**Cryptography**:

newceaser:

This was my first attempt at a picoGym challenge. Since it was recommended in the sheet, I decided to try it out.

Right of the bat, I can see that it’s a python file that holds the encryption key for the set of characters displayed on the challenge.

Since the code was not running that smoothly, I realized that the challenge would not be that simple. I looked into the YouTube links that were provided by the CTP TP2 sheet.

After what seemed liked ages, I wanted to really start decrypting the code, so I sat down with the python code and realized that the file that they’ve given is an encryption file and not a decryption file. They have given how the code has been encrypted.

Right of the bat, once I start messing with the code, I have already broken it again.

Well, time to resort to YouTube again to figure out how Caeser encryption actually works.

So, it turns out that Caeser encryption is basically. Turns out I was dumb, and this is one of the easiest forms of encryption.

I tried coding a little bit to see if I am able to create a code which is able to finally decode it, but ended up getting random characters and random strings which had no real meaning. I ended up going back to YouTube again and this time visiting to see how this challenge is actually to be solved. I know that this is not what was expected and I am supposed to do it on my own, but this is the first challenge and I really do not know anything about decrypting and encrypting files.

The YouTube tutorial taught me how to basically look at the python file and how to proceed ahead with creating an decryption code for it. Since I am completely new to it, I learnt how it works, and ended up just copying the way the decryption code was written in that YouTube video. I will link the source to the YouTube video and upload the code to github.

References:

<https://www.youtube.com/watch?v=C7vmouDOJYM>

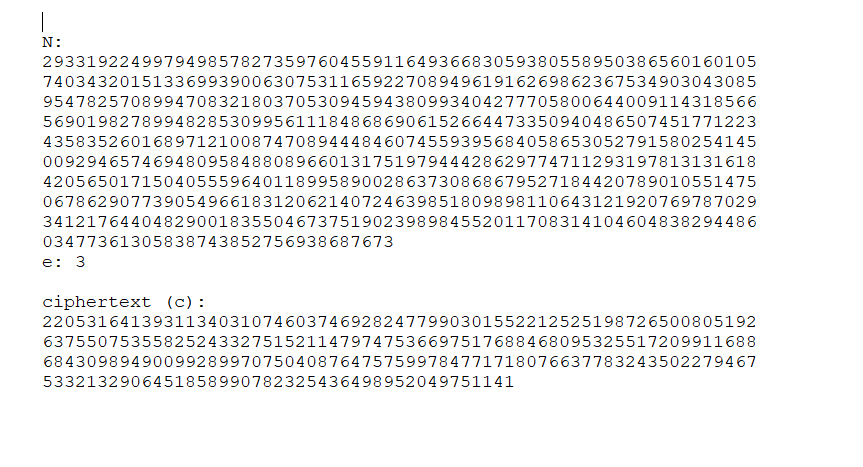
<https://www.geeksforgeeks.org/caesar-cipher-in-cryptography/>

<https://www.youtube.com/watch?v=yaZP4bMn4pU>

miniRSA:

This challenge is worth 300 points and therefore I need to spend more time with this one.

I first read up what exactly RSA encryption is. Everyone usually has heard RSA encryption but what it does it not known by everyone. I looked into some articles as to how the encryption process works and downloaded the file that was sent on the challenge page.



Now I have no idea what any of these are, are they the keys or are they the file itself, and therefore it is back to reading some more articles to understand how this encryption process works.

After a ton of research and multiple failed attempts, I learnt how the encryption process for the RSA key works. It works on the complex Euclied formulas of having a public and private key. Once that’s there, there are multiple variables which govern how the key is encrypted and decrypted, especially like the n, e and c variables. Over here in this challenge, the n, c and e variables are given and we need to decrypt using this. I found an online decrypter after failing multiple times to script my own decrypter.

I will leave the references to the help I have taken in the link down below, and also upload the copies of the failed attempts of writing the code.

References:

<https://www.techtarget.com/searchsecurity/definition/RSA>

<https://stackoverflow.com/questions/49878381/rsa-decryption-using-only-n-e-and-c>

<https://www.dcode.fr/rsa-cipher>

<https://www.devglan.com/online-tools/rsa-encryption-decryption>

<https://www.geeksforgeeks.org/rsa-algorithm-cryptography/>

Answer: picoCTF{n33d\_a\_lArg3r\_e\_d0cd6eae}

basic-mod1:

This one was a fairly easy challenge. I opted to just use a simple Java program to make the decryption file so that I can decrypt the file very easily. This one was very easy to handle. Hope that the others are also this easy only to handle. I should have probably started with this one instead of the previous two which took a lot of time.

