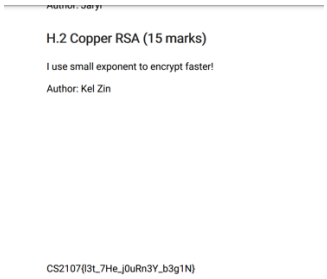
E.1 Sanity Check

I searched through the CS2107 Assignment 1.pdf until I found the flag on Pg. 4



H.2 Copper RSA (15 marks)

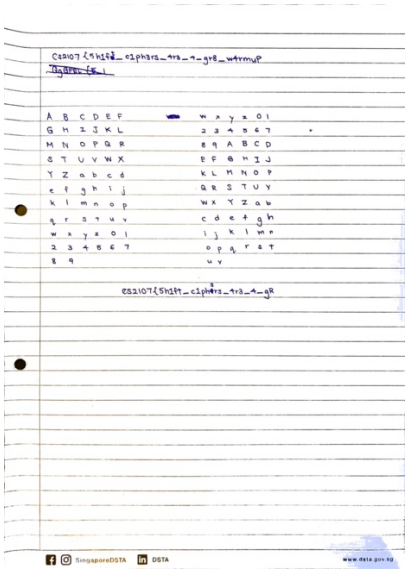
I use small exponent to encrypt faster!

Author: Kel Zin

CS2107{l3t_7He_j0uRn3Y_b3g1N}

E.2 Something's Off

I manually calculated the shift, which was -14, created the shifted table from the original table, then did substitution to find the flag.



E.3 MAC

After using the help command to understand different openssl commands. I used the command
'openssl dgst -hmac CS21072022 text.txt' to generate the MAC.

```
clover@clover-dev:~/Downloads/CTF1/MAC/dist$ openssl dgst -hmac CS21072022 text.txt
HMAC-SHA256(text.txt)= 28025dd41abceecad6056fd5b99587feac67089e6cc874679ab75e0c335a9a67
clover@clover-dev:~/Downloads/CTF1/MAC/dist$
```

E.4 Secret Penguin

After using the help command to understand the different openssl commands. I used the following command format

'openssl enc -aes-138-cbc -in {input_filename} -out {output_filename} -K 1234567890abcdef -iv abcdef1234567890abcdef1234567890'

to output the AES-encrypted file.

```
clover@clover-dev:~/Downloads$ openssl enc -aes-128-cbc -in '/home/clover/Downloads/dist/tux.png' -out '/home/clover/Downloads/dist/output.png' -K 1234567890abcdef1234567890abcdef -iv abcdef1
234567890abcdef1234567890
```

Then I used the command format

'openssl dgst -r {output_filename}

to read the SHA-256 digest of the outputted AES-encrypted file.

```
clover@clover-dev:~/Downloads$ openssl dgst -r '/home/clover/Downloads/dist/output.png'
4851ed69abe9830dda4ecca87c4634aef98ef8c2f9d7060e8ec5aaedf787a262 */home/clover/Downloads/dist/output.png
```

E.5 Prime Time

I used the RSA online calculator tool found at *decode.fr* to solve for N and subsequently decode the RSA.

M.1 Insecure OTP

1. I used the knowledge that the OTP is sufficiently short, being only 20 bytes long while the original plaintext is > 20 bytes long, to determine the OTP is insecure.
2. I re-generated the OTP by xor-ing the first 20 bytes of the message with the ciphertext.
3. I then xor-ed the ciphertext with the 20-byte OTP repeatedly, 20 bytes at a time, until I obtained the original plaintext.

```python
def encrypt(msg) :
    otp = os.urandom(20)
    print(otp.hex())      You, 2 days ago • CTF1
    print("\n")
    res = b""
    for i in range(0, len(msg), 20):
        res += xor(otp, msg[i:i+20])
    return res

def findOTP(ciphertextString, msg) :
    ciphertextByte = bytes.fromhex(ciphertextString)
    result = b""
    result += xor(msg[0:20], ciphertextByte)
    return result

def check(ciphertextString, msg) :
    otp = findOTP(ciphertextString, msg)
    res = b""
    for i in range(0, len(msg), 20):
        res += xor(otp, msg[i:i+20])
    return res

def decrypt(ciphertextString, msg) :
    ciphertextByte = bytes.fromhex(ciphertextString)
    otp = findOTP(ciphertextString, msg)
    res = b""
    for i in range(0, len(ciphertextByte), 20):
        res += xor(otp, ciphertextByte[i:i+20])
    result = res.decode()
    print(result)
```

M.2 Public Password

I followed the hints in the question to find Grandma Susan'oo's password on a picture posted on her twitter

then I checked her password against the server to find the flag.

```
clover@clover-dev:~$ nc cs2107-ctfd-i.comp.nus.edu.sg 4003
Check password: fg3b-cmn7e
Oh no!!! CS2107{p4Ssw0R6_i5_n0T_uNDeR_C0ntrOl}
```

