Lattice & CTF

先備知識

- 一點線性代數
- 一點數論
- RSA

需要的工具

- SageMath

盡量避免一些數學上的細節，而只大致上說明 Lattice 是如何在 CTF 中運用的

盡量以直觀的方式去說明為主

基本定義

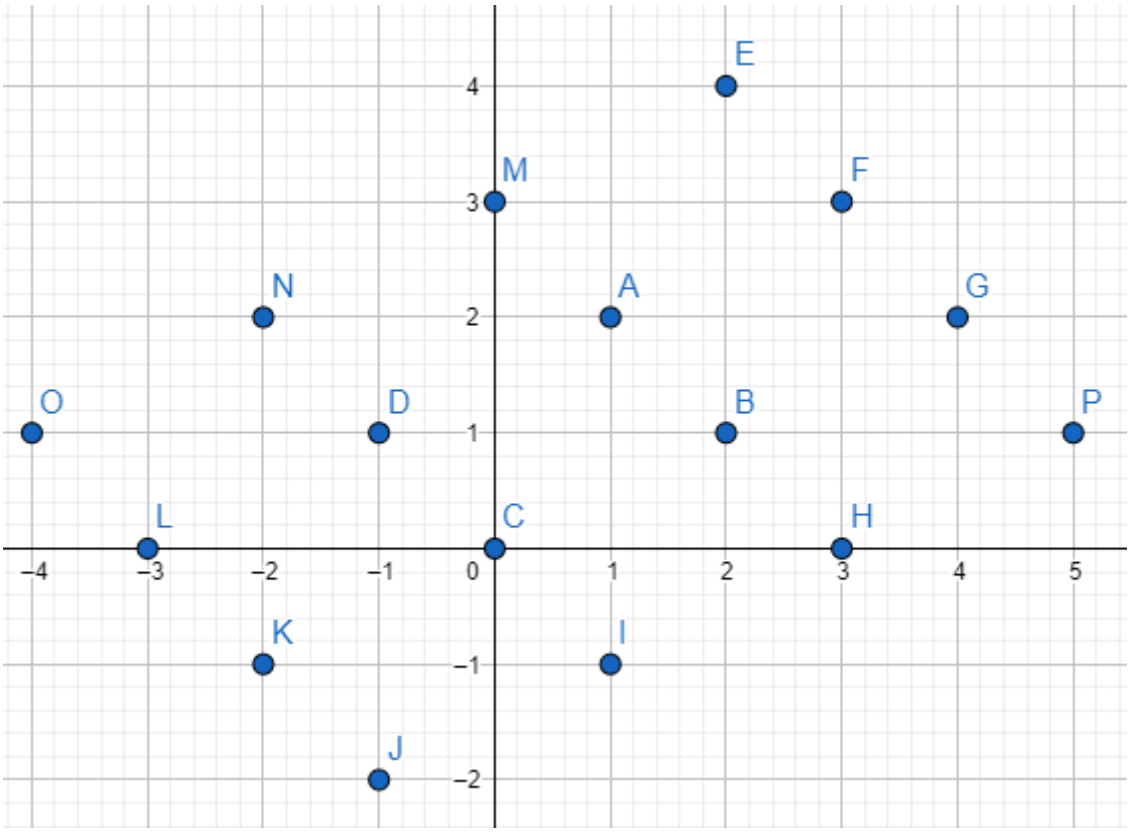
Basis (基底)是一些向量的集合，通常以矩陣的 row vector 表示

$$B = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$$

而 Lattice 是 Basis 中的向量的**整數倍**線性組合出來的點集

$$L = \left\{ \sum_{i=1}^{n} a_i v_i | a_i \in \mathbb{Z} \right\}$$

例如此圖是以 B 所展開的 Lattice $L(B)$ 的一部份



Basis 不一定要是方陣，所以 volume 就定義為

$$\mathrm{vol}(L) = \sqrt{\det(B^T B)} = |\det(B)|$$

這個在計算一些上界時有用

Shortest Vector Problem (SVP)

給予一個 B，找出 $L(B)$ 中**最短的非零向量**

Closest Vector Problem (CVP)

