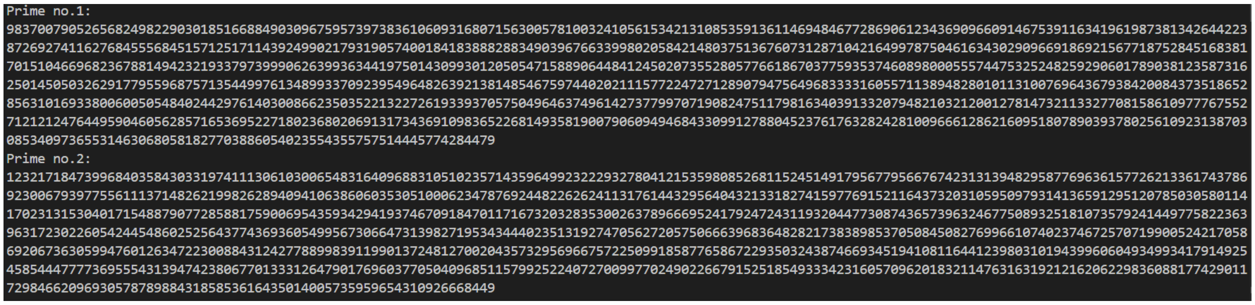
Question-1

Overview

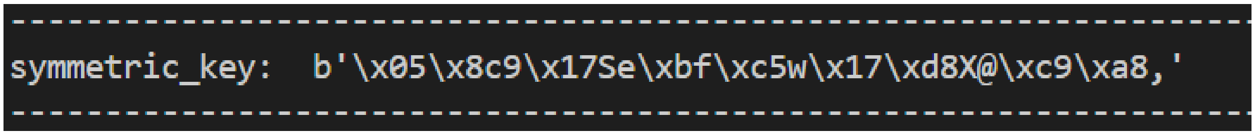
This code is giving a cryptographic system that combine RSA(asymmetric encryption) and Salsa20 (symmetric encryption) to ensure secure communication between two persons, Alice and Bob. RSA algorithm securely exchanges a symmetric key, this key is then used with salsa20 for encryption and decryption of the message.

Take two large prime numbers as input
These below are the two prime numbers I used:



```
Prime no.1:
9837007905265682498229030185166884903096759573973836106093168071563005781003241056153421310853591361146948467728690612343690966091467539116341961987381342644223
8726927411627684555684515712517114392499021793190574001841838882883490396766339980205842148037513676073128710421649978750461634302909669186921567718752845168381
7015104669682367881494232193379739990626399363441975014309930120505471588906448412450207355280577661867037759353746089800055574475325248259290601789038123587316
2501450503262917795596875713544997613489933709239549648263921381485467597440202111577224727128907947564968333316055711389482801011310076964367938420084373518652
8563101693380060050548402442976140300866235035221322726193393705750496463749614273779970719082475117981634039133207948210321200127814732113327708158610977767552
7121212476449590460562857165369522718023680206913173436910983652268149358190079060949468433099127880452376176328242810096661286216095180789039378025610923138703
085340973655314630680581827703886054023554355757514445774284479
Prime no.2:
1232171847399684035843033197411130610300654831640968831051023571435964992322293278041215359808526811524514917956779566767423131394829587769636157726213361743786
9230067939775561113714826219982628940941063860603530510006234787692448226262411317614432956404321331827415977691521164373203105950979314136591295120785030580114
1702313153040171548879077285881759006954359342941937467091847011716732032835300263789666952417924724311932044773087436573963246775089325181073579241449775822363
9631723022605424454860252564377436936054995673066473139827195343444023513192747056272057506663968364828217383898537050845082769966107402374672570719900524217058
6920673630599476012634722300884312427788998391199013724812700204357329569667572250991858776586722935032438746693451941081164412398031019439960604934993417914925
4585444777736955543139474238067701333126479017696037705040968511579925224072700997702490226679152518549333423160570962018321147631631921216206229836088177429011
729846620969305787898843185853616435014005735959654310926668449
```

Alice generates a 16-byte random symmetric key using the Random.get_random_bytes(16) function. This key will later be encrypted using RSA and securely transmitted to Bob. This key later used to encrypt and decrypt the data.

