🔒 Cryptography In CTF

TeamH4C | 박현

$ whoami

😀 이름 : 박현

👨‍👨‍👦 소속 : TeamH4C

💻 분야 : Pwn, Crypto, Problem Solving

📷 @__gosegu__

$ whoami

⚔️ CTF

Codegate 2022 Junior Final (고양이가 세상을 구한다)

WACON 2022 Junior Final (이세계아이들)

Whitehat Contest 2021 Junior Final (에밀리아)

The Hacking Championship Junior 2021 장려상 (씹덕팀명멈춰!)

BISC CTF 2022 crypto 출제자

ETC

백준 solved.ac Platinum V

🔒 What is Cryptography? (in CTF)

cryptocurrency != cryptography

원하지 않는 대상이 중요한 메시지를 읽지 못 하도록 **암호화**하는 기술을 연구하는 분야

caeser cipher, stream cipher, block cipher, RSA, Diffie-hellman, ...

암호화된 플래그를 **취약점**을 이용해 복호화 한다

Cryptography Category

LLL

RSA

Symmetric ciphers

Diffie-hellman

hash functions

Elliptic curves

MATH

⋮

LLL

solve equation on $\mathbb{R}$: newton-rapshon method, gronber basis, ...

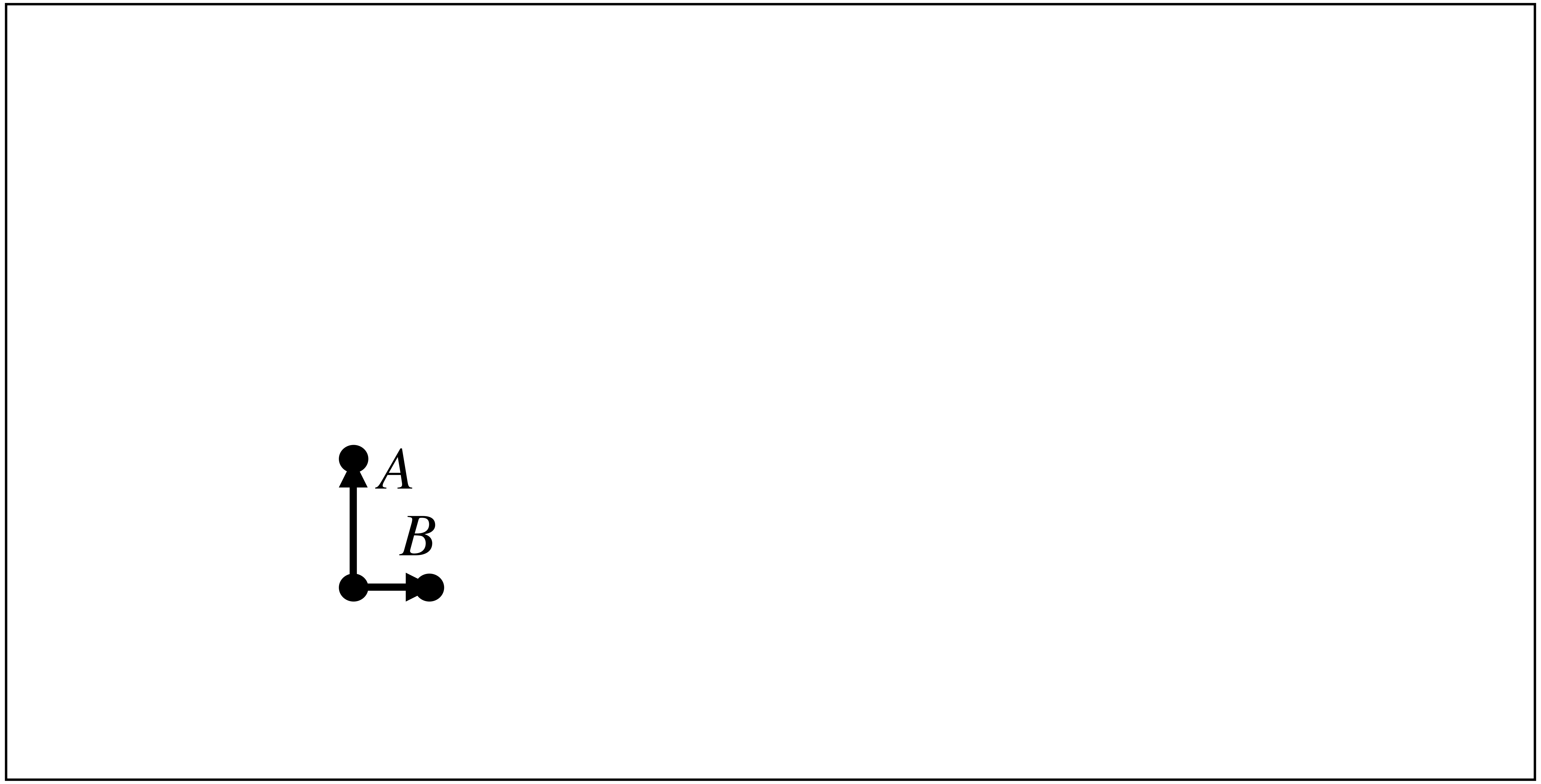
solve equation on $\mathbb{Z}$: universal tool?

$$\Big\downarrow$$

문제를 lattice 위에서 풀어보자!

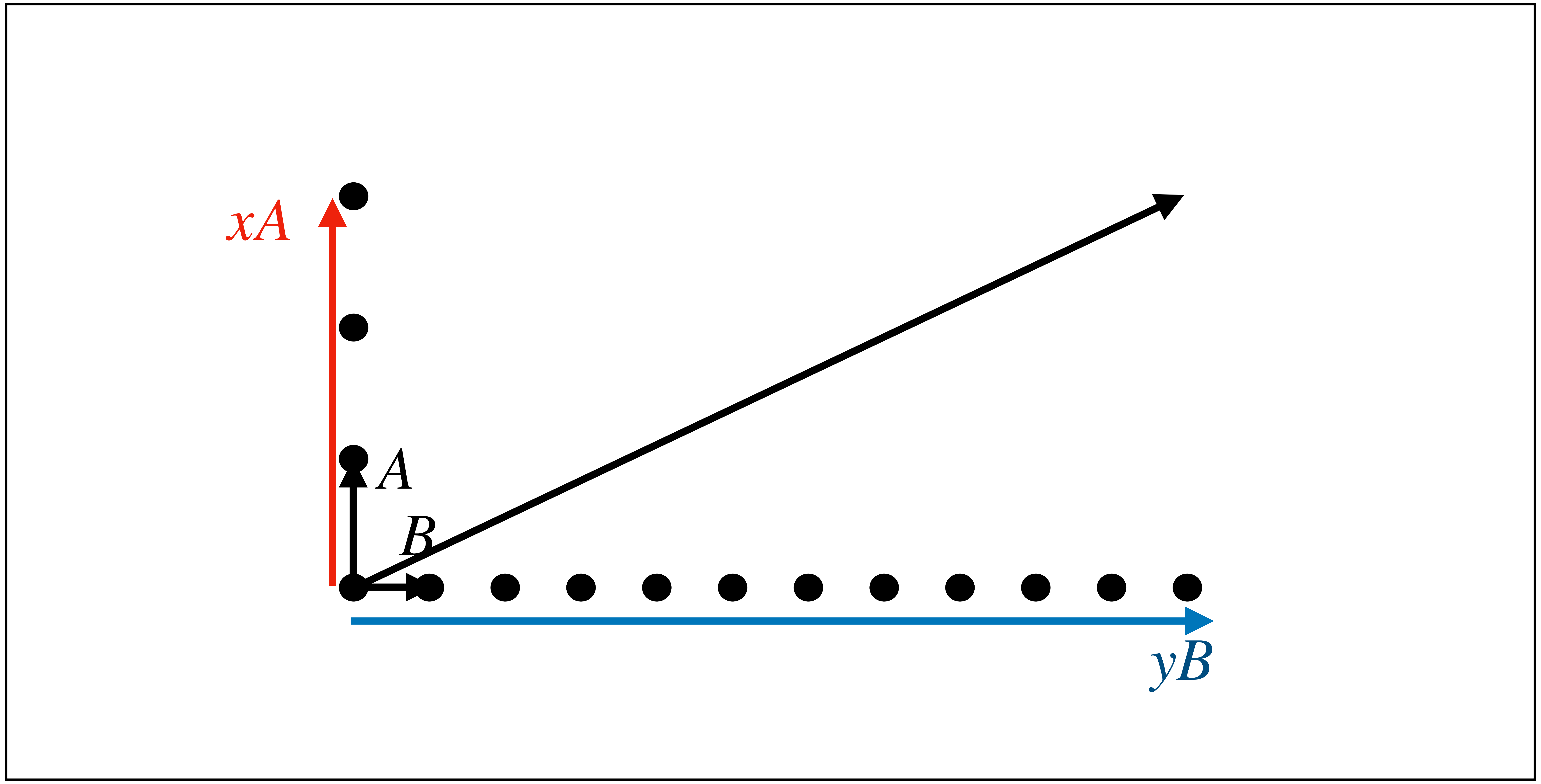
LLL

What is lattice?



