

**Secure Programming**

**RSA Challenges**

**Primrose Nyagwaya (B0099882)**

***School of Informatics***

***Department of Informatics and Engineering***

***Institute of Technology, Blanchardstown***

***Dublin 15.***

**Actual Word Count:1313**

**Bachelor of Science**

**Computing in Information Technology**

**19/04/2019**

Contents

[Table of Figure 2](#_Toc6677217)

[Introduction 3](#_Toc6677218)

[RSA Level 0: None Shall Enter 3](#_Toc6677219)

[RSA Level 1 4](#_Toc6677220)

[Level 1 rsa1.txt 5](#_Toc6677221)

[Level 1 Encrypted Message 5](#_Toc6677222)

[Solution 5](#_Toc6677223)

[RSA Level 2 6](#_Toc6677224)

[RSA Level 2 Solution 9](#_Toc6677225)

[RSA Level 3 9](#_Toc6677226)

[RSA Level 3 solution 11](#_Toc6677227)

[RSA Level 4 12](#_Toc6677228)

[RSA Level 4 Solution 13](#_Toc6677229)

[RSA Level 4,5 14](#_Toc6677230)

[RSA Level 4.5View Hint 14](#_Toc6677231)

[RSA Level4.5 Solution 15](#_Toc6677232)

[RSA Level 5 15](#_Toc6677233)

[Level 5 keytxt 16](#_Toc6677234)

[RSA Level 5 Solution 16](#_Toc6677235)

[RSA Level 6 17](#_Toc6677236)

[RSA Level 6 Key text 17](#_Toc6677237)

[RSA Level 6 Solution 18](#_Toc6677238)

[RSA Level 7 19](#_Toc6677239)

[RSA Level 7 Challenge Public Key 19](#_Toc6677240)

[RSA Level 7 Challenge Solution 19](#_Toc6677241)

[Conclusion 20](#_Toc6677242)

[References 20](#_Toc6677243)

[References 20](#_Toc6677244)

## Table of Figure

[Figure 1 Level 0 4](#_Toc6581363)

[Figure 2 Level 1 Challenge 5](#_Toc6581364)

[Figure 3 Printing decrypted cyphertext and plain text 5](#_Toc6581365)

[Figure 4 Level 1 rsa1.txt 6](#_Toc6581366)

[Figure 5 Level 1 Solution 6](#_Toc6581367)

[Figure 6 Level 2 Challenge 7](#_Toc6581368)

[Figure 7 Level 2 Solution 9](#_Toc6581369)

[Figure 8 Level 3 Challenge 10](#_Toc6581370)

[Figure 9 Level 3 key 11](#_Toc6581371)

[Figure 10 printing n,e and d 11](#_Toc6581372)

[Figure 11 Level 3 solution 12](#_Toc6581373)

[Figure 12 RSA Level 4 Challenge 13](#_Toc6581374)

[Figure 13 Level 4 mykey3 13](#_Toc6581375)

[Figure 14 Level 4 Solution 14](#_Toc6581376)

[Figure 15 level 4.5 15](#_Toc6581377)

[Figure 16 Level 4.5 Hint 15](#_Toc6581378)

[Figure 17 Level 4.5 Solution 16](#_Toc6581379)

[Figure 18 RSA Level 5 Challenge 16](#_Toc6581380)

[Figure 19 Level 5 Keytxt 17](#_Toc6581381)

[Figure 20 Level 5 Solution 17](#_Toc6581382)

[Figure 21 RSA Level 6 Challenge 18](#_Toc6581383)

[Figure 22 Level 6 Solution 19](#_Toc6581384)

[Figure 23 RSA Level 7 Challenge 20](#_Toc6581385)

[Figure 24 level 7 public key 20](#_Toc6581386)

[Figure 25 Level 7 solution 21](#_Toc6581387)

# Introduction

We are to solve a list of RSA challenges the number not known because you can only see the next challenge after solving the first one. RSA is said to be one of the mostly used encryption because of its ability to provide high level of encryption. RSA is the abbreviation of the author Rivest–Shamir–Adleman. (What is RSA, n.d.)

The website do the challenges is <https://zerodays.ctfd.io/challenges>

# RSA Level 0: None Shall Enter

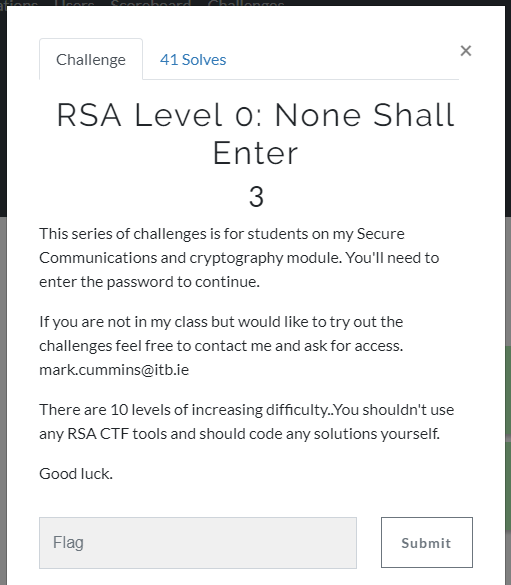


Figure 1 Level 0

Level 0 was all about entering the password to be able to do the challenges.Password entered was letmedothechallenges. After entering the password RSA Level 1opened.

## RSA Level 1

This level is where the challenges begin. To begin solving this challenge a file rsa.txt was provided, in the file there were list of values n,e,d,p.q. In this challenge we are told to encrypt he message RSA isn’t really hard as seen in Figure 2 below.

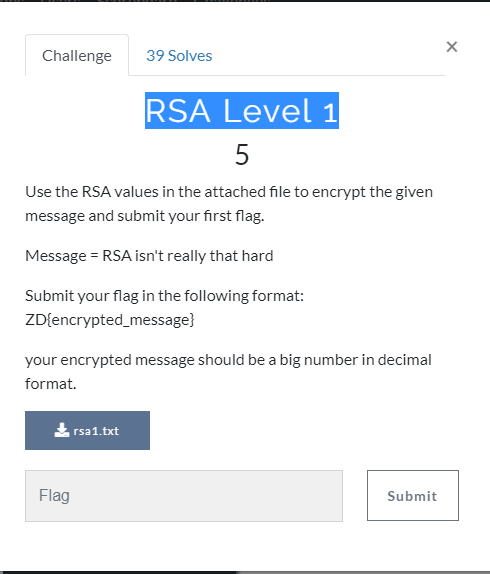


Figure 2 Level 1 Challenge

To encrypt the message RSA isn’t really hard

Figure 3 Printing decrypted cyphertext and plain text

### Level 1 rsa1.txt

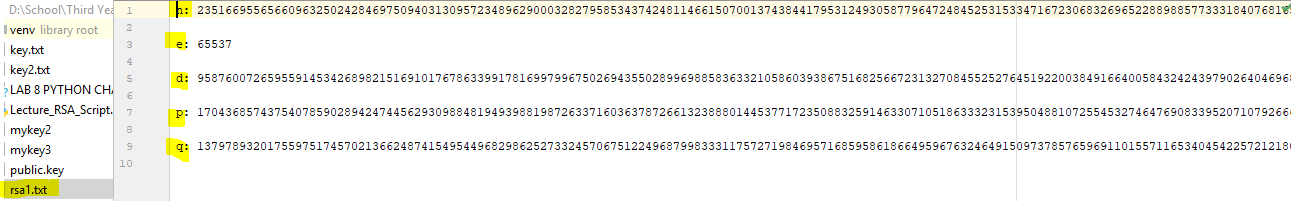


Figure 4 Level 1 rsa1.txt

### Level 1 Encrypted Message

Below is the formula used for the encryption. Of course binascii was imported at the beginning and a functions string2int was called and when we run the program the cyphertext was printed.

m = string2int(message)  
ciphertext = pow(m, e, n) print (ciphertext)

### Solution

The cyphertext of the message RSA isn’t really hard is

The ciphertext is 13309476856206179288137278795001286676504235122200291222905951541015281640474228799375180129564959032261555135231635439690367282451635413048574322588643043250005501837597608399627442074603517951858976430767446724730937928672932493206869274420288717036712376949408229648116702610597844919828482630797157003777363091998366855062763360538948110895070725322039940644906900772757193759215740687066380017485804644723367158972689710477927318380335919282326398046586751715463059075476044138690978986063001880735783893361380726584661054926968590764176030209214513123458853087059980258593405395678238799024217478961749328706800



