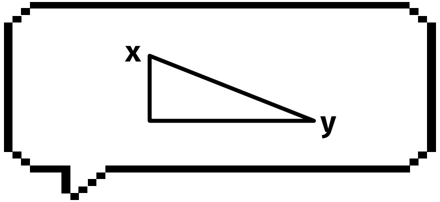
Praktikum Kriptografi

Pertemuan - 08

Topik: RSA

Review

- **Elgamal**



RSA

(Rivest Shamir Adleman)

Apa itu RSA?

RSA adalah sebuah algoritma berdasarkan skema **public-key cryptography**. Diberi nama RSA sebagai inisial para penemunya: Ron Rivest, Adi Shamir, dan Leonard Adleman.

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

Key Generation

1. Pilih 2 bilangan prima sembarang, **p** dan **q**. (rahasia)
2. Hitung **n = p x q** (tidak rahasia)
3. Hitung **m = (p – 1)(q – 1)** (rahasia)
4. Pilih e, sebuah bilangan bulat sebagai **kunci publik**. (tidak rahasia)

   Syarat : (gcd (e,m) = 1)
5. Hitung d, kunci privat, sedemikian agar **(d x e) mod m = 1**. (rahasia)

   d = e^{-1} mod m

Maka diperoleh :

Kunci publik adalah pasangan **(e,n)**. Bersifat tidak rahasia.

Kunci private adalah pasangan **(d,n)**. Bersifat rahasia

Contoh Key Generation

Diketahui:

p = 47 ; q = 71

Maka didapat :

n = p x q
  = 47 x 71 = **3337**

m = (p-1)(q-1)
  = (47-1)(71-1)
  = 46 x 70
  = **3220**

Pilih e yg relatif prima dengan 3220

e = 79

Hitung d = e^{-1} mod m

Gcd (e,m) = 1

Gcd(79,3220) = 1

3220 = 79(40) + 60
79 = 60(1) + 19
60 = 19(3) + 3
19 = 3(6) + 1
3 = 1(3) + 0

q1 = 40 ; q2 = 1 ; q3 = 3 ; q4 =6

t0 = 0 ; t1 = 1

t2 = (t0 - (q1 . t1)) mod m
   = 0 - (40) mod 3220 = 3180
t3 = (t1 - (q2 . t2)) mod m
   = 1 - (3180) mod 3220 = 41
t4 = (t2 - (q3 . t3)) mod m
   = 3180 - (41*3) mod 3220
   = 3057
t5 = (t3 - (q4 . t4)) mod m
   = 41 - (3057*6) mod 3220
   = **1019**

e^{-1} = 1019

Maka d = 1019

Contoh Key Generation

Maka diperoleh pasangan kunci :

publik (e,n) = (79,3337)
private (d,n) = (1019,3337)

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

Enkripsi

1. Ambil kunci publik penerima pesan, **e dan n**
2. Nyatakan plainteks M menjadi blok-blok **M1, M2, …,** sedemikian sehingga setiap blok merepresentasikan nilai di dalam selang **[0, n – 1]**
3. Setiap blok $\mathbf{M_i}$ dienkripsi menjadi blok Ci dengan rumus
   $\mathbf{Ci = (M_i)^e \bmod n}$

Dekripsi

1. Harus mempunyai private key dari langkah-langkah sebelumnya **(d,n)**
2. Setiap blok cipherteks Ci tadi didekripsi kembali menjadi blok $\mathbf{M_i}$

   $\mathbf{M_i = (Ci)^d \bmod n}$

Contoh Enkripsi

*menggunakan kunci dari contoh sebelumnya

1. Diketahui plainteks → **m = “HARI INI”**

   M nya ini diubah dulu ke ASCII

   → m = 7265827332737873

2. **M dipecah 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

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

Char	Dec	Oct	Hex
@	64	0100	0x40
A	65	0101	0x41
B	66	0102	0x42
C	67	0103	0x43
D	68	0104	0x44
E	69	0105	0x45
F	70	0106	0x46
G	71	0107	0x47
H	72	0110	0x48
I	73	0111	0x49
J	74	0112	0x4a
K	75	0113	0x4b
L	76	0114	0x4c
M	77	0115	0x4d
N	78	0116	0x4e
O	79	0117	0x4f
P	80	0120	0x50
Q	81	0121	0x51
R	82	0122	0x52
S	83	0123	0x53
T	84	0124	0x54
U	85	0125	0x55
V	86	0126	0x56
W	87	0127	0x57
X	88	0130	0x58
Y	89	0131	0x59
Z	90	0132	0x5a
[	91	0133	0x5b
\	92	0134	0x5c
]	93	0135	0x5d
^	94	0136	0x5e
_	95	0137	0x5f

Contoh Enkripsi

3. Kunci publik diketahui, yaitu e = 79 dan n = 3337.

   Sehingga dapat dilakukan enkripsi sebagai berikut :

   $c^1 = 726^{79} \bmod 3337 = 215$

   $c^2 = 582^{79} \bmod 3337 = 776$

   $c^3 = 733^{79} \bmod 3337 = 1743$

   $c^4 = 273^{79} \bmod 3337 = 933$

   $c^5 = 787^{79} \bmod 3337 = 1731$

   $c^6 = 003^{79} \bmod 3337 = 158$

4. Jadi, cipherteks yang dihasilkan adalah

   c = 215 776 1743 933 1731 158.

