Syahrul Imardi, MT

STMIK Amik Riau

#23

MATAKULIAH

KEAMANAN PERANGKAT LUNAK

PKC - R S A

& El Gamal

P23

STMIK
Amik Riau

MATAKULIAH

KEAMANAN PERANGKAT LUNAK

Syahrul Imardi, MT

Public Key Cryptography

P23 : Algoritma RSA & El Gamal

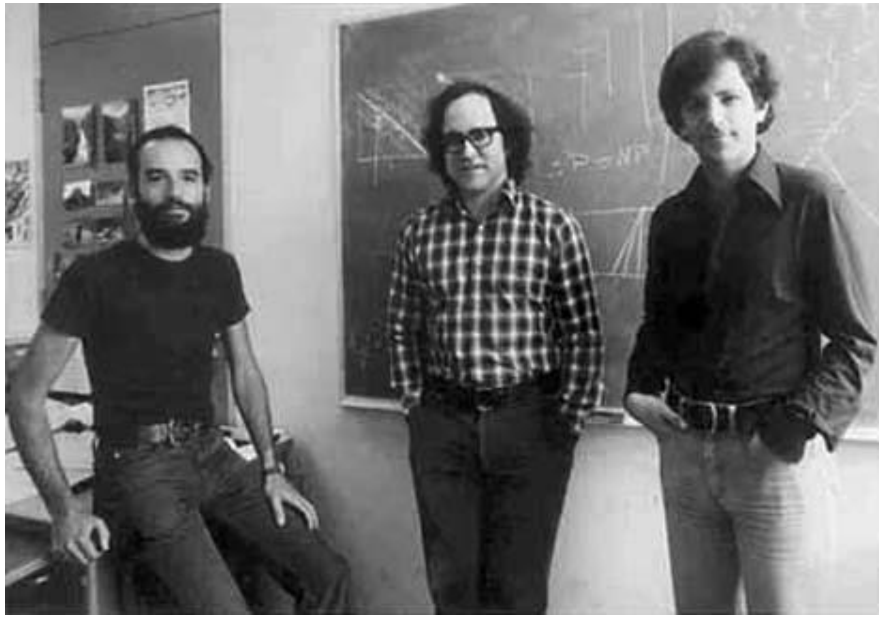
Algoritma RSA

Pendahuluan

- RSA merupakan algoritma kunci-publik yang paling terkenal dan paling banyak aplikasinya.

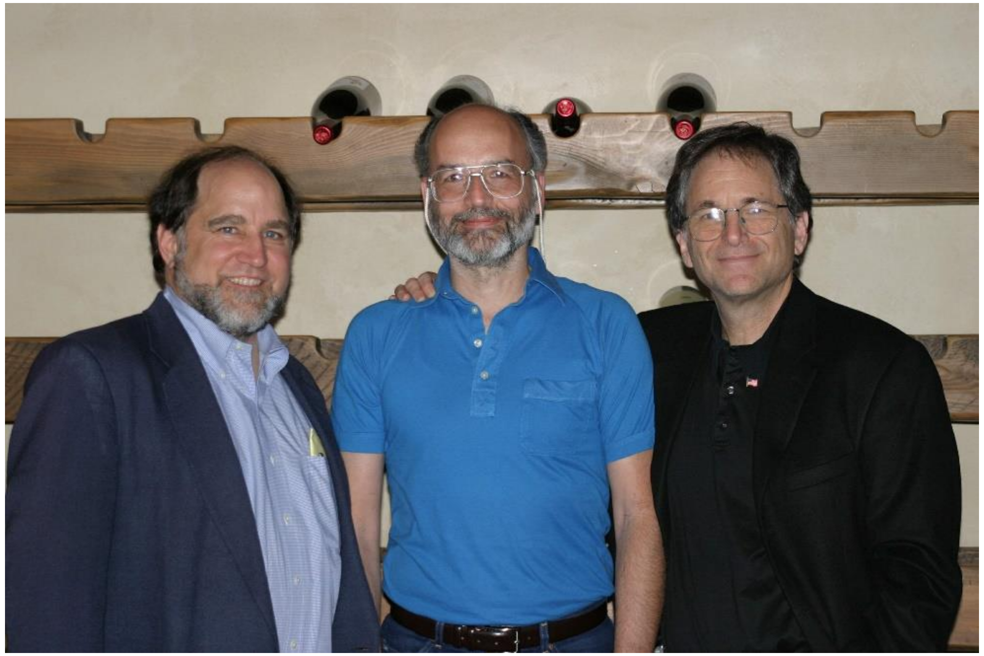
- Ditemukan oleh tiga peneliti dari *MIT* (*Massachussets Institute of Technology*), yaitu Ronald Rivest, Adi Shamir, dan Len Adleman, pada tahun 1976.
- RSA = Rivest-Shamir-Adleman

