Breaking a Few Trivial RSA Padding Schemes

August 24, 2023

1 Introduction

This challenge is actually composed of five small challenges: the flag is broken into five chunks, and each chunk is encrypted using a different padding scheme. The given public key's public exponent e is equal to 3, meaning that as soon as we have figured out a way of breaking each of the padding schemes, we can simply take the third root to obtain the respective chunk of the flag.

I cracked chunks 1 to 4 in crack.py, for the fifth part I applied Coppersmith's theorem and relied on SageMath, see part5.sage.

2 Chunk 1

Here, no padding at all has been applied. We can simply take the third root and exit.

3 Chunk 2

With this chunk/encryption method, the padding consists in multiplying the chunk with a fixed number r. Now we can take advantage of the fact that RSA is multiplicative. Thus, computing $c * (r^{-1})^e \bmod n$, which is equal to $(m * r)^e * (r^{-1})^e \bmod n$, actually gives us $m^e \bmod n$, and we can again take the third root to retrieve the chunk.

4 Chunk 3

Next, we have a padding strategy that fills with zeroes. Adding additional zeroes however reduces to the same case as with chunk 2 (multiplication with a fixed number), since it corresponds to multiplying by a power of 2. The power of two we can derive since the flag is broken into equal-length chunks. Thus, set $r = 2^l$ and proceed as in case 2.

5 Chunk 4

The fourth chunk's padding consists in repeating the chunk itself a number of times: $m' = m||m||...|m$. Mathematically, that is equivalent to $m' = m * 10..010..01..01$, where each block of the form 0..1 is of length $\lceil log_2(m) \rceil$. Therefore, this case again reduces to case 2.

6 Chunk 5

Here, I applied Coppersmith's theorem for finding a polynomial root, i.e., I cheated?? Might add more info about Coppersmith's here in the future. I derived my approach from this blog post.