Attacks on RSA cryptosystem

김동현, 박 현

발표자

이름: 김동현

소속: Anti-root

활동:

- 2019 영남대학교 SW개발 경진대회(알고리즘 부문 동상
- 영남대학교 Expert 16기 상반기 부회장
- K-shield junior 5기 보안사고 분석대응 과정 16위 (KISA 원장상)
- 2020 동계 정보보호 학술대회 논문 투고

발표자

이름: 박 현

소속: N0Named

활동:

- 2017 Digital Media HighSchool Teenager Hacking Defense Contest in Middle School 1st
- 2018 Digital Media HighSchool Teenager Hacking Defense Contest in Middle School 3rd
- 2018 Hansei Cyber Defense HighSchool in Midd School 1st
- 2018 Layer7 CTF in Middle School 1st
- 2019 The HackingChampionship Junior 3rd
- AnuCTF 3rd

Q&A용

https://www.facebook.com/profile.php?id=100005835038786

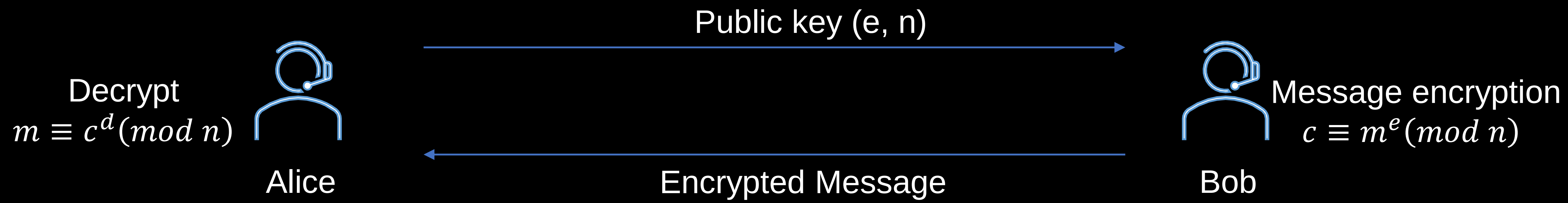
https://www.facebook.com/profile.php?id=100009136532072

Contents

- 암호수학
- 취약한 공개키
- LLL Attack
- 비밀키 일부 유출
- Q&A

암호수학

RSA


Public key (e, n)
Decrypt
$m \equiv c^d (mod\ n)$
Alice
Message encryption
$c \equiv m^e (mod\ n)$
Bob
Encrypted Message

암호수학

오일러 정리

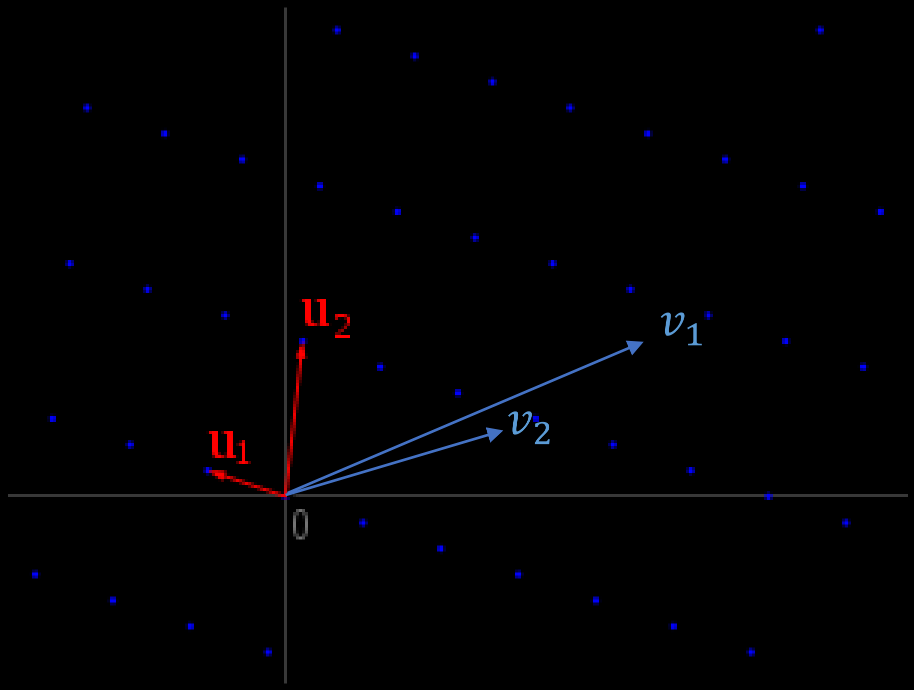
$$\forall a, n \in \mathbb{Z},\ \gcd(a, n) = 1 \ \rightarrow \mathrm{a}^{\phi(n)} \equiv 1\ (mod\ n)$$