Figure 5 Level 1 Solution

# RSA Level 2

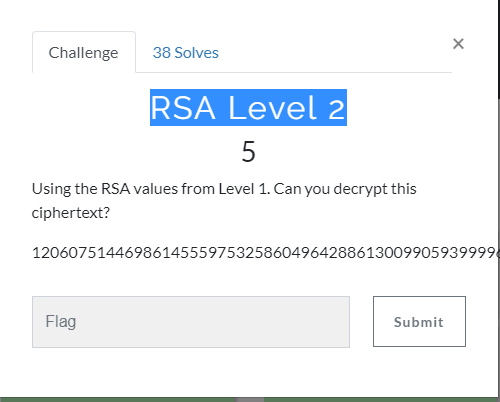
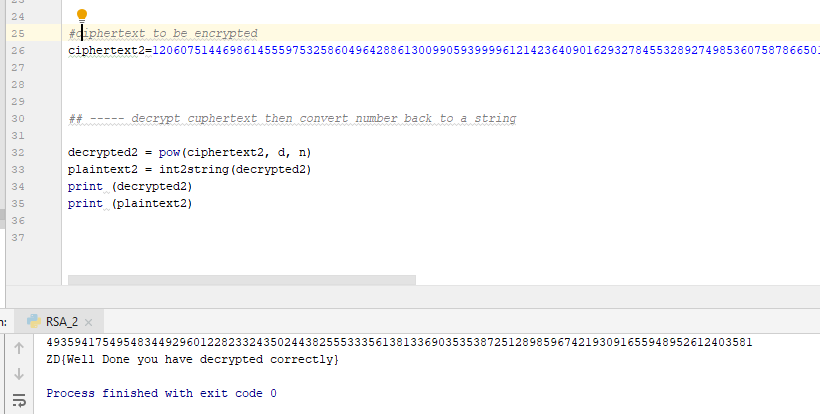


Figure 6 Level 2 Challenge

For RSA level two we have to use the values provided in Level one to decrypt the cyphertext. Below are the values provided in level 1

n: 23516695565660963250242846975094031309572348962900032827958534374248114661507001374384417953124930587796472484525315334716723068326965228898857733318407681656604325744994115789416012096318656034667361976251100005599211469354510367804546831680730445574797161330145320706346512982316782618118878428893337849886890813813050423818145497040676697510093220374542784895778086554812954376689653727580227087363619223145837820593375994747273662064715654881379557354513619477314410917942381406981452545764657853425675230343749326640073923166795823683203941972393206970228647854927797483660176460658959810390117898333516129469397  
  
e: 65537  
  
d: 9587600726595591453426898215169101767863399178169979967502694355028996988583633210586039386751682566723132708455252764519220038491664005843242439790264046968625524201298469258242007220372280857992847470031480553726983707671745159488070659256258857978134570602562717609180653377092666963295822401721181836384326336158085408894694549470434424808812412260714422693522311366681659987060925945689943522825747715934700712908720597323076354591388316712970722935035250113120539406041972135508540472211484760814740089404942374666334486855389174327639061106567747152104666795257954039030591097174242386069752606041990644663125  
  
p: 170436857437540785902894247445629309884819493988198726337160363787266132388801445377172350883259146330710518633323153950488107255453274647690833952071079266615535462115718628529996080297946386916054952930963525522668498855400580516951309863503734146131687670337990358661269686138903141878297721385390421204703  
  
q: 137978932017559751745702136624874154954496829862527332457067512249687998333117572719846957168595861866495967632464915097378576596911015571165340454225721218087595428364080801400548238088288742249145662369868461078198744980520572785232341389134600070345564258064842348774203427257497319140459851255774165194699

RSA Level 2 calls for the given cyphertext to be decrypted. For clarity sake I decided to print the decrypted cyphertext and the plain text which was converted from number to string as shown below. This was just to get a better understanding of what the formula has been doing when implemented.