給予一個 B 和一個 $v \in \mathrm{span}(B)$，找出 $L(B)$ 中**距離 v 最近的向量**

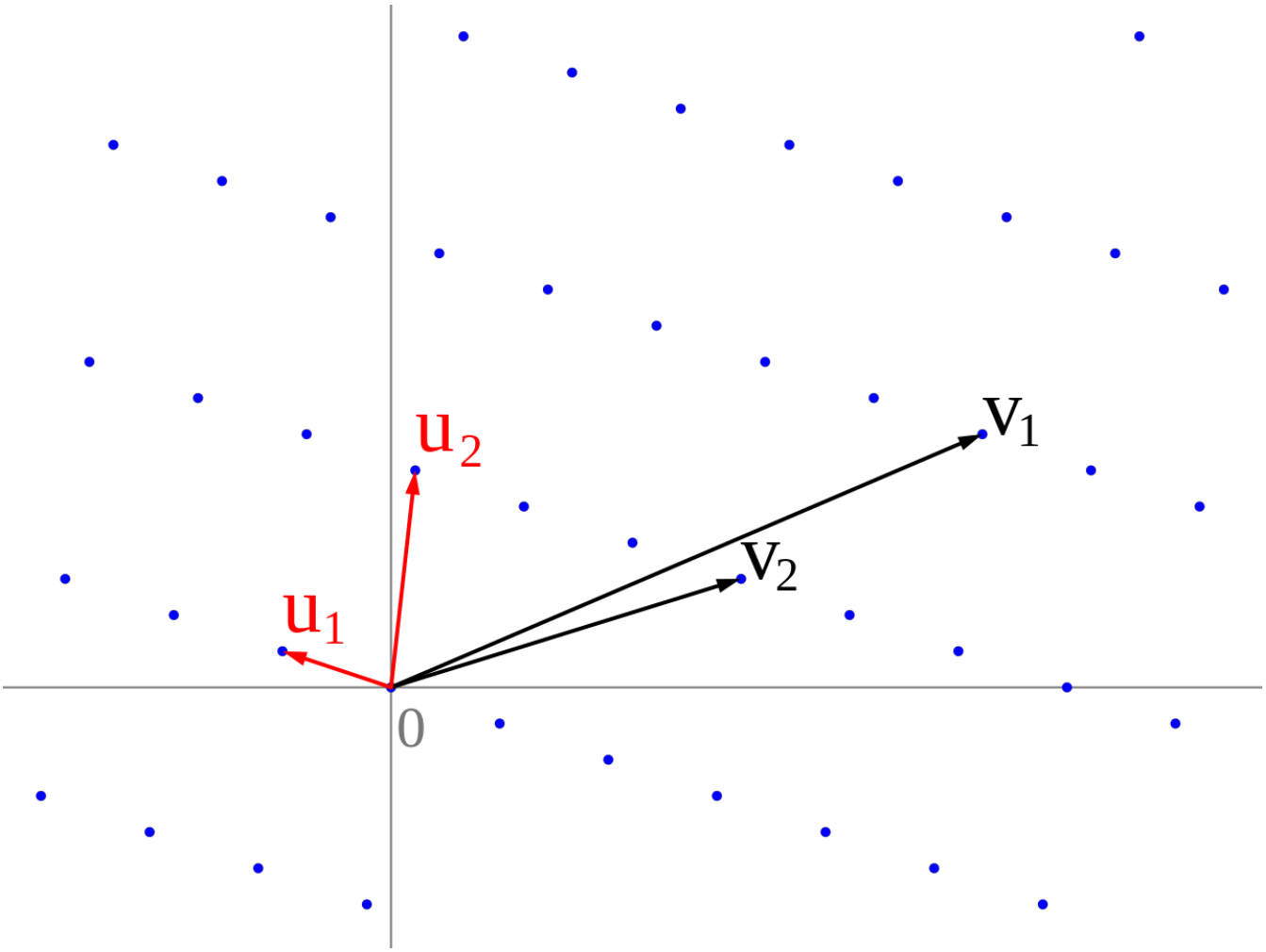
SVP 算是 CVP 的一個特例 $(v = \vec{0})$

雖然 SVP 和 CVP 兩個問題都是 **NP-Hard** 的問題

但是有演算法能求這兩個問題的**近似解**

Lattice Reduction

給予一個 B，找到一組**短**且近乎垂直的 B'