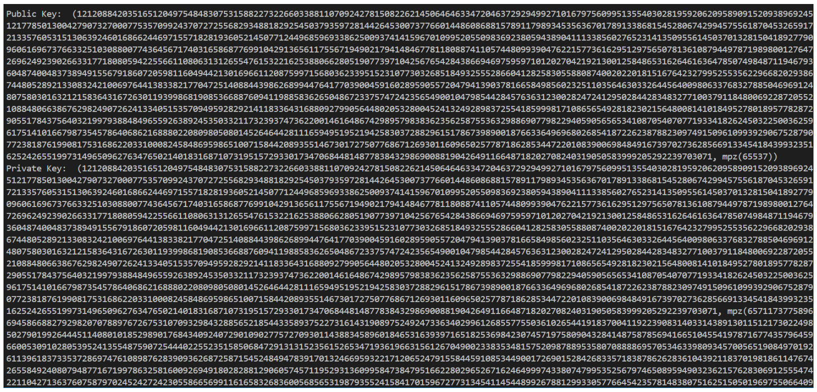


```
-----------------------------------------------------------------------
symmetric_key:  b'\x05\x8c9\x17Se\xbf\xc5w\x17\xd8X@\xc9\xa8,'
-----------------------------------------------------------------------
```

Now Bob generates the RSA key pair using the given prime numbers.

- Compute n = p * q
- Compute Euler’s totient function: z = (p-1) * (q-1)
- Choose a public exponent e = 65537
- Compute the private exponent d, the modular inverse of e mod z

