

BabyRSA[:::-1]

# ASRybaB

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

# Challenge Description

X

## ASRybaB

Just RSA

asrybabquals2019.ooverflow.io 1280

Files:

- [challenge.py](#)

# Challenge Description

```
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  
 $\text{Pow}(\text{sol}, e, n) == v?$
- 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("6300000000d000000070000004300000073df01000  
    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!

```
# uncompile6 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")

```
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...

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

# Two key points

- About d's range ->  $d0 = \text{range}(N^{\text{limit}1}, N^{\text{limit}2})$
- $\text{ppp} = \text{rand}(2^{45}, 2^{45} + 2^{12})$
- There are only 115 possible primes for ppp (brute-forcable)

# Simple, huh?



$N^{0.292}$



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](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

- $e * d = 1 \pmod{\phi(N)}$
- $(e * ppp) * d_0 = 1 \pmod{\phi(N)}$
- We can easily recover "d0" because  $d_0 < N^{0.292}$ , right?

# Actually, what is RSA?

- $\text{Pow}(m, e, n) = c$
- 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*

$$(1) \quad g(x_0) = 0 \bmod b^m \text{ where } |x_0| < X$$

$$(2) \quad \|g(xX)\| < \frac{b^m}{\sqrt{n}}$$

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

**Proof:** We have

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

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



How can we make polynomial  
with small norm?

# Lattice

- Subgroup of additive group  $\mathbb{R}^n$
- Isomorphic to additive group  $\mathbb{Z}^n$
- Simply, Integer span of basis
- $\{(2, 1), (1, 0)\}$  spans  $\{a, b \text{ is integer} \mid a \cdot (2, 1) + b \cdot (1, 0)\}$

# LLL Reduction algorithm

## LLL reduction [\[ edit \]](#)

---

The precise definition of LLL-reduced is as follows: Given a [basis](#)

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

define its [Gram-Schmidt process](#) orthogonal basis

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

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  $\mathbf{B}$  is LLL-reduced if there exists a parameter  $\delta$  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  $k = 1, 2, \dots, n$  :  $\delta \|\mathbf{b}_{k-1}^*\|^2 \leq \|\mathbf{b}_k^*\|^2 + \mu_{k,k-1}^2 \|\mathbf{b}_{k-1}^*\|^2$ .

... 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

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

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

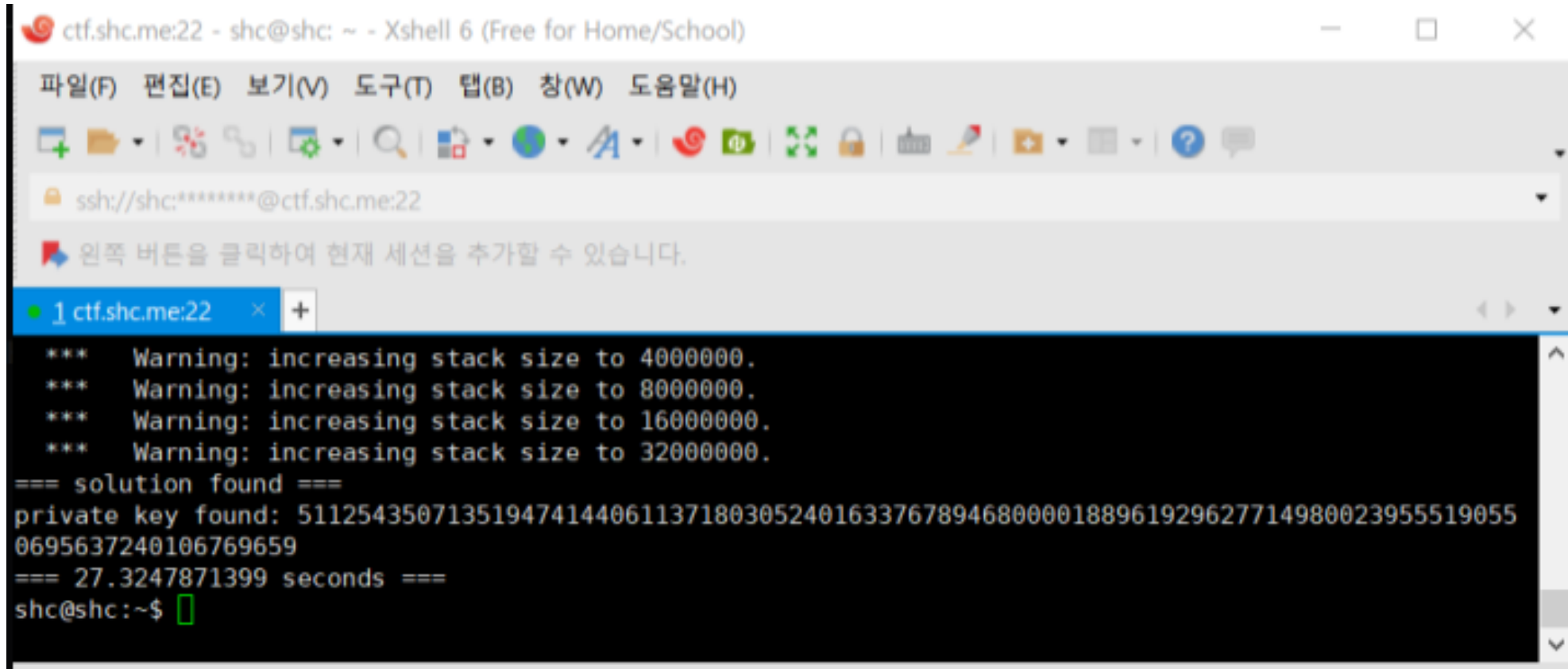
With some optimization and calculation,  
we can get  $d^{0.292}$  is maximum boundary.

# Solving challenge

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



# Private key found!



The image shows a screenshot of an Xshell 6 terminal window. The title bar indicates the connection is to 'ctf.shc.me:22' as 'shc@shc'. The menu bar is in Korean, and the toolbar contains various icons. The terminal session shows four warnings about increasing the stack size, followed by the message '=== solution found ==='. The private key is displayed as a long hexadecimal string. The session took 27.3247871399 seconds to complete. The prompt 'shc@shc:~\$' is shown with a green cursor.

