sharsable write up

@kurenaif

problem

$$m : \text{FLAG}$$

$$p, q : \text{prime}$$

$$\phi = (p-1)(q-1)$$

$$n = p \times q < 2^{1024}$$

$$d_1, d_2 < n^{0.16}$$

$$e_1 d_1 + e_2 d_2 \equiv 1 \mod \phi$$

$$c_1 \equiv m^{e_1} \mod n$$

$$c_2 \equiv m^{e_2} \mod n$$

$$\text{FLAG} \equiv c_1^{d_1} \times c_2^{d_2} \mod n$$

1. pick up equations from source code.

problem

$$m : \mathrm{FLAG}$$

$$p, q : \mathrm{prime}$$

$$\phi = (p-1)(q-1)$$

$$n = p \times q < 2^{1024}$$

$$d_1, d_2 < n^{0.16}$$

$$e_1 d_1 + e_2 d_2 \equiv 1 \mod \phi$$

$$c_1 \equiv m^{e_1} \mod n$$

$$c_2 \equiv m^{e_2} \mod n$$

$$\mathrm{FLAG} \equiv c_1^{d_1} \times c_2^{d_2} \mod n$$

1. picking up equations from source code.

2. organizing equations.

red: unknown

blue: known

problem

$$m : \text{FLAG}$$

$$p, q : \text{prime}$$

$$\phi = (p-1)(q-1)$$

$$n = p \times q < 2^{1024}$$

$$d_1, d_2 < n^{0.16}$$

$$e_1 d_1 + e_2 d_2 \equiv 1 \mod \phi$$

$$c_1 \equiv m^{e_1} \mod n$$

$$c_2 \equiv m^{e_2} \mod n$$

$$\text{FLAG} \equiv c_1^{d_1} \times c_2^{d_2} \mod n$$

1. picking up equations from source code.

2. organizing equations.

red: unknown

blue: known

note: it's not normal RSA

$$e_1 d_1 \not\equiv 1 \mod \phi$$

$$e_2 d_2 \not\equiv 1 \mod \phi$$

problem

m : FLAG

p, q : prime

$\phi = (p-1)(q-1)$

$n = p \times q < 2^{1024}$

$d_1, d_2 < n^{0.16}$

If I can get d_1 & d_2,

flag will be got soon!

$$\mathrm{FLAG} \equiv c_1^{d_1} \times c_2^{d_2} \mod n$$

1. picking up equations

from source code.

2. organizing equations.

red: unknown

blue: known

problem

$$m : \mathrm{FLAG}$$

and these equations

are probably important

$$n = p \times q < 2^{1024}$$

$$d_1, d_2 < n^{0.16}$$

$$e_1 d_1 + e_2 d_2 \equiv 1 \mod \phi$$

$$c_1 \equiv m^{e_1} \mod n$$

$$c_2 \equiv m^{e_2} \mod n$$

$$\mathrm{FLAG} \equiv c_1^{d_1} \times c_2^{d_2} \mod n$$

1. picking up equations

from source code.

2. organizing equations.

red: unknown

blue: known

d is small so....

Is Wiener's attack available?

$$d_1, d_2 < n^{0.16} < n^{0.25} = n^{1/4}$$

Small private key [edit]

In the RSA cryptosystem, Bob might tend to use a small value of *d*, rather than a large random number to improve the RSA decryption performance. However, Wiener's attack shows that choosing a small value for *d* will result in an insecure system in which an attacker can recover all secret information, i.e., break the RSA system. This break is based on Wiener's Theorem, which holds for small values of *d*. Wiener has proved that the attacker may efficiently find *d* when $d < \frac{1}{3}N^{\frac{1}{4}}$.[2]

Wiener's paper also presented some countermeasures against his attack that allow fast decryption. Two

https://en.wikipedia.org/wiki/Wiener%27s_attack

d is small so....

Is Wiener's attack available?

$$d_1, d_2 < n^{0.16} < n^{0.25} = n^{1/4}$$

Small private key [edit]

In the RSA cryptosystem, Bob might tend to use a small value of *d*, rather than a large random number to improve the RSA decryption performance. However, Wiener's attack shows that choosing a small value for *d* will result in an insecure system in which an attacker can recover all secret information, i.e., break the RSA system. This break is based on Wiener's Theorem, which holds for small values of *d*. Wiener has proved that the attacker may efficiently find *d* when $d < \frac{1}{3}N^{\frac{1}{4}}$.[2]

Wiener's paper also presented some countermeasures against his attack that allow fast decryption. Two

https://en.wikipedia.org/wiki/Wiener%27s_attack

NO.

Because...

$$\begin{aligned} e_1 d_1 &\not\equiv 1 \mod \phi \\ e_2 d_2 &\not\equiv 1 \mod \phi \end{aligned}$$

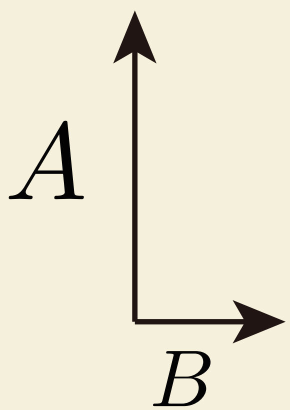
So,

nomal RSA techniques

are not available

lattice

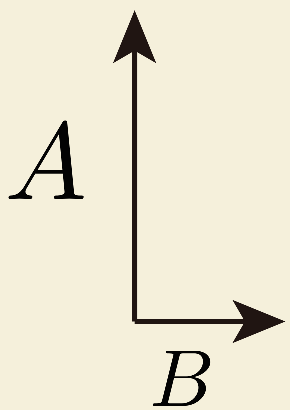
make two vectors! (but, requires linear independence)



lattice

make two vectors! (but, requires linear independence)

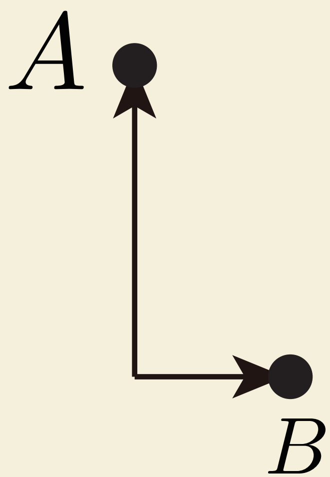
and, List the points that be made by linear combination.