Public key and Private key

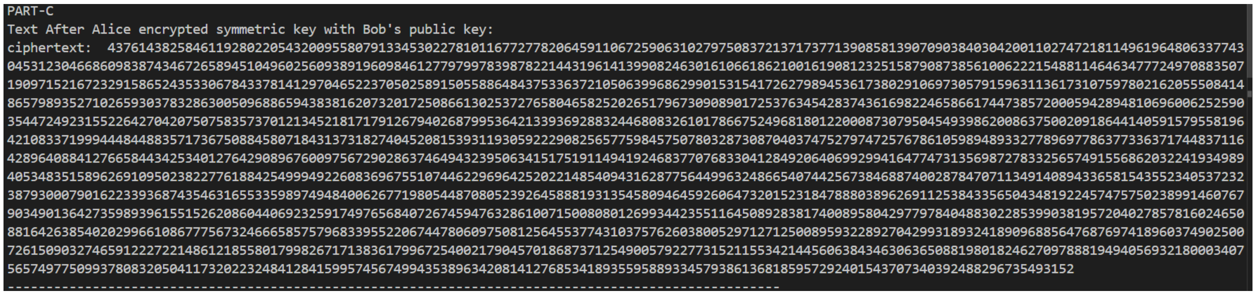


```
Public Key:  (1212088420351651204975484830753158822732266033881107092427815082262145064646334720463729294992710167975609951355403028159206209589091520938969245
1217785013004279073270007753570992437072725568293488182925450379359728144264530073776601448608688157891179893453563670178913386815452806742994575561870453265917
2133576053151306392460168662446971557182819360521450771244968596933862500937414159670109952055098369238059438904111338560276523141350955614503701328150418927790
9606169673766332510308800774364567174031658687769910429136561175567194902179414846778118088741105744809939047622157736162951297565078136108794497871989800127647
2696249239026633177180805942255661108063131265547615322162538806628051907739710425676542843866946975959710120270421921300125848653162646163647850749848711946793
6048740048373894915567918607205981160494421301696611208759971568036233951523107730326851849325552866041282583055880874002022018151676423279952553562296682029386
7448052892133083242100697644138338217704725140884439862689944764177039004591602895905572047941390378166584985602325110356463033264456400980633768327885046969124
8075803016321215836431672630119399868190853668876094119885836265048672337574724235654900104798544284576363123002824724129502844283483277100379118480069228720552
1088480663867629824907262413340515357094959282921411833643168809279905644802053280045241324928983725541859998171086565492818230215648008141018495278018957782872
9055178437564032199793884849655926389245350332117323937473622001461648674298957983836235625875536329886907798229405905656534108705407077193341826245032250036259
6175141016679873545786406862168880220809805080145264644281116594951952194258303728829615178673989001876633649696802685418722623878823097491509610993929067528790
7723818761990817531686220331000824584869598651007158442089355146730172750776867126930116096502577871862853447220108390069848491673970273628566913345418439932351
6252426551997314965096276347650214018316871073195157293301734706844814877838432986900881904264911664871820270824031905058399920529223970 3071, mpz(65537))
Private Key:  (121208842035165120497548483075315882273226603388110709242781508226214506464633472046372929499271016797560995135540302815920620958909152093896924
5121778501300427907327000775357099243707272556829348818292545037935972814426453007377660144860868815789117989345356367017891338681545280674299457556187045326591
7213357605315130639246016866244697155718281936052145077124496859693386250093741415967010995205509836923805943890411133856027652314135095561450370132815041892779
0960616967376633251030880077436456717403165868776991042913656117556719490217941484677811808874110574480993904762215773616295129756507813610879449787198980012764
7269624923902663317718080594225566110806313126554761532216253880662805190773971042567654284386694697595971012027042192130012584865316264616364785074984871194679
3604874004837389491556791860720598116049442130169661120875997156803623395152310773032685184932555286604128258305588087400202201815167642327995255356229668202938
6744805289213308324210069764413833821770472514088443986268994476417703900459160289590557204794139037816658498560232511035646303326445640098063376832788504696912
4807580301632121583643167263011939986819085366887609411988583626504867233757472423565490010479854428457636312300282472412950284428348327710037911848006922872055
2108848066386762982490726241334051535709495928292141183364316880927990564480205328004524132492898372554185999817108656549281823021564800814101849527801895778287
2905517843756403219979388484965592638924535033211732393747362200146164867429895798383623562587553632988690779822940590565653410870540707719334182624503225003625
9617514101667987354578640686216888022080980508014526464428111659495195219425830372882961517867398900187663364969680268541872262387882309749150961099392906752879
0772381876199081753168622033100082458486959865100715844208935514673017275077686712693011609650257787186285344722010839006984849167397027362856691334541843993235
1625242655199731496509627634765021401831687107319515729330173470684481487783843298690088190426491166487182027082403190505839992052922397 03071, mpz(6571173775896
6945866882792982070788976726753107093298432885652185443358937522731614319089752492473363402996126855775503610265441918370041192239083140331438913011512173022498
5027901992644451140801018529890176843409240729010902775727093011438834589601846531639397165182536984230745719758090432841487587856941665104554197871677435796459
6600530910280539524135548759072544402255235158506847291313152356152653471936196631561267049002338353481575209878895358070888869570534633980934570056519084970192
6113961837335372869747610898762839093626872587154524849478391701324669593221712065247915584459108534490017269015284268335718387862628361043921183701981861147674
2655849240807948771671997863258160092694918028288129060574571195293136099584738479516622802965267162464999743380747995352567974650895949032362157628306912555474
2211042713637607587970245242724230558665699116165832683600568565319879355241584170159672773134541145448992678812993305776645423578148380751625150501969755066409
```

Now Alice encrypts the symmetric key using Bob’s public key (n, e). The symmetric key (a byte string) is converted to an integer before encryption

Encryption Formula:

- C=M^e(modn)

