Mandatory assignment 1

Thomas Hoffmann Kilbak (thhk)

01/03/2022

The problem

The problem with having a small `e` like 7, and a big modulo n, such as one that is 4096 bits long, it means that `m^e` is less than n, in which case calculating `m^e % n` will be the same as `m^e`.

This means that to get the original message, we can simply do the reverse operation on the ciphertext to retrieve the original message. That is `c^(1/e)`.

The problem is therefor a combination of the fact that a small e has been chosen, and also a very big n.

Step-by-step solution

1. Identify that n is longer than c, or that n is longer than `c^(1/e)`.
2. Calculate `a = c^(1/e)`.
3. Convert the number `a` to a byte array with big endian ordering.
4. Decode these bytes to text, yielding `HKN{...}`

Code to solve

A python program that does this, could like this:

```
c = 1050...
e = 7
res = pow(c, 1/e)
bytes = int(res).to_bytes(4096, 'big')
decoded = bytes.decode('utf-8')
print(decoded)
```

This code does not work entirely though, as python does not use enough decimal places to achieve enough precision. This yields the result `HKN{b...` which is the start of the result, but due to missing preceision in the number, there are missing bytes of accuracy.

To fix this, one can use a calculator with high precision, or use binary search to search the range 0 to n for a value x where `x^7 = n`.

Other problems with RSA

I am a bit unsure what this part of the exercise is about. I have chosen to solve it, by researching other problems there can be when using a small public exponent.

One such problem is that, if a small e is chosen, then the number of possible d-values that satisfy $de \equiv 1 \pmod{\varphi(n)}$ are lower. This makes it easier for an attacker to brute force all the possible d-values, and thereby get the private key.