**Cryptography Assignment - 9**

**Pramod Aravind Byakod**

**Question 1:**

**Suppose a RSA public key is: n = 1259531756783983515701499777642110356794201569384295868500005799617750548880147110509521944049285041602433244172023804646590835427723055191592144638318476432867385429617360121**

**e = 65537**

**(a) What is the cipher text if you encrypt your student ID number using the textbook RSA algorithm?**

**Program:**

n = 1259531756783983515701499777642110356794201569384295868500005799617750548880147110509521944049285041602433244172023804646590835427723055191592144638318476432867385429617360121

e = 65537

id = 113436879

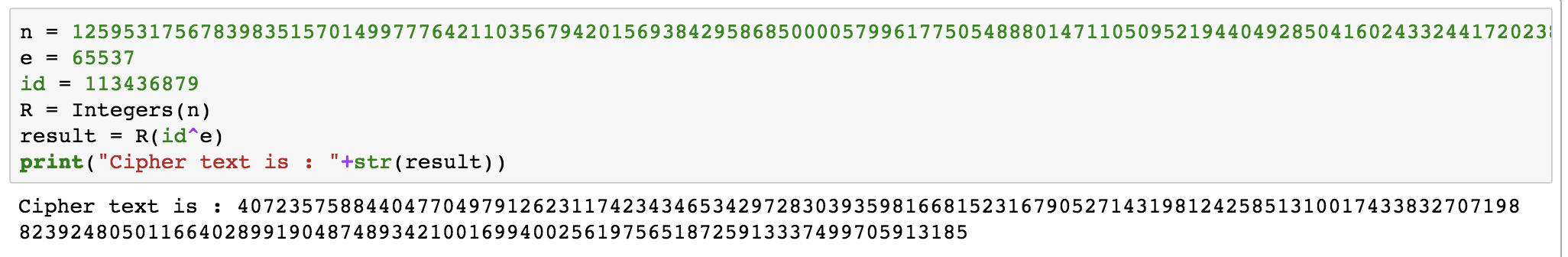
R = Integers(n)

result = R(id^e)

print("Cipher text is : "+str(result))

**Output:**

Cipher text is : 407235758844047704979126231174234346534297283039359816681523167905271431981242585131001743383270719882392480501166402899190487489342100169940025619756518725913337499705913185



**(b) Explain why the textbook RSA is not safe for encrypting student ID numbers. Is the attack cipher text only, known plaintext, chosen plaintext, or chosen cipher text? How can you improve the security of the textbook RSA?**

For short messages m we might have me < n where me is the cipher text and n is the product of p & q, making it possible to decrypt it as eth root extraction which leads to the finding of the plain text for the given cipher text also Chinese reminder theorem can be implemented for breaking the algorithm. Thus, for making the decryption difficult we need to have the messages longer as a result of which the logarithm operation is difficult, for making the message length longer we can follow a technique called padding which adds random digits to the message. Attack is cipher text and known plaintext.

**Question 2:**

**Examine the certificates of your browser, and find the RSA public key, n and e (in decimal) for** [**https://www.google.com**](https://www.google.com)

Public key (Hexadecimal number):

9c 2a 04 77 5c d8 50 91 3a 06 a3 82 e0 d8 50 48 bc 89 3f f1 19 70 1a 88 46 7e e0 8f c5 f1 89 ce 21 ee 5a fe 61 0d b7 32 44 89 a0 74 0b 53 4f 55 a4 ce 82 62 95 ee eb 59 5f c6 e1 05 80 12 c4 5e 94 3f bc 5b 48 38 f4 53 f7 24 e6 fb 91 e9 15 c4 cf f4 53 0d f4 4a fc 9f 54 de 7d be a0 6b 6f 87 c0 d0 50 1f 28 30 03 40 da 08 73 51 6c 7f ff 3a 3c a7 37 06 8e bd 4b 11 04 eb 7d 24 de e6 f9 fc 31 71 fb 94 d5 60 f3 2e 4a af 42 d2 cb ea c4 6a 1a b2 cc 53 dd 15 4b 8b 1f c8 19 61 1f cd 9d a8 3e 63 2b 84 35 69 65 84 c8 19 c5 46 22 f8 53 95 be e3 80 4a 10 c6 2a ec ba 97 20 11 c7 39 99 10 04 a0 f0 61 7a 95 25 8c 4e 52 75 e2 b6 ed 08 ca 14 fc ce 22 6a b3 4e cf 46 03 97 97 03 7e c0 b1 de 7b af 45 33 cf ba 3e 71 b7 de f4 25 25 c2 0d 35 89 9d 9d fb 0e 11 79 89 1e 37 c5 af 8e 72 69

n (2048 bits) = 19713895149719550196537065661910573762693934593220985668782860735427060889140793885919063737778303548724916253252606564904177491762533295616984617709378739783748100146882543612565825906799282133510087546060971220666055151463898734279731009956582933624646298029265838127046200538496591314458940937082185029845612274584845875286257057247598474925565775989866310636633768255501748172403430876460228793912189332026189491067186811703150477068536877439284697584041860237489395099402658887745588613142391209024263265842301844868193180477031165936332420984796347731387363914950895491332976177715889375379088870580457661428329

e (24 bits) = 65537

Program screenshot:

