# Penggunaan Aritmetika Modulo dan Balikan Modulo pada Modifikasi Algoritma *Knapsack*

**Sesdika Sansani – NIM 13507047**
Jurusan Teknik Informatika ITB, Bandung, Jl. Ganesha 10, email: if117047@students.if.itb.ac.id

***Abstract*** *– Makalah ini membahas mengenai penggunaan aritmetika modulo, relatif prima, dan balikan modulo pada algoritma* knapsack*. Selain itu, makalah ini juga memberikan penjelasan yang cukup banyak mengenai permasalahan* knapsack*, algoritma* knapsack *itu sendiri yaitu sistem kriptografi Merkle-Hellman, dan serangan terhadap* knapsack*.*

***Kata Kunci****:* knapsack*, Merkle-Hellman, teori bilangan, aritmetika modulo, relatif prima, balikan modulo*

## 1. PENDAHULUAN

Kehidupan kita saat ini dilingkupi oleh kriptografi. Mulai dari transaksi di mesin ATM, transaksi di bank, transaksi dengan kartu kredit, percakapan melalui telepon genggam, mengakses internet, sampai mengaktifkan peluru kendali pun menggunakan kriptografi. Begitu pentingnya kriptografi untuk keamanan informasi (*information security*), sehingga jika berbicara mengenai masalah keamanan yang berkaitan dengan penggunaan komputer, maka orang tidak bisa memisahkannya dengan kriptografi.

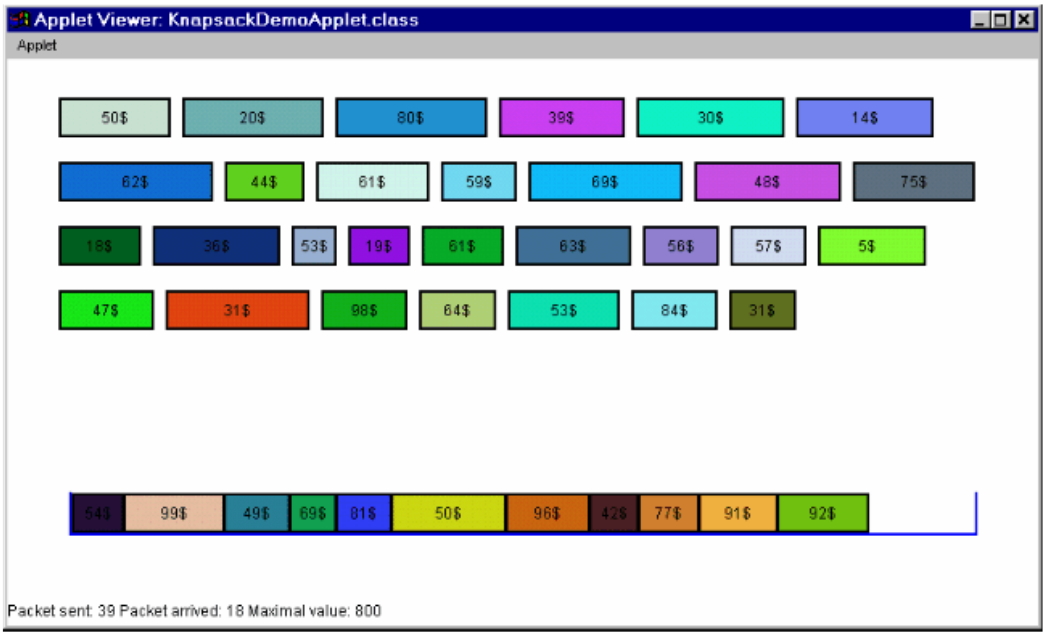
Salah satu teori yang sering digunakan dalam kriptografi adalah teori bilangan (*number theory*). Banyak permasalahan dalam teori bilangan yang digunakan pada kriptografi, misalnya permasalahan RSA (Rivest, Shamir, Adleman), logaritma diskrit, Diffie-Hellman, dan *subset sum problem*.

Seperti yang sudah disebutkan di atas, *subset sum problem* adalah salah satu persoalan dalam teori bilangan. *Subset sum problem* merupakan salah satu persoalan khusus dari persoalan *Knapsack*. Persoalan *knapsack* sendiri cukup banyak menggunakan teori bilangan dalam modifikasinya, antara lain aritmetika modulo, relatif prima, dan balikan modulo.

