

## ① 部分和問題

与えられた  $n$  個の整数  $V_0, V_1, \dots, V_{n-1}$  からある値  $W$  を構成する部分和を見つける問題。

例)

$V = (1, 3, 7, 15, 40)$ ,  $W = 23$  のとき,  
 $1 + 7 + 15 (= 23)$  で構成できる。

部分和問題は NP 完全である。

多段式時間で答えを見つけることができない、  
 $\Rightarrow$  れが大きいと現実的な時間で  
 解くのが困難

れがそこまで大きな場合は  
 動的計画法(DP)で解くことができる。

## ② ナップサック(knapsack)暗号

一方向性密数に部分和問題を利用。

入力  $\rightarrow$  出力の計算: 簡単

出力  $\rightarrow$  入力の計算: 難しい

### ③ 木密鍵

$$a = \{(a_0, a_1, \dots, a_{n-1}) \mid a_i \in \mathbb{Z}\}$$

$$p, q \in \mathbb{Z}$$

(ただし,  $p$  と  $q$  は互いに素)  
 $q > \sum_{i=0}^{n-1} a_i$

### ④ 公開鍵

$$b = \{(b_0, b_1, \dots, b_{n-1}) \mid b_i = p \cdot a_i \bmod q\}$$

## ① 平文

$$m = \{(m_0, m_1, \dots, m_{n-1}) \mid m_i \in \{0, 1\}\}$$

### ② 暗号化

$$c = \sum_{i=0}^{n-1} l_i \cdot m_i$$

$\Rightarrow$  このままで復号できないので、トップドアが必要

Merkle-Hellman (MH) ナップサック暗号と呼ばれる  
 暗号では、トップドアに超増加数列を使う。  
超増加数列

$$\alpha_{x+1} > \sum_{i=0}^x a_i \text{ を満たす数列 } (a_0, a_1, \dots, a_{n-1})$$

自身の数字が、自身より前にある全要素の合計  
 よりも大きくなる数列

超増加数列を用いた部分和問題は  
 簡単に解けることが分かっている。  
 $\Rightarrow$  解説壳ができる

### ③ 復号

まず、 $S = c \cdot p^{-1} \bmod q$  を計算する。

$$\begin{aligned} S &= \left\{ \sum_{i=0}^{n-1} (p \cdot a_i \bmod q) \cdot m_i \right\} \cdot p^{-1} \bmod q \\ &= \sum_{i=0}^{n-1} a_i \cdot m_i \bmod q \end{aligned}$$

①  $x = n-1$  とする。

②  $S \geq a_x$  であれば,  $m_x = 1$  とし,  $S = S - a_x$   
 $S < a_x$  であれば,  $m_x = 0$  とする。

③  $x = x-1$

④  $x \geq 0$  なら, ② に戻る

## ① 格子

$\mathbb{R}^m$  を  $m$  次元ベクトル空間とする。  
 $w$  本の  $m$  次元ベクトルを  $b_0, b_1, \dots, b_{w-1} \in \mathbb{R}^m$  として、  
 線形独立であるとする。

行ベクトル  
[ ... ]

\* 線形独立とは

$a_1 v_1 + a_2 v_2 + \dots + a_n v_n = 0$   
 の角字が  $a_1 = a_2 = \dots = a_n = 0$  だけであるとき、  
 $v_1, v_2, \dots, v_n$  は線形独立（一次独立）であるという。  
 1つのベクトルを残りのベクトルで表せない)

このとき、基底  $\{b_0, \dots, b_{w-1}\}$  で張られる格子を

$L(b_0, \dots, b_{w-1}) = \left\{ \sum_{i=0}^{w-1} x_i b_i \mid x_0, \dots, x_w \in \mathbb{Z} \right\}$

で定義する。

基底の整数倍の和

$w$  を格子の次元と呼ぶ。

$w = m$  のとき、格子は full-rank であるという。

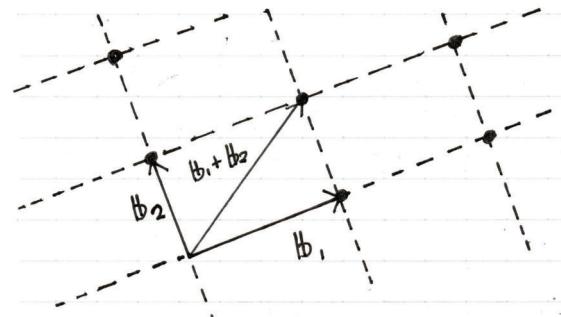
$w$  本の基底  $b_0, \dots, b_{w-1}$  を縦に並べた行列を  
 基底行列と呼ぶ。

$$B = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{w-1} \end{bmatrix} = \begin{bmatrix} b_{01} & b_{02} & \cdots & b_{0m} \\ b_{11} & b_{12} & \cdots & b_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ b_{w1} & b_{w2} & \cdots & b_{wm} \end{bmatrix} \quad w \times m \text{ 行列}$$

など、 $B$  で規定される格子は  
 $L(B) = \{x B \mid x \in \mathbb{Z}^w\}$   
 と書くことができる。

$$\left( \begin{aligned} x B &= x_0 b_0 + x_1 b_1 + \cdots + x_{w-1} b_{w-1} \\ &= \sum_{i=0}^{w-1} x_i b_i \end{aligned} \right)$$

格子の定義と同じ



$$\Leftrightarrow L(b_1, b_2)$$

同じ格子を張る基底は複数存在する。

ユニモジュラ行列  $U$  ( $| \det(U) | = 1$  かつ 整数行列)  
 に対して、 $UB$  で与えられる基底行列に文する  
 基底も同じ格子を張る  
 つまり、 $L(B) = L(UB)$  が成立する。