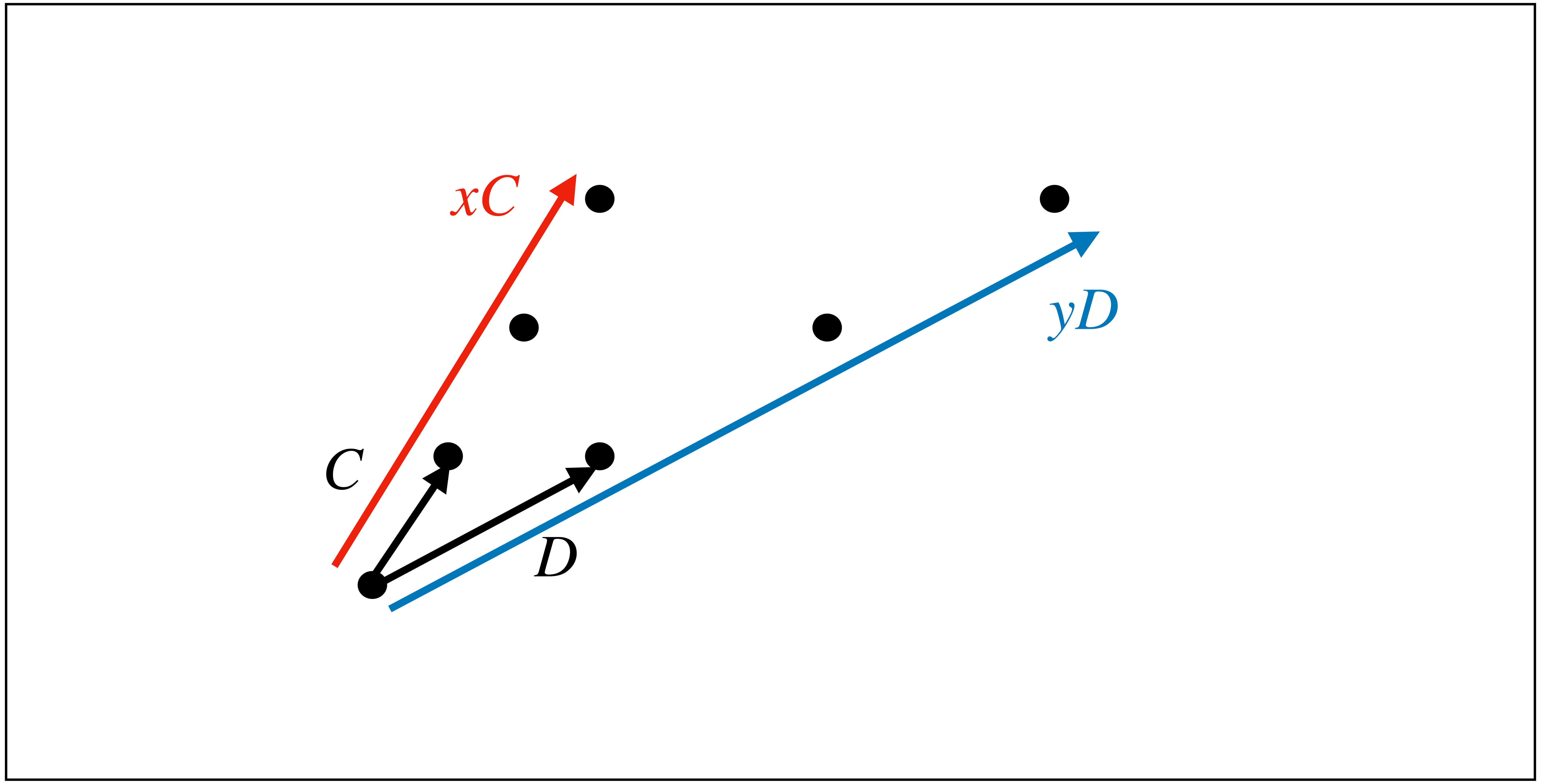
LLL

What is lattice?



LLL

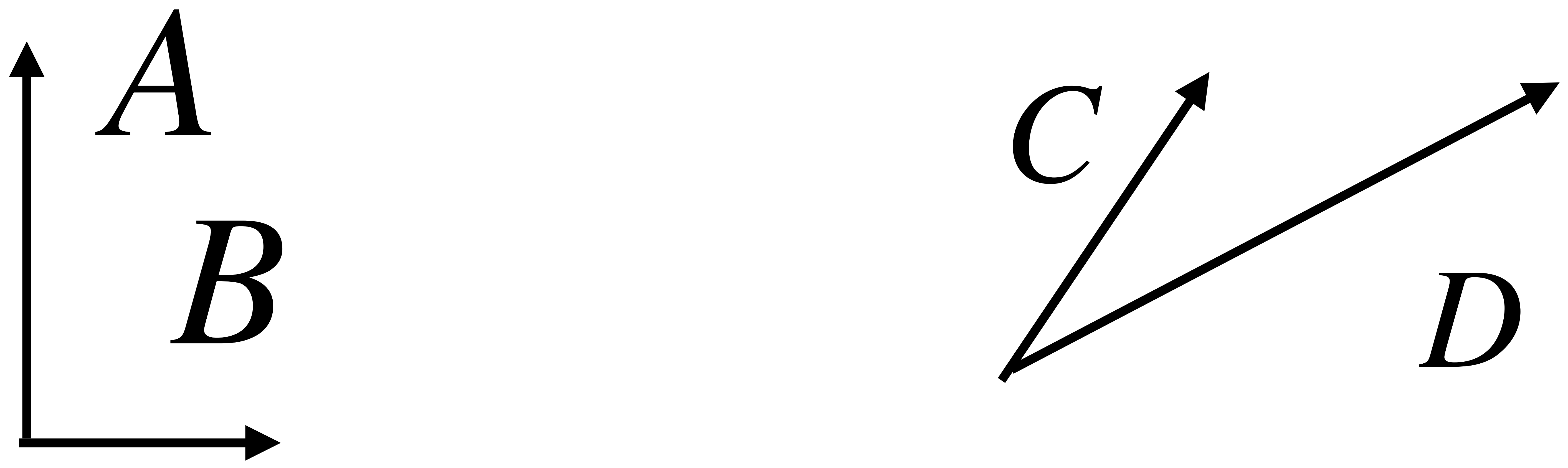
What is lattice?



LLL

What is lattice?