## 2. *KNAPSACK PROBLEM [1]*

***Knapsack problem*** adalah suatu persoalan dalam optimisasi kombinatorial. Namanya berasal dari permasalahan maksimisasi dari pilihan terbaik dari barang-barang yang perlu dibawa sehingga memenuhi satu tas penuh untuk dibawa dalam perjalanan. Diberikan beberapa barang, masing-masing memiliki bobot (*weight*) dan harga (*value*), tentukan jumlah masing-masing barang untuk dimasukkan dalam suatu kumpulan sehingga jumlah bobot kurang dari batasan bobot yang diberikan dan jumlah harga setinggi mungkin.

**Contoh 1**. Seorang pencuri merampok sebuah toko dan menemukan $n$ barang. Sebanyak $i$ barang memiliki harga $v_i$ dolar dan berbobot $w_i$ pound ($v_i$ dan $w_i$ adalah angka riil), tetapi dia hanya bisa membawa paling banyak $W$ pound di dalam buntilannya. Barang apa yang seharusnya dia ambil?



**2.1. Definisi**

Terdapat $n$ jenis barang, 1 – $n$. Setiap barang $j$ mempunyai harga $v_j$ dan bobot $w_j$. Diasumsikan bahwa semua harga dan bobot adalah bilangan nonnegatif. Bobot maksimum yang dapat dibawa adalah $W$.

Ada tiga macam dari persoalan knapsack secara umum, yaitu:

- ✓ *0-1 knapsack problem* membatasi jumlah tiap barang $x_j$ dengan nol atau satu. Secara matematika *0-1-knapsack problem* dapat diformulasikan sebagai berikut:

  Optimasi $\sum_{j=1}^{n} p_j x_j$

  dengan syarat $\sum_{j=1}^{n} w_j x_j \le W, \quad x_j \in \{0,1\}, \ j = 1,\ldots,n$
- ✓ *Bounded knapsack problem* membatasi jumlah tiap barang $x_j$ dengan harga maksimal integer $b_j$. Secara matematika *bounded knapsack problem* dapat diformulasikan sebagai berikut:

  Optimasi $\sum_{j=1}^{n} p_j x_j$ dengan syarat

  $\sum_{j=1}^{n} w_j x_j \le W, \ x_j \in \{0,1, \ldots, b_j\}, \ j = 1, \ldots, n$
- ✓ *Unbounded knapsack problem* tidak memberikan batas atas untuk jumlah masing-masing barang.

## 3. ALGORITMA *KNAPSACK [6]*

Algoritma *Knapsack* merupakan salah satu algoritma kriptografi kunci-publik. Disebut kriptografi kunci publik (*public-key cryptography*) karena kunci untuk enkripsi tidak rahasia dan dapat diketahui oleh siapapun (diumumkan ke publik), sementara kunci untuk dekripsi hanya diketahui oleh penerima pesan (karena itu rahasia) [9].

Dalam teori algoritma, persoalan *knapsack* termasuk ke dalam kelompok *NP-complete*. Suatu persoalan *C* disebut *NP-complete* jika [3]:

1. *C* adalah *NP*

   *C* adalah *NP* dapat ditunjukkan dengan memperlihatkan bahwa calon solusi dari *C* dapat dipecahkan dalam orde waktu polinomial.

   [4] Secara intuitif, **NP** adalah sekumpulan dari persoalan yang jawaban 'yes'-nya memiliki pembuktian sederhana terhadap fakta bahwa jawabannya memang 'yes'. Untuk menjelaskannya, misal diberikan himpunan dari bilangan integer, yaitu {−7, −3, −2, 5, 8}, dan kita ingin mengetahui apakah jumlah dari beberapa bilangan dari himpunan tersebut memiliki jumlah = 0. Pada contoh ini, jawabannya adalah 'yes', karena himpunan bagian dari bilangan integer {-3, -2, 5} dihubungkan dengan penjumlahan (-3) + (-2) + 5 = 0. Persoalan menentukan apakah jumlah dari himpunan bagian memiliki jumlah = 0 disebut dengan *subset sum problem*.