- Keamanan algoritma RSA terletak pada sulitnya memfaktorkan bilangan bulat yang besar menjadi faktor-faktor prima.



dahulu

The authors of RSA: Rivest, Shamir and Adleman



sekarang

Properti 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)

   Syarat: PBB(e, $\phi(n)$) = 1 , PBB = pembagi bersama terbessar = *gcd*
5. d (kunci dekripsi) (rahasia)

   d dihitung dari $d \equiv e^{-1} \bmod (\phi(n))$
6. m (plainteks) (rahasia)
7. c (cipherteks) (tidak rahasia)

Penurunan Rumus RSA

- Prinsip: Teorema Euler $a^{\phi(n)} \equiv 1 \pmod{n}$
- Syarat:

1. a harus relatif prima terhadap n
2. $\phi(n)$ = Toitent Euler = fungsi yang menentukan berapa banyak dari bilangan-bilangan 1, 2, 3, …, n yang relatif prima terhadap n.

   Contoh: $\phi(20) = 8$, sebab terdapat 8 buah yang relatif prima dengan 20, yaitu 1, 3, 7, 9, 11, 13, 17, 19.

Jika $n = pq$ adalah bilangan komposit dengan p dan q prima, maka

$$\phi(n) = \phi(p)\,\phi(q) = (p - 1)(q - 1).$$

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

$\downarrow$ (pangkatkan kedua ruas dengan k)

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

$\downarrow$

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

$\downarrow$ (ganti a dengan m)

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

$\downarrow$ (kalikan kedua ruas dengan m)

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

- Misalkan e dan d dipilih sedemikian sehingga

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

  atau

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

  Maka

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

  $$\downarrow$$

  $$m^{e \cdot d} \equiv m \pmod{n} \;\rightarrow\; (m^e)^d \equiv m \pmod{n}$$

- Enkripsi: $\boxed{E_e(m) = c = m^e \bmod n}$

- Dekripsi: $\boxed{D_d(c) = m = c^d \bmod n}$

Prosedur Pembangkitan Sepasang Kunci

1. Pilih dua bilangan prima, p dan q
2. Hitung $n = pq$.
3. Hitung $\phi(n) = (p - 1)(q - 1)$.
4. Pilih sebuah bilangan bulat e sebagai kunci publik, e harus relatif prima terhadap $\phi(n)$.
5. Hitung kunci dekripsi, d, dengan persamaaan

$$ed \equiv 1 \ (\textbf{mod}\ \phi(n)) \text{ atau } d \equiv e^{-1} \ \text{mod}\ (\phi(n))$$

Hasil dari algoritma di atas:

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

Enkripsi

1. Nyatakan pesan menjadi blok-blok plainteks: m_1, m_2, m_3, …

   (syarat: $0 \leq m_i < n - 1$)

2. Hitung blok cipherteks c_i untuk blok plainteks m_i menggunakan kunci publik e dengan persamaan

$$c_i = m_i^e \textbf{ mod } n$$

Dekripsi

1. Misalkan blok-blok cipherteks adalah c_1, c_2, c_3, …

2. Hitung kembali blok plainteks m_i dari blok cipherteks c_i menggunakan kunci privat d dengan persamaan

$$m_i = c_i^d \textbf{ mod } n,$$

Contoh pembangkitan kunci oleh Alice

- Misalkan Alice memilih p = 47 dan q = 71 (keduanya prima), maka dapat dihitung:

  $n = p \times q = 3337$

  $\phi(n) = (p - 1)\times(q - 1) = 3220.$

- Alice memilih kunci publik e = 79 (yang relatif prima dengan 3220 karena pembagi bersama terbesarnya adalah 1).