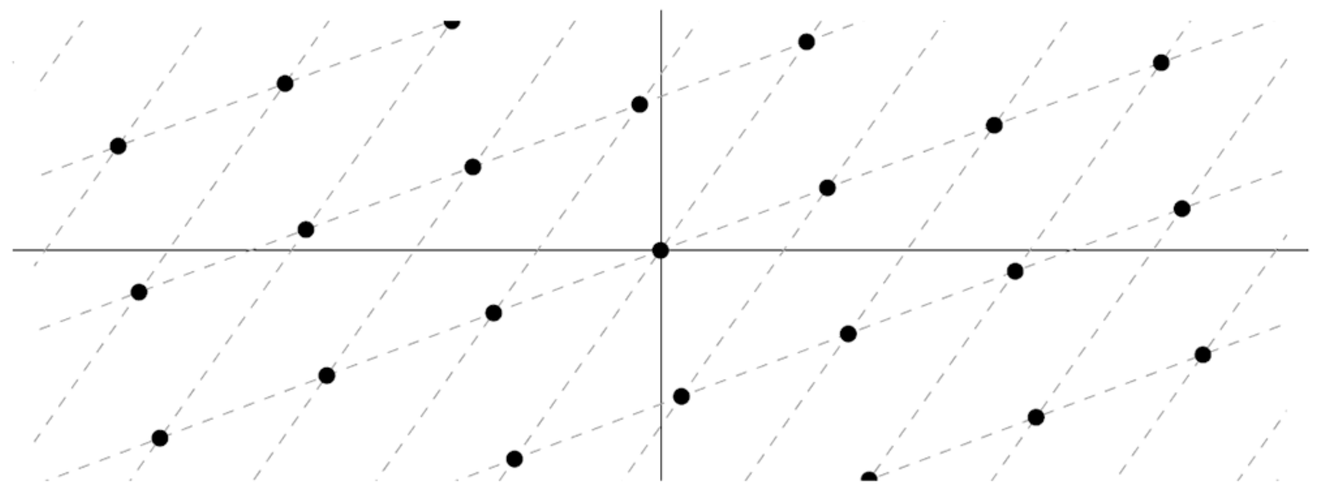
"lattice basis vector"

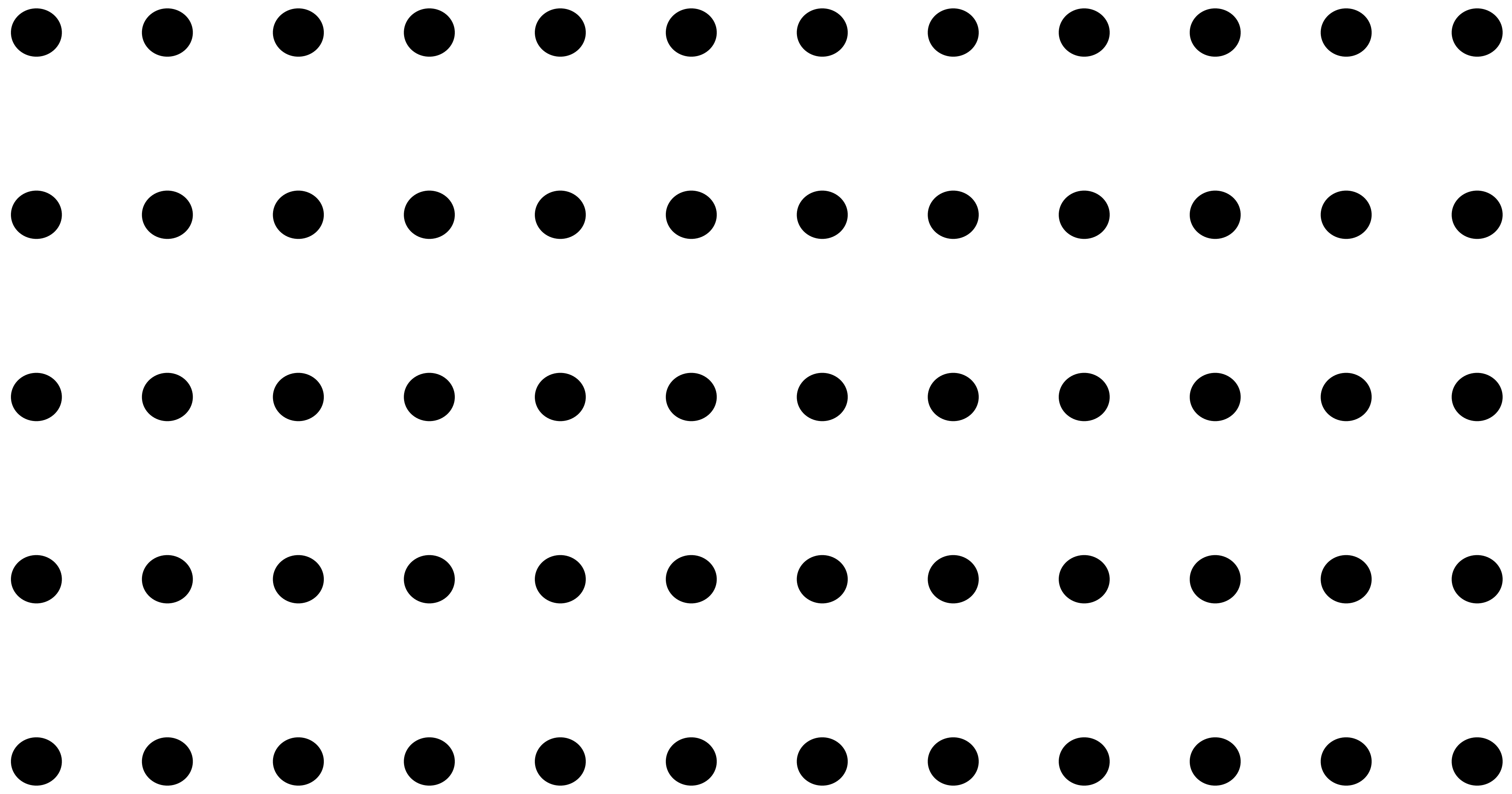


LLL

What is lattice?

= set of all integer linear combinations of vectors from a lattice basis vector

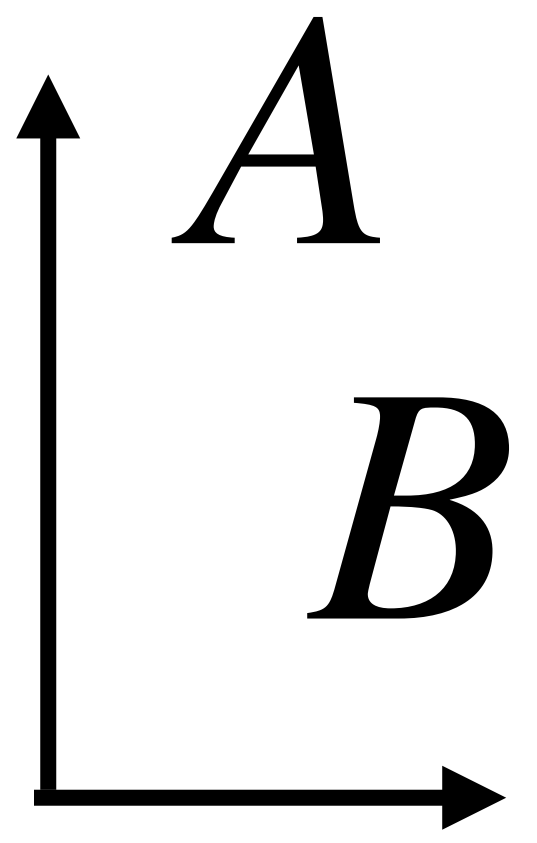


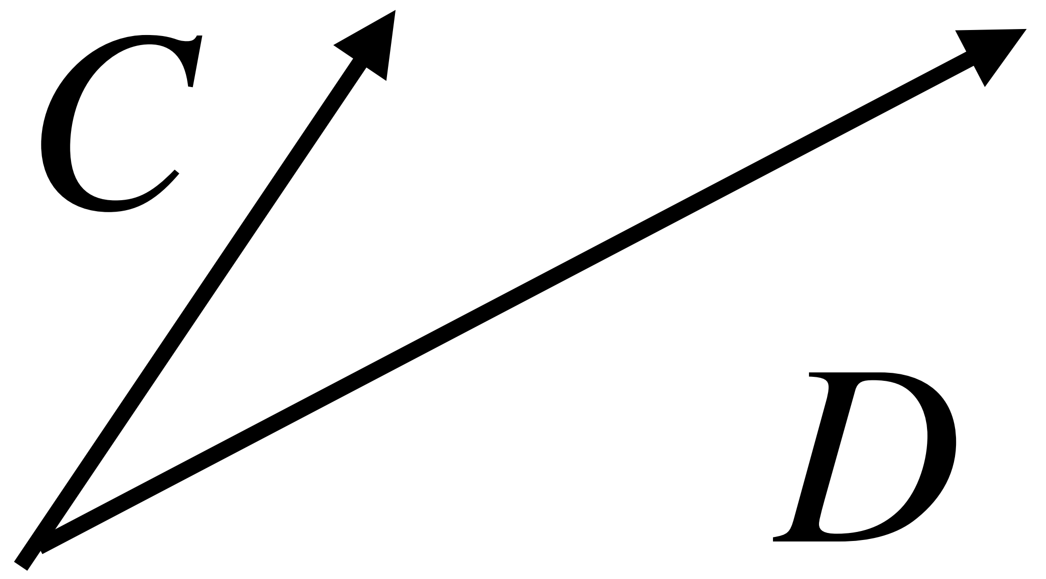


LLL

SVP (shortest vector problem)

who is shortest vector?