   Selagi bilangan integer yang diberikan pada algoritma bertambah besar, jumlah dari himpunan bagian juga bertambah secara eksponensial, dan memang pada kenyataannya *subset sum problem* adalah *NP-complete*. Bagaimanapun, perhatikan jika diberikan suatu himpunan bagian tertentu (sering disebut dengan *sertifikat*), kita dapat dengan mudah mengeceknya atau memperlihatkannya (*verify*) apakah jumlah dari himpunan bagiannya = 0 hanya dengan menjumlahkan bilangan integer dari himpunan bagian itu. Jadi jika memang jumlah = 0, himpunan bagian tersebut merupakan bukti (*proof*) atau saksi mata (*witness*) untuk kenyataan bahwa jawabannya adalah 'yes'. Algoritma yang memperlihatkan apakah himpunan bagian yang diberikan memiliki jumlah = 0 disebut *verifier*. Suatu persoalan disebut **NP** jika dan hanya jika terdapat suatu *verifier* untuk persoalan tersebut yang membuthkan waktu pengerjaan dalam orde waktu polinomial, yang merupakan alasan bahwa *subset sum problem* merupakan **NP**.
2. Setiap persoalan pada *NP* dapat direduksi menjadi *C*.

   Persoalan *K* dapat direduksi menjadi *C* jika terdapat waktu-polinomial reduksi, sebuah algoritma deterministik yang mengubah $k \in K$ menjadi $c \in C$, sehingga jawaban *c* adalah YES jika dan hanya jika jawabah *k* adalah YES. Untuk membuktikan bahwa persoalan *NP A* merupakan persoalan *NP-complete* cukup dengan menunjukkan bahwa persoalan *NP-complete* yang telah diketahui direduksi menjadi *A*.

Persoalan yang termasuk *NP-complete* tidak dapat dipecahkan dalam orde waktu polinomial sehingga sulit dipecahkan.

**2.1. Algoritma *Knapsack* Sederhana**

Ide dasar dari algoritma kriptografi *knapsack* adalah mengkodekan pesan sebagai rangkaian solusi dari persoalan *knapsack*. Setiap bobot $w_i$ di dalam persoalan *knapsack* merupakan kunci privat, sedangkan bit-bit plainteks menyatakan $b_i$.

**Contoh 2**. Misalkan $n = 6$ dan $w_1 = 1$, $w_2 = 5$, $w_3 = 6$, $w_4 = 11$, $w_5 = 14$, dan $w_6 = 20$.

Plainteks: 111001010110000000011000

Plainteks dibagi menjadi blok yang panjangnya $n$, kemudian setiap bit di dalam blok dikalikan dengan $w_i$ yang berkorepsonden:

- Blok plainteks ke-1 : 111001
  *Knapsack* : 1, 5, 6, 11, 14, 20
  Kriptogram : (1 x 1) + (1 x 5) + (1 x 6) + (1 x 20) = 32
- Blok plainteks ke-2 : 010110
  *Knapsack* : 1, 5, 6, 11, 14, 20
  Kriptogram : (1 x 5) + (1 x 11) + (1 x 14) = 30
- Blok plainteks ke-3 : 000000

*Knapsack* : 1, 5, 6, 11, 14, 20
Kriptogram : 0

- Blok plainteks ke-4 : 011000
  *Knapsack* : 1, 5, 6, 11, 14, 20
  Kriptogram : (1 x 5) + (1 x 6) = 11

Jadi, cipherteks yang dihasilkan: 32 30 0 11

Sayangnya, algoritma *knapsack* sederhana di atas hanya dapat digunakan untuk enkripsi, tetapi tidak dirancang untuk dekripsi. Misalnya, jika diberikan kriptogram = 32, maka tentukan $b_1$, $b_2$, …, $b_6$ sedemikian sehingga

$$32 = b_1 + 5b_2 + 6b_3 + 11b_4 + 14b_5 + 20b_6 \quad (2)$$

Solusi persamaan (2) ini tidak dapat dipecahkan dalam orde waktu polinomial dengan semakin besarnya *n* (dengan catatan barisan bobot tidak dalam urutan menaik). Namun, hal inilah yang dijadikan sebagai kekuatan algoritma *knapsack*.

### 2.2. *Superincreasing Knapsack*

*Superincreasing knapsack* adalah persoalan *knapsack* yang dapat dipecahkan dalam orde $O(n)$ (polinomial). Ini adalah persoalan *knapsack* yang mudah sehingga tidak disukai untuk dijadikan sebagai algoritma kriptografi yang kuat.

Jika senarai bobot disebut barisan *superincreasing*, maka kita dapat membentuk *superincreasing knapsack*. Barisan *superincreasing* adalah suatu barisan di mana setiap nilai di dalam barisan lebih besar daripada jumlah semua nilai sebelumnya. Misalnya {1, 3, 6, 13, 27, 52} adalah barisan *superincreasing,* tetapi {1, 3, 4, 9, 15, 25} bukan.

