

Q1 Teamname

0 Points

crpytoEngineers

Q2 Commands

15 Points

List the commands used in the game to reach the ciphertext.

1. exit2
2. exit4
3. exit3
4. exit1
5. exit4
6. exit4
7. exit2
8. exit2
9. exit1
10. read

We get this complete sequence of commands to reach panel after forming meaning combination of the hexadecimal numbers(after converting them into ASCII) encountered in the path.

Q3 Analysis

60 Points

Give a detailed description of the cryptanalysis used to figure out the password. (Explain in less than 150 lines and use Latex wherever required. If your solution is not readable, you will lose marks. If necessary, the file upload option in this question must be used TO SHARE IMAGES ONLY.)

As we reach the panel, it was clearly mention that there is an RSA encryption with exponent e (public key) = 5,

$N =$

8436444373572503486440255453382627917470389343976334334386326034275667
860921689509377926302880924650595564757217668266944527000881648177170141
7554768871285020442403001649254405058303439906229201909599348669565697
534331652019516409514800265887388539283381053937433496994442146419682027
649079704982600857517093

and encrypted password i.e. cipher text $C =$

23701787746829110396789094907319830305538180376427283226295906585301889
54399653341053938177968436688097089627901880710053017665162508698865521
085855413334590627256102779817144092314796016509489198045275785268570702
0289384698322665347609905744582248157246932007978339129630067022987966
706955482598869800151693 .

As we know that, in RSA, for e as public key and d as private key,

Encryption works as $C = M^e \bmod N$

Decryption works as $M = C^d \bmod N$

Now, to decrypt the password, we need prime factors of N . Since, N is very large, hence, it is not possible to find the factors. Second approach is to compute 'd' for which we require $\phi(N)$. But, as it is not possible to factorize N , hence we cannot find $\phi(N)$. We checked that $C^{\frac{1}{e}}$ is not an integer, which shows that there is some padding added.

Let p be the padding, our encryption equation converts as follows

$$C = (p + M)^e \bmod N$$

We also observed that e is very small and padding is added (described later) and hence, low exponent - Stereo typed message attack (i.e. Coppersmith's attack) can be performed.

Now, this problem gets translated to following polynomial,

$$f(x) = (p + x)^e - C \pmod{N} \quad \text{eq(1)}$$

and root of this polynomial will be our required password.

Coppersmith's theorem states that, Let N be an integer of unknown factorization, which has a divisor $b \geq N^\beta$, $0 < \beta \leq 1$. Let $f(x)$ be a univariate monic polynomial of degree δ and let $c \geq 1$.

Then we can find in time $O(c\delta^5 \log^9(N))$ all solutions x_0 of the equation

$$f(x_0) \equiv 0 \pmod{b} \text{ with } |x_0| \leq c.N^{\frac{\beta^2}{\delta}}$$

Now, for $x_0 < N^{\frac{1}{e}}$ and x_0 be the root of equation (1), this root will be the required password.

Now, finding roots of polynomial ($f(x)$) over Ring of Integer Modulo N using Coppersmith's Algorithm is hard. For that, Nicholas Howgrave-Graham, proposed an alternative and efficient technique for finding small roots of univariate modular polynomials.

Howgrave-graham theorem states that

Let $g(x)$ be an univariate polynomial with n monomials. Further, let m be a positive integer. Suppose that

$$g(x_0) \equiv 0 \pmod{N^m} \text{ where } |x_0| \leq X$$
$$||g(xX)|| < \frac{N^m}{\sqrt{n}}$$

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

According to Howgrave, we need to find a polynomial that shares the same root as our function f but modulo N^m .

Now, we referred to the steps from finding such polynomial to generating desired roots, described in the alternative technique proposed by Howgrave-Graham in this paper [1].

This steps are as follows :

1. Creating polynomial $f(x)$ over Ring of Integers modulo N .
2. Generating polynomial $g(x)$ over Ring of Integers from $f(x)$.
3. Constructing lattice L from coefficients of polynomial $g(x)$ (basis vector).

For $h=3$ and natural number X , defining lattice L of $(hk) \times (hk)$ dimension. Each entry $L(i,j)$ (i and j starting from 0) is given by $e_{i,j} X^j$, where $e_{i,j}$ is the coefficient of x^j of polynomial

$$g(x) = N^{h-1-v} x^u (f(x))^v \text{ where } v = \text{ceil}(i/e) \text{ and } u = i - e*v$$

4. Performing LLL Reduction on L to get shortest vector.
5. Converting shortest vector into polynomial.
6. Finding roots of the shortest vector polynomial.
7. This root will be our required password.

Padding :

Now to find the padding, we thought that the hexadecimal encountered on entering commands to reach panel can be helpful. We collected them and converted into ASCII. After trying various combination of them, we got a meaning-full sentence.

$p = \text{"You see a Gold-Bug in one corner. It is the key to a treasure found by"}$

Now, we tried this padding, but get undesired roots.

Then we tried padding with a space at last.

$p = \text{"You see a Gold-Bug in one corner. It is the key to a treasure found by "}$

With this padding (mentioned explicitly in the code), we get desirable roots.

Also, we tried various versions of this padding by left shifting a bit one by one.

As, root $x_0 < N^{\frac{1}{e}}$, x_0 cannot be more than 250 bits.

Hence, we shifted the bits atmost 250 times.

Conversion :

We get root = **4773930458381642785**.

Then, we generated binary of the root. We get 63 bit binary number. To make it 64 bit we added 0 bit at the MSB. Then, we performed ASCII conversion on the binary using code *decryptPassword.py*, we get our password.

Password = B@hubAl!


Programs :

smallExpRSA.py : It tries various padding bits and generate required root.
decryptPassword.py : To get ASCII conversion of root after converting root into binary.

Note : Installation of sage and sage activation will be required to run the code.
* sage is used to create Ring polynomial over Integers modulo N and to find roots.
* fpylll library is used to create Integer Matrix and for LLL reduction.

References :

- [1] <https://link.springer.com/content/pdf/10.1007/BFb0024458.pdf>
- [2] <https://github.com/mimoo/RSA-and-LLL-attacks>
- [3] Lecture 13

 No files uploaded

Q4 Password

25 Points

What was the final command used to clear this level?

B@hubAI!

Q5 Codes

0 Points

It is mandatory that you upload the codes used in the cryptanalysis. If you fail to do so, you will be given 0 marks for the entire assignment.

▼ smallExpRSA.py

 Download

```
1 import sys
```

```
2
```