*## To run type python rsa.py from the commandline (assuming you've pythonh installed***import** binascii  
  
**def** string2int(my\_str):  
 **return** int(binascii.hexlify(my\_str), 16)  
  
**def** int2string(my\_int):  
 **return** binascii.unhexlify(format(my\_int, **"x"**).encode(**"utf-8"**)).decode(**"utf-8"**)  
  
*# -------------------------------------------------------------------------*n = 23516695565660963250242846975094031309572348962900032827958534374248114661507001374384417953124930587796472484525315334716723068326965228898857733318407681656604325744994115789416012096318656034667361976251100005599211469354510367804546831680730445574797161330145320706346512982316782618118878428893337849886890813813050423818145497040676697510093220374542784895778086554812954376689653727580227087363619223145837820593375994747273662064715654881379557354513619477314410917942381406981452545764657853425675230343749326640073923166795823683203941972393206970228647854927797483660176460658959810390117898333516129469397  
e = 65537  
d = 9587600726595591453426898215169101767863399178169979967502694355028996988583633210586039386751682566723132708455252764519220038491664005843242439790264046968625524201298469258242007220372280857992847470031480553726983707671745159488070659256258857978134570602562717609180653377092666963295822401721181836384326336158085408894694549470434424808812412260714422693522311366681659987060925945689943522825747715934700712908720597323076354591388316712970722935035250113120539406041972135508540472211484760814740089404942374666334486855389174327639061106567747152104666795257954039030591097174242386069752606041990644663125  
p = 170436857437540785902894247445629309884819493988198726337160363787266132388801445377172350883259146330710518633323153950488107255453274647690833952071079266615535462115718628529996080297946386916054952930963525522668498855400580516951309863503734146131687670337990358661269686138903141878297721385390421204703  
q = 137978932017559751745702136624874154954496829862527332457067512249687998333117572719846957168595861866495967632464915097378576596911015571165340454225721218087595428364080801400548238088288742249145662369868461078198744980520572785232341389134600070345564258064842348774203427257497319140459851255774165194699  
  
  
*#ciphertext to be encrypted*ciphertext2=12060751446986145559753258604964288613009905939999612142364090162932784553289274985360758786650150438909964774489580594646537566480131089578622598287608280958826485540308546699201437758196408375494069165098540792161560520821702762571130453590350894456351542803892333747581478520236785140328470857700892653885938706372579150142033036479912415927913687580857324142121399292745144946430853334522298591089627330825281302600321017113877555269059227306217943736817867883165263245345825896646180907782570834990026416074994773400197992710509066779111650705105160806274734336871715369410205954061055784649468877193662056626998  
  
  
  
*## ----- decrypt cuphertext then convert number back to a string*decrypted2 = pow(ciphertext2, d, n)  
plaintext2 = int2string(decrypted2)  
print (plaintext2)

Therefore to decrypt the cyphertext given in RSA level two using the values given in RSA level 1.We decrypt the cyphertext to a number and then convert the number back to a string.

### RSA Level 2 Solution

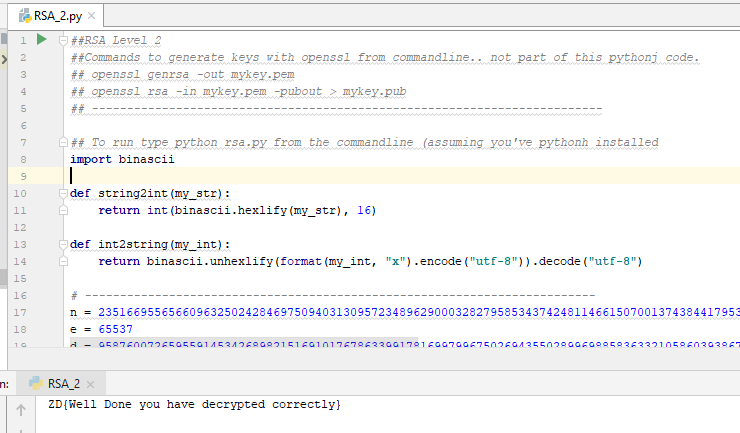


Figure 7 Level 2 Solution

The decrypted encryption for RSA Level 2 is ZD{Well Done you have decrypted correctly}

# RSA Level 3

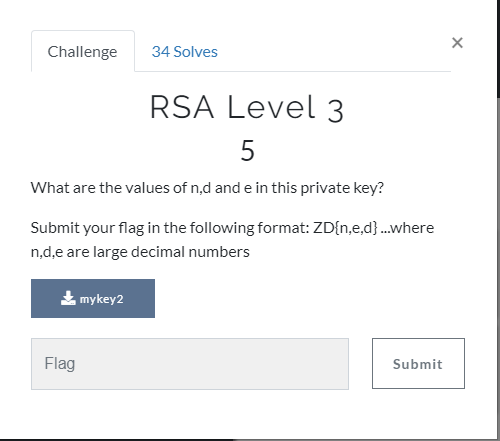


Figure 8 Level 3 Challenge

For the above challenge RSA private key with the name mykey2 was provided to help solve RSA Level 3



Figure 9 Level 3 key

RSA Level 3 requires us to print n,e and d. For this challenge we have to import a crypto public key. import\_key() is for loading the key is loaded from memory.