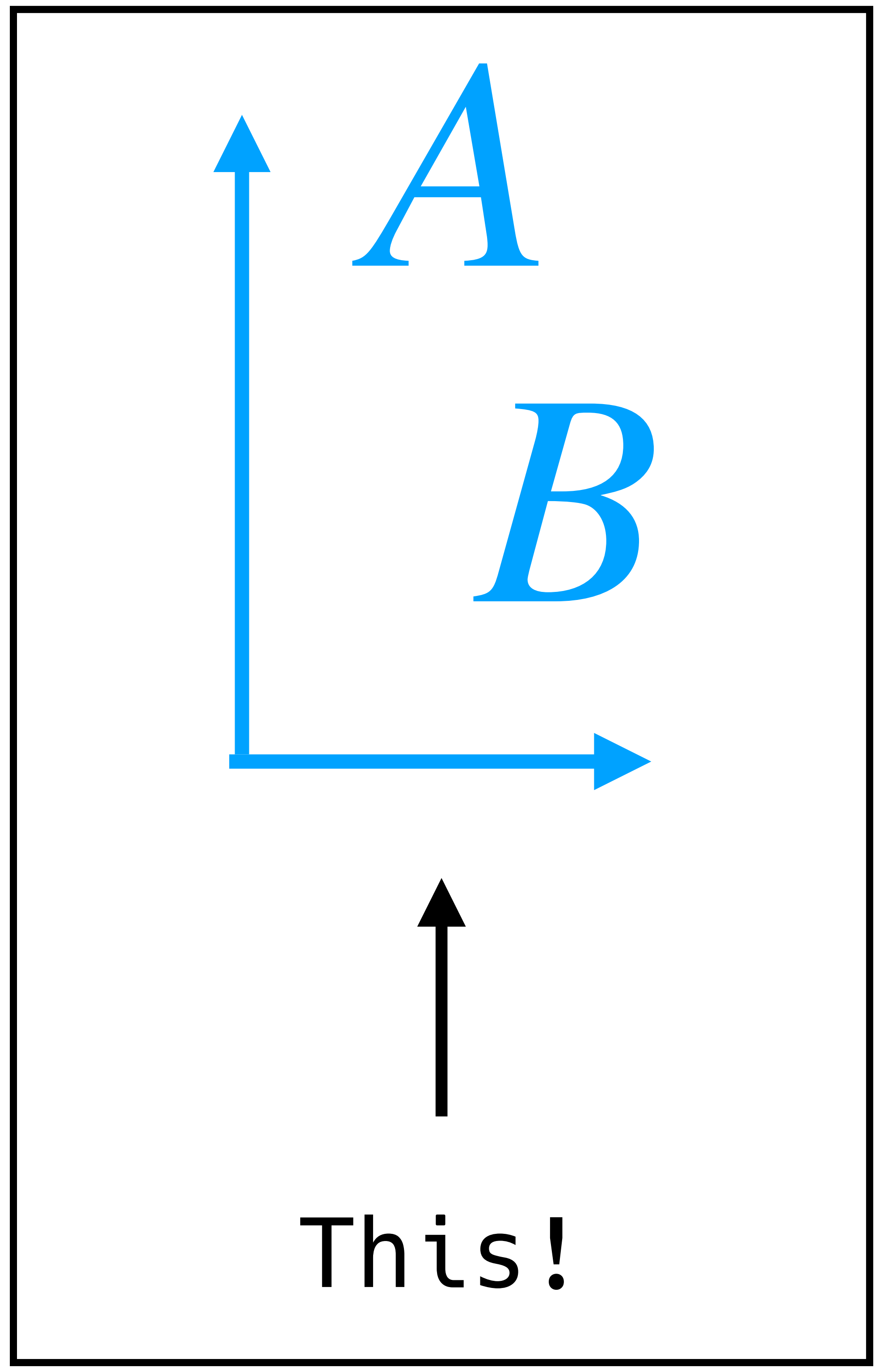




LLL

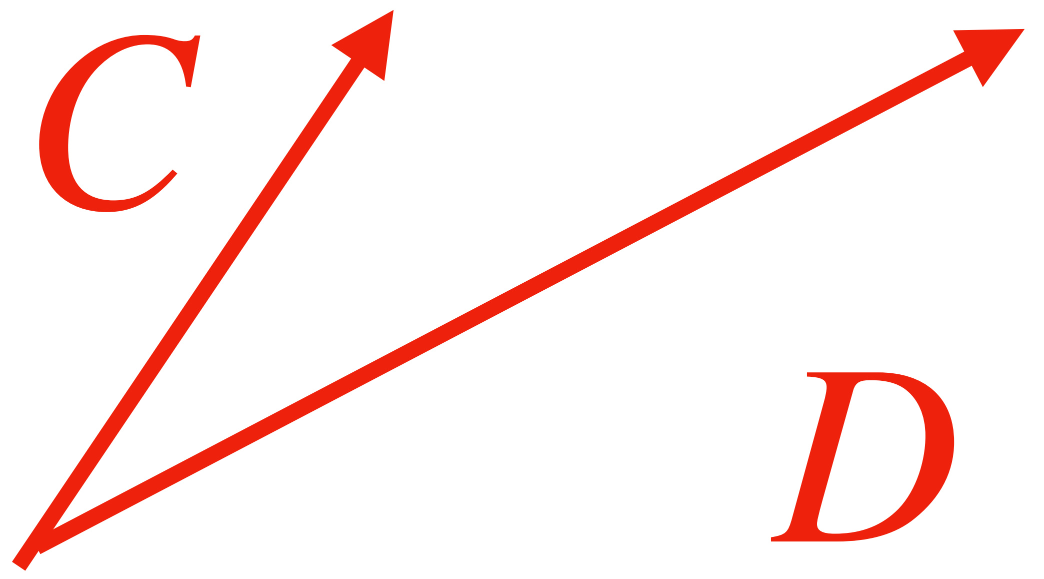
SVP (shortest vector problem)

who is shortest vector?



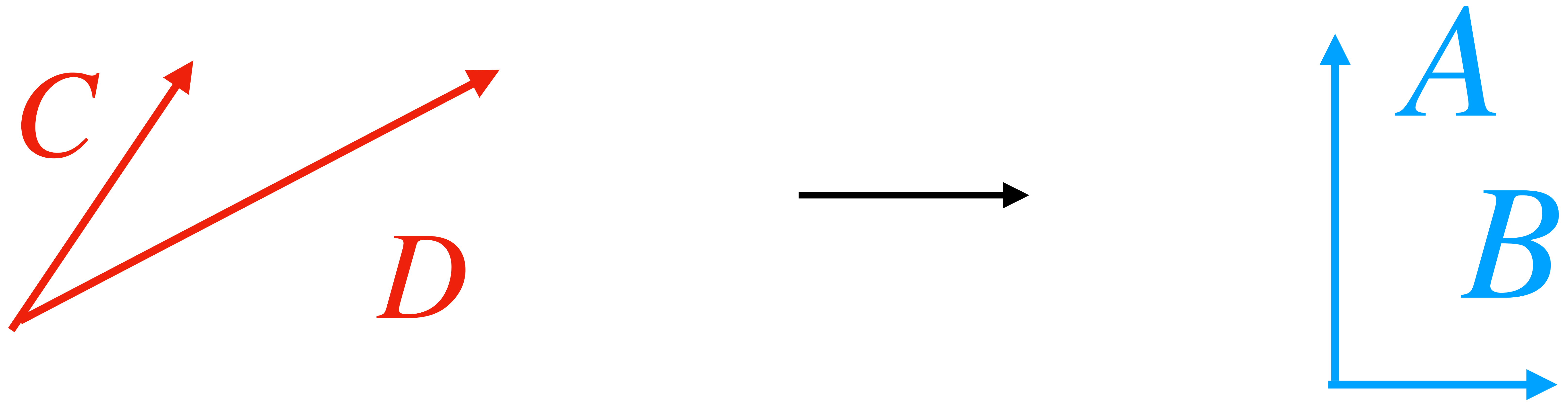
short

orthogonal



LLL

LLL algorithm is convert vector to "shortest vector"



LLL

CTF Progress

Math Problem -> SVP or CVP -> doing BKZ, LLL, ... -> solve

LLL

Let's go to the CTF!