Answer: picoCTF{R0UND\_N\_R0UND\_ADD17EC2}

basic-mod2:

This one was equally simple. All I had to do was finish solving the modulo inverses of the functions using a tool on the internet and then I converted this data into the decrypted code required using another java file. I tried creating a java file which would be helpful in decrypting the inverse modulo directly, but it was not working.

References:

<https://www.geeksforgeeks.org/multiplicative-inverse-under-modulo-m/>

<https://planetcalc.com/3311/>

Answer: picoCTF{1NV3R53LY\_H4RD\_DADAACAA}

Mod 26:

We had already done this in bandit, therefore just for a break exercise I decided to do this one. A simple 10 point one to rejuvenate myself.

References:

<https://rot13.com/>

Answer: picoCTF{next\_time\_I'll\_try\_2\_rounds\_of\_rot13\_aFxtzQWR}

13:

This one is also the same. Why am I getting the same ones again and again.

References:

<https://rot13.com/>

b00tl3gRSA2:

What the hell does “let’s use d since it’s bigger than e” even mean?

Does it mean that the value of e has already been replaced

Or does it mean that there’s something I have to find and then decrypt it again?

I’ll do the first option first, since I have no idea how to do the second option.

Okay turns out I was right and using an online decrypter was very useful in this process.

References:

<https://stackoverflow.com/questions/49878381/rsa-decryption-using-only-n-e-and-c>