Solusi dari *superincreasing knapsack* (yaitu $b_1$, $b_2$, …, $b_n$) mudah dicari sebagai berikut (berarti sama dengan mendekripsikan cipherteks menjadi plainteks semula):

1. Jumlahkan semua bobot di dalam barisan.
2. Bandingkan bobot total dengan bobot terbesar di dalam barisan. Jika bobot terbesar lebih kecil atau sama dengan bobot total, maka ia dimasukkan ke dalam *knapsack*, jika tidak, maka ia tidak dimasukkan.
3. Kurangi bobot total dengan bobot yang telah dimasukkan, kemudian bandingkan bobot total sekarang dengan bobot terbesar selanjutnya. Demikian seterusnya sampai seluruh bobot di dalam barisan selesai dibandingkan.
4. Jika bobot total menjadi nol, maka terdapat solusi persoalan *superincreasing knapsack*, tetapi jika tidak nol, maka tidak ada solusinya.

**Contoh 3**. Misalkan bobot-bobot yang membentuk barisan *superincreasing* adalah {2, 3, 6, 13, 27, 52}, dan diketahui bobot *knapsack* ($M$) = 70. Kita akan mencari $b_1$, $b_2$, …, $b_6$ sedemikian sehingga

$$70 = 2b_1 + 3b_2 + 6b_3 + 13b_4 + 27b_5 + 52b_6$$

Caranya sebagai berikut:

(i) Bandingkan 70 dengan bobot terbesar, yaitu 52. Karena 52 ≤ 70, maka 52 dimasukkan ke dalam *knapsack*.
(ii) Bobot total sekarang menjadi 70 – 52 = 18. Bandingkan 18 dengan bobot terbesar kedua, yaitu 27. Karena 27 > 18, maka 27 tidak dimasukkan ke dalam *knapsack*.
(iii) Bandingkan 18 dengan bobot terbesar berikutnya, yaitu 13. Karena 13 ≤ 18, maka 13 dimasukkan ke dalam *knapsack*.
(iv) Bobot total sekarang menjadi 18 – 13 = 5.
(v) Bandingkan 5 dengan bobot terbesar kedua, yaitu 6. Karena 6 > 5, maka 6 tidak dimasukkan ke dalam *knapsack*.
(vi) Bandingkan 5 dengan bobot terbesar berikutnya, yaitu 3. Karena 3 ≤ 5, maka 3 dimasukkan ke dalam *knapsack*.
(vii) Bobot total sekarang menjadi 5 – 3 = 2.
(viii) Bandingkan 2 dengan bobot terbesar berikutnya, yaitu 2. Karena 2 ≤ 2, maka 2 dimasukkan ke dalam *knapsack*.
(ix) Bobot total sekarang menjadi 2 – 2 = 0. Karena bobot total tersisa = 0, maka solusi persoalan *superincreasing knapsack* ditemukan.

Barisan bobot yang dimasukkan ke dalam *knapsack* adalah {2, 3, – , 13, – , 52} sehingga 70 = (1 x 2) + (1 x 3) + (0 x 6) + (1 x 13) + (0 x 27) + (1 x 52). Dengan kata lain, plainteks dari kriptogram 70 adalah 110101.

## 3. TEORI BILANGAN

### 3.1 Aritmetika Modulo [7]

Aritmetika modulo (*modular arithmetic*) memainkan peranan yang penting dalam komputasi *integer*, khususnya pada aplikasi kriptografi. Operator yang digunakan pada aritmetika modulo adalah **mod**. Operator mod, jika digunakan pada pembagian bilangan bulat, memberikan sisa pembagian. Misalnya 23 dibagi 5 memberikan hasil = 4 dan sisa = 3, sehingga kita tulis 23 mod 5 = 3. Definisi dari operator mod dinyatakan sebagai berikut:

**DEFINISI** Misalkan $a$ adalah bilangan bulat dan $m$ adalah bilangan bulat > 0. Operasi $a$ mod $m$ (dibaca "$a$ modulo $m$") memberikan sisa jika $a$ dibagi dengan $m$. dengan kata lain, $a$ mod $m = r$ sedemikian sehingga $a = mq + r$, dengan $0 \le r < m$.

> Notasi: $a$ mod $m = r$ sedemikian sehingga $a = mq + r$, dengan $0 \le r < m$.

Bilangan $m$ disebut **modulus** atau **modulo**, dan hasil aritmetika modulo $m$ terletak di dalam himpunan {0, 1, 2, …, $m$ – 1}

Jika $a$ mod $m$ = 0, maka dikatakan bahwa $a$ adalah kelipatan dari $m$, yaitu $a$ habis dibagi dengan $m$.