lattice

make two vectors! (but, requires linear independence)

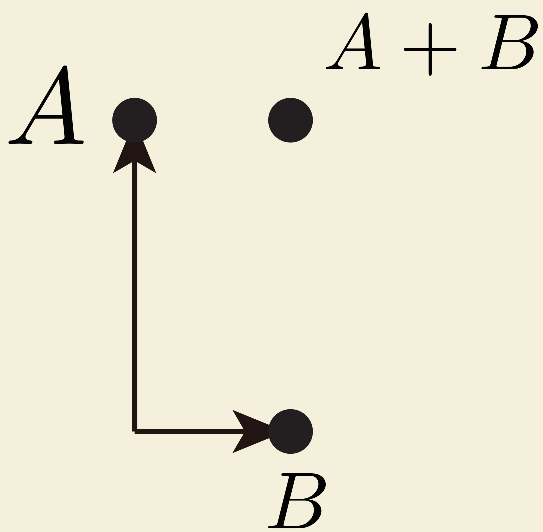
and, List the points that be made by linear combination.



lattice

make two vectors! (but, requires linear independence)

and, List the points that be made by linear combination.

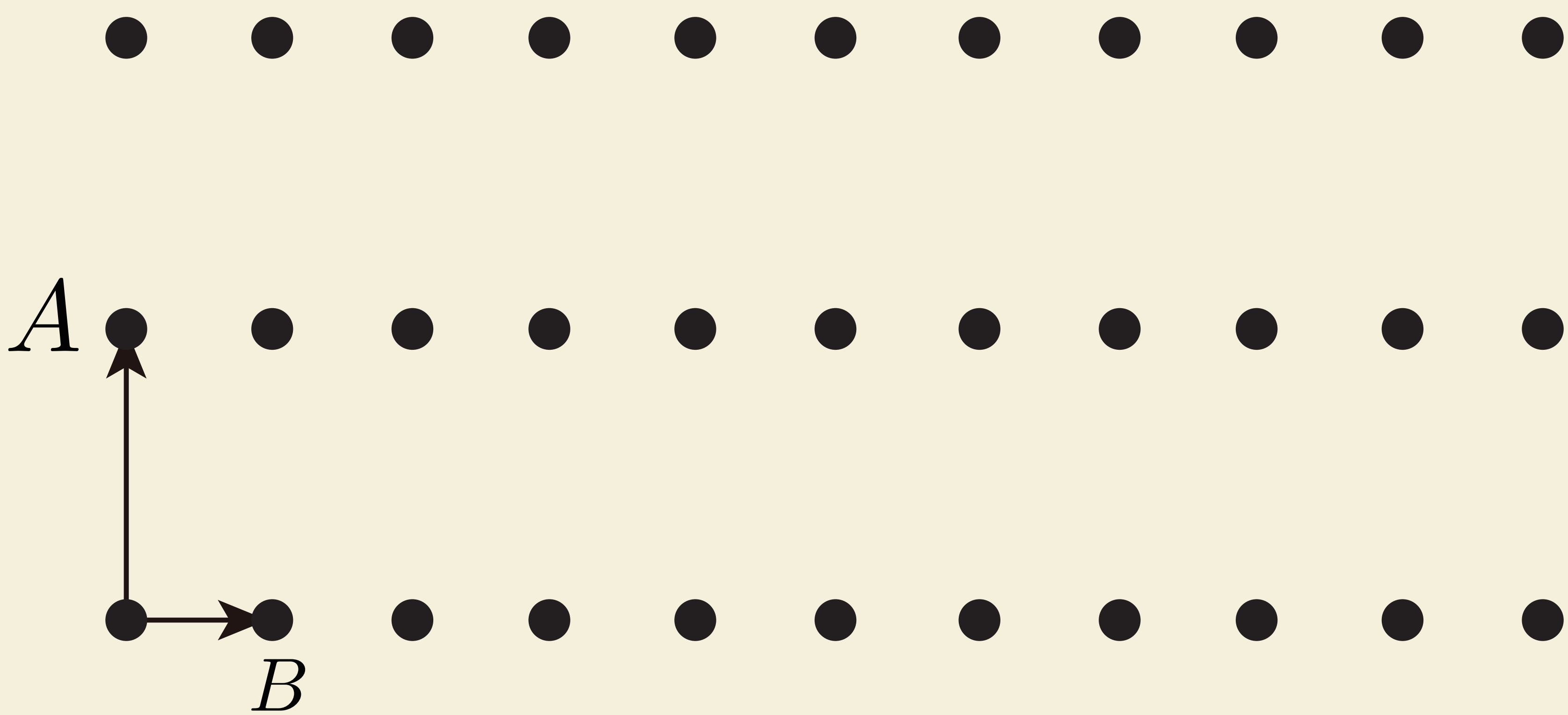


lattice

make two vectors! (but, requires linear independence)