行列

$[0 \ 1], [1 \ 0]$  が張る格子と

$[5 \ 2], [2 \ 1]$  が張る格子同じ。

## ④ SVP (Shortest Vector Problem)

格子基底が与えられたときに、その格子上で一番原点に近い格子点(ただし原点を除く)を求める問題を最短ベクトル問題(SVP)と呼ぶ。

## ⑤ CVP (Closest Vector Problem)

基底と一点  $x \in \mathbb{R}^m$  が与えられたときに、その格子上で一番  $x$  に近い格子点を求める問題を最近ベクトル問題(CVP)と呼ぶ。

SVPとCVPはNP困難であることが知られている。

NP完全と同等以上

⇒ 格子の次元が大きいと、  
現実的な時間で解くのは困難。

## ⑥ 格子簡約アルゴリズム

同じ格子を長さ「できるだけ直交した基底」を求めるアルゴリズム。

⇒ 結果的に直交した基底が求まる。

## ⑦ LLL

格子簡約アルゴリズムの一重。  
ここで短いベクトルを多回見つけて求められる。

⇒ 近似的に最短ベクトルが求められる  
(次元が小さいと実際に最短ベクトルが得られることがある)

- 番目のベクトルが、出力のベクトルの中で一番短い。

## ⑥ CLOS法

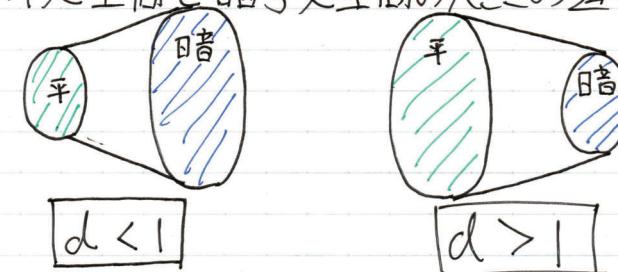
低密度攻撃とロギばれるナップサック暗号に対する攻撃手法の一重。  
密度  $d$  を次のように定義する。

$$d = \frac{\text{(平文のビット数)}}{\text{(暗号文の平均ビット数)}} = \frac{n}{\log_2 \max(l_i)}$$

最大の  $l_i$

公用鍵の要素数

$d$  は平文空間と暗号文空間の大きさの違いを表している。



CLOS法では、 $d < 0.94$  のときに暗号を解読できる。

平文を  $M = [m_0 \ m_1 \ \dots \ m_{n-1}]$ ,

公用鍵を  $\mathbf{h} = [h_0 \ h_1 \ \dots \ h_{n-1}]$ ,

暗号文を  $c$  とおく。

そして、次のような基底行列  $B$  を考える。

$$B = \begin{bmatrix} h_0 & 2 & 0 & \cdots & 0 \\ h_1 & 0 & 2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_{n-1} & 0 & 0 & \cdots & 2 \\ -c & -1 & -1 & \cdots & -1 \end{bmatrix} = \begin{bmatrix} h_0 & 2I \\ -c & -1I \end{bmatrix}$$

転置(行と列を交換する)

行ベクトルが基底になっている

この格子に左から  $A = [M \ I]$  をかけると\*

$$AB = \left[ \left( \sum_{i=0}^{n-1} l_{ii} M_i \right) - c \quad 2m_0-1 \quad 2m_1-1 \cdots 2m_{n-1}-1 \right]$$

$c = \sum_{i=0}^{n-1} l_{ii} M_i$  であったので、

$$AB = [0 \quad 2m_0-1 \quad 2m_1-1 \cdots 2m_{n-1}-1]$$

$M_i$  はビットなので、0 または 1 を取る。

よって、

$$2m_i-1 = \begin{cases} 1 & (m_i = 1) \\ -1 & (m_i = 0) \end{cases}$$

となり、

$$AB = [0 \quad 1 \text{ or } -1 \quad 1 \text{ or } -1 \cdots 1 \text{ or } -1]$$

ベクトル  $AB$  の長さ (ノルム)

$\|AB\| = \sqrt{c}$  であるから、十分短いベクトルだと言える。

また、 $AB$  は  $L(B)$  上のベクトルであるから、 $B$  に LLL を適用して得られた短い基底の中に含まれると考えられる。

( $d < 0.94$  なら、確実に含まれる)

まとめると、

$B$  を作る  $\rightarrow B$  に LLL を適用する  $\rightarrow$  その中から  $AB$  を見つけ、

1 なら  $m_i = 1$

-1 なら  $m_i = 0$

とする。

\*について

この操作は、基底をそれぞれ  $M_0, M_1, \dots, M_{n-1}$  1 倍して足し合わせることに等しい。

$$\begin{aligned} & \left[ \begin{matrix} l_{00} & 2 & 0 & \cdots & 0 \end{matrix} \right] \times M_0 \\ & \left[ \begin{matrix} l_{11} & 0 & 2 & \cdots & 0 \end{matrix} \right] \times M_1 \\ & \vdots \\ & \left[ \begin{matrix} l_{n-1} & 0 & 0 & \cdots & 2 \end{matrix} \right] \times M_{n-1} \\ & + \underline{\left[ \begin{matrix} -c & -1 & -1 & \cdots & -1 \end{matrix} \right] \times 1} \\ & \left[ l_{00}M_0 + l_{11}M_1 + \cdots + l_{n-1}M_{n-1} - c \quad 2m_0-1 \quad 2m_1-1 \cdots 2m_{n-1}-1 \right] \end{aligned}$$