Başak Amasya – 26628

CS411 – HW4 – SOLUTIONS

1) Please refer to Q1.py for the solution.

I generated a random integer (x) which is relatively prime to N (gcd(x,N) = 1) so that I can compute its modular inverse. Then computed c_ = (c * (x^e (mod N))) mod N and sent it to the server to get the corresponding plaintext m_. m_ = (c * x ^e (mod N))^d) mod N = c^d * x ^ ed (mod N) = (c^d * x) mod N= m * x (mod N) so we can write m as (m_ * x_inverse) mod N. (where $d \equiv e^{-1} \bmod \Phi(n)$)

Corresponding plaintext is 609582572245086581955173486556226059889437853770629680858438291721089382181432413886171267314993856833992501 and when we convert it to string using to_bytes, I got: b'Bravo: you find it. Your secret code is 57175'.

2) Please refer to Q2.py for the solution.

I performed an exhaustive search on possible PINs. 4 decimal digit pin is in range [0000,9999]. Random number R is 8-bit unsigned integer, so R is in range [2^7,2^8-1]. By using the given RSA_OAEP_Enc function, I computed corresponding ciphertext for each pin and R pair, with given e and N values until I obtained the given ciphertext. Corresponding R is obtained as 142 and the PIN is 7146.

3) Please refer to Q3.py for the solution.

If we can find out k, we can find the message and to do so we can perform exhaustive search on k, which is between 1 and q-1. We can say that flaw in this algorithm is that k is not large enough. r is g^k mod p, and we know g, p and r. so the k value which gives our r value will be our session key. Then we will calculate our message from (t * h^-k) mod p since we already know t and h as well.

k is found as 31659. m in integer form is 4577933565607691214689963654891646595903581846406907626077086469304820420435450020534356449723013351214400119651700145422949033503144323329091288273353023. When converted to string using to_bytes, the message is: b'Why is Monday so far from Friday, and Friday so close to Monday?'

4) Please refer to Q4.py for the solution.

We observe that r1 and r2 are same. So, k must be same (k is used twice). As explained in slide 15 in digital signatures lecture slides, we can obtain the secret key using s1, s2, h1, h2, r (r1=r2) and q. Corresponding formula is $a = (s_i h_j - s_j h_i)(r(s_j - s_i))^{-1} \pmod q$, taking i=2 and j=1. (Modular inverse does not exist when we take i=1 and j=2). Using the given modinv function, the private key is obtained as 16887419846051932713464453144375211173350562631553254703155613922671. I also checked my result by computing the corresponding public key beta and checked if it’s the same with public key given which it is.

5) Please refer to Qbonus.py for the solution.

r1 and r2 are not same but it is stated that in the hint part that the sender went out of random numbers so I thought there could be some dependence between the session keys. So, I performed an exhaustive search on possible coefficient values (x) starting from 2. I checked it for both k1 = x*k2 and k2 = x * k1. By using the formula given in slide 16 in digital signatures lecture slides $a = (s_i h_j - s_j h_i x)(s_j r_i x - s_i r_j)^{-1} \pmod{q}$, I calculated the private key for possible combinations where modular inverses exists. To identify the private key, I calculated beta (public key) by g^a mod p since I know beta, g and p. Corresponding private key should give beta. I obtained the beta value when x is 127, k2 = 127*k1 (i=2, j=1). Private key is 384680223193444082342876995407780976557870169632461355085385664072.