### 3.2. Relatif Prima [7]

**DEFINISI** Dua buah bilagnan bulat $a$ dan $b$ dikatakan relatif prima jika PBB($a$, $b$) = 1

Sebagai contoh, 20 dan 3 relatif prima sebab PBB(20, 3) = 1. Tetapi 20 dan 5 tidak relatif prima sebab PBB(20, 5) = 5 ≠ 1.

Jika $a$ dan $b$ relatif prima, maka dapat ditemukan bilangan bulat $m$ dan $n$ sedemikian sehingga

$$ma + nb = 1$$

**Contoh 4**. Bilangan 20 dan 3 adalah relatif prima karena PBB(20, 3) = 1, atau dapat ditulis

$$2 \cdot 20 + (-13) \cdot 3 = 1$$

dengan $m = 2$ dan $n = -13$.

### 3.3. Balikan Modulo (*Modulo Invers*) [8]

Jika $a$ dan $m$ relatif prima (PBB($a$, $m$) = 1) dan $m > 1$, maka kita dapat menemukan balikan (*invers*) dari $a$ modulo $m$. Balikan dari $a$ modulo $m$ adalah bilangan bulat $a^{-1}$ sedemikian sehingga

$$a\,a^{-1} \equiv 1 \pmod{m}$$

Dari definisi relatif prima diketahui bahwa PBB($a$, $m$) = 1 sehingga terdapat bilangan bulat $p$ dan $q$ sedemikian sehingga

$$pa + qm = 1$$

yang mengimplikasikan bahwa

$$pa + qm \equiv 1 \pmod{m}$$

Karena $qm \equiv 0 \pmod{m}$, maka

$$pa \equiv 1 \pmod{m}$$

**Contoh 5**. Tentukan balikan modulo dari 4 (mod 9)

*Jawab*: Karena PBB(4, 9) = 1, maka balikan dari 4 (mod 9) ada. Dari algoritma Euclidean diperoleh bahwa

$$9 = 2 \cdot 4 + 1$$
$$-2 \cdot 4 + 1 \cdot 9 = 1$$

Dari persamaan terakhir ini kita peroleh -2 adalah balikan dari 4 modulo 9. Periksalah bahwa

$-2 \cdot 4 \equiv 1 \pmod 9$ (9 habis membagi $-2 \cdot 4 - 1 = -9$)

## 5. PENGGUNAAN TEORI BILANGAN PADA SISTEM KRIPTOGRAFI MERKLE-HELLMAN

[2] Kriptografi Merkle-Hellman merupakan sistem kunci nirsimetri, yang berarti bahwa dalam sistem komunikasi diperlukan dua kunci, yaitu kunci publik dan kunci privat. Berbeda dengan RSA, kunci publik digunakan hanya untuk enkripsi, sedangkan kunci privat hanya digunakan untuk dekripsi. Karena itu sistem kriptografi ini tidak dapat dipakai untuk otentikasi dengan kriptografi.

Sistem Merkle-Hellman didasarkan pada *subset sum problem* (kasus khusus dari persoalan *knapsack*). Misal diberikan himpunan bilangan dan suatu bilangan yang merupakan jumlah dari himpunan bagian dari himpunan bilangan tersebut. Pada umumnya, persoalan ini termasuk *NP-complete*. Akan tetapi, jika himpunan bilangan yang digunakan (disebut sebagai *knapsack*) merupakan *superincreasing*, persoalan menjadi 'mudah' dan dapat dipecahkan dalam orde waktu polinomial dengan algoritma Greedy sederhana.

Sudah dijelaskan pada pembahasan sebelumnya bahwa algoritma *superincreasing knapsack* adalah algoritma yang lemah, karena cipherteks dapat didekripsi menjadi plainteksnya secara mudah dalam waktu lanjar ($O(n)$). Sedangkan algoritma *non-superincreasing knapsack* atau *normal knapsack* adalah kelompok algoritma *knapsack* yang sulit (dari segi komputasi) karena membutuhkan waktu dalam orde eksponensial untuk memecahkannya. Namun, Martin Hellman dan Ralph Merkle menemukan cara untuk memodifikasi *superincreasing knapsack* menjadi *non-superincreasing knapsack* dengan menggunakan kunci publik (untuk enkripsi) dan kunci privat (untuk dekripsi). Modifikasi dilakukan dengan menggunakan perhitungan aritmetika modulo.

