BabyRSA[::-1]

THEORI

ASRybaB

- ASRybaB == BabyRSA[::-1]
- It means not baby
- Let’s start!

Challenge Description

X

ASRybaB

Just RSA

`asrybab.quals2019.oooverflow.io 1280`

Files:

- challenge.py

Challenge Description

```python
def main():
    print "Welcome to ASRybaB"
    sys.stdout.flush()

    sys.stdin.readline()

    print "1) get challenges"
    print "2) solve challenges"
    sys.stdout.flush()


    option = int(sys.stdin.readline().strip())
    if option == 1:
        send_challenges()
        sys.exit(0)
    elif option == 2:
        test_challenges()
        sys.exit(0)
    sys.exit(21)
```

What occurs in send / test?

Challenge Description

- “Send” function creates 3 tuples of (n, e, v)
- “Test” function get solution and validate it
  Pow(sol, e, n) == v?
- There are some hash checks for prevent unintended sol.
- 940 seconds of timeout

WTF can RSA be broken?

Reverse key creation algorithm

```
def send_challenges():

    code = marshal.loads("630000000000d000000070000004300000073df01000
    create_key = types.FunctionType(code, globals(), "create_key")

    ck = create_key

    challenges = []
    for _ in xrange(NCHALLENGES):
        n, e = ck()
        v = number.getRandomInteger(NSIZE-1)
        challenges.append((n, e, v))
```

Magic!

```
# uncompyle6 version 3.3.2
# Python bytecode 2.7
# Decompiled from: Python 2.7.15 (default, Oct 15 2018, 15:26:09)
# [GCC 8.2.1 20180801 (Red Hat 8.2.1-2)]
# Embedded file name: /originalchallenge.py
Nsize = NSIZE
pqsize = Nsize / 2
N = 0
while N.bit_length() != Nsize:
    while True:
        p = number.getStrongPrime(pqsize)
        q = number.getStrongPrime(pqsize)
        if abs(p - q).bit_length() > Nsize * 0.496:
            break

    N = p * q

phi = (p - 1) * (q - 1)
limit1 = 0.261
limit2 = 0.293
while True:
    d = number.getRandomRange(pow(2, int(Nsize * limit1)), pow(2, int(Nsize * limit1) + 1))
    while d.bit_length() < Nsize * limit2:
        ppp = 0
        while not number.isPrime(ppp):
            ppp = number.getRandomRange(pow(2, 45), pow(2, 45) + pow(2, 12))

        d *= ppp

    if number.GCD(d, phi) != 1:
        continue
    e = number.inverse(d, phi)
    if number.GCD(e, phi) != 1:
        continue
    break

zzz = 3
return (N, e)
```

The key code (generating "d")

```python
limit1 = 0.261
limit2 = 0.293
while True:
    d = number.getRandomRange(pow(2, int(Nsize * limit1)), pow(2, int(Nsize * limit1) + 1))
    while d.bit_length() < Nsize * limit2:
        ppp = 0
        while not number.isPrime(ppp):
            ppp = number.getRandomRange(pow(2, 45), pow(2, 45) + pow(2, 12))

        d *= ppp
```

So the algorithm is…

- limit1 = 0.261, limit2 = 0.293
- d0 = rand(N^limit1, N^limit2)
- ppp = rand(2^45, 2^45 + 2^12)
- d = d0 * ppp

Two key points

- About d's range -> d0 = range(N^limit1, N^limit2)
- ppp = rand(2^45, 2^45 + 2^12)
- There are only 115 possible primes for ppp (brute-forcable)

Simple, huh?

Google

N^0.292

전체 지도 이미지 뉴스 동영상 더보기 설정 도구

검색결과 약 2,540,000개 (0.34초)

Cryptanalysis of RSA with private key d less than N 0.292

https://crypto.stanford.edu/~dabo/pubs/abstracts/lowRSAexp.html ▾ 이 페이지 번역하기

Cryptanalysis of RSA with private key d less than $N^{0.292}$. Authors: D. Boneh and G. Durfee. Abstract: We show that if the private exponent d used in the RSA ...

Cryptanalysis of RSA with Private Key d Less than N 0.292 - Springer

https://link.springer.com/chapter/10.1007/3-540-48910-X_1 - 이 페이지 번역하기

D Boneh 저술 - 1999 - 566회 인용 - 관련 학술자료

1999. 4. 15. - We show that if the private exponent d used in the RSA public-key cryptosystem is less than $N^{0.292}$ then the system is insecure. This is the first ...

Solving challenge

- e * d = 1 (mod piN)

- (e * ppp) * d0 = 1 (mod piN)

- We can easily recover “d0” because d0 < N^0.292, right?

Actually, what is RSA?

- Pow(m, e, n) = c
- Solving equation $x^e - c = 0 \pmod{n}$
- Not “all” RSA keys are safe, right?

Breaking RSA is **very** difficult

- How about some easy condition?
- If e is small? How about d is small?
- …or Maybe we can leak private key partially…
- How about p and q are too close?