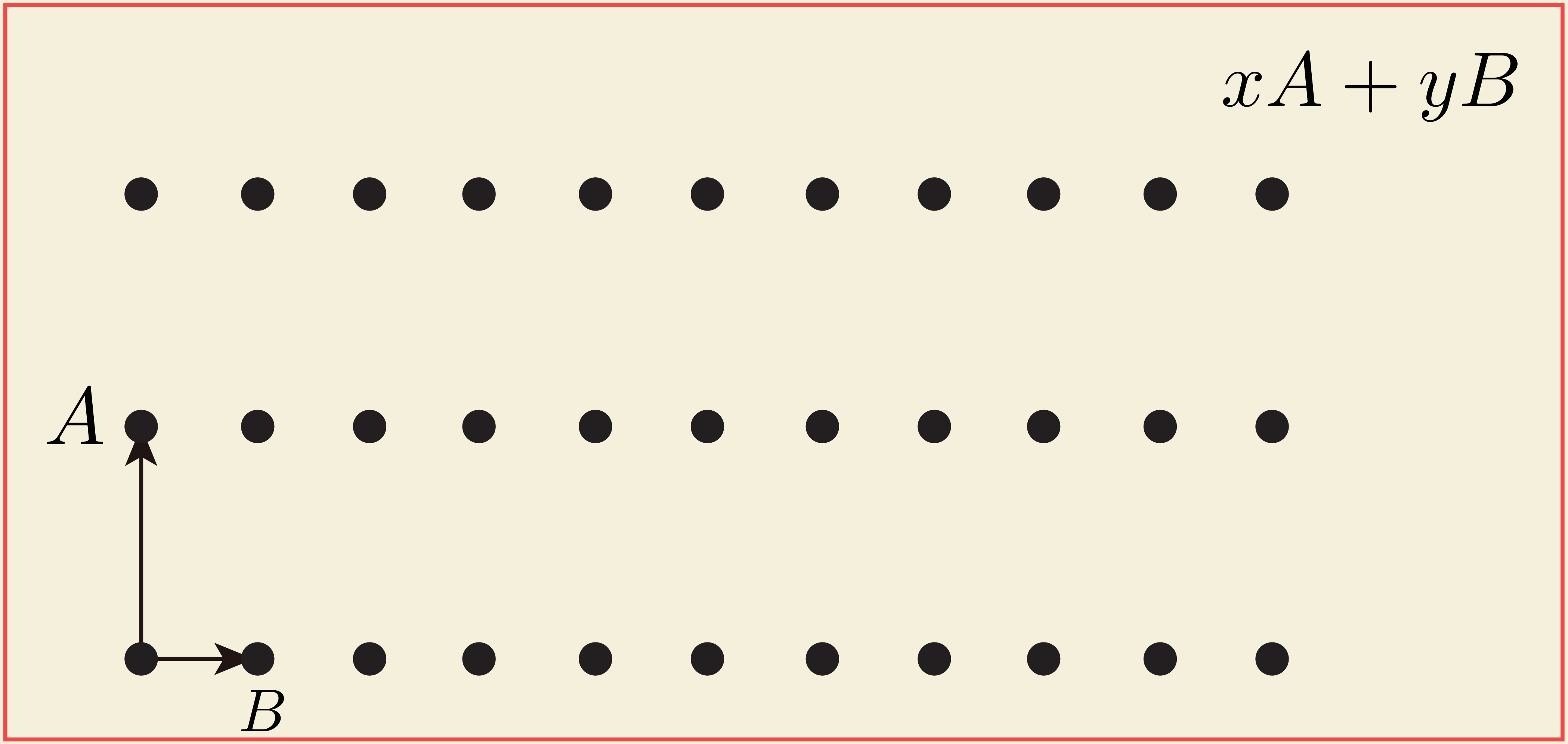
and, List the points that be made by linear combination.

$$xA + yB$$



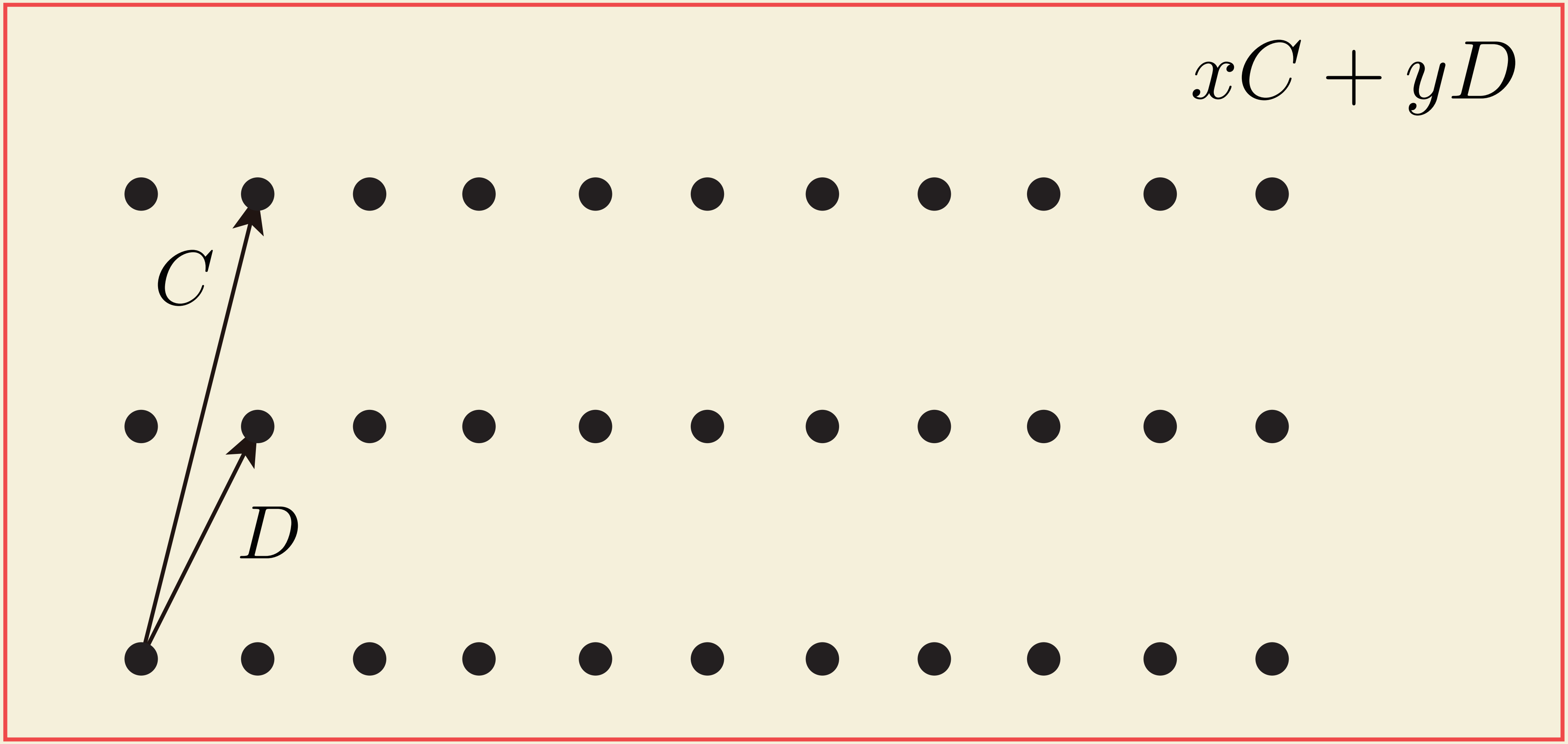
lattice

two vectors make 2D lattice!