$$\mathrm{a}^{k*\phi(n)+1} \equiv a\ (mod\ n)$$

암호수학

격자 (Lattice)

$$L = \{\sum_{i=0}^{n} a_i * v_i \mid a_0, a_1, \ldots, a_n \in \mathbb{Z}\}$$



암호수학

Howgrave-Graham Theorem

Let $g(x_1, \ldots, x_n) \in \mathbb{Z}[\mathrm{x}_1, \ldots, \mathrm{x}_\mathrm{n}]$ be an integer polynomial with at most ω monomials.
Suppose that

1. $g(y_1, \ldots, y_n) \equiv 0 \ (mod\ p^m)\ for\ |y_1| \leq X_1, \ldots, |y_n| \leq X_n$
2. $||g(x_1X_1, \ldots, x_nX_n)|| < \frac{p^m}{\sqrt{\omega}}$

Then, $g(y_1, \ldots, y_n) = 0$ holds over the integers.

취약한 공개키

Fermat's factorization

$$n = x^2 - y^2$$

$$p = x + y$$

$$q = x - y$$

$$n = \left(\frac{p+q}{2}\right)^2 - \left(\frac{p-q}{2}\right)^2$$

취약한 공개키

Fermat’s factorization

ex)

```python
from Crypto.Util.number import getPrime, bytes_to_long
from gmpy2 import *

def key_generation():
    p = getPrime(1024)
    q = next_prime(p)
    e = 0x10001
    n = p * q
    return e, n


def encrypt(m, e, n):
    return (pow(m, e, n))


if __name__ == "__main__":
    m = b"Hello World"
    pubKey = key_generation()
    e, n = pubKey
    m = bytes_to_long(m)
    c = encrypt(m, e, n)
    print("n: {}".format(n))
    print("e: {}".format(e))
    print("c: {}".format(c))
```

취약한 공개키

Fermat’s factorization

ex)

```python
from Crypto.Util.number import long_to_bytes, inverse, GCD
from gmpy2 import *

n = 10338065320880842840714962411843807252488879489247568758
e = 65537
c = 21431153245847574295730115113628524636579854888233782972

# fermat's factorization
a = isqrt(n)
b2 = square(a) - n

while not is_square(b2):
    a += 1
    b2 = square(a) - n
p = a + isqrt(b2)
q = a - isqrt(b2)

phi = (p - 1) * (q - 1)
d = inverse(e, phi)
m = pow(c, d, n)
m = long_to_bytes(m)
m = m.decode()
print (m)
```

```
$ python3 ex.py
Hello World
$ 
```

취약한 공개키

Common
Modulus

$$c_1 \equiv m^{e_1} \ (mod\ n)$$
$$c_2 \equiv m^{e_2} \ (mod\ n)$$
$$\gcd(e_1, e_2) = 1$$

취약한 공개키

Scenario

Calculate s, t ∈ $\mathbb{Z}$ such that $e_1 * s + e_2 * t = 1$

$m \equiv (c_1^{-1})^{-s} * c_2^{t} \ (mod\ n)\ or\ c_1^{s} * (c_2^{-1})^{-t}\ (mod\ n)$