How to use LLL in CTF?

Integer Problems

$a_1, a_2, \cdots, a_n \in \mathbb{R}$에 대해서 다음을 만족하는 비자명한 해 $x_1, x_2, \cdots, x_n \in \mathbb{Z}$ 를 구하라.

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = 0$$

Subset sum Problem

집합 $A \in \mathbb{Z}$와 자연수 M이 주어졌을 때, $\sum S = M$을 만족시키는 A의 부분집합 S를 구하라.

$$A = \{a_1, a_2, \ldots, a_n\}$$

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n - M = 0 \quad x_i = 1 \ or \ 0$$

$$S = \{a_i | x_i = 1\}$$

How to use LLL in CTF?

💡 **Idea**

Integer Problems -> SVP!

How to use LLL in CTF?

solve subset sum problem using SVP

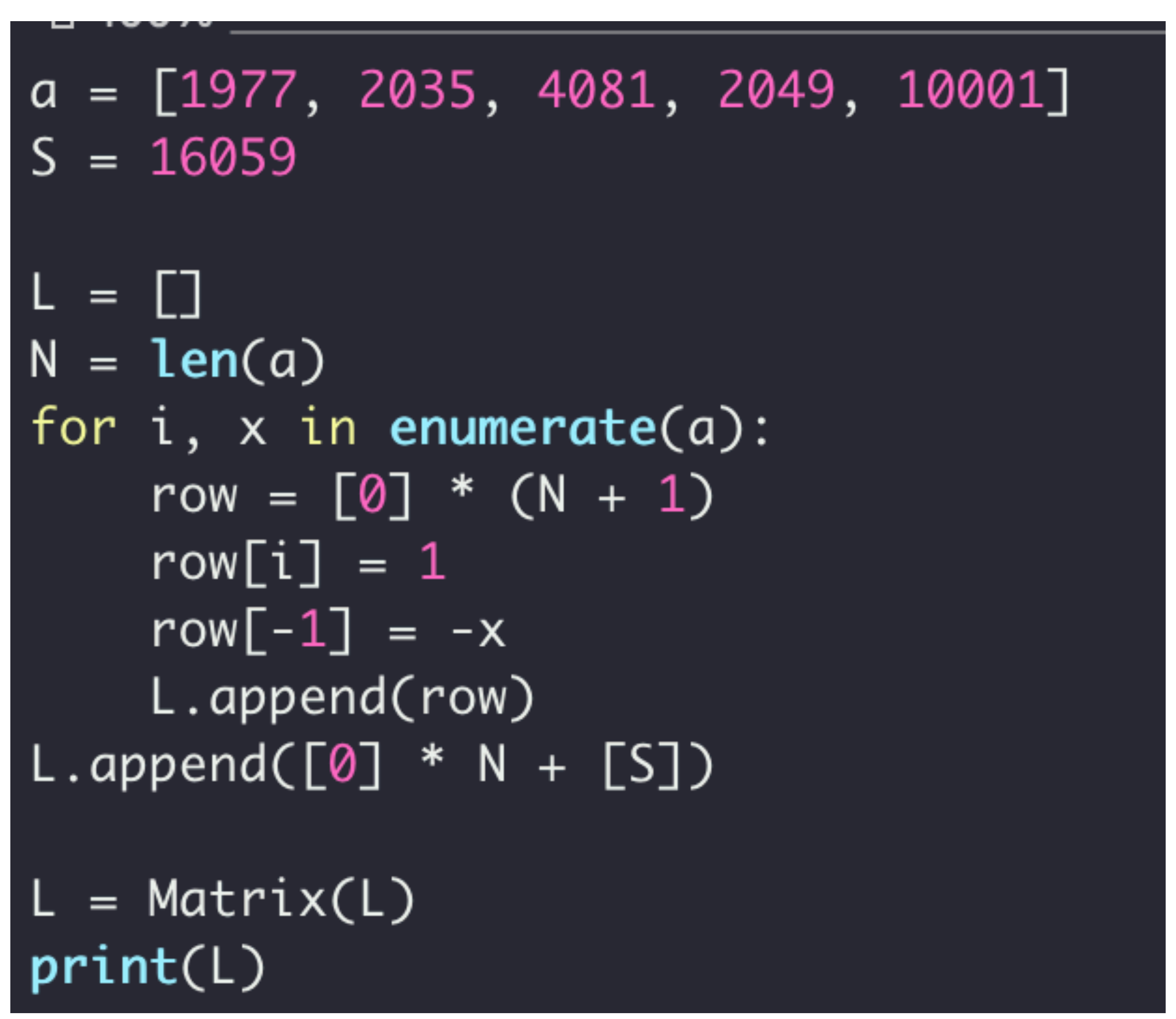
$$L = \begin{bmatrix} 1 & 0 & \cdots 0 & -a_1 \\ 0 & 1 & \cdots 0 & -a_2 \\ & & \ddots & \vdots \\ 0 & 0 & \cdots 1 & -a_n \\ 0 & 0 & \cdots 0 & M \end{bmatrix}$$

shortest vector!

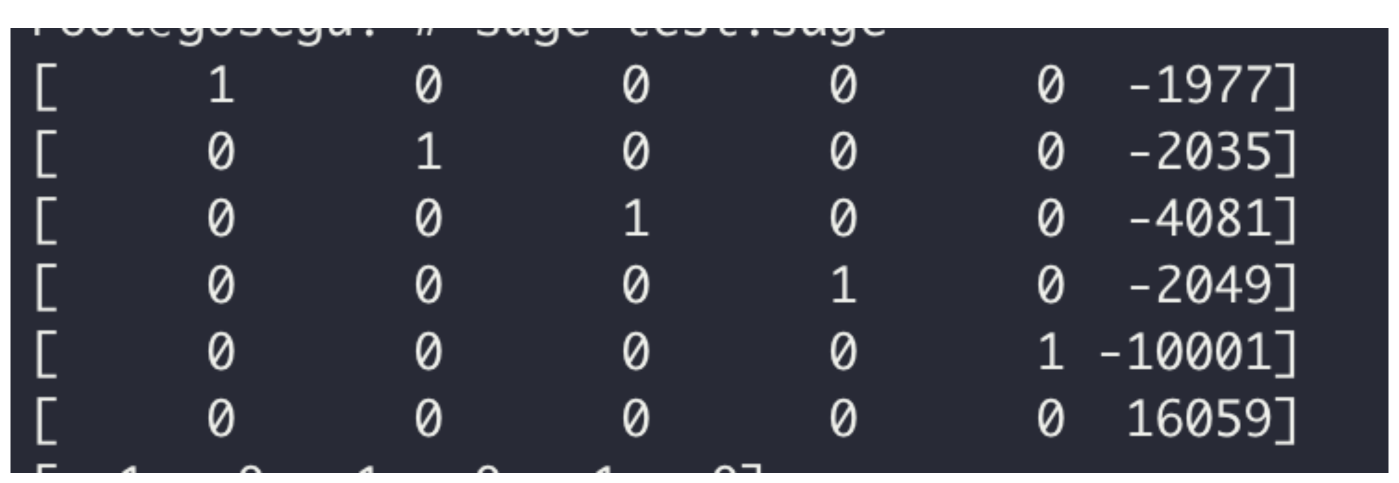
$$x_1 b_1 + \cdots + x_n b_n = (x_1, \cdots, x_n, M - \sum x_i a_i) = \boxed{(x_1, \cdots, x_n, 0)}$$