Minkowski's theorem 給出了**最短非零向量**的上界為

$$||x||_2 \leq \sqrt{n}, |\operatorname{vol}(L)|^{1/n}$$

二維 Lattice

Lagrange-Gauss algorithm

通用: LLL

Lenstra–Lenstra–Lovász lattice basis reduction algorithm

LLL 有個額外的參數 $\delta \in (0.25, 1)$

當 $\delta = 1$ 的時候不一定能在多項式時間內完成

得到的 Basis 叫 δ-LLL-Reduced Basis

LLL-Reduced 的 B 中最短向量 b_1 符合

$$||b_1|| \leq (2/\sqrt{4\delta - 1})^{n-1} \cdot \lambda_1(L)$$
$$||b_1|| \leq (2/\sqrt{4\delta - 1})^{(n-1)/2} \cdot |\det(L)|^{1/n}$$

不過實際上對於隨機的 Basis 來說，平均狀況下 LLL 可以達到

$$1.02^n |\det(L)|^{1/n}$$

LLL on the Average

這兩個演算法可以先當成是個 Blackbox，只要記得:

$$B \xrightarrow{LatticeReduction} B'$$

在適當條件下可以得到一個**短**的 Basis B' 即可

範例一

RSA 中有個 Wiener Attack 可以在 $d < \frac{1}{3}n^{\frac{1}{4}}$ 的情況可以還原出 d

而這個範例是在 $e \approx n, d < \frac{1}{\sqrt{5}}n^{\frac{1}{4}}$ 的情況可以還原出 d 的攻擊

Cryptanalysis of RSA and Its Variants 5.1.2.1

註: crypto 中的 $\approx$ 通常指的是量級

例如 $e \approx 2^{1022}, n \approx 2^{1024}$ 的時候就可能說是 $e \approx n$

$$ed \equiv 1 \pmod{\varphi(n)}$$
$$ed = 1 + k\varphi(n) = 1 + k(n - s)$$
$$ed - kn = 1 - ks$$

其中的 $s = p + q - 1, k \approx d$

此時對於這個 Basis 來說:

$$B = \begin{bmatrix} e & \sqrt{n} \\ n & 0 \end{bmatrix}$$

可以看出 $v = (1 - ks, d\sqrt{n})$ 在 Lattice 裡面。

此時 Lattice 中最短向量

$$\lambda_1 \leq \sqrt{2}|\operatorname{vol}(L)|^{1/2} = \sqrt{2}n^{3/4}$$

假設 $p+q<\sqrt{3n}$ 和 $k<d$，則有

$$\begin{aligned}|v|^2 &= (1-ks)^2+(d\sqrt{n})^2\\ &\leq k^2s^2+d^2n\\ &\leq 9nk^2+d^2n\\ &\leq 10nd^2\end{aligned}$$

若是 $|v|=d\sqrt{10n}<\sqrt{2}n^{3/4}$，即 $d<\frac{1}{\sqrt{5}}n^{1/4}$

那麼 v 有機會是 L 中的最短向量

如果 v 或是 $-v$ 是 L 中的最短向量，使用 Lagrange-Gauss algorithm 可得 $v=(l,d\sqrt{n})$，也就得到了 d

---

在 CTF 中重要的是要學會怎麼構造 Lattice Basis

簡而言之，Lattice 有用的時候是在你的問題可以化成**向量的整系數線性組合**，且目標向量是個**短**向量的時候很有用

範例二

Knapsack cryptosystem 是個比 RSA 更早的一個公鑰加密系統，它的核心是利用 Subset sum problem 作為 trapdoor 提供安全性的