```
ctf.shc.me:22 - shc@shc: ~ - Xshell 6 (Free for Home/School)

파일(F) 편집(E) 보기(V) 도구(T) 탭(B) 창(W) 도움말(H)

ssh://shc:*****@ctf.shc.me:22

왼쪽 버튼을 클릭하여 현재 세션을 추가할 수 있습니다.

1 ctf.shc.me:22 x +

*** Warning: increasing stack size to 4000000.
*** Warning: increasing stack size to 8000000.
*** Warning: increasing stack size to 16000000.
*** Warning: increasing stack size to 32000000.
=== solution found ===
private key found: 5112543507135194741440611371803052401633767894680000188961929627714980023955519055
0695637240106769659
=== 27.3247871399 seconds ===
shc@shc:~$
```

:thinking:

- 30 seconds / 1 test
- 3 challenges \* 115 primes
- $3 \cdot 115 \cdot 30 = 10350 \ggggggggggggggggggg 940$  (time limit)

# Finally...

The image displays four screenshots of Xshell terminal windows, arranged in a 2x2 grid, showing the process of solving a CTF challenge. Each window is titled 'ctf.shc.me:22 - shc@shc: ~ - Xshell 6 (Free for Home/School)'. The top-left window shows the initial state with a terminal prompt 'shc@shc:~\$' and a message '왼쪽 버튼을 클릭하여 현재 세션을 추가할 수 있습니다.' (You can add the current session by clicking the left button). The top-right window shows the same state but with a terminal tab titled '1 ctf.shc.me:22' open. The bottom-left window shows the terminal output of a brute-force attack. It displays four warnings: 'Warning: increasing stack size to 4000000.', 'Warning: increasing stack size to 8000000.', 'Warning: increasing stack size to 16000000.', and 'Warning: increasing stack size to 32000000.'. Below these, it says '=== solution found ===', followed by 'private key found: 51125435071351947414406113718030524016337678946800001889619296277149800239555190550695637240106769659' and '=== 27.0872139931 seconds ==='. The bottom-right window shows the same output but with a different time: '=== 26.7282350063 seconds ==='. The terminal prompt is 'shc@shc:~\$'.

OOO{Br3akingLimits?}

Questions?