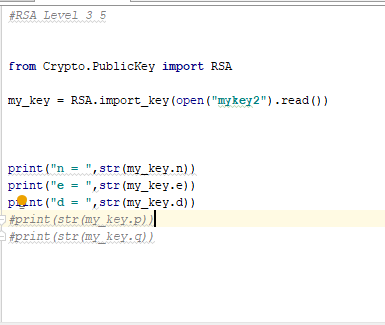


Figure 10 printing n,e and d

### RSA Level 3 solution

Below are the keys n,e and d printed after running the program.

n = 25804750360904248224329381618104859031736073222395248860499133051128513648896523325052435992237099000924849024564826844234697029915822571039994777064514824635243492963301478168480948385273501322600327794162141312212647635059765199895820389754402745012084674140342116327728380193544092902468871988407101113210818201750378837933282937750636399588218292079918162950294282004250547116764896324653457065316101365613347914317959505469586731762453963212022535424803572822031093301253062385003468990260150084465026582297271243286096414015105084255301102689693963566336024015793743350044649378818363190804537712326729609503119

e = 65537

d = 25564173244610971210351548452585197331194535758994673657884351015114666175874020376311922413781312689534869581593541780807512387877301314484073132668426835287971970400465565582293880630212968962130535153634911375813942663745522611172864711921423815900248860235223648392420965446176110834882493594752879241898541637225074107295433979711279703743206408857411263301020537809364614125840424406332180800207630227259634428188684178985515460114123914453650465520564263249152086786817100938246123146166512963816583276055667323176816331368813221646162976565638562106256552057858358804329568634212548081342870432443412497223185

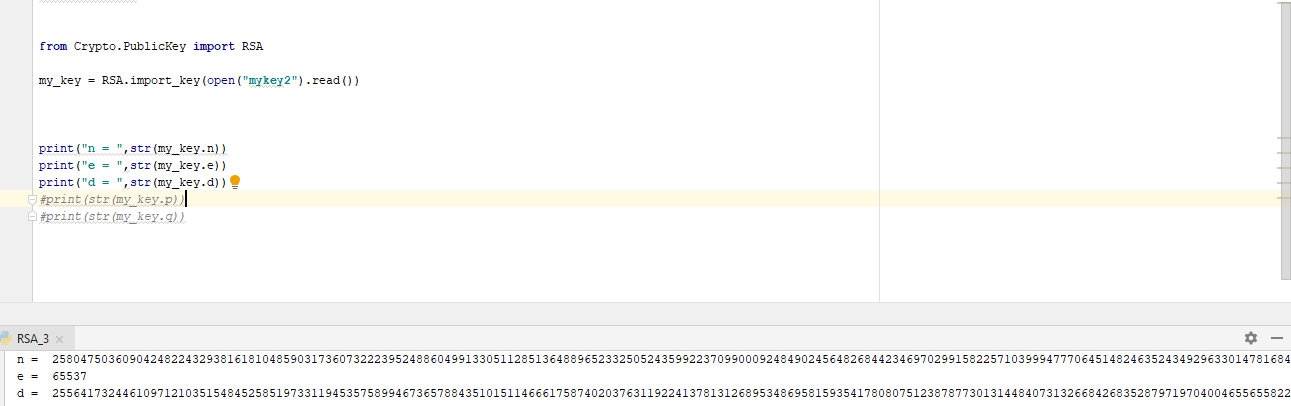


Figure 11 Level 3 solution

# RSA Level 4

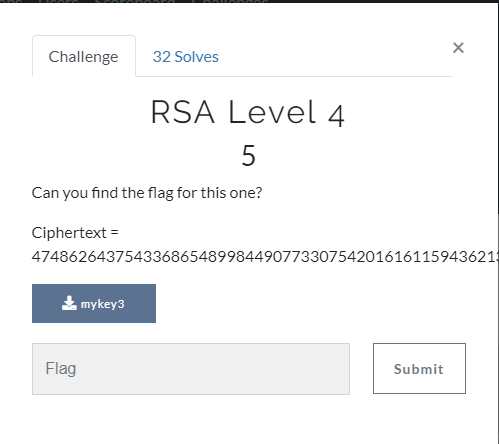


Figure 12 RSA Level 4 Challenge

Given the above ciphertext above and mykey3 below to find the flag.



