Kriptosistem Knapsack

Disusun Oleh :

Akik Hidayat [1]

Universitas padjadjaran

Bandung

2007

1. Jurusan Matematika FMIPA Universitas Padjadjaran
Jl. Raya Bandung Sumedang Km 21 Jatinangor Tlp/Fax 022-7794696

I. PENDAHULUAN

Sejarah Merkle-Hellman Knapsack Kriptosistem

Merkle-Hellman Knapsack merupakan Kriptosistem yang dibuat oleh Merkle dan Hellman pada tahun 1980 Walaupun sistem ini, dan beberapa variannya, telah dipecahkan sekitar awal tahun 1980, tetapi masih layak untuk dipelajari dengan berbagai alasan.

Masalah yang mendasari matematika adalah masalah penjumlahan himpunan bagian dimana sangat berhubungan dengan masalah knapsack dari operasi pencarian (dengan demikian, "Knapsack" dalam nama dari system ini merupakan misnomer). Suatu masalah bisa dideskripsikan sebagai berikut. Jika setiap elemen dari himpunan S adalah suatu bilangan integer positif. Diberikan suatu himpunan bagian dari S, penjumlah dari elemen terdekat dari bagian himpunan bilangan menghasilkan bilangan integer yang berkoresponden dengan himpunan bagiannya.

Masalah penjumlahan himpunan bagian adalah kebalikannya, untuk itu, diberikan bilangan integer T, apakah ada himpunan bagian dari S yang dijumlahkan sama dengan T? Penyelesaian masalah ini (hanya membutuhkan respon ya atau tidak) apakah NP merupakan masalah yang lengkap. Selanjutnya adalah hubungan dengan pencarian masalah, diberikan suatu T yang mana jawabannya adalah Ya, cari suatu himpunan bagian (atau himpunan-himpunan bagian, jawabannya tidak mungkin unik) yang merupakan hasil jumlah. Seperti dengan masalah NP apapun, meskipun itu adalah bukan algoritma polynomial untuk penyelesaian masalah secara umum. Beberapa soal mungkin bisa terselesaikan dengan mudah, ini adalah soal dengan masalah himpunan penjumlahan. Dan memperolehkan pencarian dari suatu trapdoor.

Disini akan merumuskan suatu masalah, sebagai contoh : diberikan s1,s2,..,sn adalah himpunan bilangan positif (ukuran pemanggilan) dan T adalah bilangan positif, penyelesaian masalah adalah dengan mencari suatu vector 0-1 (x1,x2,..,xn) yang dapat dinyatakan sebagai berikut:

$$x_1s_1 + x_2s_2 + ... + x_ns_n = T.$$

Bentuk Knapsack

Ada dua macam bentuk atau tipe Knapsack yaitu :

- General Knapsacks
- Superincreasing knapsacks

General Knapsacks

Masalah knapsack membahas tentang serangkaian $a1, a2, \ldots, a_n$ dari bilangan bulat positif dan jumlah, T. Masalahnya adalah dengan mencari sebuah vector dari bit 0 dan 1 yang dijumlahkan dari bilangan bulat positif terdekatnya dengan bit 1 dibandingkan dengan T. Yaitu, diberikan $S = [a1, a2, \ldots, an]$, dan T, carilah vector V dari nilai 0 dan 1 seperti:

$$\sum_{i=1}^{n} a_i \ * \ \text{vi} = \text{T}$$

Superincreasing knapsacks

Himpunan bilangan bulat positif S harus dari suatu rangkaian pengurutan dari bilangan terkecil sampai terbesar**.** Setiap bilangan adalah bilangan terbesar daripada jumlah semua bilangan awalnya. Selanjutnya, setiap bilangan a_k akan dapat dinyatakan sbb :

$$a_k > \sum_{j=1}^{k-1} a_j$$

Penyelesaiannya dimulai dengan T. dicocokkan dengan bilangan paling besar dalam himpunan S . Jika bilangan ini adalah lebih besar dari T tidak dijumlahkan, selanjutnya dilakukan pengurutan posisi dalam V dijadikan 0. Jika bilangan terbesar adalah kurang dari atau sama dengan T, maka bilangan tersebut adalah dijumlahkan,selanjutnya dilakukan posisi pengurutan dalam$V$ menjadi nilai 1 dan penurunan T dari bilangan. Diulangi untuk semua sisa bilangan dalam S.

Super-Increasing Sets

Suatu himpunan dari nilai dikatakan super-increasing jika suatu bilangan lebih besar dari penjumlahan bilangan-bilangan sebelumnya.

$$s_k > \sum_{j=1}^{k-1} s_j$$

Sebagai contoh 3, 7 , 12, 30, 60, 115 adalah suatu himpunan super-increasing sedangkan 3, 5, 7, 9, 11 bukan. Diambil kesimpulan yang kuat dari beberapa angka yang terlihat selalu mempunyai bentuk batasan himpunan super-increasing (misal, 1, 3, 9, 27, 81,..), sebagai jawaban dari masalah himpunan penjumlahan untuk suatu batasan himpunan super-increasing, hanya satu kunci pembanding untuk T. Mengubah nilai dari T ke bentuk T' dan kemudian melakukan proses perulangan untuk T'. selanjutnya dari cara ini dan jika kami akhiri dengan 0 kemudian T adalah himpunan penjumlahan, dan banyaknya perubahan dari himpunan, sebaliknya adalah bukan suatu himpunan penjumlahan. Dengan demikian, untuk T=82 kita kalkulasikan : 82 – 60 = 22; 22 – 12 = 10; 10 – 7 = 3; 3 – 3= 0 dan selanjutnya 82 adalah himpunan penjumlahan dan himpunan ini diberikan dari vector(1,1,1,0,1,0) menjadi 3, 7, 12, 60. Dari bentuk lain , T = 80 akan memberikan:

80 – 20 = 60; 20 – 12 = 8; 8 – 7 = 1 dan 80 adalah bukan dari himpunan penjumlahan. Untuk suatu ukuran himpunan super-increasing , suatu solusi, jika itu ada, berbentuk unik. Juga dari beberapa bilangan hanya bisa digunakan sekali, banyaknya kalkulasi bisa diakhiri bagian awal. Sebagai contoh, dengan T =85 akan diperoleh 85 – 60 = 25;

25 – 12 = 13 dan kita bisa berhenti ketika menggunakan 12 kembali dan ini tidak bisa memberikan suatu penyelesaian.

Kriptosistem

Terdapat masalah himpunan penjumalahan dalam suatu kriptosystem. Bob menciptakan dan membuat umumnya dari deret ukuran s1, s2, …, sn. Untuk berhubungan dengan Bob, Alice mengambil pesannya dan menuliskan dalam suatu binary string. Dia mulai

memecah string kedalam ukuran blok n (blok tersebut diakhiri dengan 0 jika dibutuhkan). Beberapa blok digunakan sebagai karakteristik vector dari suatu himpunan, dan dia mengkalkulasikan penjumlahan himpunan dan mengirimkan itu ke Bob (bahwasanya, dia mengkalkulasikan jumlah dari banyaknya s_i' yang disesuaikan dengan banyaknya 1' di dalam blok). Oscar, yang menghalangi pesan ini, harus memecahkan masalah himpunan penjumlahan dari beberapa blok untuk mengganti pesan dan ini berhubungan dengan masalah NP jika ukuran dari system cukup besar. Dari bentuk lain, ini adalah tepat jika Bob harus melakukan dengan baik dalam mendekrip pesan. Namun, Bob mengetahui suatu rahasia!! Ukuran kunci publicnya adalah merupakan himpunan super-increasing dengan penyamaran yang pintar. Dan semua dia lakukan dengan menjadikan kembali urutan ke himpunan super increasing yang asli dan menyelesaikan masalah yang sederhana.

II. Knapsack Cryptosystem

Dalam kriptosistem Merkle-Hellman, 'penyamaran' diakhiri dengan penelusuran: diberikan s1, s2,..,sn merupakan sebuah ukuran himpunan super-increasing. Memilih suatu bilangan prima p yang lebih besar dari pada jumlah semua s_i, dan suatu bilangan b dengan $1 < b < p$. Sekarang Bob mengkalkulasikan ukuran himpunan kunci public :

$$t_i = bs_i \bmod p.$$

ketika Bob menerima sebuah blok dari Alice, dia mulai menggandakan itu dari $b^{-1} \bmod p$, yang mengubah s_i' kembali ke t_i' dan kemudian memecahkan masalah himpunan penjumlahan yang menggunakan t_i'. Bob menyimpan nilai p dan b sebagai kunci rahasia. System ini sangat popular sejak bisa secara cepat diimplementasikan. Namun, diawal tahun 1980, Shamir, menggunakan Lenstra algoritma pemrograman linier cepat, mampu membuang jauh penyamaran ini dan memperoleh himpunan super-increasing milik Bob(atau sesuatu yang ekuivalen dengan itu). Banyak metode lain untuk melakukan penyamaran himpunan super-increasing yang telah dicoba, tapi sebagian besar dari itu dapat di buka. Satu metode, diketahui sebagai kriptosistem **Chor-Rivest**, menggunakan nilai manipulasi berhingga dan ini masih dianggap aman untuk digunakan.

III. Algoritma Knapsack

Apa yang kita butuhkan :

- **S = (s1,…,sn) bilangan integer superincreasing**
- $p > \sum_{i=1}^{n} s_i$ **bilangan prima**
- **a, $1 \le a \le p$-1**
- **t = a si mod p**

Public Key: t

Private Key: si, p, a,

Encode:

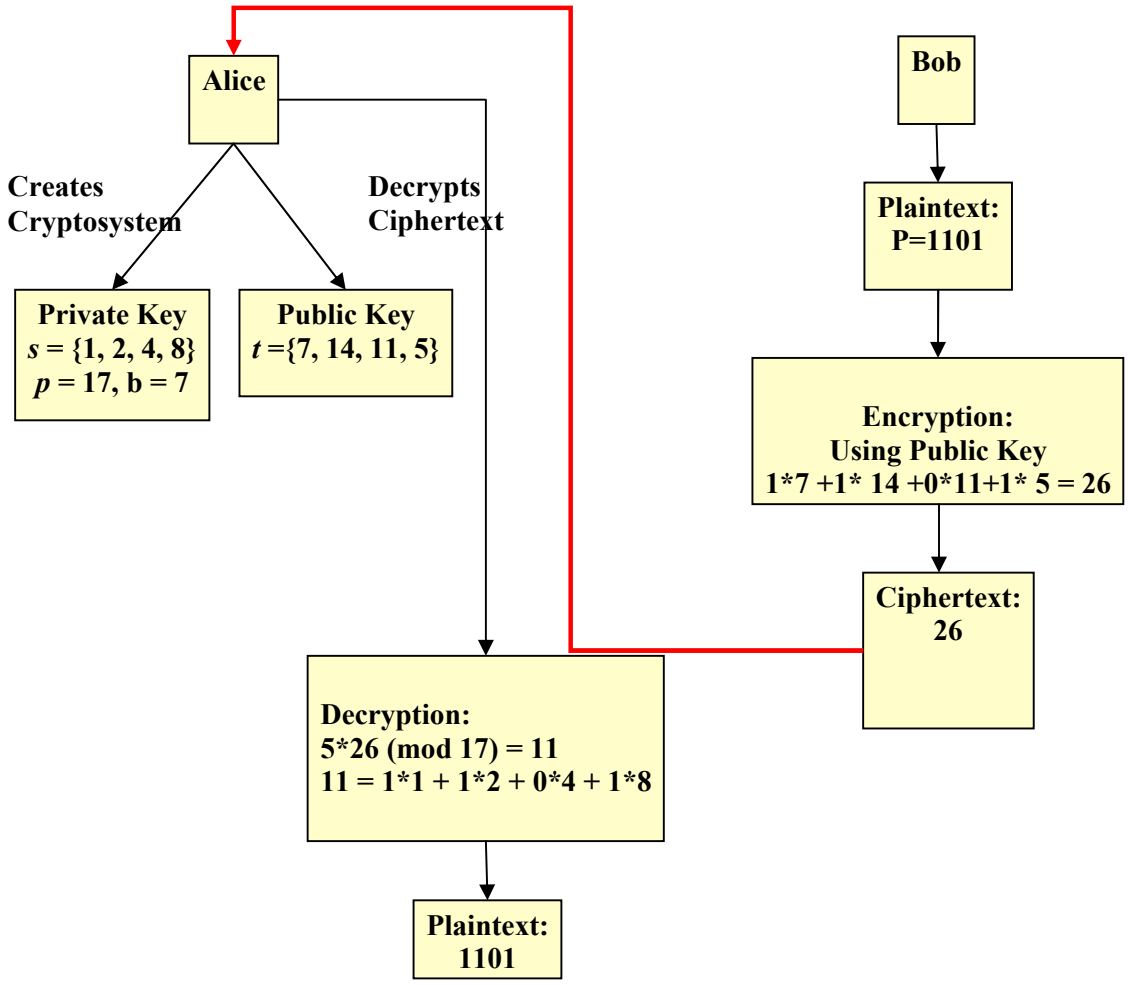
$e_s(x_1,\ldots,x_n) = \sum_{i=1}^{n} x_i t_i$

Decode:

$z = a^{-1} y \bmod p$

- penyelesaian masalah *subset* $(s_1,\ldots,s_n,Z)$ diberlakukan untuk $d_K(y) = (x_1,\ldots,x_n)$.

IV. Cara Kerja Knapsack



Mencari Knapsack superincreasing :

1. Pertama mengambil serangkaian ***superincreasing*** s bilangan bulat positif dengan cara pilih bilagan inisial(terkecil). Pilih bilangan selanjutnya dengan bilangan yang lebih besar daripada yang pertama. Kemudian pilih bilangan yang lebih besar daripada penjumlahan bilangan pertama dan kedua. Teruskan proses ini dari memilih bilangan-bilangan baru yang lebih besar daripada jumlah semua bilangan yang sebelumnya dipilih.

2. Setelah memilih knapsack yang sederhana $S = [s_1, s_2,\ldots,s_m]$, memilih sebuah bilangan pengali *b* dan di modulus *p.*
    - Bilangan mod seharusnya adalah angka yang lebih besar daripada jumlah semua s_i dan merupakan bilangan prima
    - Pengali tidak mempunyai factor persekutuan dengan modulus.
3. Mencari kunci publik , kita mengganti setiap bilangan s_i dalam knapsack sederhana dengan ketentuan.

$$t_i = a * s_i \ mod \ p$$

S = [1, 2, 4, 8] dan diubah dari pengali *b* dan kemudian di mod *p* dimana b = 7 dan p = 17:

1 * 7 = 7 mod 17 = 7

2 * 7 = 14 mod 17 = 14

4 * 7 = 28 mod 17 = 11

9 * 7 = 63 mod 17 = 5

knapsack t= [7,14,11,5]

Enkripsi Knapsack

1. Pesan plaintext *P* bisa dituliskan dalam bentuk:

$P = [p_1, p_2,..., p_k]$.

2. Membagi pesan ke dalam blok bit-bit *m* , $P_0 = [p_l, p_2,..., p_m]$, $P_1 = [p_{m+1},..., p_{2m}]$, dan selanjutnya. *(m* adalah bilangan pembatas dalam knapsack)
3. Memilih nilai dengan mengubah dari bentuk 1 bit kedalam P_i selanjutnya P_i disajikan sebagai vector yang dipilih untuk element t.
4. Nilai ciphertext merupakan:

Pi * t, target menggunakan blok P_i untuk memilih vector.

Proses Dekripsi :

Penerima tahu knapsack sederhana dan nilai dari ***a*** dan ***p*** yang ditranformasi ke dalam knapsack sulit.

- Dengan nilai a^{-1} kemudian $a^* a^{-1} = 1 \bmod p$. Dalam contoh kami, $7^{-1} \bmod 17$ adalah 5, mulai $5 * 8 \bmod 17 = 40 \bmod 17 = (17 * 7) + 1 \bmod 17 = 1$.
- Ingat bahwa *H* adalah knapsack sulit yang terjadi dari knapsack sederhana*S. H* adalah memperoleh *S* dengan

$H = w * S \bmod n$

- Pesan ciphertext juga di dapat dari algoritma enkripsi:
  $\boldsymbol{C = H * P = w * S * P \bmod n}$
- Untuk mengubah cipher, pengali *C* dari w^{-1}, mulai
  $\boldsymbol{w^{-1} * C = w^{-1} * H * P =}$
  $\boldsymbol{w^{-1} * w * S * P \bmod n =}$
  $\boldsymbol{S * P \bmod n}$

- Sekarang penerima dapat memecahkan masalah knapsack sederhana dengan knapsack *S* dan target $w^{-1} * C_i$ untuk beberapa bilangan ciphertext C_i.
- Dimulai $w^{-1} * C_i = S * P \bmod n$, solusi untuk target $w^{-1} * C_i$ adalah blok plaintext P_i, dimana adalah pesan asli yang di enkripsi.
- *QED*

V. Contoh Kasus Knapsack

Masalah:

Diberikan Private key :

s = (1,2,5,11,32,87,141)

a = 200

p = 307

Plaintext (x) :SYANE MARANNU THANA

Pertanyaan :

Chipertext : ?????

Jawaban :

Enkripsi :

Perhitungan Public Key (t) :

ti = a* si mod p

$t_1 = a*s_1 \bmod p = 200 * 1 \bmod 307 = 200$

$t_2 = a*s_2 \bmod p = 200 * 2 \bmod 307 = 93$

$t_3 = a*s_3 \bmod p = 200 * 5 \bmod 307 = 79$

$t_4 = a*s_4 \bmod p = 200 * 11 \bmod 307 = 51$

$t_5 = a*s_5 \bmod p = 200 * 32 \bmod 307 = 260$

$t_6 = a*s_6 \bmod p = 200 * 87 \bmod 307 = 208$

$t_7 = a*s_7 \bmod p = 200 * 141 \bmod 307 = 263$

Didapatkan

t = (200,93,79,51,260,208,263)

Plaintext :

SYANE MARANNU THANA

Dimasukkan dalam kode ASCII

x = 83 89 65 78 69 77 65 82 65 78 78 85 84 72 65 78 65

Masing – masing kode ASCII tersebut dikonversi ke biner

S → 83 : 1010011

Y → 89 : 1011001

A → 65 : 1000001

N → 78 : 1001110

E → 69 : 1000101

M → 77 : 1001101

A → 65 : 1000001

R → 82 : 1010010

A → 65 : 1000001

N → 78 : 1001110

N → 78 : 1001110

U → 85 : 1010101

T → 84 : 1010100

H → 72 : 1001000

A → 65 : 1000001

N → 78 : 1001110

A → 65 : 1000001

Plaintext di bagi dalam block sesuai dengan banyaknya s,
pada contoh ini banyaknya s adalah 7 digit.

1010011→ y = 200 + 79 + 208 + 263 = 750

1011001→ y = 200 + 79 + 51 + 263 = 593

1000001→ y = 200 + 263 = 463

1001110→ y = 200 + 51 + 260 + 208 = 719

1000101→ y = 200 + 260 + 263 = 723

1001101→ y = 200 + 51 + 260 + 263 = 774

1000001→ y = 200 + 263 = 463

1010010→ y = 200 + 79 + 208 = 487

1000001→ y = 200 + 263 = 463

1001110 → y = 200 + 51 + 260 + 208 = 719

1001110→ y = 200 + 51 + 260 + 208 = 719

1010101→ y = 200 + 79 + 260 + 263 = 802

1010100→ y = 200 + 79 + 260 = 539

1001000→ y = 200 + 51 = 251

1000001→ y = 200 + 263 = 463

1001110→ y = 200 + 51 + 260 + 208 = 719

1000001→ y = 200 + 263 = 463

Ciphertext :