### 5.1 Penggunaan Aritmetika Modulo dan Relatif Prima pada Pembangkitan Kunci Merkle-Hellman

[2] Pada kriptografi Merkle-Hellman, kunci yang digunakan terdiri dari *knapsack*. Kunci publik merupakan *'hard' knapsack*, sedangkan kunci privat privat merupakan yang 'mudah' (*superincreasing knapsack*), yang dikombinasikan dengan dua bilangan tambahan, yaitu pengali (*multiplier*) dan modulus yang digunakan untuk mengubah *superincreasing knapsack* menjadi *hard knapsack*. Bilangan-bilangan yang sama digunakan untuk mengubah jumlah dari himpunan bagian dari *hard knapsack* menjadi jumlah dari himpunan bagian dari *superincreasing knapsack* yang dapat dipecahkan dalam orde waktu polinomial.

Untuk mengenkripsikan $n$-bit pesan, caranya adalah sebagai berikut:

i. Tentukan barisan *superincreasing* $w = (w_1, w_2, ..., w_n)$ dari bilangan bukan nol.
ii. Pilih salah satu bilangan integer $q$ sehingga memenuhi $q > \sum_{i=1}^{n} w_i$ dan salah satu angka integer bilangan integer $r$ secara acak sehingga PBB($r$, $q$) = 1 ($r$ relatif prima dengan $q$). Bilangan $q$ dipilih dengan cara di atas untuk memastikan keunikan dari chiperteks. Jika bilangan yang digunakan lebih kecil, lebih dari satu plainteks akan dienkripsi menjadi chiperteks yang sama. Sedangkan $r$ harus tidak memiliki persekutuan dengan $q$ karena jika tidak maka balikan modulo dari $r$ mod $q$ tidak dapat ditemukan. Bilangan yang merupakan balikan modulo dari $r$ mod $q$ adalah penting agar memungkinkan dekripsi.
iii. Kemudian hitung barisan $\beta = (\beta_1, \beta_2, ..., \beta_n)$ yang memenuhi $\beta_i \equiv r w_i \bmod q$

Kunci publik adalah $\beta$, sedangkan kunci privat adalah ($w$, $q$, $r$).

**Contoh 6**. Misalkan barisan *superincreasing* adalah {2, 3, 6, 13, 27, 52}, $q$ = 105, dan $r$ = 31. Barisan *non-superincreasing* (normal) *knapsack* dihitung sebagai berikut:

$2 \times 31 \bmod 105 = 62$
$3 \times 31 \bmod 105 = 93$
$6 \times 31 \bmod 105 = 81$
$13 \times 31 \bmod 105 = 88$
$27 \times 31 \bmod 105 = 102$
$52 \times 31 \bmod 105 = 37$

Jadi, kunci publik adalah {62, 93, 81, 88, 102, 37}, sedangkan kunci privat adalah {2, 3, 6, 13, 27, 52, 105, 31}.

### 5.2. Penggunaan Balikan Modulo pada Deskripsi

***Enkripsi [2]***

Terdapat $n$-bit pesan $\alpha = (\alpha_1, \alpha_2, ..., \alpha_n)$ dengan $\alpha_i$ adalah bit ke-$i$ dari pesan dan $\alpha_i \in \{0, 1\}$. Cara untuk mengenkripsi pesan tersebut adalah sebagai berikut:

i. Pilih himpunan bagian dari *normal knapsack* (kunci publik) yang berkorespondensi dengan 1 pada plainteks dan mengabaikan bagian yang berkorespondensi dengan 0 pada plainteks.
ii. Elemen dari himpunan bagian yang telah dipilih dijumlahkan dan hasilnya menjadi chiperteks.

$$c = \sum_{i=1}^{n} \alpha_i \beta_i$$

**Contoh 7**. Misalkan terdapat suatu plainteks plainteks: 011000110101101110 dan kunci publik yang digunakan seperti pada Contoh 5. Plainteks dibagi menjadi blok yang panjangnya 6, kemudian setiap bit di dalam blok dikalikan dengan elemen yang berkorepsponden di dalam kunci publik:

- Blok plainteks ke-1 : 011000
  Kunci publik : 62, 93, 81, 88, 102, 37
  Kriptogram : (1 x 93) + (1 x 81) = 174
- Blok plainteks ke-2 : 110101
  Kunci publik : 62, 93, 81, 88, 102, 37
  Kriptogram : (1 x 62) + (1 x 93) + (1 x 88) + (1 x 37) = 280
- Blok plainteks ke-3 : 101110
  Kunci publik : 62, 93, 81, 88, 102, 37
  Kriptogram : (1 x 62) + (1 x 81) + (1 x 88) + (1 x 102) = 333