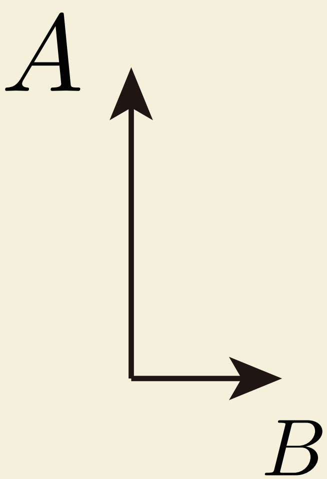


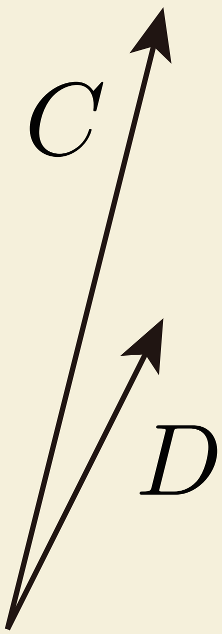
lattice

and, vector C and D make same lattice X)



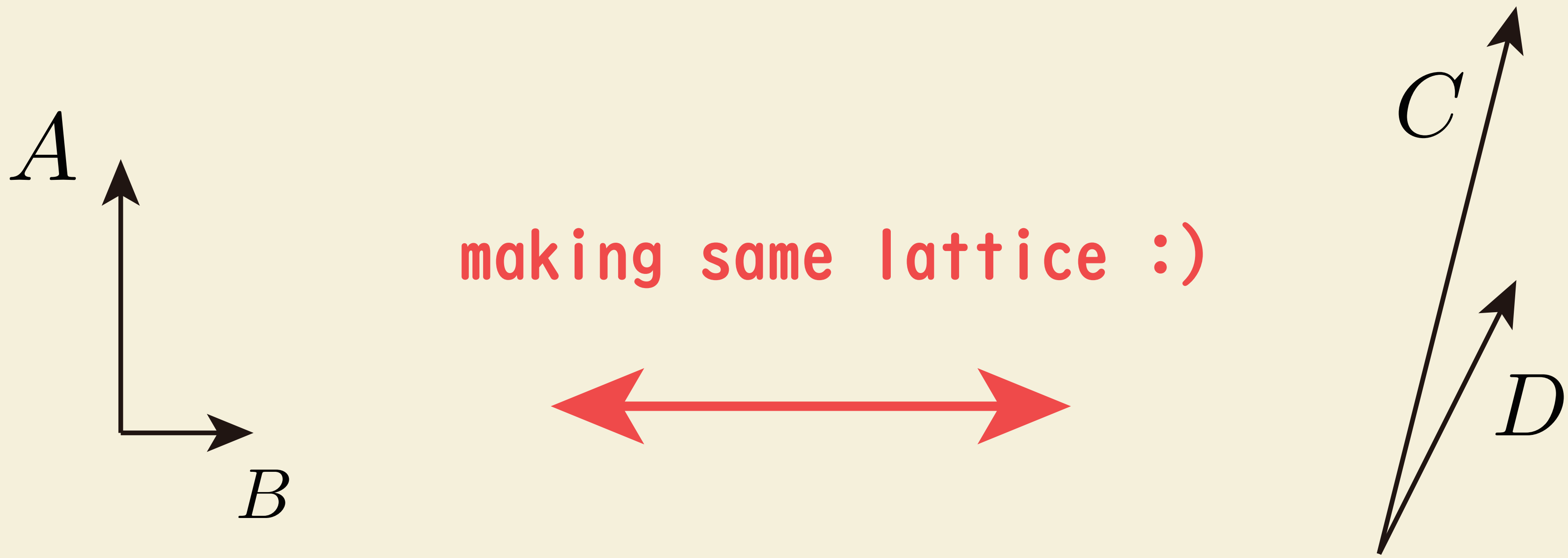
they are called lattice basis vectors





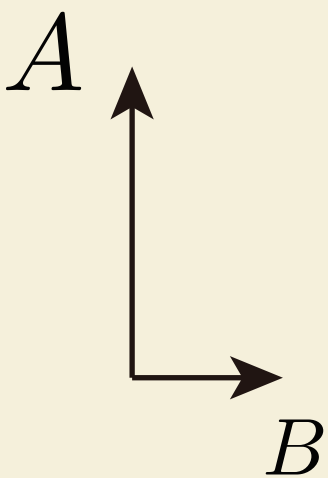
simple lattice basis vectors

they are called lattice basis vectors

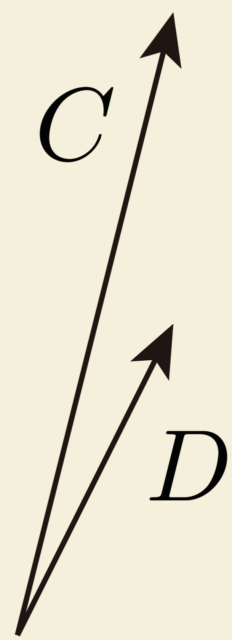


simple lattice basis vectors