취약한 공개키

Coppersmith

$$f_b(x) \equiv 0 \pmod{b} \ (b \mid N)$$

$$\Downarrow$$

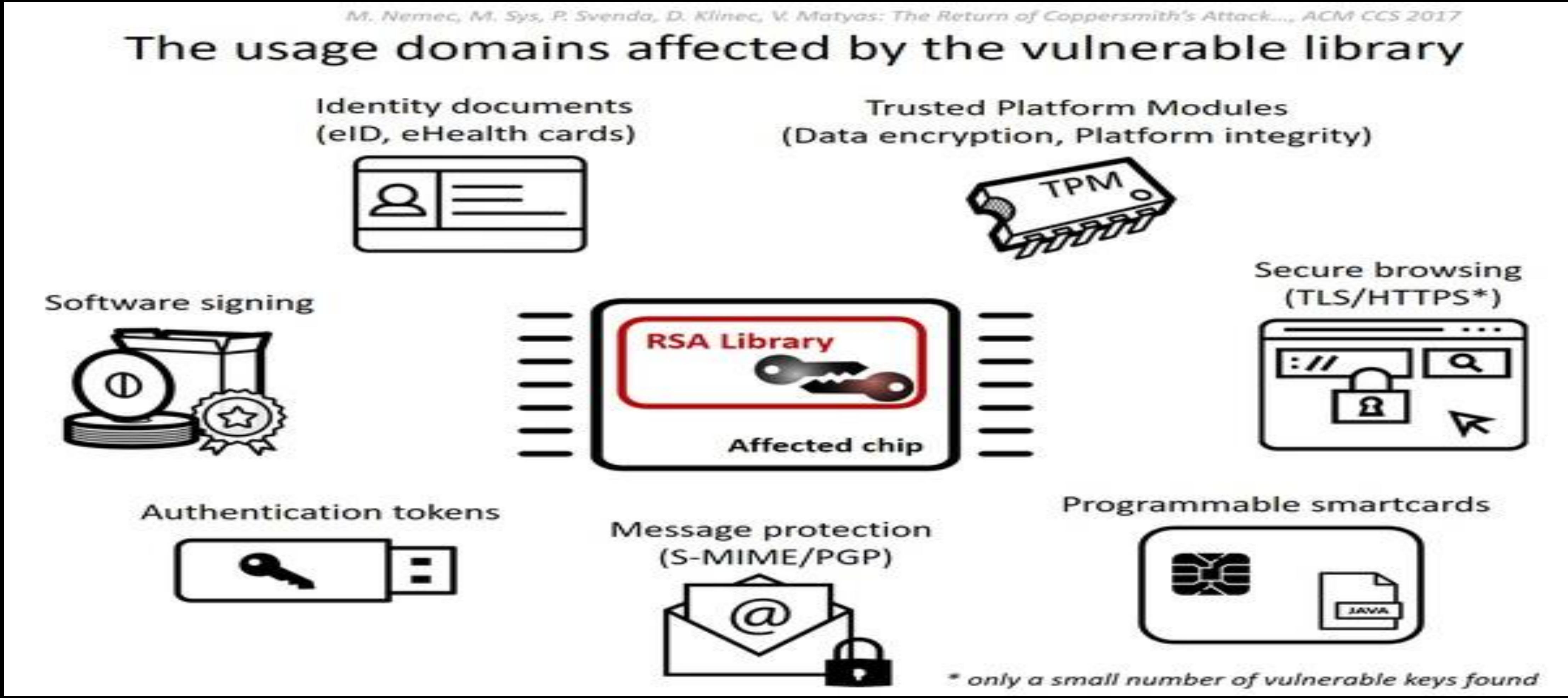
$$f(x) \equiv 0 \pmod{b^m}$$

$$\Downarrow$$

$$f(x) = 0$$

취약한 공개키

ROCA



M. Nemec, M. Sys, P. Svenda, D. Klinec, V. Matyas: The Return of Coppersmith's Attack..., ACM CCS 2017
The usage domains affected by the vulnerable library
Identity documents
(eID, eHealth cards)
Trusted Platform Modules
(Data encryption, Platform integrity)
TPM
Secure browsing
(TLS/HTTPS*)
Software signing
RSA Library
Affected chip
Authentication tokens
Message protection
(S-MIME/PGP)
Programmable smartcards
JAVA
* only a small number of vulnerable keys found