- Nilai e dan n dapat dipublikasikan ke umum.

- Selanjutnya Alice menghitung kunci privat *d* dengan kekongruenan:

  $ed \equiv 1 \pmod{\phi(n)}$

  d adalah balikan *e* dalam modulus $\phi(n)$

  d dapat dihitung dengan algoritma Euclidean atau dengan rumus:

$$d = \frac{1 + k\phi(n)}{e}$$

  Dengan mencoba nilai-nilai k = 1, 2, 3, …, diperoleh nilai *d* yang bulat adalah 1019. Ini adalah kunci privat (untuk dekripsi).

- Misalkan Bob akan mengirim plainteks *M* = 'HELLO ALICE' kepada Alice

- Dengan memisalkan A = 00, B = 01, …, Z = 25, maka pesan *m* dikodekan ke dalam *integer* (spasi diabaikan) menjadi

$$M = 0704111114001108 0204$$

Pecah *M* menjadi blok yang 4 digit:

$m_1 = 0704$ $m_4 = 1108$

$m_2 = 1111$ $m_5 = 0204$

$m_3 = 1400$

(Perhatikan, m_i masih terletak di dalam selang [0, 3337 – 1]

- Bob mengenkripsi setiap blok dengan menggunakan kunci public Alice (e = 79):

$$c_1 = 704^{79} \bmod 3337 = 328;$$

$$c_2 = 1111^{79} \bmod 3337 = 301;$$

$$c_3 = 1400^{79} \bmod 3337 = 2653;$$

$$c_4 = 1108^{79} \bmod 3337 = 2986;$$

$$c_5 = 204^{79} \bmod 3337 = 1164;$$

Cipherteks: C = 0328 0301 2653 2986 1164

.

- Bob mengirim cipherteks C kepada Alice

- Alice mendekripsi cipherteks dengan menggunakan kunci privatnya, yaitu d = 1019

$m_1 = 328^{1019} \bmod 3337 = 704 = 0704$

$m_2 = 301^{1019} \bmod 3337 = 1111$

$m_3 = 2653^{1019} \bmod 3337 = 1400$

$m_4 = 2986^{1019} \bmod 3337 = 1108$

$m_5 = 1164^{1019} \bmod 3337 = 204$

- Alice memperoleh kembali plainteks dari Bob

$M = 07041111140011080204$

yang dikodekan kembali menjadi

M = HELLO ALICE

Keamanan RSA

- Keamanan algoritma *RSA* terletak pada tingkat kesulitan dalam memfaktorkan bilangan bulat n faktor-faktor prima (p dan q), yang dalam hal ini $n = p \times q$.

- Sekali n berhasil difaktorkan menjadi p dan q, maka

$$\phi(n) = (p - 1)\times(q - 1) \text{ dapat dihitung.}$$

Selanjutnya, karena kunci enkripsi e diumumkan (tidak rahasia), maka kunci dekripsi d dapat dihitung dari kekongruenen

$$ed \equiv 1 \pmod{\phi(n)}.$$

- 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.

- Usaha untuk mencari faktor 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).

- Algoritma pemfaktoran yang tercepat saat ini memiliki kompleksitas

$$O(\exp(\sqrt[3]{\frac{64}{9} b(\log(b)^2}))$$

  untuk bilangan bulat *n* sepanjang b-bit.

- Hingga saat ini belum ditemukan algoritma pemfaktoran bilangan bulat besar dalam waktu polinomial.

- Fakta inilah yang membuat algoritma *RSA* dianggap masih aman untuk saat ini. Semakin panjang bilangan bulatnya, maka semakin lama waktu yang dibutuhkan untuk memfaktorkannya.

