Pirates of the RSAbia

Jonathan Brooks

21.09.2020

Task 1

Cargo port: Antwerp
Session key 01: fefea443a59ac24377a072e307c0b0ee
Session key 02: acd2a482f084d4d3db37abd807fdabd4
Session key 03: b47a45662b2fe6a702aacfd4a4206847
Session key 04: c2b05b0a319163fcff62462520beb58e

Task 2

Very early in the process I noticed that there was something weird with the way that the method created the prime numbers. And after some thinking I came to the conclusion that that was the vulnerability. There is only one randomly generated number and the rest are just nextprime(2**last generated prime*). Then I noticed that the last modulus created, consisted of two really close primes.

$$N4 = P3 * P4 \tag{1}$$

When I saw that I checked if I could solve the equation to find P3.

$$N4 = P3 * P4 \tag{2}$$

$$N4 = P3 * (2 * P3 + x) \tag{3}$$

x is an arbitrary number that defines the length from 2*P3 to P4 (3).

$$N4 = 2 * P3^2 + P3 * x \tag{4}$$

$$N4 \approx 2 * P3^2 \tag{5}$$

I then decided to ignore x, because it was so small compared to N4 and P3 (5).

$$\frac{N4}{2} \approx P3^2 \tag{6}$$

$$\sqrt{\frac{N4}{2}} \approx \sqrt{P3^2} \tag{7}$$

$$\sqrt{\frac{N4}{2}} \approx P3 \tag{8}$$

This equation gave me a number very close to P3. Which I used in a loop which checked if the greatest common divisor between N4 and the current number I was on was 1. If it was 1, I just set the current number to be the previous prime. I let it run until I found a number that had a GCD different from 1, since then I would have found P3.
This then gave me P3, which I used to easily calculate all the other primes. I also calculated all the private keys (d) for each modulus.
To decrypt the binary files I first extracted the hex values for each file, and converted them to decimal. Then I just had to use modular exponentiation to calculate the message.

$$c^d \equiv m \pmod{n} \tag{9}$$

Then I converted m to bytes and got the data that the pirates managed to steal.