they are called lattice basis vectors

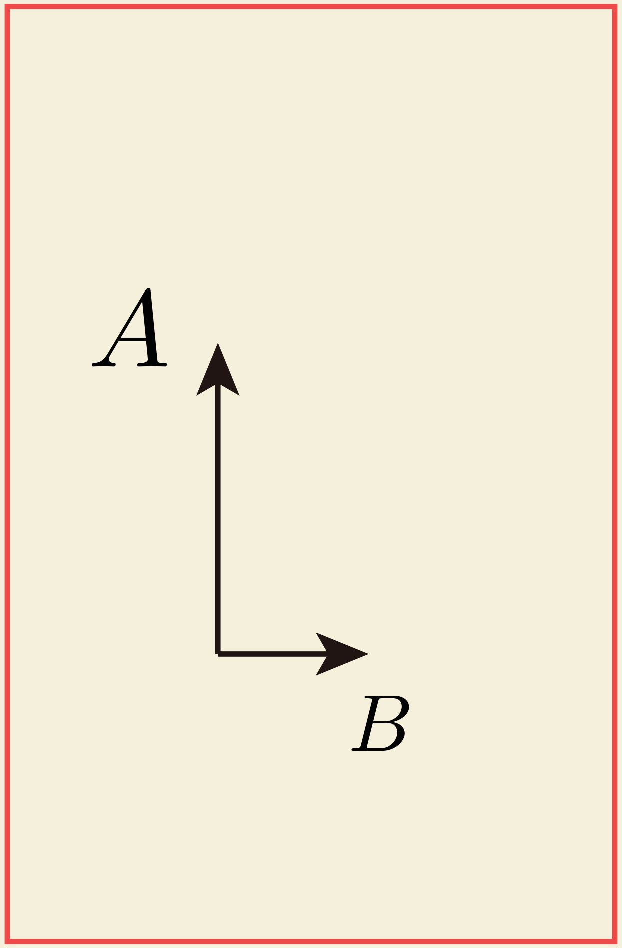


which do you like?



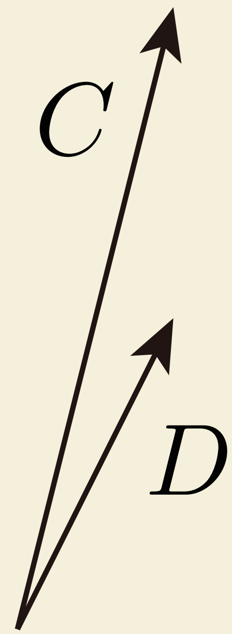
simple lattice basis vectors

they are called lattice basis vectors



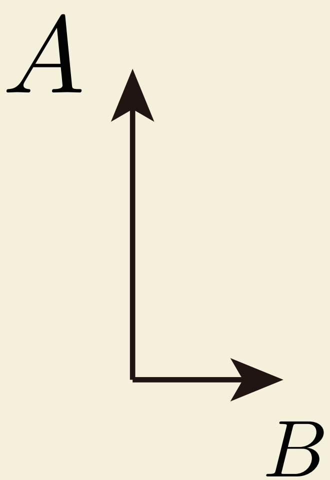
I like this :)

which do you like?



simple lattice basis vectors

they are called lattice basis vectors

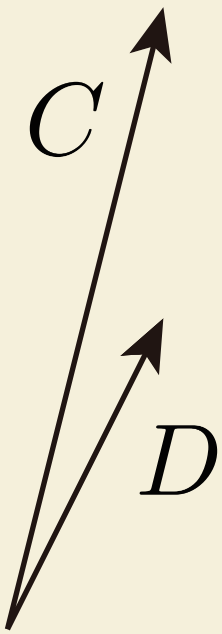


I like this :)