M.3 Offline Password Cracking

I downloaded the John the Ripper software, followed its instructions on the usage and cracked the password.

```
clover@clover-dev:~/Downloads/CTF1/Offline Password/dist$ cat stolenshadow.txt
bob:$6$YSksHCmqScpJ9MRg$.pqQEW9RaAr40YsRi3z3WBqQY7NYLOHA.5EZhy0f0V2L5vFiwEv.f/5s30ghkArSHpS777TlomwKCvUvWgIhD1:1003:1003:Bob,,,:/home/bob:/bin/bash
clover@clover-dev:~/Downloads/CTF1/Offline Password/dist$ john stolenshadow.txt
Created directory: /home/clover/.john
Loaded 1 password hash (crypt, generic crypt(3) [?/64])
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:10 90% 1/3 0g/s 220.0p/s 220.0c/s 220.0C/s Bob222222..bbob1993
0g 0:00:00:14 0% 2/3 0g/s 225.3p/s 225.3c/s 225.3C/s brenda..keith
abcd1234         (bob)
1g 0:00:00:15 100% 2/3 0.06570g/s 227.0p/s 227.0c/s 227.0C/s sniper..bigben
Use the "--show" option to display all of the cracked passwords reliably
Session completed
clover@clover-dev:~/Downloads/CTF1/Offline Password/dist$ --show
--show: command not found
clover@clover-dev:~/Downloads/CTF1/Offline Password/dist$ ls
stolenshadow.txt
clover@clover-dev:~/Downloads/CTF1/Offline Password/dist$ john stolenshadow.txt --show
bob:abcd1234:1003:1003:Bob,,,:/home/bob:/bin/bash

1 password hash cracked, 0 left
clover@clover-dev:~/Downloads/CTF1/Offline Password/dist$
```

M.5 Perfect AES, Imperfect Key

I made use of the fact that the first 6 bytes of the plaintext is definitely "CS2107".

1. I generated a SHA-512 key and a cipher using the randomly generated key and the known IV
2. I used the cipher to decrypt the ciphertext to see if it matches the plaintext.
3. Using a while loop, I repeat steps 1-2 until the the obtained plaintext has the first 6 bits "CS2107".

```
def decryption() :
    while True :
        iv = "4b0fb9a4dfbabe6810b2fb01d2012b84"
        ct = "c089a2553fdcbb0bbdbd7655fc34c75eb7f2ccd28fc801480c5a15b7f366f8737a30aa3e845d79e509486ffd
        key = sha512(os.urandom(20)[:3]).digest()[:16]
        cipher = AES.new(key, AES.MODE_CBC, bytearray.fromhex(iv))
        pt = cipher.decrypt(bytearray.fromhex(ct))
        if pt[:6] == "CS2107".encode('utf-8') :
            print(pt.decode('utf-8'))

decryption()
```

M.6 Substitution Cipher

1. I first obtained iterated through the entire ciphertext to obtain the frequencies of every alphabet in the ciphertext.
2. I then compared it to the frequency of alphabets table on the internet and used it to make my first guess
3. I then made substitutions by using known facts such as the last line must start with “CS2107” and that bullet points are in the format “I, II, III, IV”, and sub-bullet points are in the “A, B, C, D” format to derive the rest of the plaintext.

```
49  for i in f :
50      if (i in alphabet):
51          alphabetArray[ord(i) - ord('A')] += 1
52
53  for i in f :
54      if (i in alphabet) :
55          encrypted += conversion(ord(i) - ord('A'))
56      else :
57          encrypted += i
58
59  print(alphabetArray)
60  g = open("decrypted.txt", 'w')
61  g.write(encrypted)
```

H.2 Copper RSA

1. I used the fact that the exponent is a low number (3) to implement the Chinese Remainder Theorem, obtaining a value X which is the numerical value of the original plaintext, after going through the quadratic equation, to the power of 3.
2. I found the quadrated value of the original plaintext by taking the cube root of X
3. I then used binary search to solve the quadratic equation and find the original plaintext, in numerical form.
4. I then converted the plaintext to byte form, and then to a string, to find the original plaintext.

```
36   def decrypt() :
37       X = c1*n2*n3*modInverse(n2*n3, n1) + c2*n1*n3*modInverse(n1*n3, n2) + c3*n1*n2*modInverse(n1*n2, n3)
38       pt = X % (n1*n2*n3)
39       print(pt)
40       M = findCubeRoot(pt)
41       print(M)
42       print("\n")
43       n = findQuadratic(1, M, M)
44       print(n)
45       coded = long_to_bytes(n)
46       print(coded.decode())
47       """
48       for y in range(M // 2):
49           if 4 * (y ** 2) + 521 * y + 47829 == M:
50               coded = long_to_byte(y)
51               print(coded.decode())
52               break
53       print("not found")
54       """
```