Cs 411

1a)x^6 + x^5+1

1b)You can see both of the problem’s solution in pdf’s that attached in the file.

2)When we reduct the first digest we get the “TUP,.V” and in the rainbow table its pair is ‘RSURBK’. When we hash the “TUP,.V” value and make a reduction for t times, we get the same as its pairs. So for the first digest our password is ‘RSURBK’. For the second digest we get ‘SQOHSG’ and we could not find it in rainbow table, so we need to hash it again and we should reduct it again until we find that password in rainbowtable. When we reduct ‘SQOHSG’ we get ‘OXCQNR’, and its pair in rainbow table is ‘KJMB?D’. For the third digest we need hash it and reduce it for twice and we get a ‘G.LPYY’ , its rainbow pair is ‘LHBTKY’ and when we check is it is true or not we get the ‘The test is passed’ ☺.So our third password is LHBTKY. In the 4th digest we also found in the rainbow table after hash it and digest it 2 times. Our password is ‘NKMMOI’ and its pair is ‘,PRRHR’ when we check it we get that it is true. For the next digests password is in below and I write that after how many reduction and hash I found. However when I try to solve 7 by manually I understand that I need to write a code. You can see my code in rainbow\_table.py folder

Digest[0]= RSURBK; 0

Digest[1]= KJMB?D 1

Digest[2]= LHBTKY 2

Digest[3]= ,PRRHR 2

Digest[4]=YQLLZB 1

Digest[5]=UPYC!L 2

Digest[6]=EA?!WT 2

Digest[7]= !BURIB -

Digest[8]= EOAFLL -

Digest[9]= TVGH.O -

3a) p\*q is n and we know that p and q are prime. And since cp= kp^e, cp is divisible by k and p. It gives us the fact that if we take gcd of n and cp we can get the p since it is only common element. After calculating p we can also get q by dividing n with p.

3b)when we get the egcd of n and cp we get the p and same for n and cq we get the q. And both are prime.

P=45931572870827881561956359348907507610206389661943955042830107463735662349972722791061733124489189272240130584647783528115980820171539864787746408093357313112739577731204137447836378064314806894823016698356673012051408104260148385313472774949091208519062499104153730513034259359276259882037652364138616878279

Q=162714425555590299907582727396980097499459248735262849323615961586880482025872588759372594209176495245117290424311606087206252036379802275425817797958224085089194159244862342658014419330000145704525568894322127427547322440610723007470284010301550512933411159828723823852019169776225493115796888791016517378863

Since we can calculate the phi of (p-1)\*(q-1) and taking inverse modulo of e we et d value. When we get the cm^d modulo n we get the message, when we convert it we get the message as

‘Insanity is doing the same thing, over and over again, but expecting different results.’

You can check DeterministicRSA.py

3b)

The rest is calculating phi which is (p-1)\*(q-1) and taking inverse modulo of e. Because in the RSA to obtain decryption key we have to obey and apply “e x d = 1 mod(phi(n))” formula.

The result of inverse modulo is d and cipherd = message mod(n). When we cast the bytes to string we obtain: Message: b'Insanity is doing the same thing, over and over again, but expecting different results.'

4a)Since using the original key is forbidden we need a number that gives us the same solution as c. if we add n to c we will have (c+n)^d since its modulo n is also c. c+n should be our query key.

4b)Since program does not allow us to put c in the program we can put c+n since we are taking modulus n. when we put c+n into the program we get ‘4417062138181194427281180308946019215465220166377446460595634949992122329451246465252624109212290215526754540917661445855047970294529542156359354106257673625954139874363269542681174266025116819672400000963736042091690661832469240538804272742072330150888432766371284831039467717420035447095937444167110449516493356646070748237250918629198860417814812318904272404504584949746159024326478932467825813171171369227410600977878532890648929622701524048807393570087991330881487823438654155687665951485310328031281646903154944809921847760184656414912306875762229124493640611111232858898623505870494745371313488332182879196816’

And when we turned them into plaintext we get

-'You discovered my verry secret message:) Bravo'

You can check the last lines of my DeterministicRSA.py folder