它的金鑰生成方式不是重點，重點在於加密方式上

Public key 是一個數列

$$B=(b_1,\cdots,b_n)$$

訊息以二進位的 bits 表示

$$m=(m_1,\cdots,m_n), m_i\in 0,1$$

加密就很單純的將 bits 和 Public key 內積:

$$C=B\cdot m=\sum_{i=1}^{n}b_im_i$$

給予 C 和 B 求 m 的這個問題就是 Subset sum problem，是一個 **NP-Complete** 的問題

不過這個問題在某些情況下是可以用 Lattice Reduction 解決的

定義密度為:

$$d(a) = \frac{n}{\log_2(\max a_i)}$$

目前最佳的情況是已知 $d < 0.9408$ 的時候可以用 Lattice 解決這個問題

一個比較容易想到的 Lattice 可以構造為下:

$$B = \begin{bmatrix} 1 & & & & b_1 \\ & 1 & & & b_2 \\ & & \ddots & & \vdots \\ & & & 1 & b_n \\ 0 & 0 & \cdots & 0 & -C \end{bmatrix}$$

這個 Lattice 中包含了 $v = (m, 0)$ 這個向量，且它的長度比其他許多向量都要短，所以 LLL 之後有機會可以找到

一個可改進的方向是把最後一個 row 的 0 換成 $-\frac{1}{2}$，這樣目標向量會更短

另一個方向是把最後的 column 乘上某個常數，這樣 $\mathrm{vol}(L)$ 會變大，但是目標向量不會改變

$$B = \begin{bmatrix} 1 & & & & Nb_1 \\ & 1 & & & Nb_2 \\ & & \ddots & & \vdots \\ & & & 1 & Nb_n \\ -\frac{1}{2} & -\frac{1}{2} & \cdots & -\frac{1}{2} & -NC \end{bmatrix}$$

根據這篇，取 $N > \frac{\sqrt{n}}{2}$ 會比較好

Babai CVP

這個是一個利用 Lattice Reduction 之後做一些調整去**逼近** CVP 問題解的一個演算法

```
def Babai_closest_vector(B, target):
    # Babai's Nearest Plane algorithm
    M = B.LLL()
    G = M.gram_schmidt()[0]
    small = target
    for i in reversed(range(M.nrows())):
        c = ((small * G[i]) / (G[i] * G[i])).round()
        small -= M[i] * c
    return target - small
```

這個是在 Sage 中這個演算法的使用方式

一樣，建議可以先當一個 Blackbox

給予一個 B 和目標向量 v，兩個丟進去那個函數之後就會得到一個近似的 CVP 的解

```
B = # ...
v = # ...
ans = Babai_closest_vector(B, v)
```

範例三

有個問題叫做 Hidden Number Problem，定義為給予下方的 oracle 函數

$$f(x) = \mathrm{MSB}_k(ag^x \bmod p)$$

要利用這個函數求出 $a \in [1, p)$ 的值

也可以理解為有個向量 $\vec{t} = (t_1, \cdots, t_n)$ ，以及 $\vec{u} = a\vec{t} \bmod p$

如何在只有 $\vec{u}$ 的 MSB 以及 $\vec{t}$ 的情況下求 a 的問題

一個做法是利用下方的 Lattice:

$$B = \begin{bmatrix} p & & & 0 \\ & \ddots & & \vdots \\ & & p & 0 \\ t_1 & \cdots & t_n & 1 \end{bmatrix}$$

可知目標向量 $\vec{v} = (u_1, \cdots, u_n, a) \in L(B)$

只是 v 中的 a 是未知數，所以沒辦法直接用 Babai CVP

一個方法是微調一下變成

$$B = \begin{bmatrix} p & & & 0 \\ & \ddots & & \vdots \\ & & p & 0 \\ t_1 & \cdots & t_n & \frac{1}{p} \end{bmatrix}$$

這樣目標向量是 $\vec{v} = (u_1, \cdots, u_n, \frac{a}{p})$ ，其中的 $\frac{a}{p}$ 可以估計為 0.5

這樣使用 Babai CVP 去找出接近 $\vec{v} = (u_1, \cdots, u_n, \frac{a}{p})$ 就有機會找到 $\vec{v}$