취약한 공개키

ROCA

ex) CryptoHack fast primes

```python
def sieve(maximum=10000):
    marked = [False]*(int(maximum/2)+1)

    for i in range(1, int((math.sqrt(maximum)-1)/2)+1):
        for j in range(((i*(i+1)) << 1), (int(maximum/2)+1), (2*i+1)):
            marked[j] = True

    primes.append(2)

    for i in range(1, int(maximum/2)):
        if (marked[i] == False):
            primes.append(2*i + 1)


def get_primorial(n):
    result = 1
    for i in range(n):
        result = result * primes[i]
    return result


def get_fast_prime():
    M = get_primorial(40)
    while True:
        k = random.randint(2**28, 2**29-1)
        a = random.randint(2**20, 2**62-1)
        p = k * M + pow(e, a, M)

        if is_prime(p):
            return p
```

```python
sieve()

e = 0x10001
m = bytes_to_long(FLAG)
p = get_fast_prime()
q = get_fast_prime()
n = p * q
phi = (p - 1) * (q - 1)
d = inverse(e, phi)

key = RSA.construct((n, e, d))
cipher = PKCS1_OAEP.new(key)
ciphertext = cipher.encrypt(FLAG)

assert cipher.decrypt(ciphertext) == FLAG

exported = key.publickey().export_key()
with open("key.pem", 'wb') as f:
    f.write(exported)

with open('ciphertext.txt', 'w') as f:
    f.write(ciphertext.hex())
```

취약한 공개키

ROCA

ex) CryptoHack fast primes

$p' \equiv k * M + 65547^{a} \ (mod\ M)\ (a, k \in\ \mathbb{Z}\ \&\&\ \text{unknown})$

$n \equiv (k\ * M + 65537^{a}(mod\ M))\big(l\ * M + 65537^{b}(mod\ M)\big)$

$n \equiv 65537^{c}\ (mod\ M), c = a + b$

M′ = 0x1b3e6c9433a7735fa5fc479ffe4027e13bea

$f(x) = M' * x + \big(65547^{a'}\ (mod\ M')\big)$

$c' \equiv \log_{65537} p\ \ (mod\ M')$

$\left(\frac{c'}{2} \leq a' \leq \frac{c' + ord_{M'}(65537)}{2}\right)$

$\Rightarrow$ Get p

Tool: https://gitlab.com/jix/neca

취약한 공개키

Hastad's broadcast

$$c_1 \equiv m^e \ (mod\ n_1)$$
$$c_2 \equiv m^e \ (mod\ n_2)$$
$$c_3 \equiv m^e \ (mod\ n_3)$$
$$\vdots$$
$$c_e \equiv m^e \ (mod\ n_e)$$

$\Downarrow$ Use Chinese Remainder Theorem

$$m^e \equiv c' \ (mod\ \Pi n_i)$$

$\Downarrow$ $m^e < \Pi n_i$

$$m^e = c'$$

취약한 공개키

Boneh-durfee

$$d < n^{0.292}$$
$$e * d = k * \phi(n) + 1$$
$$k * (n - p - q + 1) + 1 \equiv 0 \ (mod\ e)$$

$$\Big\downarrow \ x = k, y = -p - q$$

$$x * (n + y + 1) = 0 \ (mod\ e)$$

$$\Big\downarrow \ \text{LLL}$$

$$g_{i,k}(x, y) = x^i * f^k(x, y) * e^{m-k} \ (0 \le i \le m - k \ \&\& \ 0 \le k \le m)$$
$$h_{j,k}(x, y) = y^j * f^k(x, y) * e^{m-k} \ (0 \le j \le t \ \&\& \ 0 \le k \le m)$$

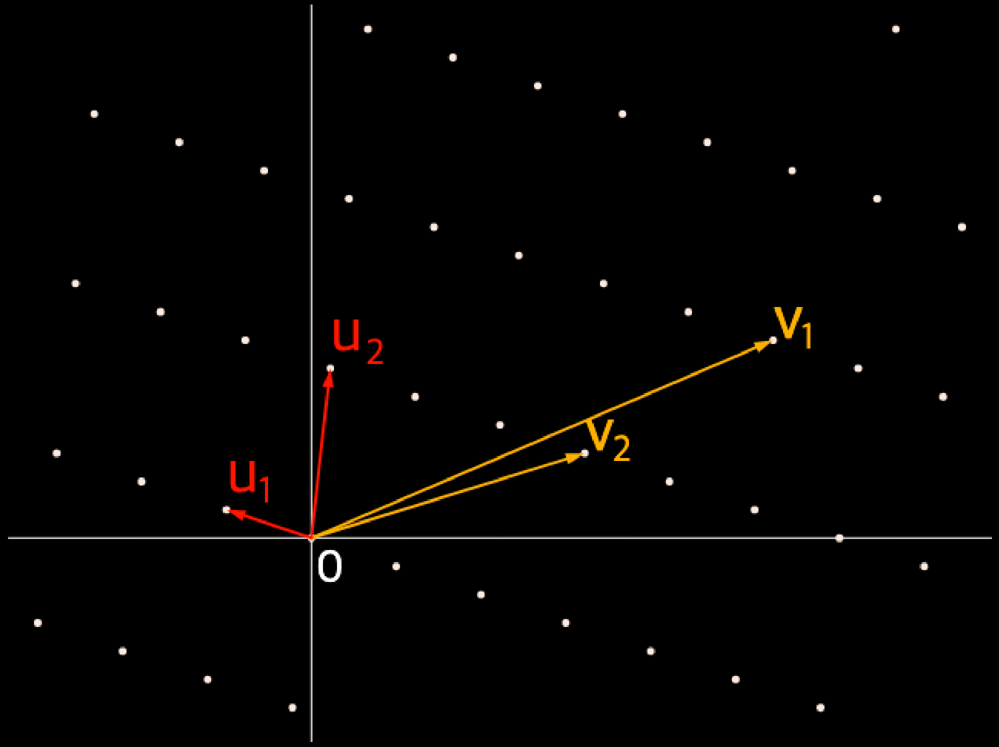
Tool: https://github.com/mimoo/RSA-and-LLL-attacks/blob/master/boneh_durfee.sage

LLL Attack

What's a Lattice?

Given a set of linearly independents $\boldsymbol{vectors}\ v_1, v_2, \ldots.\, v_n \ \in\ \mathbb{R}^m$
The lattice L **generated by** $v_1, v_2, \ldots.\, v_n$ **is the set of linearly independent**
Vectors $v_1, v_2, \ldots.\, v_n$ **with integer coefficents.**

