```
#he variables e and n represent the public key components: e is the public exponent, and n is the modulus.
#ct is cipher text
# b"the string" ---->represent byte object
#int.from_bytes() is a method in Python that converts a sequence of bytes into an integer. It takes two parameters: the bytes to convert and the byte order ('big' or 'little').
#"big" is used as the byte order parameter, which means that the most significant byte is at the beginning of the byte array.
# f = pt.to_bytes(pt.bit_length(),"big")---> to convert back to byte array
#ϕ = (p-1)(q-1),n = p*q
# gympy2 for arithmetic precision.
#msg = pow(signature,e,n) --> decrypts


# e = 65537 #n=1357710300952485642173335487080859767066354702016933829773598844057850532336246692097984881143326212510718962230463038903561779272603619225672552782635164311103480359123
# string = b'CottonCandyCyan'
# t = int.from_bytes(string,"big")
# s = pow(t,e,n)
# print(s)


# ct = 65730873122241583977501942857521551655862419814142537467794605641487141508168112705849405463650251324036527189894154256188004834922930955553552265141530965811260529590272915364
# d = 892512843416404890969189225917012576464519172143244378532361470121370126393153136977200863985957188446938202992120539523534794006068352469868528761114395236552599867193758253282
# n = 100004469514414305589755093689388362495734643502738599480039961817993219308305825163403681010510644997857734671729532909478372020474784776573386509518129821108046890669622337566
# pt = pow(ct,d,n)
# print(pt)
# f = pt.to_bytes(pt.bit_length(),"big")
# print(f)

# p = 285880804885505715088365434821249117591
# q = 223548505081667812376946559606986057893
# phi = (p-1)*(q-1)
# print(phi)

# e = 65537
# phi = 60405307365679473695011595103128719128176824853854670380659392210778713718677526218707654799173300354387162857343160755894610700871253076565877380345494 24
# d = (pow(e,-1,phi))
# print(d)

# ct = 5119667904388996554913075691965298077574451428525647832421139040755532148564677239610936209568188199266675953600861923623367821617006732308102397661359638
# n = 6910782171213996658397253121710630123795235324197483219005527369298065059694620267925472189305202037262152652525687783466320955431741311010423968405213211
# e = 65537
# p = 77875213153738183457169866835133610155772314072823133021250433414301022076677
# q = 88741743249819454803256128724422664779772307593666473114032370133748568669343
# phi = (p-1)*(q-1)
# d = pow(e,-1,phi)
# print(d)
# pt = pow(ct,d,n)
# ct = pt.to_bytes(pt.bit_length(),"big")
# print(ct)

# import gmpy2
# n = 11707471274924331410412020652320882823973935637081327123181242084265975742755345528255937121587508168181175373813507557449750499918842051892515504623503575163940438907183555373 21
# e = 3
# ct = 82895006021025315212194766818959778180071540025724099235413209819072766522409462780959875360431145495518560461472745549100306588074106771552399979846135716227942541180509399193 4
# dec = gmpy2.iroot(ct,e)[0]
# dec_int = int(dec)
# decoded_msg = dec_int.to_bytes(dec_int.bit_length(),"big")
# print(decoded_msg)

#FIND THE MESSAGE FROM THE SIGNATURE =>

# p = 14142576175088243436611365603746552806444693459074754759191307124755807147852306777542433937990254914068238812124659214675533210970490138667155888686046871028957355803052736168 78
# q = 15700605155155474944794846115416606928074123353990903887712229662815525047202777873958621029128492467226439817899770483875319707358855377189482808032161924249558621406803643285 53
# e = 65537
# signature = 21127256972405957923387011048400011431059653140091568867584307537291932809179496459498207512119297630904379849803010372724832459871218654257328645476196081067340401201618
# n = p*q
# msg = pow(signature,e,n)
# print(msg)
# ans = msg.to_bytes(msg.bit_length(),"big")
# print(ans)
```