Jadi, cipherteks yang dihasilkan : 174, 280, 333

***Dekripsi [2]***

Untuk mendekripsi cipherteks $c$, penerima harus menemukan pesan dalam bentuk $\alpha_i$ sehingga memenuhi $c = \sum_{i=1}^{n} \alpha_i \beta_i$

Ini akan menjadi persoalan yang sulit jika $\beta_i$ merupakan nilai acak karena penerima harus memecahkan permisalan dari permasalahan penjumlahan dari himpunan bagian, yang diketahui merupakan *NP-hard*. Walaupun demikian, nilai $\beta_i$ dipilih sehingga dekripsi mudah dilakukan jika kunci privat ($w$, $q$, $r$) diketahui.

Hal yang penting dari dekripsi adalah menemukan suatu bilangan integer $s$ yang merupakan balikan modulo (*modular inverse*) dari $r$ modulo $q$. Ini berarti $s$ memenuhi persamaan $sr \bmod q = 1$ atau $sr \equiv 1 \pmod q$ atau terdapat bilangan integer $k$ sehingga $sr = kq + 1$. Karena $r$ dipilih sehingga memenuhi persamaan PBB($r$, $q$) = 1, maka $s$ dan $k$ mungkin ditemukan dengan menggunakan perhitungan balikan modulo yang memenuhi $sr \equiv 1 \pmod q$. Kekongruenan ini dapat dihitung dengan cara yang sederhana sebagai berikut:

$sr \equiv 1 \pmod q$
$\Leftrightarrow sr = kq + 1$
$\Leftrightarrow s = (1 + kq)/r$, $k$ sembarang bilangan bulat

Kalikan setiap kriptogram dengan $s$ mod $m$, lalu nyatakan hasil kalinya sebagai penjumlahan elemen-elemen kunci privat untuk memperoleh plainteks dengan menggunakan algoritma pencarian solusi superincreasing knapsack.

**Contoh 8.** Kita akan mendekripsikan cipherteks dari Contoh 7 dengan menggunakan kunci privat {2, 3, 6, 13, 27, 52}. Di sini, $r$ = 31 dan $q$ = 105. Nilai $s$ diperoleh sebagai berikut: $s = (1 + 105k)/31$

Dengan mencoba $k$ = 0, 1, 2, …, maka untuk $k$ = 18 diperoleh $s$ bilangan bulat, yaitu $s = (1 + 105 \times 18)/31 = 61$

Cipherteks dari Contoh 7 adalah 174, 280, 222. Plainteks yang berkoresponden diperoleh kembali sebagai berikut:

174 × 61 mod 105 = 9 = 3 + 6, berkoresponden dengan 011000
280 × 61 mod 105 = 70 = 2 + 3 + 13 + 52, berkoresponden dengan 110101
333 × 61 mod 105 = 48 = 2 + 6 + 13 + 27, berkoresponden dengan 101110

Jadi, plainteks yang dihasilkan kembali adalah: 011000 110101 101110

## 6. SERANGAN PADA SISTEM KNAPSACK [5]

Ketika Ralph Merkle mengajukan sistem di atas pada tahun 1976, dia yakin bahwa sistem tersebut aman, walaupun dalam kasus iterasi tunggal, dan menawarkan $100 untuk siapa saja yang mampu memecahkannya.

Pada tahun 1982, Adi Shamir menemukan serangan pada sistem *Knapsack* iterasi tunggal. Serangan tersebut sedikit terbatas, dan tidak lama setelahnya diumumkan cara untuk mengubah skema umum yang kemudian mencegah penyerangan. Walaupun demikian, Merkle tetap membayar $100 sebagaimana telah dijanjikan. Dan yang lebih penting, ini merupakan terobosan yang sangat penting untuk kehancuran dari sistem *Knapsack*.

Untuk sementara, diasumsikan tidak ada permutasi yang digunakan. Kemudian untuk setiap $i$, berlaku

$$\beta_i \equiv w_i r \bmod q$$

Dengan definisi dari kekongruenan modulo, harus ada bilangan integer sehingga untuk setiap $i$ berlaku

$$s\beta_i - qk_i = w_i$$

dengan $s$ adalah balikan modulo dari $r$ mod $q$.