Basis of Lattice



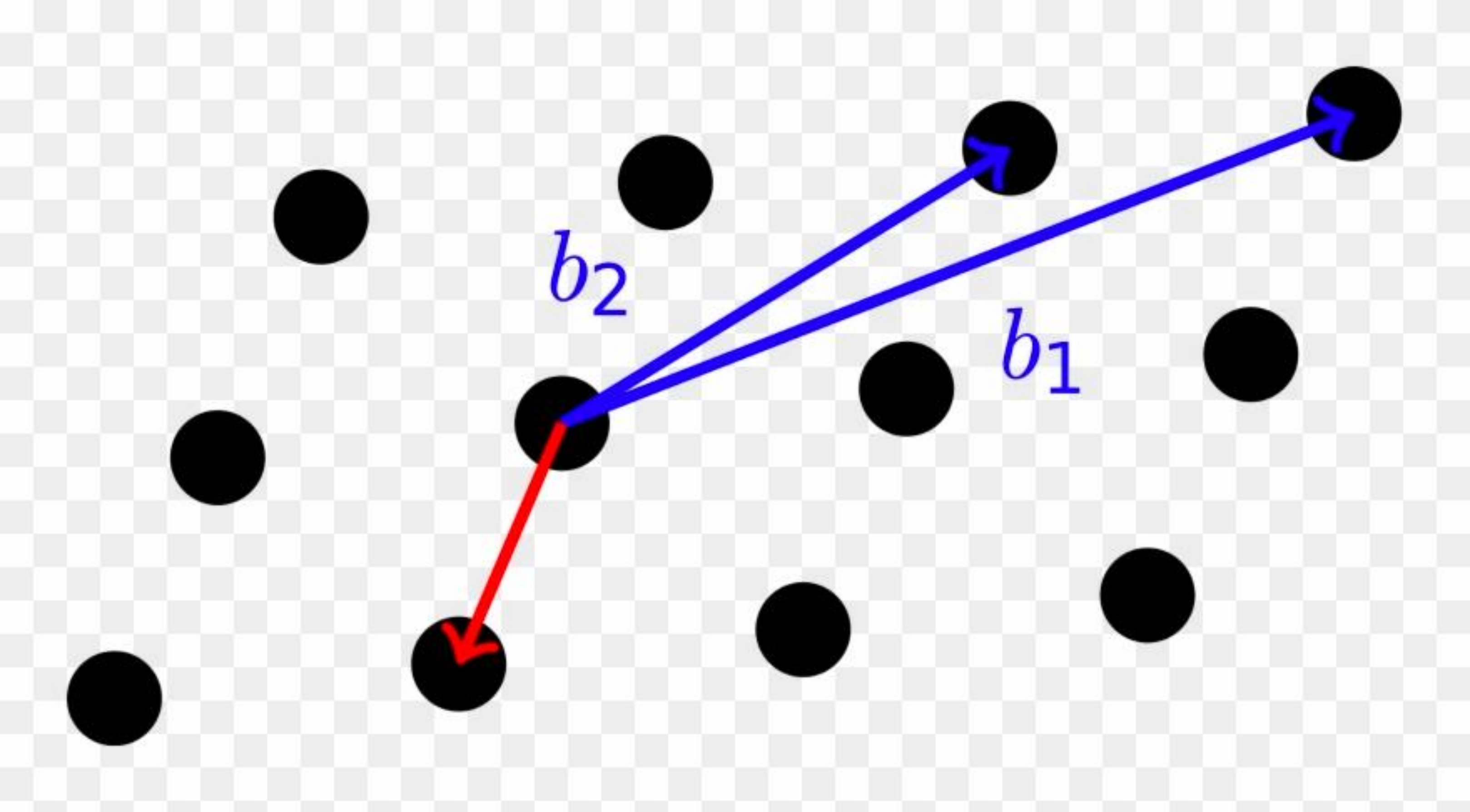
Lattice Problem

1. SVP (Shortest Vector Problem)

2. CVP (Closest Vector Problem)

SVP

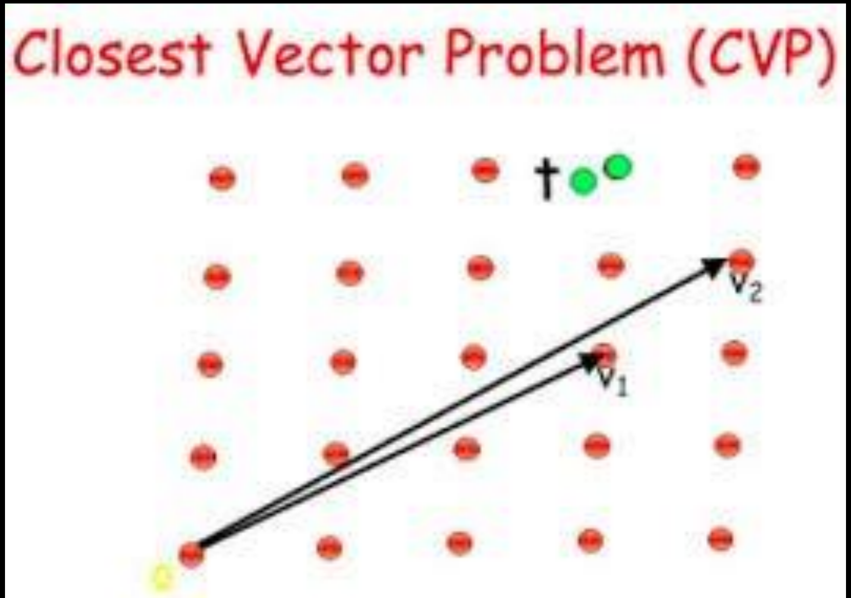
Shortest Vector Problem : Find the shortest non-zero vector in a Lattice L



CVP

Closest Vector Problem : Given a vector $w \in \mathbb{R}^m$ **that is not** , L

Find the vector that is closest to w



Cryptohack - Find the Lattice

```python
def gen_key():
    q = getPrime(512)
    upper_bound = int(math.sqrt(q // 2))
    lower_bound = int(math.sqrt(q // 4))
    f = random.randint(2, upper_bound)
    while True:
        g = random.randint(lower_bound, upper_bound)
        if math.gcd(f, g) == 1:
            break
    h = (inverse(f, q)*g) % q
    return (q, h), (f, g)


def encrypt(q, h, m):
    assert m < int(math.sqrt(q // 2))
    r = random.randint(2, int(math.sqrt(q // 2)))
    e = (r*h + m) % q
    return e


def decrypt(q, h, f, g, e):
    a = (f*e) % q
    m = (a*inverse(f, g)) % g
    return m


public, private = gen_key()
q, h = public
f, g = private

m = bytes_to_long(FLAG)
e = encrypt(q, h, m)

print(f'Public key: {(q,h)}')
print(f'Encrypted Flag: {e}')
```

generate key

512bit prime q

$$2 < f < \sqrt{\frac{q}{2}}$$

$$\sqrt{\frac{q}{4}} < g < \sqrt{\frac{q}{2}}$$

$$f^{-1}g \equiv h \pmod{q}$$

encrypt

1. choose random number $2 < r < \sqrt{\frac{q}{2}}$

2. $e \equiv rh + m \pmod{q}$

decrypt

$$a \equiv fe \pmod{q}$$

$$m \equiv af^{-1} \pmod{g}$$

Known

$$\boxed{\begin{gathered} e \equiv rh + m \pmod{q} \\ f^{-1}g \equiv h \pmod{q} \end{gathered}}$$

Unknown

$$\boxed{f, g}$$

Goal

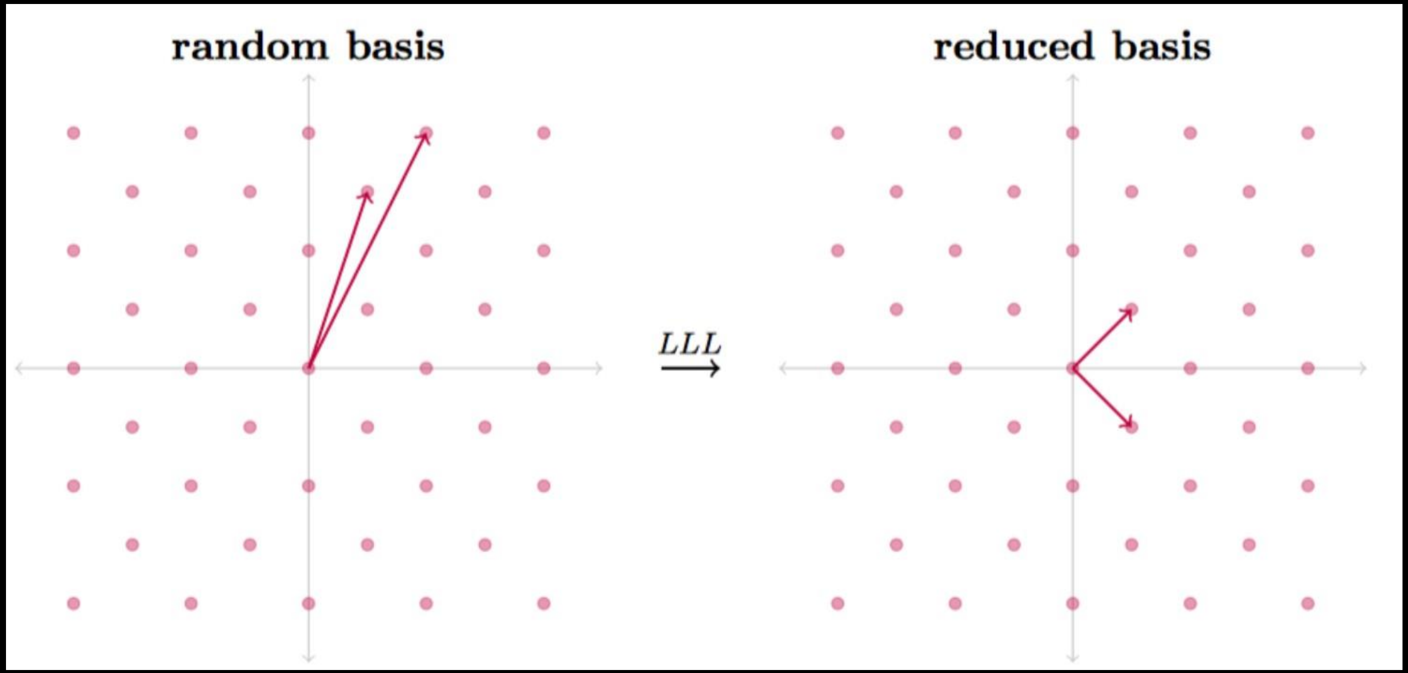
How to get f, g?