How to use LLL in CTF?

solve subset sum problem using SVP



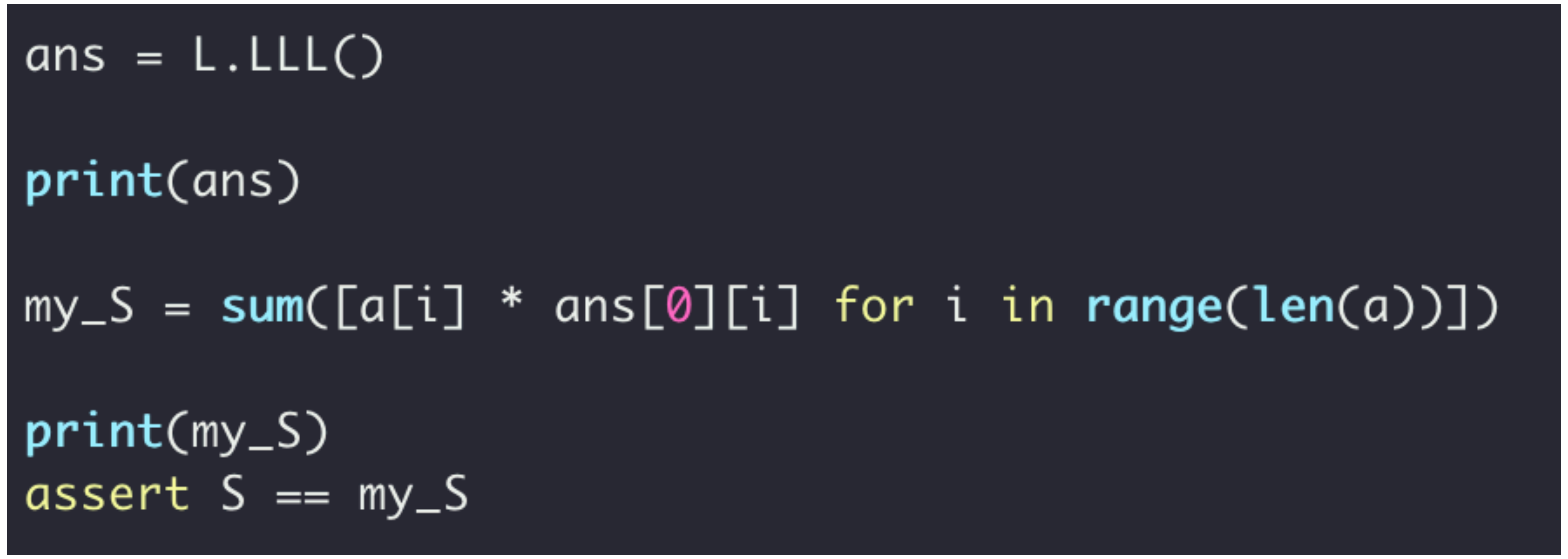
```
a = [1977, 2035, 4081, 2049, 10001]
S = 16059

L = []
N = len(a)
for i, x in enumerate(a):
    row = [0] * (N + 1)
    row[i] = 1
    row[-1] = -x
    L.append(row)
L.append([0] * N + [S])

L = Matrix(L)
print(L)
```



```
[     1      0      0      0      0  -1977]
[     0      1      0      0      0  -2035]
[     0      0      1      0      0  -4081]
[     0      0      0      1      0  -2049]
[     0      0      0      0      1 -10001]
[     0      0      0      0      0  16059]
```

How to use LLL in CTF?

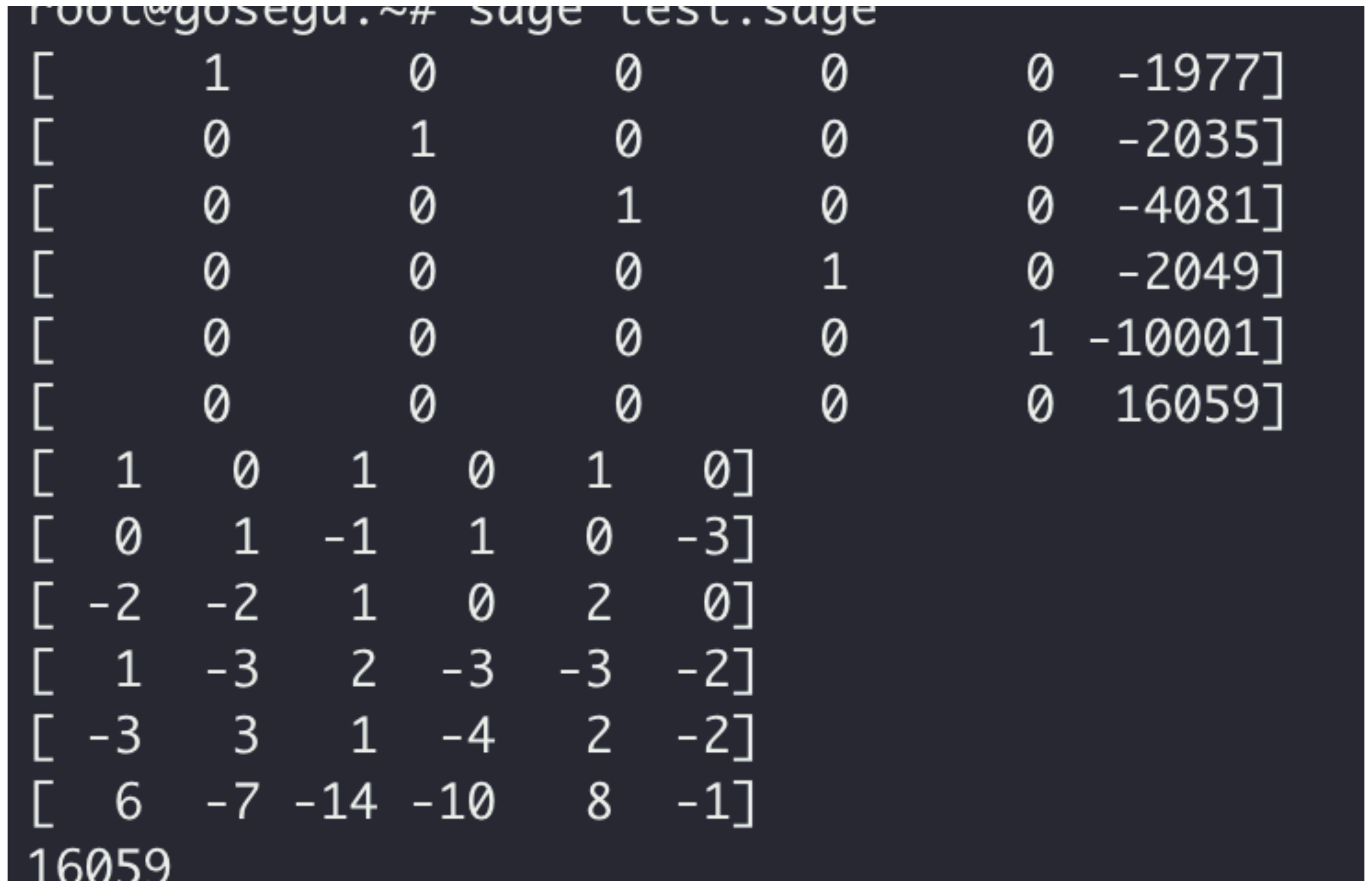
solve subset sum problem using SVP



```
ans = L.LLL()

print(ans)

my_S = sum([a[i] * ans[0][i] for i in range(len(a))])

print(my_S)
assert S == my_S
```

How to use LLL in CTF?

solve subset sum problem using SVP



```
root@gosegu:~# sage test.sage
[     1      0      0      0      0  -1977]
[     0      1      0      0      0  -2035]
[     0      0      1      0      0  -4081]
[     0      0      0      1      0  -2049]
[     0      0      0      0      1 -10001]
[     0      0      0      0      0  16059]
[  1   0   1   0   1   0]
[  0   1  -1   1   0  -3]
[ -2  -2   1   0   2   0]
[  1  -3   2  -3  -3  -2]
[ -3   3   1  -4   2  -2]
[  6  -7 -14 -10   8  -1]
16059
```

LLL

Naive solution : $\mathcal{O}(2^n)$

LLL solution : polynomial time!

