menghitung

$$a = g^k \bmod p = 2^{1520} \bmod 2357 = 1430$$
$$b = y^k m \bmod p = 1185^{1520} \cdot 2035 \bmod 2357 = 697$$

Jadi, cipherteks yang dihasilkan adalah (1430, 697). *B* mengirim cipherteks ini ke *A*.

Dekripsi: *A* mendekripsi cipherteks dari *B* dengan melakukan perhitungan sebagai berikut:

$$(a^x)^{-1} = a^{p-1-x} \bmod p = 1430^{605} \bmod 2357 = 872$$

$$m = b/a^x \bmod p = 697 \cdot 872 \bmod 2357 = 2035$$

Paten

- Algoritma ElGamal tidak dipatenkan. Tetapi, algoritma ini didasarkan pada algoritma Diffie-Hellman, sehingga hak paten algoritam Diffie-Hellman juga mencakup algoritma ElGamal. Untunglah hak paten algoritma Diffie-Hellman berakhir pada bulan April 1997, sehingga algoritma ElGamal dapat diimplementasikan untuk aplikasi komersil.