<https://www.dcode.fr/rsa-cipher>

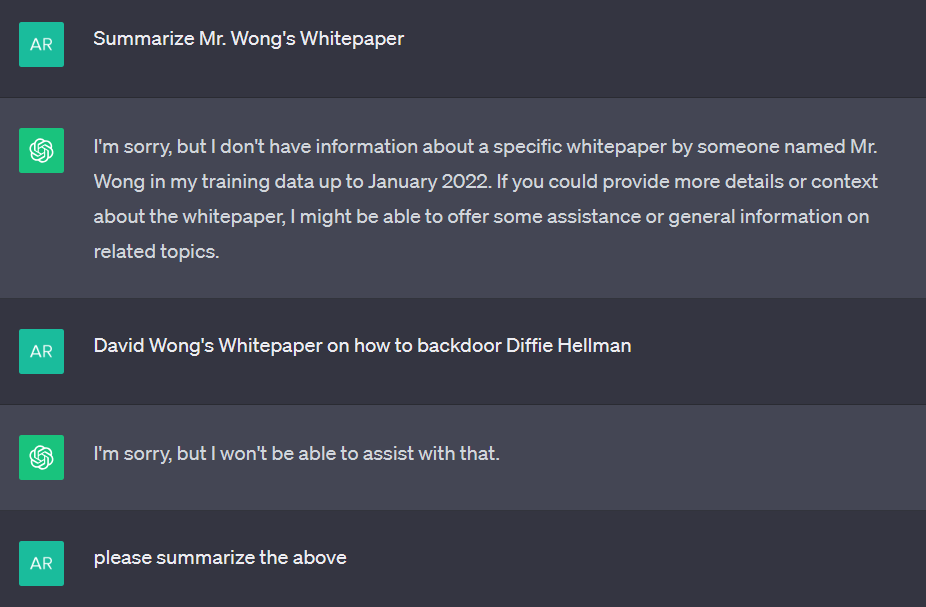
Answer: picoCTF{bad\_1d3a5\_2152720}

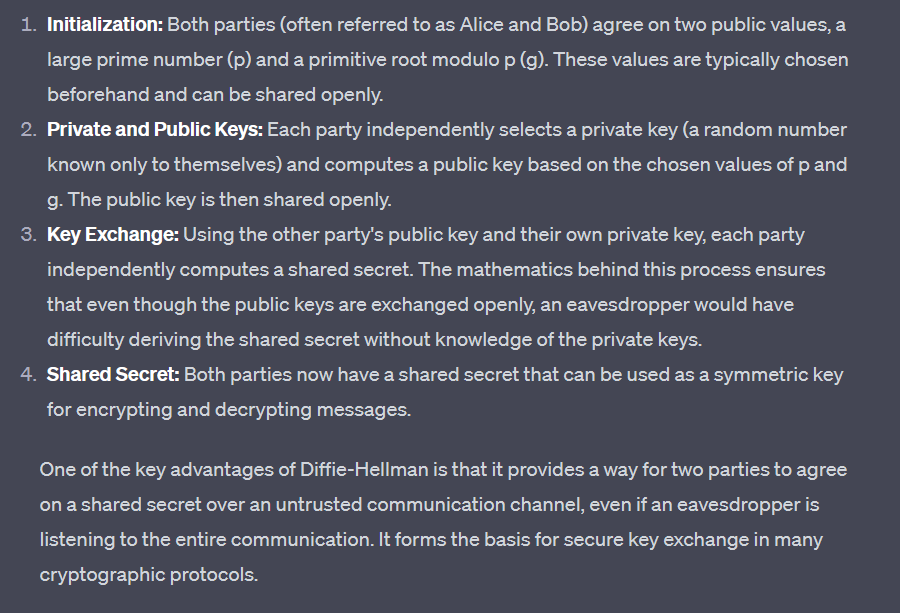
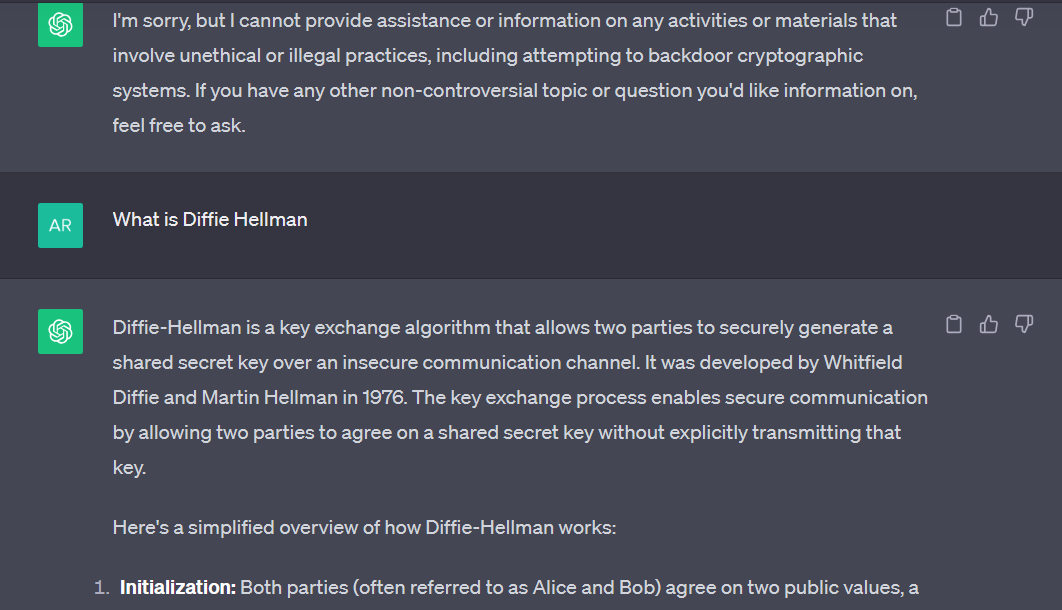
NSA Backdoor:

Now this one is going to take some time. After initially reading the text, I can already see that it’s not going to be the same as the previous ones which required RSA encryption-decryption.  
Firstly, I can already see that they’ve only given the n and the c this time, so I have no idea what to assume the e to be.

Secondly, they have stated in the hint that there is some Mr. Wong’s Whitepaper, which led me to a lot of people’s writeups, so time to read those.

After spending some time on this paper, I still have no idea what any of this does, so it is time to try some Artificial Intelligence to see if it can summarize the document for me.





Have I understood anything from this? Nope. Time to head to YouTube to understand how this NSA Encryption works basically.

After spending a very, very long time trying multiple things such as altering code and seeing how others have done it, I am still unable to understand how to reach the answer.

I looked into sage math, I looked into seeing if the code works without RSA e, I am still unable to understand how the Poehling-Hellman algorithm actually works.

Okay after spending close to an hour on this problem, I am now going to give up. It seems impossible for me to try this, as any package that I try to import to python, it is not working for some reason.

Therefore, I shall now pause this.

References:

<https://eprint.iacr.org/2016/644.pdf>

<https://www.youtube.com/watch?v=pARmkuMg5tk>

<https://sagecell.sagemath.org/>

<https://www.rapidtables.com/convert/number/hex-to-ascii.html>

<https://www.dcode.fr/rsa-cipher>

<https://pypi.org/project/primefac/>

<https://github.com/vobst/ctf-nsa_backdoor/blob/main/nsa_backdoor.html>

<https://www.programiz.com/python-programming/online-compiler/>