這樣就能得到我們的目標 a

這個方法能否找到 $\vec{v}$ 和 k 與 n 有關，當 k 越多的時候自然越容易找到

如果在 k 比較小的時候也可以利用提升 n 來解決

這是因為輸出的向量 v 符合 $|v-t| \leq \frac{1}{4}\sum_{i=1}^{n}||b_i||$

當 k 越小的時候誤差 $|v-t|$ 會變大，所以可以透過提升 n 把容許的誤差提高

Coppersmith method

Coppersmith method 是個可以找到多項式的 **small roots** 的方法

它的原理也是和 Lattice Reduction 有關

假設有個整數多項式 $f(x)$ 有一個根 x_0 符合 $f(x_0) \equiv 0 \pmod{N}$

當 $|x_0| < N^{1/d}, d = \deg f(x)$ ，Coppersmith method 可以求出 x_0

它的原理大致上是利用 LLL 去找另一個多項式 $q(x)$ 在**整數**上和 $f(x) \equiv 0 \pmod{N}$ 有相同的根 x_0

詳細原理推薦觀看 Kuruwa 大神的說明

範例四

假設今天有有個 RSA 加密的 flag，n 有 1024 bits 且 $e = 3$

flag format 已知為:

FLAG{????????}

設未知部分的值為 x，則 $m = a + bx$

以 RSA 的加密方式可知 $m^e \equiv (a+bx)^3 \equiv c \pmod{n}$

所以可得 $f(x) = (a+bx)^3 - c$ 這個多項式

$f(x) \equiv 0 \pmod{n}$ 有一根 x_0 為 ???? 的部分

假設 ???? 夠短，例如只有 40 bytes，則

$$|x_0| < 2^{320} = 2^{960/3} < n^{1/3} = n^{1/d}$$

所以用 Coppersmith 可得 x_0

實際使用的話 Sage 本身就有個 `small_roots` 可以使用

它支援單變數的 Coppersmith

Coppersmith method - Extra

其實 Coppersmith method 還有其他的用法，這邊會簡單介紹一下

這是可以用在**未知 modulus** 的狀況

已知 N 有一個未知因數 $b < N^\beta$，且 $f(x) \equiv 0 \pmod b$ 有一根 x_0

當 $|x_0| < N^{\beta^2/d}$ 的時候也可以求出 x_0

另一個是**多變數**的狀況，以雙變數為例

$f(x,y) \equiv 0 \pmod N$ 有一根 (x_0, y_0)

當 $|x_0 y_0| < N^{2/3d}$ 時也可求出根

一個 n 變數的多項式的大致上界為

$$|\prod_{i=1}^{n} x_i| < N^{\frac{2}{(n+1)d}}$$

其實我沒很確定這個的正確性...

前者 Sage 本身就有支援，但後者只能自己實作

個人推薦直接使用 defund/coppersmith 的 script

範例五

RSA 的 $n = pq$，其中 p 和 q 的 MSB 已知一部份

例如 1024 bits 的 n，其中 512 bits 的 p, q 的前 300 bits 已知

設 b_p, b_q 分別為 p, q 的前 300 bits，可寫出多項式:

$$f(x,y) = (2^{212} b_p + x)(2^{212} b_q + y)$$

能找到一組小的根 $|x_0 y_0| < 2^{424} < n^{1/2}$ 符合 $f(x_0, y_0) \equiv 0 \pmod n$

另一個做法是用未知 modulus 的做法:

$$f(x) = 2^{212}b_p + x$$

則 $f(x) \equiv 0 \pmod{p}$，$p < n^{0.5}$ 有一根

$$|x_0| < 2^{212} < n^{0.5^2/1}$$

範例六

Boneh and Durfee Attack

在 $d < n^{0.292}$ 的時候可以還原 private key

$$ed \equiv 1 \pmod{\varphi(n)}$$
$$ed = 1 + k\varphi(n) = 1 + k(n - s)$$

其中的 $s = p + q - 1, k \approx d$

這邊看起來和範例一很像，不過這次我們兩邊 $\bmod e$

$$1 + k(n - s) \equiv 0 \pmod{e}$$

因為 $e \approx n$、$s < 3\sqrt{n}$ 以及 $k \approx d$，所以 (k, s) 是組 small roots

---

其他相關的用途

- (EC)DSA biased nonce
- Truncated LCG
- RSA (some cases)
- Inequality solving
- ...

END

相關連結

- Coppersmith 的上界(最後一頁)
- Sage Coppersmith
- Inequality Solving with CVP