$\downarrow$

Use LLL!

LLL?

find shortest basis vector!



$$f^{-1}g \equiv h \pmod{q}$$

$$\downarrow$$

$$fh \equiv g \pmod{q}$$

$$\downarrow$$

$$fh = g + qk$$

Main Idea
h, q만 가지고 f, g 가 존재하는 lattice를 만들자!

$$L = \begin{pmatrix} 1 & h \\ 0 & q \end{pmatrix}$$

$$L \begin{pmatrix} f \\ -k \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}$$

많은 경험이 필요한 부분 ㅜㅜㅜㅜㅜ

```python
from Crypto.Util.number import long_to_bytes
q = 7638232120454925879231554234011842347641017888219021175304217358715878636183252433454896490677496516149889316745664606749499241420160898019203925115292257
h = 2163268902194560093843693572170199707501787797497998463462129592239973581462651622978282637513865274199374452805292639586264791317439029535926401109074800

enc = 5605696495253720664142881956908624307570671858477482119657436163663663844731169035682344974286379049123733356009125671924280312532755241162267269123486523

def decrypt(q, h, f, g, e):
    a = (f*e) % q
    m = a*(pow(f, -1, g)) % g
    return m
M = MatrixSpace(ZZ, 2)([
  [1, h],
  [0, q],
])

f, g = M.LLL().rows()[0]

print(long_to_bytes(decrypt(q, h, f, g, enc)))
```

```
ubuntu@ip-172-26-34-187:~/CryptoHack/Mathematics/FindTheLattice$ sage solve.sage
crypto{[redacted]
```

else..

$$\begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & -a_1 \\ 0 & 1 & 0 & 0 & \cdots & -a_2 \\ 0 & 0 & 1 & 0 & \cdots & -a_3 \\ 0 & 0 & 0 & 1 & \cdots & -a_4 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & 1 & \cdots & S \end{bmatrix}$$

knapsack cryptography

Coppersmith Theorem

p의 상위 비트를 알 때

$$x + p' \equiv 0 \pmod{p}$$

p의 하위 비트를 알 때

$$2^k x + p' \equiv 0 \pmod{p}$$

m의 상위 비트를 알 때

$$(x + m')^e - c \equiv 0 \pmod{N}$$

m의 하위 비트를 알 때

p의 하위 비트와 동일하게 2의 제곱수를 곱해주면 됨

d의 하위 l 비트를 알고 있을 때

$$ed = k(N - p - q + 1) + 1 = k(N - p - \frac{N}{p} + 1) + 1$$

$k \le e$ 이므로, 모든 0, … , e에 대해 순회 하면서 다음 방정식을 풀자

$$kp^2 + (ed' - kN - k - 1)p + kN \equiv 0 \pmod{2^l}$$

그럼 이 이후는 p의 하위비트를 알고 있을 때의 문제와 같다!

Reference

http://www.secmem.org/blog/2020/10/23/SVP-and-CVP/

https://cryptohack.org/

http://blog.rb-tree.xyz/2020/03/10/coppersmiths-method/ https://www

.math.uni-frankfurt.de/~dmst/teaching/WS2015/Vorlesung/Alex.May.pdf

https://www.semanticscholar.org/paper/The-Return-of-Coppersmith's-Attack%3A-Practical-of-Nemec-S%C3%BDs/
0b978f224b8520c8e3d9b2eb55431262fcb16c05

모두 다 엄청엄청 좋은 글이므로 무조건 읽는 것을 권장!

Q & A