Contoh Dekripsi

1. Dekripsi dilakukan dengan menggunakan kunci privat **d = 1019**

   Blok-blok cipherteks didekripsikan sebagai berikut:

   $m_1 = 215^{1019} \bmod 3337 = 726$

   $m_2 = 776^{1019} \bmod 3337 = 582$

   $m_3 = 1743^{1019} \bmod 3337 = 733$

   $m_4 = 933^{1019} \bmod 3337 = 273$

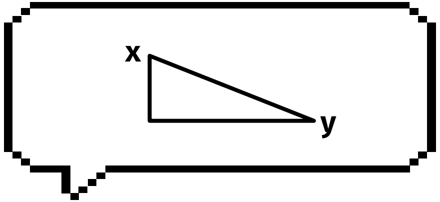
   $m_5 = 1731^{1019} \bmod 3337 = 787$

   $m_6 = 158^{1019} \bmod 3337 = 003$

2. Selanjutnya, m digabungkan menjadi

   m = 7265827332737873, terus dikodein deh pake ASCII.

3. Sehingga dihasilkan bahwa hasil dekripsi/plainteksnya itu adalah **“HARI INI”**



RSA Cryptanalysis

How to Attack an RSA Encryption

Cryptanalysis??

Cryptanalysis merupakan sebuah studi cryptosystem yang bertujuan untuk mengerti **cara kerja dari algoritma** yang bersangkutan dan **teknik untuk memecahkan/melemahkannya**.

Dilakukan dengan cara mencoba melakukan dekripsi Ciphertext (CT) **tanpa mengetahui Plaintext (Pt)** dari sumbernya, **encryption key, dan algoritma** yang digunakan.

Objective: *Malicious Activities, Security Improvements*

Jenis Cryptanalysis

Ciphertext-only Attack: Punya akses hanya ke ciphertext, tidak ada pengetahuan tentang plaintext, algoritma enkripsi, atau cryptographic key.

Known Plaintext Attack: Punya akses ke plaintext dari ciphertext, bertujuan untuk mencari kunci enkripsi yang digunakan.

Chosen Plaintext Attack: Punya pengetahuan tentang algoritma maupun akses ke perangkat yang melakukan enkripsi. Lalu, memilih sebuah plaintext dan mengenkripsinya, untuk mendapat informasi terkait key yang digunakan.

Differential Cryptanalysis: Tipe Chosen Plaintext Attack pada block cipher, dimana beberapa plaintext digunakan (tidak hanya 1). Umumnya untuk mencari tahu response dari sebuah algoritma untuk tipe data yang berbeda.

Jenis Cryptanalysis

Integral Cryptanalysis: Mirip dengan differential, tetapi menggunakan set dari plaintext yang dimodifikasi sebagian. Efektif untuk algoritma tipe jaringan substitusi-permutasi.

Side Channel Attack: Memanfaatkan info yang didapat dari physical system (misal: runtime, consumed power, radiasi elektromagnetik, dsb).

Dictionary Attack: Semi bruteforce attack, umumnya digunakan dengan menggunakan sebuah wordlist hasil enkripsi dari password yang umum digunakan.

MiTM Attack: Dilakukan dengan memposisikan diri diantara jalur komunikasi, ciphertext akan ditangkap oleh attacker, dan dicoba untuk dibaca atau diubah.

Pada RSA??

RSA sampai momen penulisan PPT ini, masih dapat dikatakan sebagai algoritma yang cukup kuat dan baik untuk digunakan.

Cryptanalysis RSA dapat dikatakan rumit dan akan memakan waktu milyaran tahun pada komputer biasa (atau 10 detik pada *"Perfect Quantum Computer"*, tapi hal itu belum ada).

Untuk keperluan studi, kita akan mencoba melakukan cryptanalysis pada RSA yang lebih disederhanakan.

Technically Possible, Realistically *(somewhat)* Impossible.

Ciri algoritma kriptografi yang dapat dikatakan sebagai cukup baik, terhadap posibilitas cryptanalysis pada dunia nyata.

Letsgoo

Pertama, cari posibilitas serangan selain *factoring n*, dengan observasi apakah kita dapat mengkomputasi **$\varphi(n)$.** Jika **n** dan **$\varphi(n)$ diketahui,** maka produk 2 prima **p, q, dan n** dapat difaktorisasi dengan mudah, dengan persamaan (untuk mencari variabel ***p*** dan ***q***) :

$$n = pq$$

$$\varphi(n) = (p-1)(q-1)$$

Lalu, jika kita substitusi ***q = np*** ke persamaan kedua, kita dapat membentuk persamaan kuadratik dengan value ***p*** yang tidak diketahui:

$$P^2 - (n - \varphi(n) + 1)p + n = 0$$

Letsgoo

p dan ***q*** menjadi akar dari persamaan dari faktor ***n***. Sehingga, jika nilai **φ(*n*)** dapat dipelajari, maka ***n*** dapat difaktorkan dan sistem dapat dipecahkan.

Bingung?! Ini contohnya:

Misal: diketahui ***n = 84773093*** dan **φ(*n*) = *84754668***.
Dengan menggunakan informasi ini, kita dapat membentuk persamaan kuadratik:

$$P^2 - 18426p + 84773093 = 0$$

Dengan formula kuadratik, kita dapat menemukan akar 9539 dan 8887, yang merupakan kedua faktor dari ***n***.

Mencari Eksponen Dekripsi

Kita akan mencoba membuktikan bahwa eksponen dekripsi ***a*** dapat digunakan untuk mencari faktor ***n***. Sehingga jika ***a*** diketahui, maka modulus ***n*** juga harus diganti. Untuk menggambarkannya, kita dapat menggunakan **Las Vegas Algorithm :**

Misalkan ***0 < ∈ < 1*** adalah bilangan riil. Maka, untuk setiap instance I, akan dicari sebuah jawaban yang merupakan sebuah probabilitas. Jawaban **tidak selalu di-return, tapi jika iya, maka jawaban pasti benar.**

Algoritma akan terus diulang hingga jawaban ditemukan.

Mencari Eksponen Dekripsi

Algoritma tersebut didasarkan dari fakta akar dari **1 (mod n)** dimana ***n = pq*** merupakan produk dari 2 bilangan prima ganjil yang berbeda.

Reminder: kongruensi $x^2 \equiv 1 \pmod p$ punya 2 solusi, yaitu $x \equiv \pm 1 \pmod p$, begitu juga dengan $x^2 \equiv 1 \pmod q$.

Karena, $x^2 \equiv 1 \pmod n$ didapatkan jika dan hanya jika $x^2 \equiv 1 \pmod p$ dan $x^2 \equiv 1 \pmod q$. **Maka,** akan ada 4 akar dari $1 \pmod n$. **Hasil** yang didapatkan untuk 2 solusi adalah $x = \pm 1 \pmod n$ **(trivial)** dan 2 hasil lainnya adalah negatif dari masing-masing modulo n **(non-trivial)**.

Bingung kan? Ini contohnya

Misalkan: ***n = 403 = 13 x 31***
Maka, 4 akar dari 1 mod 403 = **1, 92, 311,** dan **402**

92 → ***x ≡ 1 (mod 13), x ≡ -1 (mod 31)***, dengan **CRT (non-trivial)**
311 → 403 - 92 = 311 (non-trivial), atau ***x ≡ -1 (mod 13), x ≡ 1 (mod 31)***

Jika kita anggap x sebagai **non-trivial square root**, maka didapat:

$$n \mid (x - 1)(x + 1)$$

n tidak memfaktorkan sisi lawannya, tetapi mengikuti aturan :

$$\gcd(x + 1, n) = p \text{ atau } q \;\|\; \gcd(x - 1, n) = p \text{ atau } q$$

Mencari Eksponen Dekripsi

Pengetahuan akan **akar non-trivial** dari ***1 (mod n)*** dapat menentukan faktorisasi dari ***n***, hanya dengan jumlah komputasi polinomial, yang menentukan berbagai hasil dari kriptografi.

Misal: Pada Slide 23, ***gcd(93,403) = 31*** dan ***gcd(312,403) = 13***

Berikutnya, kita akan menggunakan contoh untuk menggambarkan aplikasi algoritma dimana **A** mengkomputasi **eksponen dekripsi a** dari **eksponen enkripsi b**.

Mencari Eksponen Dekripsi

Misal: *n* = 89855713, *b* = 34986517, *a* = 82330933, dan diberikan random value ***w* = 5**.