Figure 13 Level 4 mykey3

### RSA Level 4 Solution

Given the cyphertext to find the flag. I first decrypted the cyphertext and converted the int to string and below is the flag found.

*ZD{OK time to move onto some harder stuff}*

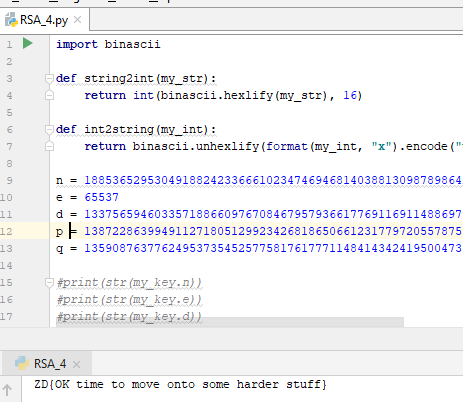


Figure 14 Level 4 Solution

# RSA Level 4,5

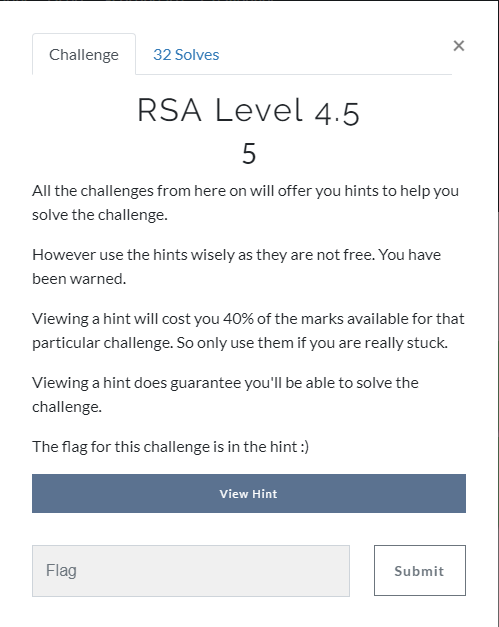


Figure 15 level 4.5

## RSA Level 4.5View Hint

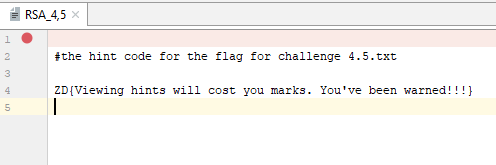


Figure 16 Level 4.5 Hint

### RSA Level4.5 Solution

The solution for level 4 was all in the hint. Clicking the hint and giving it a try that was that was needed for this challenge.

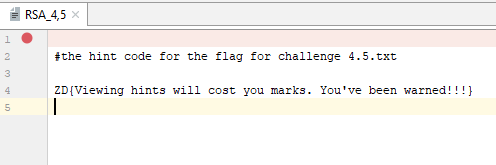


Figure 17 Level 4.5 Solution

# RSA Level 5

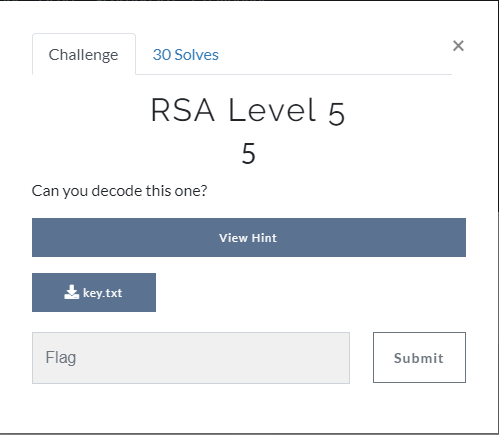
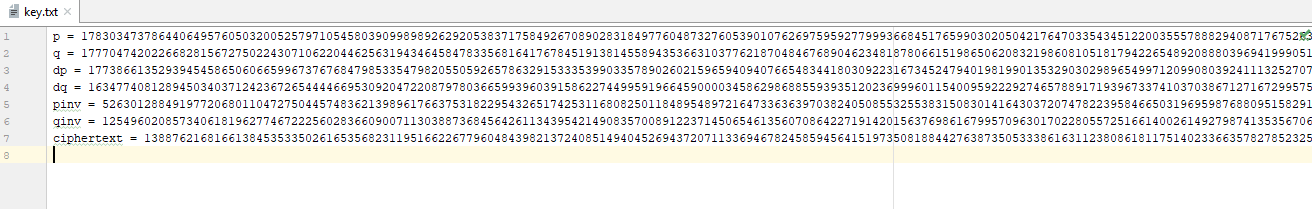