Contoh parameter RSA

- Modulus *n* sepanjang 1024 bit (setara 300 angka decimal
- Bilangan prima *p* dan *q* masing-masing panjangnya sekitar 154 angka decimal
- Sumber: https://www.di-mgt.com.au/rsa_alg.html

- `n =`
```
119294134840169509055527211331255649644606569661527638012067481954943056851150333806315957037715620297305000118628770846689969112892212245457118060574995989517080042105263427376322274266393116193517839570773505632231596681121927337473973220312512599061231322250945506260066557538238517575390621262940383913963
```

- `p =`
```
10933766183632575817611517034730668287155799984632223454138745671121273456287670008290843302875521274970245314593222946129064538358581018615539828479146469
```

- `q =`
```
10910616967349110231723734078614922645337060882141748968209834225138976011179993394299810159736904468554021708289824396553412180514827996444845438176099727
```

- Secara umum dapat disimpulkan bahwa RSA hanya aman jika *n* cukup besar.

- Jika panjang *n* hanya 256 bit atau kurang, ia dapat difaktorkan dalam beberapa jam saja dengan sebuah komputer *PC* dan program yang tersedia secara bebas.

- Jika panjang *n* adalah 512 bit atau kurang, ia dapat difaktorkan dengan beberapa ratus komputer

- Tahun 1977, 3 orang penemu *RSA* membuat sayembara untuk memecahkan cipherteks dengan menggunakan RSA di majalah *Scientific American*.

- Hadiahnya: $100

- Tahun 1994, kelompok yang bekerja dengan kolaborasi internet berhasil memecahkan cipherteks hanya dalam waktu 8 bulan.

Kelemahan RSA

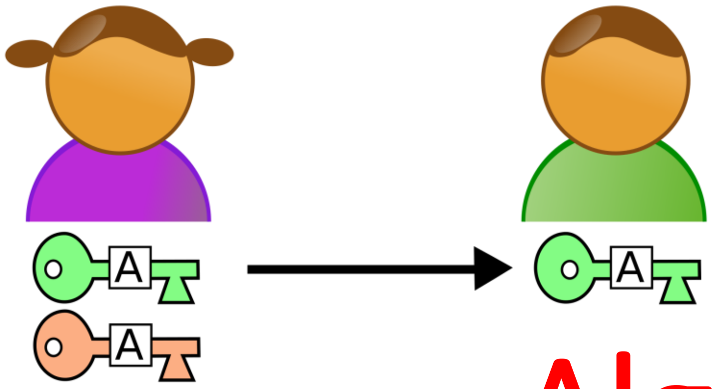
- *RSA* lebih lambat daripada algoritma kriptografi kunci-simetri seperti *DES* dan *AES*

- Dalam praktek, *RSA* tidak digunakan untuk mengenkripsi pesan, tetapi mengenkripsi kunci simetri (kunci sesi) dengan kunci publik penerima pesan.