750 593 463 719 723 774 463 487 463 719 719 802 539 251 463 719 463

Dekripsi :

Hitung Z

$Z = a^{-1} y \bmod p$

200^{-1} =???? → dengan algoritma extended Euclidian

gcd (307,200) =

307 = 1 * 200 + 107

$200 = 1 * 107 + 93$

$107 = 1 * 93 + 14$

$93 = 6 * 14 + 9$

$14 = 1 * 9 + 5$

$9 = 1 * 5 + 4$

$5 = 1 * 4 + 1$

$4 = 4 * 1 + 0$

Selanjutnya:

$t_0 = 0$

$t_1 = 1$

$t_2 = t_0 - q_1 t_1 = 0 - 1*1 = -1$

$t_3 = t_1 - q_2 t_2 = 1 - 1*(-1) = 2$

$t_4 = t_2 - q_3 t_3 = (-1) - 1*2 = -3$

$t_5 = t_3 - q_4 t_4 = 2 - 6*(-3) = 20$

$t_6 = t_4 - q_5 t_5 = (-3) - 1*20 = -23$

$t_7 = t_5 - q_6 t_6 = 20 - 1*(-23) = 43$

$t_8 = t_6 - q_7 t_7 = (-23) - 1*43 = -66$

$200^{-1} = 241$

- Untuk y = 750:

  $Z = 241*750 \bmod 307$

  $= 180750 \bmod 307$

  $= 234$

$234 = 1*1 + 0*2 + 1*5 + 0*11 + 0*32 + 1*87 + 1*141$

Plaintext → 1010011

- Untuk y = 593:

  Z = 241*593 mod 307

  = 142913 mod 307

  = 158

158 = 1*1 + 0*2 + 1*5 + 1*11 + 0*32 + 0*87 + 1*141

Plaintext → 1011001

- Untuk y = 463:

  Z = 241*463 mod 307

  = 111583 mod 307

  = 142

142 = 1*1 + 0*2 + 0*5 + 0*11 + 0*32 + 0*87 + 1*141

Plaintext → 1000001

- Untuk y = 719:

  Z = 241*719 mod 307

  = 173279 mod 307

  = 131

131= 1*1 + 0*2 + 0*5 + 1*11 + 1*32 + 1*87 + 0*141

Plaintext → 1001110

- Untuk y = 723:

  Z = 241*723 mod 307

  = 174243 mod 307

  = 174

174 = 1*1 + 0*2 + 0*5 + 0*11 + 1*32 + 0*87 + 1*141

Plaintext → 1000101

- Untuk y = 774:

  Z = 241*774 mod 307

  = 186534 mod 307

  = 185

185 = 1*1 + 0*2 + 0*5 + 1*11 + 1*32 + 0*87 + 1*141

Plaintext → 1001101

- Untuk y = 487:

  Z = 241*487 mod 307

  = 117367 mod 307

  = 93

93 = 1*1 + 0*2 + 1*5 + 0*11 + 0*32 + 1*87 + 0*141

Plaintext → 1010010

- Untuk y = 802:

  Z = 241*802 mod 307

  = 193282 mod 307

  = 179

179 = 1*1 + 0*2 + 1*5 + 0*11 + 1*32 + 0*87 + 1*141

Plaintext → 1010101

- Untuk y = 539:

  Z = 241*539 mod 307

  = 129899 mod 307

  = 38

38 = 1*1 + 0*2 + 1*5 + 0*11 + 1*32 + 0*87 + 0*141

Plaintext → 1010100

- Untuk y = 251:

  Z = 241*251 mod 307

  = 60491 mod 307

  = 12

12 = 1*1 + 0*2 + 0*5 + 1*11 + 0*32 + 0*87 + 0*141

Plaintext → 1001000

Plaintext Dimasukkan dalam kode ASCII

x = 83 89 65 78 69 77 65 82 65 78 78 85 84 72 65 78 65

Maka akan menjadi:

SYANE MARANNU THANA