Figure 18 RSA Level 5 Challenge

## Level 5 keytxt

Figure 19 Level 5 Keytxt

### RSA Level 5 Solution

The hint had a website <https://en.wikipedia.org/wiki/RSA_(cryptosystem)> The values found in the key.txt are named differently. After decrypting and converting int to string we got the below flag

*Those extra private key values are meant to make it easier?*

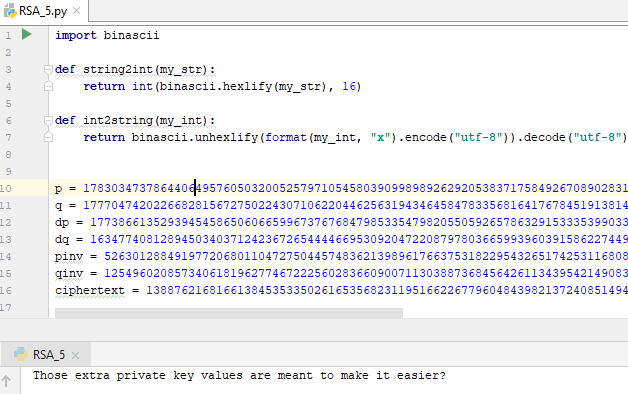


Figure 20 Level 5 Solution

# RSA Level 6

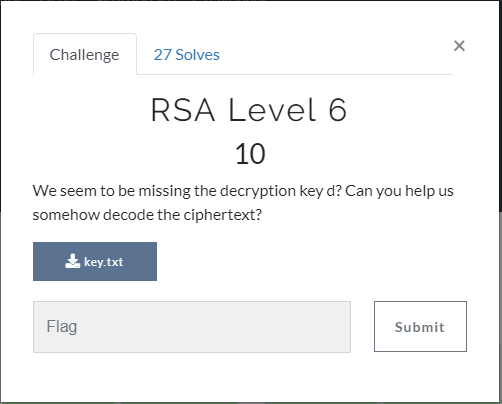
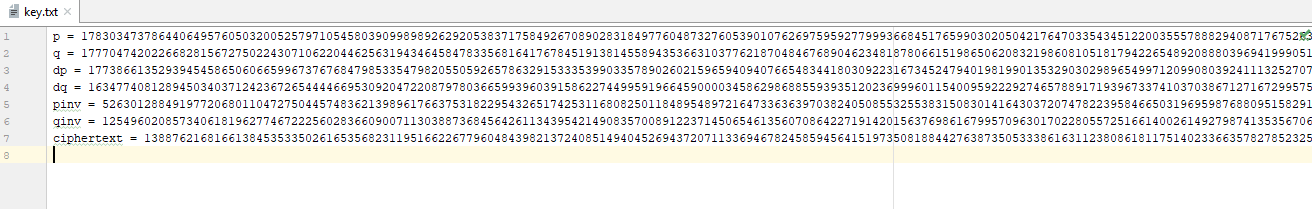


Figure 21 RSA Level 6 Challenge

## RSA Level 6 Key text



### RSA Level 6 Solution

For this challenge I looked at this website <https://en.m.wikipedia.org/wiki/Chinese_remainder_theorem>. This challenge was solved using the Chinese remainder theorem. The flag is *You are doing very well, you must be starting to understand RSA by now!*