<-

because,

* short
* right angle



simple lattice basis vectors

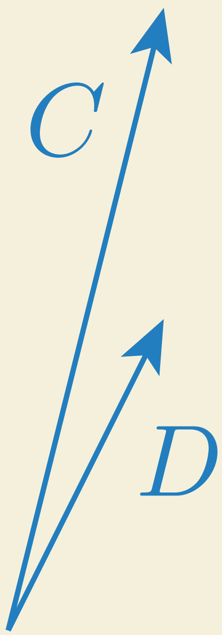
“LLL” algorithm

convert

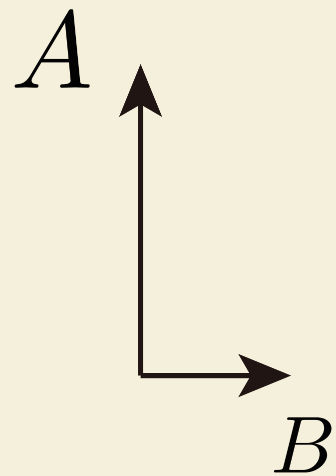
bad vectors

to

good vectors



bad vectors



good vectors

simple lattice basis vectors

“LLL” algorithm

convert

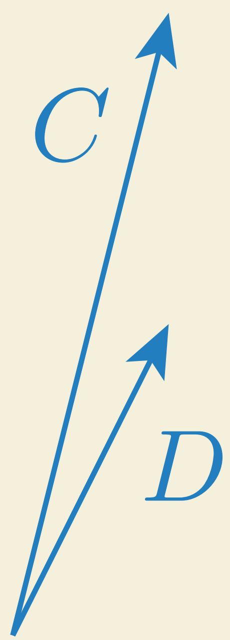
bad vectors

to

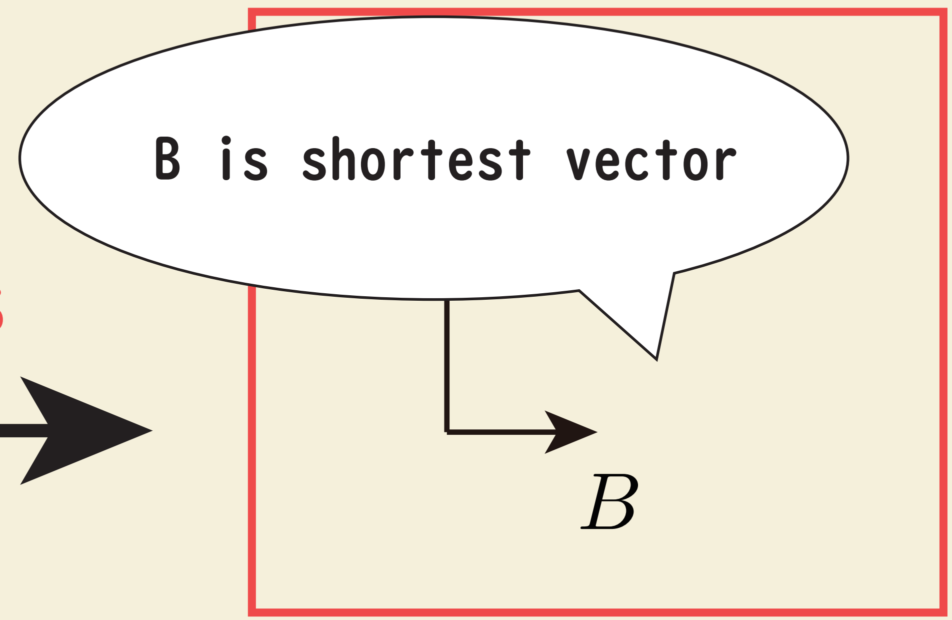
good vectors

may be contain

shortest vector



bad vectors

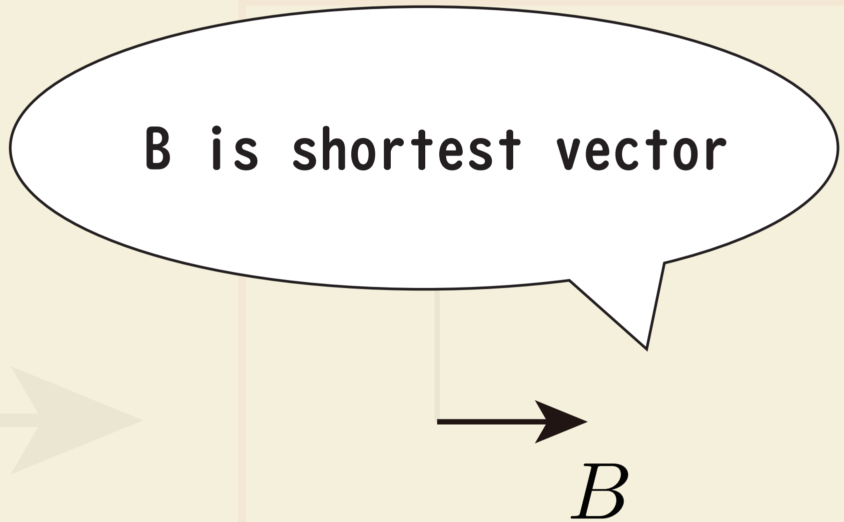


good vectors

How to solve sharsable?

decide shortest vector to solve this problem.

$$[d_1, d_2, ?] = \text{shortest vector}$$



How to solve sharsable?

decide shortest vector to solve this problem.

$$[d_1, d_2, ?] = \text{shortest vector}$$

B is shortest vector

$\longrightarrow$ B

note: d are small. so, this vector

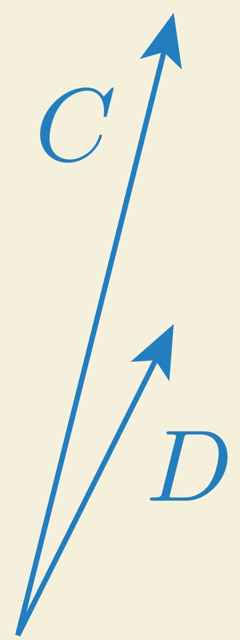
is small! may be....

$$d_1, d_2 < n^{0.16}$$

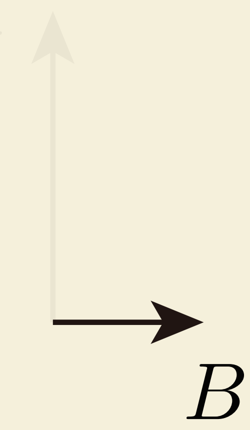
How to solve sharsable?

Make a lattice whose shortest vector is this.

$$[d_1, d_2, ?] = \text{shortest vector}$$



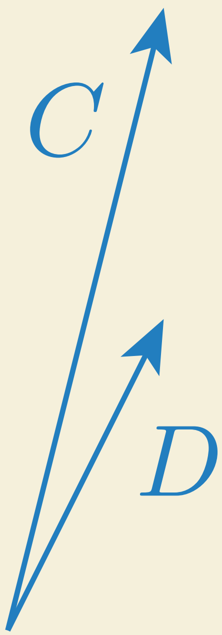
bad vectors



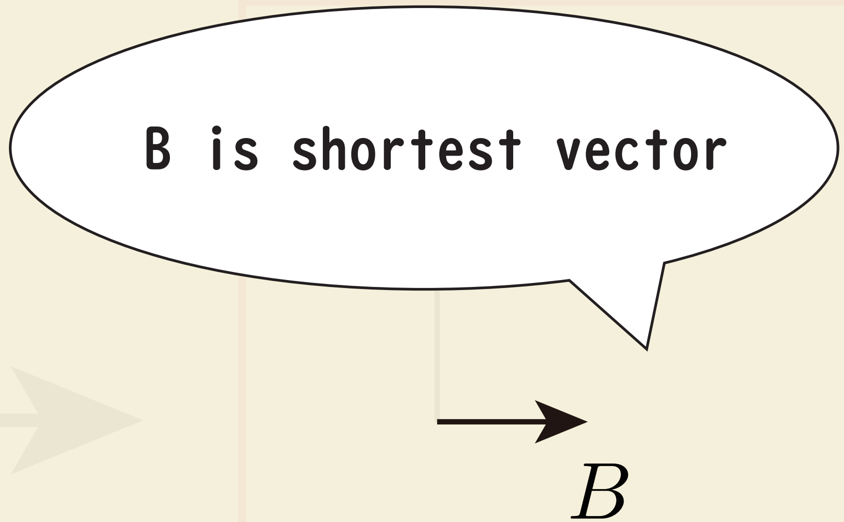
How to solve sharsable?

Make a lattice whose shortest vector is this.

$$[d_1, d_2, ?] = \text{shortest vector}$$



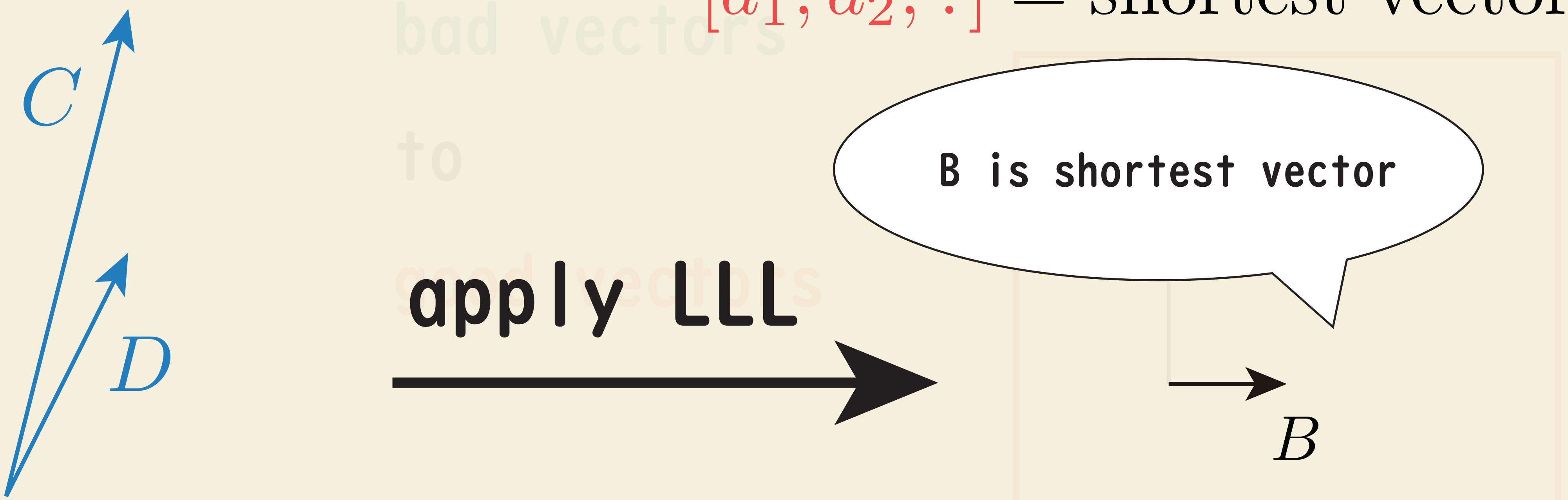
bad vectors



How to solve sharsable?

Make a lattice whose shortest vector is this.

$$[d_1, d_2, ?] = \text{shortest vector}$$



bad vectors

How to make lattice?

$$e_1 d_1 + e_2 d_2 \equiv 1 \mod \phi$$

this equation is available!

How to make lattice?

note: k is unknown

$$e_1 d_1 + e_2 d_2 \equiv 1 \mod \phi$$

$$e_1 d_1 + e_2 d_2 = 1 + k\phi$$

$$e_1 d_1 + e_2 d_2 = 1 + k((p-1)(q-1))$$

$$e_1 d_1 + e_2 d_2 = 1 + k(pq - (p+q) + 1)$$

$$e_1 d_1 + e_2 d_2 = 1 + k(n - (p+q) + 1)$$

$$e_1 d_1 + e_2 d_2 - kn = 1 + k(-(p+q) + 1)$$

How to make lattice?

so, bit(k) is
bit(min(e,d)).

$$e_1 d_1 + e_2 d_2 \equiv 1 \mod \phi$$

$$e_1 d_1 + e_2 d_2 = 1 + k\phi$$

$$e_1 d_1 + e_2 d_2 = 1 + k((p-1)(q-1))$$

$$e_1 d_1 + e_2 d_2 = 1 + k(pq - (p+q) + 1)$$

$$e_1 d_1 + e_2 d_2 = 1 + k(n - (p+q) + 1)$$

$$e_1 d_1 + e_2 d_2 - kn = 1 + k(-(p+q) + 1)$$

How to make lattice?

$$e_1d_1 + e_2d_2 - kn = 1 + k(-(p+q)+1)$$

$$p, q : 512\text{bit}$$

$$n : 1024\text{bit}$$

$$d : 164\text{bit}(\because d_1, d_2 < n^{0.16})$$

$$e : 1024\text{bit}$$

$$k : 164\text{bit}(\because k = \min(e, d))$$

How to make lattice?

$$e_1d_1 + e_2d_2 - kn = 1 + \underbrace{\underbrace{k}_{164\text{bit}}(-\underbrace{(p+q)}_{512\text{bit}}+1)}_{\fallingdotseq 676\text{bit}}$$

$p, q : 512\text{bit}$

$n : 1024\text{bit}$

$d : 164\text{bit}(\because d_1, d_2 < n^{0.16})$

$e : 1024\text{bit}$

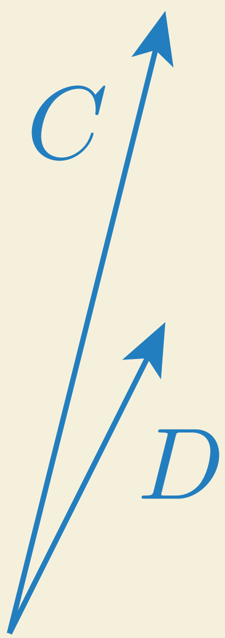
$k : 164\text{bit}(\because k = \min(e, d))$

important.

remaind please :)

make bad vectors.

$$e_1 d_1 + e_2 d_2 - kn = 1 + k(-(p+q)+1)$$



$$v_1 = [1, 0, e_1]$$

$$v_2 = [0, 1, e_2]$$

$$v_3 = [0, 0, -n]$$

make bad vectors.