Maka,

$$ab - 1 = 2^3 \times 360059073378795$$

Lakukan loop untuk mencari nilai **akar:**

Step 6 → v = 85877701

…

Step 10 → v = 1

…

Step 12 → gcd(85877702,n) = 9103

dan faktor lainnya adalah **n/9103 = 9871**

Analisis Algoritma

Jika kita beruntung dan memilih ***w*** yang merupakan kelipatan dari ***p*** dan ***q***, maka kita dapat langsung mencari faktor ***n***. Umumnya terdeteksi di **step 2**, jika w relatif prima dengan n, maka kita dapat menghitung **w^r, w^{2r}, w^{4r}, ..., hingga $w^{(2^t)r} \equiv 1 \pmod{n}$.**

Karena, $ab - 1 = 2^8 r \equiv 0 \pmod{\varphi(n)}$ maka dapat diketahui bahwa **$w^{(2^8)} \equiv 1 \pmod{n}$.**

Lalu, untuk mencari solusi kita dapat melanjutkan perhitungan jumlah solusi untuk setiap kongruensi.

Note: untuk kasus dimana pilihan ***w*** buruk, tidak akan dijelaskan di praktikum ini, karena cukup memakan waktu, silakan eksplorasi sendiri di buku **“Cryptography: Theory and Practice”** oleh **Douglas Stinson**.

Partial Information untuk PT Bits

Kadang, ada kemungkinan informasi parsial mungkin **“bocor”** oleh enkripsi RSA. Contoh informasi parsial yang dimaksud :

1) Diberikan $y = e_K(x)$, komputasi *parity*(y), dimana *parity*(y) didenotasikan ke bit dari x tingkat rendah
2) Diberikan $y = e_K(x)$, komputasi *half*(y), dimana *half*(y) = 0, jika $0 \leqslant x \leqslant n/2$ dan *half*(y) = 1 jika $n/2 \leqslant x \leqslant n-1$

Dari sana, kita akan membuktikan, jika kita **mendapat $y = e_K(x)$**, seluruh algoritma yang menghitung ***parity*(y)** atau ***half*(y)**, dapat digunakan **untuk mengkonstruksi algoritma** yang menghasilkan **plaintext *x***.

Partial Information untuk PT Bits

Secara **simple** nya, jika kita **mendapat sebuah ciphertext**, mengkomputasikan **low-order bit dari plaintext-nya** secara polinomial akan **ekuivalen dengan menentukan seluruh plaintext**.

Untuk membuktikan, dapat dilakukan komputasi ***parity*(y)** sebagai ekuivalen polinomial dari ***half*(y)**, dengan identitas:

$$\mathit{half}(y) = \mathit{parity}(y \times e_K(2) \bmod n)$$

$$\mathit{parity}(y) = \mathit{half}(y \times e_K(2^{-1}) \bmod n)$$

Dari aturan multiplicative $e_K(x_1)e_K(x_2) = e_K(x_1x_2)$

Partial Information untuk PT Bits

Untuk menghitung $x = d_K(y)$, dari algoritma yang menghitung *half*(y), dapat dilakukan dengan cara:

$$y_i = half(y \times (e_K(2))^i) = half(e_K(x.2^i)) \text{ untuk } 0 \leqslant i \leqslant \log_2 n$$

Lalu, dapat diobservasi bahwa :

$$half(e_K(x)) = 0 \Leftrightarrow x \in \left[0, \frac{n}{2}\right)$$

$$half(e_K(2x)) = 0 \Leftrightarrow x \in \left[0, \frac{n}{4}\right) \cup \left[\frac{n}{2}, \frac{3n}{4}\right)$$

$$half(e_K(4x)) = 0 \Leftrightarrow x \in \left[0, \frac{n}{8}\right) \cup \left[\frac{n}{4}, \frac{3n}{8}\right) \cup \left[\frac{n}{2}, \frac{5n}{8}\right) \cup \left[\frac{3n}{4}, \frac{7n}{8}\right),$$

dan seterusnya. **Sehingga**, kita dapat menggunakan teknik binary search untuk membuktikannya.

Partial Information untuk PT Bits

Contoh:

Misalkan diberikan ***n* = 1457, *b* = 779**, dan **Ciphertext (*y*) = 722, e_K(2)** dikomputasi dan didapatkan hasil **946.**

Dengan menggunakan algoritma untuk menemukan ***half*(y)**, dan binary search, kita dapat menemukan nilai y_i, yaitu:

i	0	1	2	3	4	5	6	7	8	9	10
y_i	1	0	1	0	1	1	1	1	1	0	0

Lalu, binary search akan berlanjut, hingga ditemukan ***plaintext x* = [999.55] = 999**

Tugas

1. Kerjakan secara manual Enkripsi, dan Dekripsi algoritma RSA, dengan diketahui :

   p = 19, q = 13

   Plaintext: HIMATIF

Format: Tugas8_NPM.pdf

Deadline Tugas: H-1 Praktikum Berikutnya, 23.59

Thank You!!

Kalau misalkan ada pertanyaan,
yaudah tanya aja

Praktikum Kriptografi 2023