2^15 = 32768
2^30 = 1073741824
2^100 = 1267650600228229401496703205376
2^1024 = 179769313486231590772930519078902473361797697894230 6...

LLL

많은 공개키 암호 시스템은 modular equation을 기반으로 함 => LLL로 깰 수 있는 확률이 높음

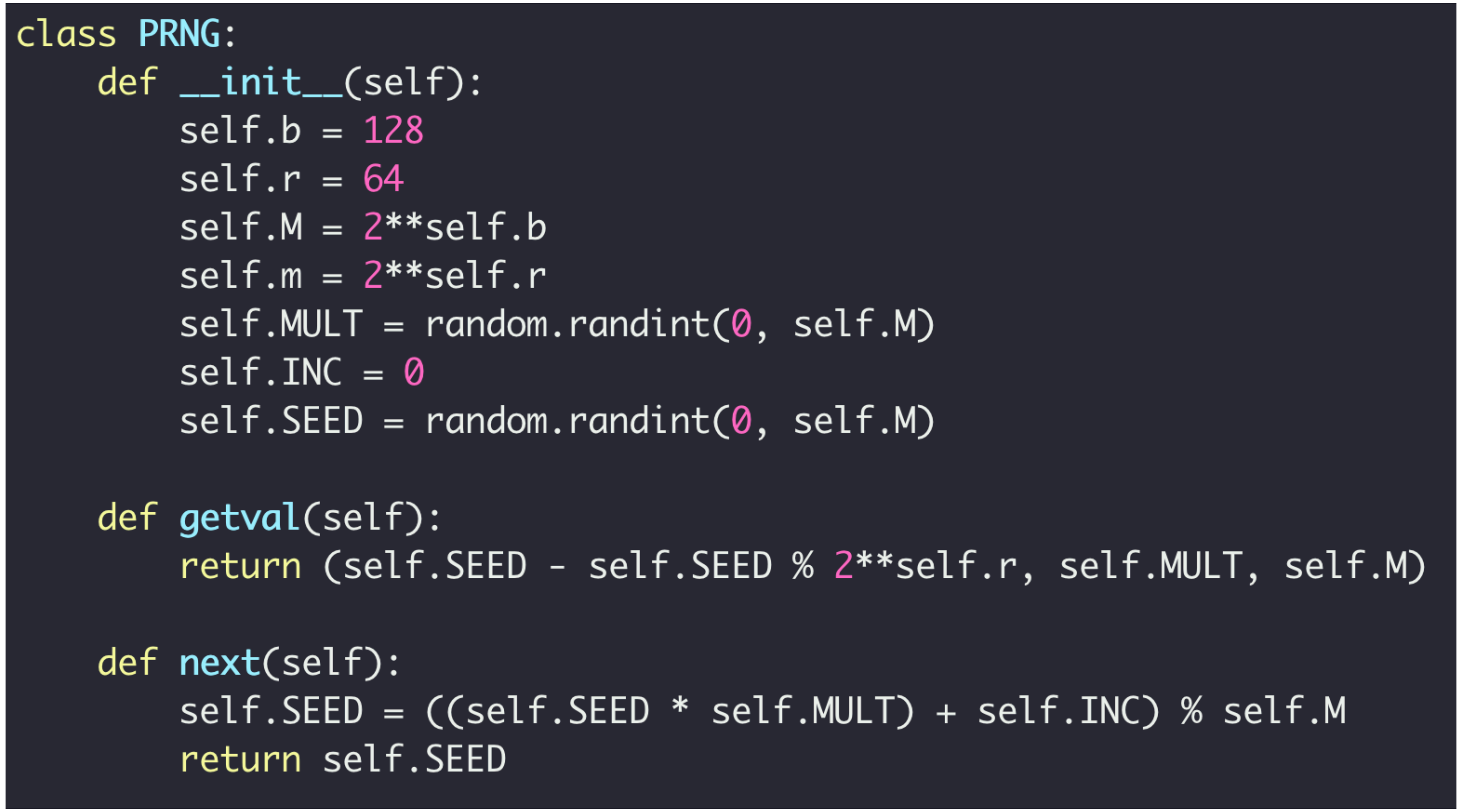
`RSA with particular setting, knapsack cryptography, NTRUEncrypt, ...,`

Practice!

```python
class PRNG:
    def __init__(self):
        self.b = 128
        self.r = 64
        self.M = 2**self.b
        self.m = 2**self.r
        self.MULT = random.randint(0, self.M)
        self.INC = 0
        self.SEED = random.randint(0, self.M)

    def getval(self):
        return (self.SEED - self.SEED % 2**self.r, self.MULT, self.M)

    def next(self):
        self.SEED = ((self.SEED * self.MULT) + self.INC) % self.M
        return self.SEED
```

BISC CTF 2022 - brokenPRNG

Practice!

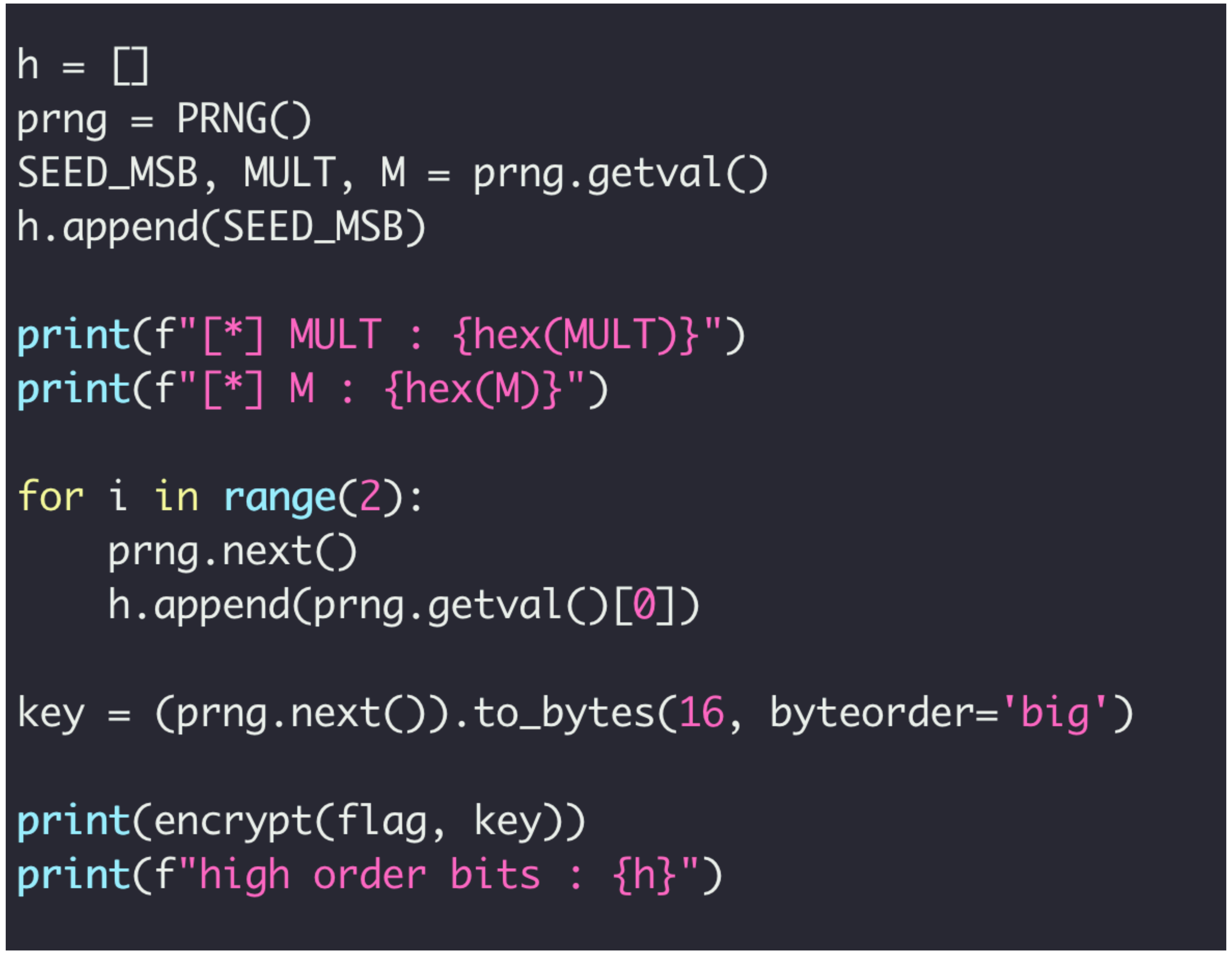


```python
class PRNG:
    def __init__(self):
        self.b = 128
        self.r = 64
        self.M = 2**self.b
        self.m = 2**self.r
        self.MULT = random.randint(0, self.M)
        self.INC = 0
        self.SEED = random.randint(0, self.M)

    def getval(self):
        return (self.SEED - self.SEED % 2**self.r, self.MULT, self.M)

    def next(self):
        self.SEED = ((self.SEED * self.MULT) + self.INC) % self.M
        return self.SEED
```