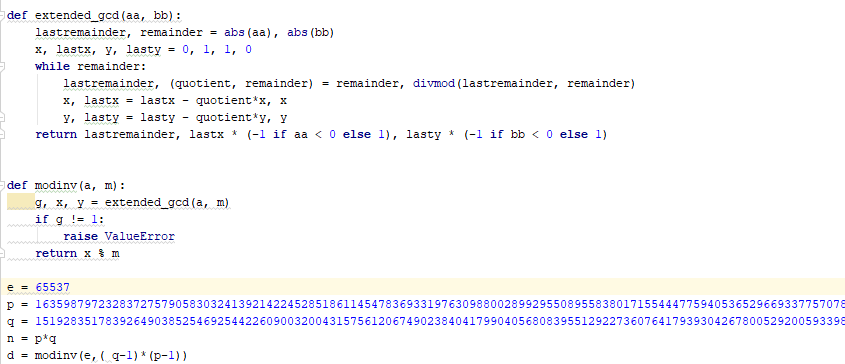


Figure 22 Level 6 Solution

# RSA Level 7

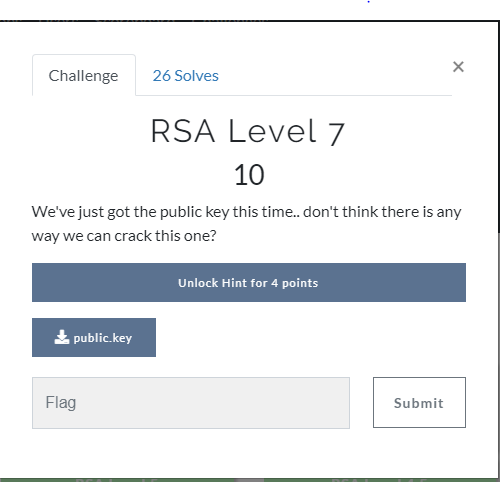
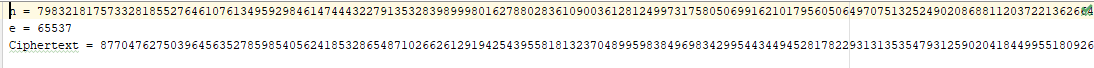


Figure 23 RSA Level 7 Challenge

## RSA Level 7 Challenge Public Key

Figure 24 level 7 public key

### RSA Level 7 Challenge Solution

The Flag found for RSA Level 7 is *Only 4 more challenges to go!*

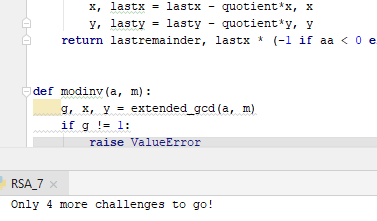


Figure 25 Level 7 solution

# Conclusion

Solving the challenges was one of the interesting parts of my learning journey. The first one was a bit easy because there were self-explanatory but as we moved further the challenges were challenging. I have learnt a lot from these challenges, the challenges imparted new knowledge. The first time I heard of the challenges from my lecturer Mark I just thought there was no way I could ever solve one but after following through I am glad I gave it a try and it was worth it.

(Modular\_inverse#Python, n.d.) (ctf-wtf, n.d.)

# References

# References

*Chinese\_remainder\_theorem*. (n.d.). Retrieved from https://en.m.wikipedia.org: https://en.m.wikipedia.org/wiki/Chinese\_remainder\_theorem

*Crypto.PublicKey.RSA-modul*. ( 2012, May 24 Thu). Retrieved from dlitz.net: https://www.dlitz.net/software/pycrypto/api/current/Crypto.PublicKey.RSA-module.html

*ctf-wtf*. (n.d.). Retrieved from ctftime.org: https://ctftime.org/ctf-wtf/

*cyber-security-capture-the-flag-ctf-what-is-it*. (n.d.). Retrieved from blogs.cisco.com: https://blogs.cisco.com/perspectives/cyber-security-capture-the-flag-ctf-what-is-it

LLC, T. N. (n.d.). */cryptography/what-is-rsa/*. Retrieved from ctf101.org: https://ctf101.org/cryptography/what-is-rsa/

*Modular\_inverse#Python*. (n.d.). Retrieved from /www.rosettacode.org: http://www.rosettacode.org/wiki/Modular\_inverse#Python

*public\_key*. (n.d.). Retrieved from pycryptodome: https://pycryptodome.readthedocs.io/en/latest/src/public\_key/public\_key.html

*What is RSA*. (n.d.). Retrieved from tf101.org: https://ctf101.org/cryptography/what-is-rsa/