Howgrave-graham lemma

- Solving modulo equation is **very** hard

- If the norm of polynomial is small enough,

- We can solve it in Integer Ring :)

Howgrave-graham lemma

Theorem 2 (Howgrave-Graham) *Let $g(x)$ be an univariate polynomial with n monomials. Further, let m be a positive integer. Suppose that*

(1) $g(x_0) = 0 \bmod b^m$ *where* $|x_0| < X$
(2) $\|g(xX)\| < \frac{b^m}{\sqrt{n}}$

Then $g(x_0) = 0$ holds over the integers.

Proof: We have

$$|g(x_0)| = \sum_i c_i x_0^i \leq \sum_i |c_i x_0^i|$$
$$\leq \sum_i |c_i| X^i \leq \sqrt{n} \|g(xX)\| < b^m.$$

But $g(x_0)$ is a multiple of b^m and therefore it must be zero. $\square$

How can we make polynomial with small norm?

Lattice

- Subgroup of additive group R^n
- Isomorphic to additive group Z^n
- Simply, Integer span of basis
- {(2, 1), (1, 0)} spans {a, b is integer | a*(2, 1) + b*(1, 0)}

LLL Reduction algorithm

LLL reduction [edit]

The precise definition of LLL-reduced is as follows: Given a basis

$$\mathbf{B} = \{\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_n\},$$

define its Gram–Schmidt process orthogonal basis

$$\mathbf{B}^* = \{\mathbf{b}_0^*, \mathbf{b}_1^*, \ldots, \mathbf{b}_n^*\},$$

and the Gram-Schmidt coefficients

$$\mu_{i,j} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle}, \text{ for any } 1 \leq j < i \leq n.$$

Then the basis B is LLL-reduced if there exists a parameter δ in (0.25,1] such that the following holds:

1. (size-reduced) For $1 \leq j < i \leq n$: $|\mu_{i,j}| \leq 0.5$. By definition, this property guarantees the length reduction of the ordered basis.
2. (Lovász condition) For k = 1,2,..,n $: \delta \|\mathbf{b}_{k-1}^*\|^2 \leq \|\mathbf{b}_k^*\|^2 + \mu_{k,k-1}^2 \|\mathbf{b}_{k-1}^*\|^2$.

… it reduces basis, get more **smaller** basis which spans same lattice XD

So, the scenario is

- Construct some polynomials that have “same root”
- Construct Matrix of polynomials
- Run LLL Reduction algorithm, get first basis (smallest norm)
- By Howgrave-Graham lemma, we can solve in Integer

Boneh-Durfee attack

$$e \cdot d = k \cdot \varphi(N) + 1 \Rightarrow k \cdot \varphi(N) + 1 = 0 \pmod{e}$$
$$\Rightarrow k \cdot (N + 1 - p - q) + 1 = 0 \pmod{e}$$

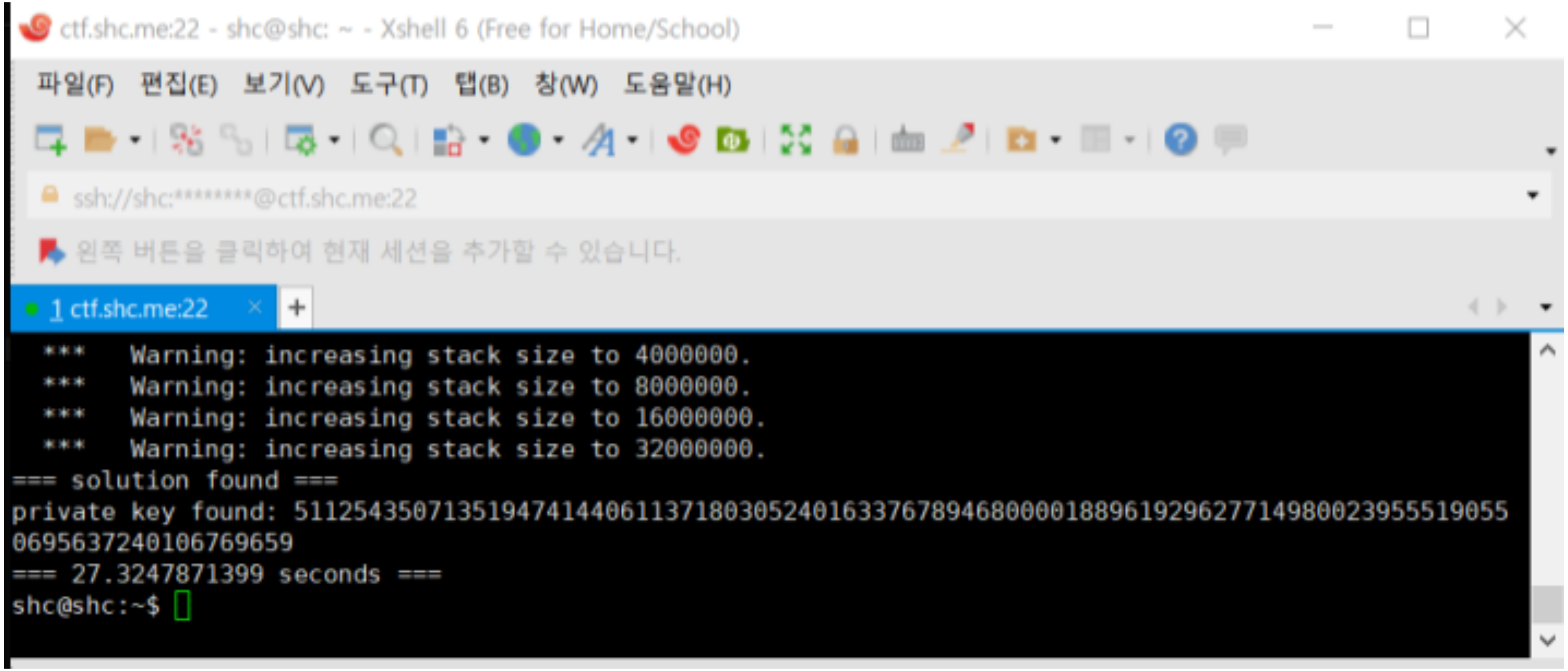
$$f(x, y) = x \cdot (N + 1 + y) = 0 \pmod{e}$$