$$x_n \equiv x_{n-1} \times MULT \pmod{2^{128}}$$

x_0 **is random**

Practice!



```python
h = []
prng = PRNG()
SEED_MSB, MULT, M = prng.getval()
h.append(SEED_MSB)

print(f"[*] MULT : {hex(MULT)}")
print(f"[*] M : {hex(M)}")

for i in range(2):
    prng.next()
    h.append(prng.getval()[0])

key = (prng.next()).to_bytes(16, byteorder='big')

print(encrypt(flag, key))
print(f"high order bits : {h}")
```

known : MSB 64bit of x_0, x_1, x_2 $MULT$

unknown : x_0, x_1, x_2

Practice!

Goals: GET key! => GET LSB

Practice!

$$x_0 = x_0$$

$$x_1 \equiv x_0 \times MULT \pmod{2^{128}}$$

$$x_2 \equiv x_1 \times MULT \equiv x_0 \times MULT^2 \pmod{2^{128}}$$

Practice!

$$Mx_0 \equiv 0 \pmod{M}$$

$$ax_0 - x_1 \equiv 0 \pmod{M}$$

$$a^2x_0 - x_2 \equiv 0 \pmod{M}$$

$$\longrightarrow$$

$$L \times (x_0, x_1, x_2) = 0$$

$$L = \begin{bmatrix} M & 0 & 0 \\ a & -1 & 0 \\ a^2 \bmod M & 0 & -1 \end{bmatrix}$$

Practice!

$B =$ **`L.LLL()`**

$$B \times (x_0, x_1, x_2) \equiv 0 \pmod{M}$$

$$B \times (x_0, x_1, x_2) = M(k_0, k_1, k_2)$$

$$B \times (MSB + LSB) = Mk$$

$$B \times LSB = M \times (k_0, k_1, k_2) - B \times MSB$$

Practice!

B is small!

(k_0, k_1, k_2) too small

$$k_i = (B_0 \times MSB_0)/M$$

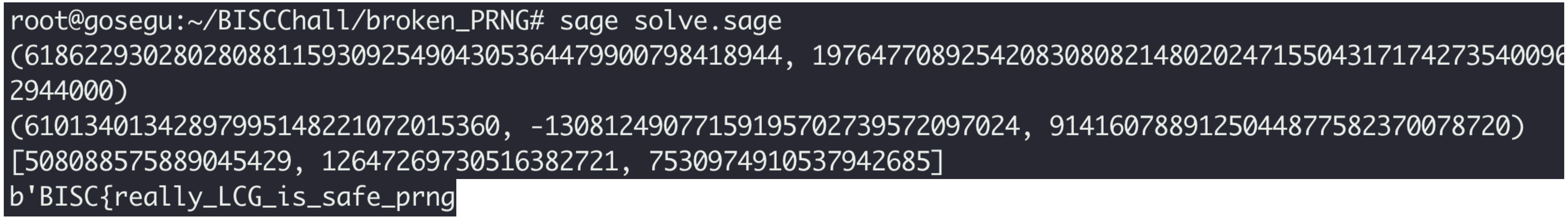
we know k.. So just solve linear equation!

$$\boxed{B \times LSB = M * (k_0, k_1, k_2) - B \times MSB}$$

Practice!

```
M = 2^128
a = 0x570a9ec8b8a9e8005d20abb2e555e29d
t = {'iv': '/5E2ciAHa0oGBEkIzoRV1A==', 'ciphertext': '+huZfkhjnNtH4sxZrItOxbJmu3RvMyOfNQH69axnX/nQcEw2iwTrZRgZyzbL8FoGB7uE5nEm
J23dHBqtA7Dp1hdj/gHjR6Ja+Ok7d4G5oPnMfN6xd79uuKjzwgt4w=='}
Y = [29186099369194997890922604909306052608, 96546435635255329419749944464313942016, 150895939852556399276298410260405682176]
iv, ciphertext = base64.b64decode(t['iv']), base64.b64decode(t['ciphertext'])

L = Matrix([[M, 0, 0], [a, -1, 0], [a^2 % M, 0, -1]])
B = L.LLL()

W1 = B * vector(Y)
print(W1)
W2 = vector([ round(RR(w) / M) * M - w for w in W1 ])

print(W2)
Z_ = list(B.solve_right(W2))

print(Z_)
key = int((a*(Z_[2] + Y[2]))%M).to_bytes(16, byteorder='big')

print(decrypt(ciphertext, key, iv))
```

Practice!



```
root@gosegu:~/BISCChall/broken_PRNG# sage solve.sage
(618622930280280881159309254904305364479900798418944, 19764770892542083080821480202471550431717427354009б
2944000)
(610134013428979951482210720153б0, -13081249077159195702739572097024, 9141607889125044877582370078720)
[508088575889045429, 12647269730516382721, 7530974910537942685]
b'BISC{really_LCG_is_safe_prng
```

⚔️ CTF RSA 맛보기

RSA 공격의 핵심 => Coppersmith Method!

⚔️ CTF RSA 맛보기

Coppersmith Method

$$F(x) = x^n + a_{n-1}x^{n-1} + \cdots + a_1x + a_0$$

Find x_0 **s.t** $F(x_0) \equiv 0 \pmod{M}, |x_0| < M^{1/n}$

모듈러 방정식이 있을 때 조건만 맞으면 방정식을 풀어낼 수 있는 매우 강력한 기법

⚔️ CTF RSA 맛보기

RSA Encryption

$$C \equiv m^e \pmod{N}$$

RSA Decryption

$$m \equiv C^d \pmod{N}$$

⚔️ CTF RSA 맛보기

RSA Encryption

$$C \equiv m^e \pmod{N}$$

m의 일부를 충분히 알고있다면 $m' < N^{1/e}$?

$$C \equiv (m' + x)^e \pmod{N}$$

$$\downarrow$$

Coppersmith theorem을 이용해 풀 수 있음

⚔️ CTF RSA 맛보기

RSA Encryption

$$C \equiv m^e \pmod{N}$$

e가 매우 작다면? -> Coppersmith theorem을 써먹을 수 있지 않을까?

$$\downarrow$$

low public exponent attack

⚔️ CTF RSA 맛보기

RSA Encryption

$$C \equiv m^e \pmod{N}$$

d가 작다면? $d < \frac{1}{3}N^{1/4}$ $d < n^{0.292}$

$$\downarrow$$

low private exponent attack

⚔️ CTF RSA 맛보기

RSA Encryption

$$C \equiv m^e \pmod{N}$$

키의 일부가 유출되었다면

$$\downarrow$$

Partial Key exposure attack

⚔️ CTF crypto tool

RSACtftool (python)

Sagemath

공부할 건 더 많다..

~~LLL~~

RSA

Symmetric ciphers

Diffie-hellman

hash functions

Elliptic curves

MATH

⋮

강의를 보고 크립토를 하고 싶어졌어요!! 더 공부 할래요!

암호학 책 사서 읽어보기

CTF 문제들 풀어보기

cryptohack 등 크립토 워게임 풀어보기

알고리즘 공부 하기

드림핵에서 암호학 커리큘럼 수강하기

🙋 Q & A

감사합니다.