$$v_1 = [1, 0, e_1]$$
$$v_2 = [0, 1, e_2]$$
$$v_3 = [0, 0, -n]$$

$$0 \cdot v_1 + 0 \cdot v_2 + v_3 = \underbrace{[0, 0, -n]}_{\text{1024 bit}}$$

$$d_1 v_1 + d_2 v_2 + k v_3 = \underbrace{[\underbrace{d_1}_{\text{164bit}}, \underbrace{d_2}_{\text{164bit}}, \underbrace{1 + k(-(p+q)+1)}_{\text{676bit}}]}_{\text{may be small!}}$$

make bad vectors.

$$v_1 = [1, 0, e_1]$$
$$v_2 = [0, 1, e_2]$$
$$v_3 = [0, 0, -n]$$

$$0 \cdot v_1 + 0 \cdot v_2 + v_3 = \underbrace{[0, 0, -n]}_{\text{1024 bit}}$$

$$d_1 v_1 + d_2 v_2 + k v_3 = \underbrace{[\underbrace{d_1}_{\text{164bit}}, \underbrace{d_2}_{\text{164bit}}, \underbrace{1 + k(-(p+q)+1)}_{\text{676bit}}]}_{\text{may be small!!}}$$

$\boxed{\text{point: making small vector is very difficult}}$

make bad vectors.

$$v_1 = [1, 0, e_1]$$
$$v_2 = [0, 1, e_2]$$
$$v_3 = [0, 0, -n]$$

$$0 \cdot v_1 + 0 \cdot v_2 + v_3 = \underbrace{[0, 0, -n]}_{\text{1024 bit}}$$

smallest vector probably contains d!

$$d_1 v_1 + d_2 v_2 + k v_3 = \boxed{[\underbrace{d_1}_{\text{164bit}}, \underbrace{d_2}_{\text{164bit}}}, \underbrace{1 + k(-(p+q)+1)}_{\text{676bit}}]$$

may be small!

point: making small vector is very difficult apply LLL! and get flag!

$$v_1 = [1, 0, e_1]$$
$$v_2 = [0, 1, e_2]$$
$$v_3 = [0, 0, -n]$$

very easy!

make good(short) vector!

```
B = Matrix(ZZ, [
    [1, 0, e1],
    [0, 1, e2],
    [0, 0, -n],
])

l = B.LLL()
row = l[0]
d1 = abs(row[0])
d2 = abs(row[1])
ans = Mod(c1, n)^d1 * Mod(c2, n)^d2
print(long_to_bytes(ans))
```

why can't get flag...?

minkowski's first theorem

$$||v|| < \sqrt{D}|\det(L)|^{1/D}$$

v : smallest vector in L

D : Dimension(D = 3 in this problem)

$$v_1 = [1, 0, e_1]$$
$$v_2 = [0, 1, e_2]$$
$$v_3 = [0, 0, -n]$$

$$L = \begin{bmatrix} 1 & 0 & e_1 \\ 0 & 1 & e_2 \\ 0 & 0 & -n \end{bmatrix}$$

$$|\det(L)| = n$$

minkowski's first theorem

$$|\det(L)| = n$$
$$D = 3$$

$$\underbrace{||v||}_{\text{676bit over}} < \sqrt{D}\underbrace{|\det(L)|^{1/D}}_{1024/3 = 341\text{bit}}$$

676bit over

1024/3 = 341bit

$$v$$

$$d_1 v_1 + d_2 v_2 + k v_3 = [\underbrace{d_1}_{\text{164bit}}, \underbrace{d_2}_{\text{164bit}}, \underbrace{1 + k(-(p+q)+1)}_{\text{676bit}}]$$

164bit 164bit 676bit

may be small!

minkowski's first theorem

$$|\det(L)| = n$$
$$D = 3$$

$$\underbrace{||v||}_{\text{676bit over}} \cancel{<} \underbrace{\sqrt{D}|\det(L)|^{1/D}}_{1024/3\,=\,341\text{bit}}$$

not satisfied :(

v

$$d_1 v_1 + d_2 v_2 + k v_3 = \underbrace{[\underbrace{d_1}_{\text{164bit}}, \underbrace{d_2}_{\text{164bit}}, \underbrace{1 + k(-(p+q)+1)}_{\text{676bit}}]}_{\text{may be small!}}$$

scaling

$$\underbrace{||v||}_{\text{676bit + small}} < \sqrt{D}\underbrace{|\det(L)|^{1/D}}_{(512+512+1024)/3}$$

676bit + small

(512+512+1024)/3

= 682 bit

$$M = 2^{512}$$

$$L = \begin{bmatrix} M & 0 & e_1 \\ 0 & M & e_2 \\ 0 & 0 & -n \end{bmatrix}$$

$$|\det(L)| = M^2 n$$

$$d_1 v_2 + d_2 v_2 + k v_3 = \underbrace{[M d_1, M d_2, 676\text{bit}]}_{\text{676bit + small}}$$

676bit + small

scaling

$$||v|| < \sqrt{D}|\det(L)|^{1/D}$$

676bit + small

(512+512+1024)/3

= 682 bit

$M = 2^{512}$

$L =$

76bit]

small

$|\det($

may be OK!

coding

```
M = 2**512

B = Matrix(ZZ, [
    [M, 0, e0],
    [0, M, e1],
    [0, 0, -n],
])

l = B.LLL()

from Crypto.Util.number import long_to_bytes

row = l[0]
d0 = abs(row[0]) // M
d1 = abs(row[1]) // M
if Mod(2, n) ** (e0 * d0 + e1 * d1) == 2:
    ans = Mod(c0, n)^d0 * Mod(c1, n)^d1
    print(long_to_bytes(ans))
~
```

coding

```
M = 2**512

B = Matrix(ZZ, [
    [M, 0, e0],
    [0, M, e1],
    [0, 0, -n],
])

l = B.LLL()

from Crypto.Util.number import long_to_bytes

row = l[0]
d0 = abs(row[0]) // M
d1 = abs(row[1]) // M
if Mod(2, n) ** (e0 * d0 + e1 * d1) == 2:
    ans = Mod(c0, n)^d0 * Mod(c1, n)^d1
    print(long_to_bytes(ans))
~
```

scaling

row[0] = M*d[0]

d[0] = row[0] // M

たいとる