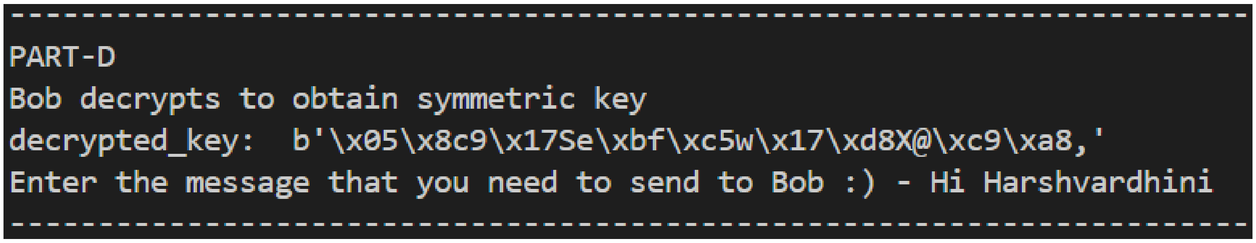


```
PART-C
Text After Alice encrypted symmetric key with Bob's public key:
ciphertext:  437614382584611928022054320095580791334530227810116772778206459110672590631027975083721371737713908581390709038403042001102747218114961964806337743
0453123046686098387434672658945104960256093891960984612779799783987822144319614139908246301610661862100161908123251587908738561006222154881146463477724970883507
1909715216723291586524353306784337814129704652237050258915055886484375336372105063996862990153154172627989453617380291069730579159631136173107597802162055508414
8657989352710265930378328630050968865943838162073201725086613025372765804658252026517967309089017253763454283743616982246586617447385720005942894810696006252590
3544724923155226427042075075835737012134521817179126794026879953642133936928832446808326101786675249681801220008730795045493986200863750020918644140591579558196
4210833719994448448835717367508845807184313731827404520815393119305922290825657759845750780328730870403747527974725767861059894893327789697786377336371744837116
4289640884127665844342534012764290896760097567290286374649432395063415175191149419246837707683304128492064069929941647747313569872783325657491556862032241934989
4053483515896269109502382277618842549994922608369675510744622969642520221485409431628775644996324866540744256738468874002878470711349140894336581543552340537232
3879300079016223393687435463165533598974948400626771980544870805239264588819313545809464592606473201523184788803896269112538433565043481922457475750238991460767
9034901364273598939615515262086044069232591749765684072674594763286100715008080126993442355116450892838174008958042977978404883022853990381957204027857816024650
8816426385402029966108677756732466658575796833955220674478060975081256455377431037576260380052971271250089593228927042993189324189096885647687697418960374902500
7261509032746591222722148612185580179982671713836179967254002179045701868737125490057922773152115534214456063843463063650881980182462709788819494056932180003407
5657497750993780832050411732022324841284159957456749943538963420814127685341893559588933457938613681859572924015437073403924882967354931 52
-----------------------------------------------------------------------------------------------------------------------
```

After Bob receives the ciphertext he decrypts the ciphertext using his private key (n, d).

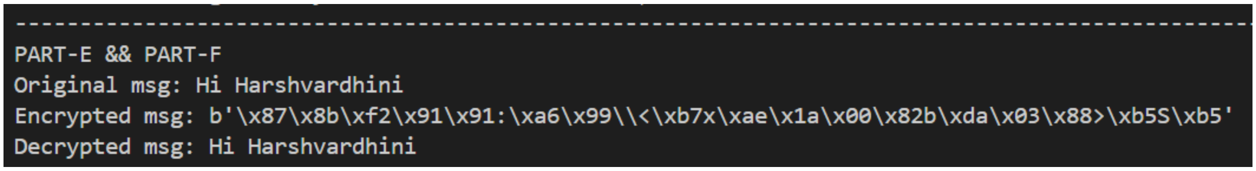
Decryption Formula:

- M = C^d mod n



```
-------------------------------------------------------------------------
PART-D
Bob decrypts to obtain symmetric key
decrypted_key:  b'\x05\x8c9\x17Se\xbf\xc5w\x17\xd8X@\xc9\xa8,'
Enter the message that you need to send to Bob :) - Hi Harshvardhini
-------------------------------------------------------------------------
```

Now Bob encrypts the message using Salsa20 with the decrypted symmetric key. AAs you can see in above image it is asking for the text that needs to be encrypted and send to Alice. Salsa20 generates a nonce and encrypts the message.



```
---------------------------------------------------------------------------------------------
PART-E && PART-F
Original msg: Hi Harshvardhini
Encrypted msg: b'\x87\x8b\xf2\x91\x91:\xa6\x99\\<\xb7x\xae\x1a\x00\x82b\xda\x03\x88>\xb5S\xb5'
Decrypted msg: Hi Harshvardhini
```

Finally Alice decrypts the received ciphertext using the same symmetric key which has been shared to bob previously. In above picture as you can see the original message and decrypted message are same which means securely shared the correct information maintaining the integrity.

This approach ensures confidentiality, integrity, and authenticity in communication.