Kemudian bagi persamaan di atas menjadi

$$\frac{s}{q} - \frac{k_i}{\beta_i} = \frac{w_i}{q\beta_i}$$

Jika $q$ bilangan yang sangat besar, persamaan di sebelah kanan akan menjadi sangat kecil, sehingga setiap persamaan dari komponen $k$ dan $\beta$ dekat dengan $u/m$. Ganti i dengan 1 dan kurangkan dari persamaan sebelumnya, didapatkan

$$\left|\frac{k_i}{\beta_i} - \frac{k_1}{\beta_1}\right| = \left|\frac{w_i}{q\beta_i} - \frac{w_1}{q\beta_1}\right|$$

Karena kedua pembagian di sebelah kanan nilainya positif, dan hasil pengurangannya sangat kecil, persamaan di atas dapat dituliskan

$$\left|\frac{k_i}{\beta_i} - \frac{k_1}{\beta_1}\right| = \frac{w_i}{q\beta_i}$$

Perhatikan bahwa *w* adalah barisan superincreasing, setiap elemennya harus lebih kecil dari setengah bilangan sebelumnya, sehingga untuk setiap *i* berlaku

$$w_i < q2^{i-n}$$

Kemudian dapat juga dinyatakan bahwa

$$\left|\frac{k_i}{\beta_i} - \frac{k_1}{\beta_1}\right| = \frac{2^{i-n}}{\beta_i}$$

Dengan menyusun ulang persamaan di atas didapatkan

$$k_i\beta_1 - k_1\beta_i = \beta_1 2^{i-n}$$

Karena $\beta$ termasuk kunci publik, hanya sedikit dari pertidaksamaan di atas (tiga atau empat) bersifat unik untuk menentukan *k*. Pertidaksamaan ini merupakan salah satu contoh dari *integer programming*, sehingga dengan algoritma *Lenstra integer linear programming* dapat ditemukan nilai *k* dengan cepat. Dan jika nilai *k* sudah diketahui, mudah untuk memecahkan sistem.

Andaikan dilakukan permutasi terhadap $\beta$ sebelum mempublikasikannya. Karena hanya dibutuhkan 3 atau 4 dari elemen pertama *k*, kita dapat mencoba semua kombinasi kemungkinan yang hanya sampai bilangan kubik atau quartik.

## 6. KESIMPULAN

Teori bilangan memiliki peranan yang sangat luas dalam bidang kriptografi. Salah satunya adalah dalam persoalan *knapsack*. Teori bilangan yang dipakai antara lain aritmetika modulo, balikan modulo, dan relatif prima.

Knapsack sendiri pada awalnya banyak digunakan pada bidang keamanan. Hal ini dikarenakan persoalan ini termasuk ke dalam *NP-complete*. Namun sudah banyak serangan yang ditemukan untuk memecahkan persoalan ini. Dan sekarang, sistem apapun yang menggunakan perkalian modular untuk menyembunyikan *superincreasing knapsack* dapat dipecahkan dengan efisien.

Walaupun demikian, seperti yang sudah dilihat, ini bukan satu-satunya pilihan untuk hanya menggunakan *knapsack* dalam bidang kriptografi. Masih banyak algoritma-algoritma mangkus yang dapat digunakan untuk menjamin sekuritas dari suatu penjagaan.

## DAFTAR REFERENSI


[1] http://en.wikipedia.org/wiki/Knapsack_problem
Waktu akses: 24 Desember 2008 pukul 20.10 WIB

[2] http://en.wikipedia.org/wiki/Merkle-Hellman
Waktu akses: 26 Desember 2008 pukul 16.01 WIB

[3] http://en.wikipedia.org/wiki/NP-complete
Waktu akses: 24 Desember 2008 pukul 20.10 WIB

[4] http://en.wikipedia.org/wiki/NP_(complexity)
Waktu akses: 26 Desember 2008 pukul 15.26 WIB

[5] http://www.derf.net/knapsack/#Attacks
Waktu akses: 28 Desember 2008 pukul 14.07 WIB

[6] Ir. Rinaldi Munir, M.T., *Algoritma Knapsack*, Bandung, hal. 2-10
http://www.informatika.org/~rinaldi/Kriptografi/2006-2007/Algoritma%20Knapsack.doc
Waktu akses: 21 Desember 2008 pukul 10.02 WIB

[7] Ir. Rinaldi Munir, M.T., *Diktat Kuliah IF2091 Struktur Diskrit*, Bandung, 2003, hal. V-13

[8] idem, hal. V-16

[9] www.informatika.org/~rinaldi/Buku/Kriptografi/Bab-1_Pengantar%20Kriptografi.pdf
Waktu akses: 21 Desember 2008 pukul 09.50 WIB