With some optimization and calculation,
we can get d^0.292 is maximum boundary.

Solving challenge

- but (e * ppp) > N…
- We need to **optimize Lattice well**
- Making Lattice **compact** took more time… : (

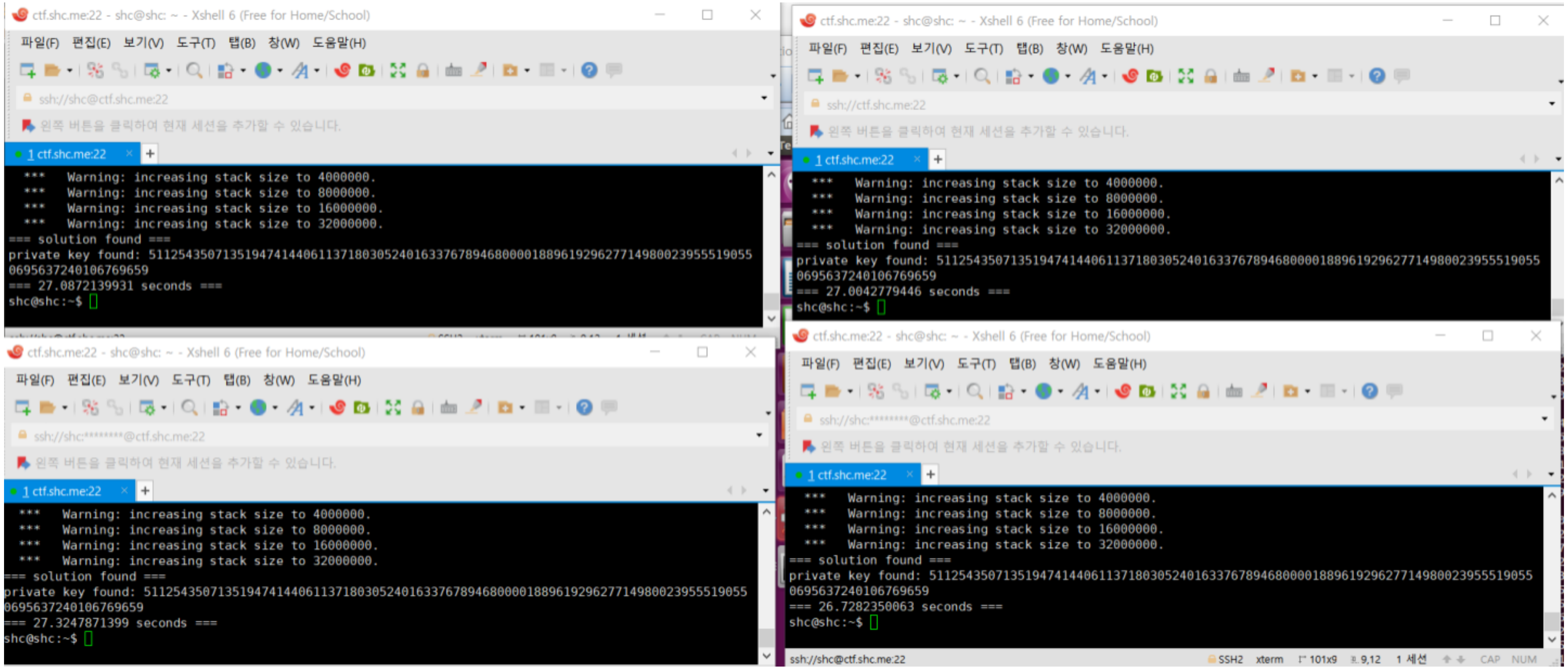
Private key found!



:thinking:

- 30 seconds / 1 test
- 3 challenges * 115 primes
- 3 * 115 * 30 = 10350 >>>>>>>>>>>>>>>>>> 940 (time limit)

Finally…



OOO{Br3akingL!mits?}

Questions?