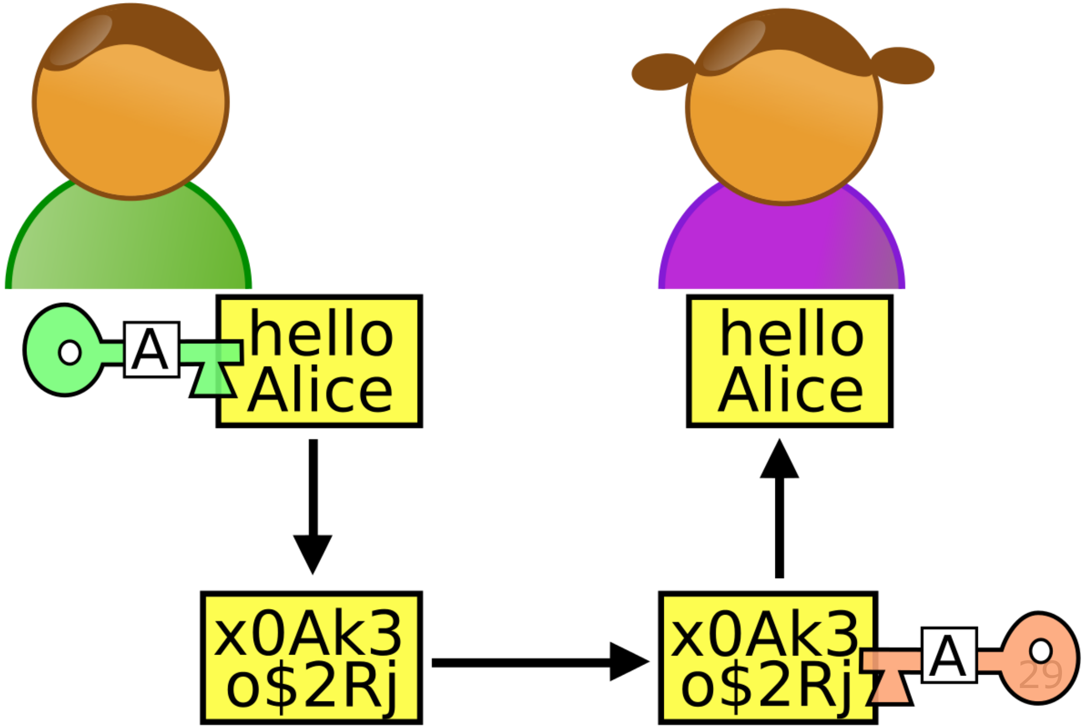
- Pesan dienkripsi dengan algoritma simetri seperti DES atau AES.

- Pesan dan kunci simetri dikirim bersamaan.

- Penerima mendekripsi kunci simetri dengan kunci privatnya, lalu mendekripsi pesan dengan kunci simetri tersebut.



Algoritma Elgamal



Pendahuluan

- Algoritma Elgamal dibuat oleh Taher Elgamal (1985). Pertama kali dikemukakan di dalam makalah berjudul "*A public key cryptosystem and a signature scheme based on discrete logarithms*"

- 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

$$g^x \equiv y \pmod{p}$$

Properti algoritma ElGamal:

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, kunci publik)
5. m (plainteks)	(rahasia)
6. a dan b (cipherteks)	(tidak rahasia)

Prosedur Pembangkitan 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)

Prosedur Enkripsi

1. Susun plainteks menjadi blok-blok m_1, m_2, …, (nilai setiap blok 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.

Prosedur Dekripsi

1. Gunakan kunci privat x untuk menghitung $(a^x)^{-1} = a^{p-1-x} \bmod p$

2. Hitung plainteks m dengan persamaan:

   $m = b/a^x \bmod p = b(a^x)^{-1} \bmod p$

Contoh: Alice membangkitkan kunci publik dan kunci privatnya. Bob mengengkripsi pesan dengan menggunakan kunci publik Alice.

(a) Pembangkitan kunci (Oleh Alice)

Misal p = 2357, g = 2, dan x = 1751.

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

Hasil: Kunci publik: (y = 1185, g = 2, p = 2357)

Kunci privat: (x = 1751, p = 2357).

Alice memberitahu kunci publik ini kepada Bob

(b) Enkripsi (Oleh Bob)

Misalkan pesan m = 2035 (nilai m masih berada di dalam selang [0, 2357 – 1]).

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

Bob 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).

Bob mengirim cipherteks ini kepada Alice.

(c) Dekripsi (Oleh Alice)

Alice menghitung:

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

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

Alice mendapatkan kembali plainteks m = 2035 yang dikirim oleh Bob.

Referensi utama :

>> Michael Felderer , Riccardo Scandariato (editor) - Exploring Security in Software Architecture and Design, 2018.

>> Nancy R. Mead, Carol Woody - Cyber Security Engineering_ A Practical Approach for Systems and Software Assurance-Addison-Wesley Professional (2016)

>> James Helfrich - Security for Software Engineers-CRC Press (2019)

>> Pete Loshin - Simple Steps to Data Encryption_ A Practical Guide to Secure Computing-Syngress (2013)

>> Tevfik Bultan,Fang Yu,Muath Alkhalaf,Abdulbaki Aydin (auth.) - String Analysis for Software Verification and Security (2017

Ada pertanyaan?