Q1 Team Name

0 Points

INFINITY

Q2 Commands

10 Points

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

exit1, exit3, exit4, exit4, exit1, exit3, exit4, exit1, exit3, exit2, read

Q3 Analysis

60 Points

Give a detailed description of the cryptanalysis used to figure out the password. (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.)

On the last screen panel we have the following data
" You see the following written on the panel:

n =
84364443735725034864402554533826279174703893439763343343863260342756678609216895093779263028809246505955647572176682669445270008816481771701417554768871285020442403001649254405058303439906229201909599348669565697534331652019516409514800265887388539283381053937433496994442146419682027649079704982600857517093

INFINITY: This door has RSA encryption with exponent 5 and the password is
50178623865464437705117894233684051163813370400911458370665447586087316913420453473147267127501500015268357303406362797757594192085386118589199026764728144285648441781750590996960989276697083883546011278089668087634138747937486517183338383654747494705690147895462073667848010755833844425024779162363493605520"

From above data it can be concluded that
$c =$
$50178623865464437705117894233684051163813370400911458370665447586087$
$n =$
$84364443735725034864402554533826279174703893439763343343863260342756$
$e = 5$
Here, we know the public key($n,\ e = 5$) and the ciphertext c.
● In RSA, encryption and decryption is defined as follows,
For encryption, $c = m^e\ mod\ n$
For decryption, $m = c^d\ mod\ n$
where, m is the original message.
● The public key modulus n is long enough and hence cannot be factorized into their prime factors $p\ and\ q$ easily.
● As the encryption system uses a small public exponent ($e = 5$), the $\textbf{Coppersmith Attack}$ can be used to obtain a complete message given that we know partial bits of message (we can call that padding).
● While moving to the panel screen, on entering different exit commands, we are provided with some space-separated hex values at each gate. Hexadecimally decoding them to respective ASCII character, we get ($\textbf{without the quote symbol}$):
$exit1$: 59 6f 75 20 73 65 65 = $'You\ see'$
$exit3$: 20 61 20 47 6f 6c 64 = $'\ a\ Gold'$
$exit4$: 2d 42 75 67 20 69 6e = $'-Bug\ in'$
$exit4$: 20 6f 6e 65 20 63 6f = $'one\ co'$
$exit1$: 72 6e 65 72 2e 20 49 = $'rner.\ I'$
$exit3$: 74 20 69 73 20 74 68 = $'t\ is\ th'$
$exit4$: 65 20 6b 65 79 20 74 = $'e\ key\ t'$
$exit1$: 6f 20 61 20 74 72 65 = $'o\ a\ tre'$
$exit3$: 61 73 75 72 65 20 66 = $'asure\ f'$
$exit2$: 6f 75 6e 64 20 62 79 = $'ound\ by'$

On concatenating the above partitioned data, we get :
$\textbf{You see a Gold-Bug in one corner. It is the key to a treasure found by}$

$\textbf{Process to break RSA}$

$\textbf{1.}$ Here, we are assuming we are given $\textbf{padding}$ m_0
So our original message is of the form $m = m_0 + x$, where x are the unknown bits of the message.
$\textbf{2.}$ Now, according to coppersmith theorem : "Given an integer N and a polynomial $f(x)$ of degree t , then one can recover the root x_0 such that $f(x_0) = 0\ modN$ efficiently, provided that $x_0 < N^{1/t}$ ". So, let's try to apply Coppersmith Algorithm in our case.
$\textbf{3.}$ Construct polynomial $f(x) = (m_0 + x)^e - c$. Here, m_0(padding) , e(public exponent) and c (ciphertext) are already known and so the degree of the polynomial of the function f is 5.
$\textbf{4.}$ $N^{1/d} \approx 204$ bits for all the paddings used, so we can efficiently find the unknown message part x as it is less than $25\ bytes(204/8 \approx\ 25)$. In our case, the length of x is unknown. So, we $\textbf{brute-forced}$ every possible length from 1 to 25, and checked whether a solution exists for that particular length. We proceeded as follows to construct the polynomial and find its root:
  $\textbf{a)}$ Convert each character of the padding message to the binary of its ASCII. So we now have $length\ of\ padding \times 8\ bits$. But this is not the actual padding. These are just the $most\ significant\ bits$ of padding.
  $\textbf{b)}$ Let i be the number of bytes of x in the current iteration. We assumed that each character of x has an ASCII value greater than $32(00100000)$ which is the ASCII for $\textbf{whitespace(` ')}$. So effective padding becomes: $M = binary\_to\_int(m_0 + 00100000 * i)$. The selection of whitespace helps in further reducing the small root of the polynomial x. So effectively, we are finding roots of the polynomial
$f(x') = (M + x')^e - c$.
  $\textbf{c)}$ The maximum value of x' is assumed to be integer representation of $`\sim' * i$ i.e. $binary\_to\_int(str\_to\_bin(`\sim' * i))$ as it is the last ASCII table character that can be represented in the message.
  $\textbf{d)}$ The $inbuilt$ function $\textbf{small\_roots()}$ of SAGE is used to find the root of the polynomial for each iteration.

● We used different sentences to be used as the padding:
$\textbf{1)}$ "You see a Gold-Bug in one corner. It is the key to a treasure found by "
$\textbf{2)}$ "you see a gold-bug in one corner. it is the key to a treasure found by "
$\textbf{3)}$ "YOU SEE A GOD-BUG IN ONE CORNER. IT IS THE KEY TO A TREASURE FOUND BY "
$\textbf{4)}$ "INFINITY: This door has RSA encryption with exponent 5 and the password is "

● We first thought to use concatenated decrypted message found above as the padding. But for first three there were no roots found. So, we searched for all the sentences that came on the screen which can be used as the padding. Then we tried the 4th padding which was on the last screen panel and for this padding for length of x = 10 results in the unique root.
● The root obtained is $x_0 = \textbf{165729669577494468957753}$. So our actual unknown message is obtained by adding 32(ASCII of ' ') to each byte of x_0.
$x = x_0 + binary\_to\_int('00100000' * 10) = 317438007725212639901273.$
● Converting x to string(by packing 8 bits into a byte), the unknown message obtained is "C8YP7oLo6Y". So the actual message is :
m = INFINITY: This door has RSA encryption with exponent 5 and the password is C8YP7oLo6Y. Hence the password for the current level is $\textbf{C8YP7oLo6Y.}$

No files uploaded

Q4 Password

10 Points

What was the final command used to clear this level?

C8YP7oLo6Y

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 for the entire assignment.

▾ INFINITY.zip Download

```
Binary file hidden. You can download it using the button above.
```