Solutions to CNS Exercises

# Exercise 1

**SECRET SHARING - BASIC EXAMPLE**

Assume a (3,5) Secret Sharing scheme, with standard numbering of parties. i.e., Pi 🡪 xi = {1,2,3,4,5}. Using arithmetic modulus 11, the dealer hides a secret S in the range [0,10]. At reconstruction time, Parties 1,2 and 4 show their shares:

P1 🡪 share1 = 4

P2 🡪 share2 = 7

P4 🡪 share4 = 9

What is the secret?

**Solution:**

Remembering the general formula for the Lagrange Coefficient in the (**t,n**) case: having 3 out of 5 shares the secret is completely revealed and is computable through the means of the above formula as follows:

Now we can compute the secret as:

**Remark: Multiplicative inverse mod n**

mod 11 = ? we need to find

in this case is 4:

So the Solution can be written as follows:

# Exercise 2

**2: SECRET SHARING – Information leakage**

Assume the same SS scheme as in exercise 1 above but in this case using **ordinary arithmetic – NO MODULUS**! The dealt secret S was the range [0,10]. At reconstruction time, Parties 2 and 4 show their shares:

P2 🡪 share2 = 29

P4 🡪 share4 = 75

Prove that even if a third share is missing, this information is sufficient to completely reveal the secret!

[*hint: write an expression parametric in the hidden secret S, and show that only one value in the range [0,10] satisfies a condition…*]

**Solution:**

In this case we have only 2 out of 5 shares so we need to use a trick in order to guess, in this case completely reveal, the secret. Let’s calculate as before shares for x = 1, 2 ,4

Now we can’t derive the secret because the share bind to x=1 is not known BUT we know something else about the solution: the share **must be** an integer value, so let’s write out the parametric function as hinted by the professor.

Now given we can guess until we find a value of S that yields an integer value for x

* S = 1
* S = 2
* S = 3
* S = 4
* S = 5
* S = 6
* S = 7
* S = 8
* S = 9
* S = 10

Given the educated guess we’ve made, the secret is 7 because, in this case, is the only result which leads to an integer share value.

# Exercise 3

**3: Common Modulus Attack**

An RSA scheme uses modulus n = 77; A same message M is RSA-encrypted using two different public keys e1 = 17 and e2 = 23, but same modulus n=77. The two resulting ciphertexts are: c1=60 and c2=53. Decrypt the message applying the Common Modulus Attack.

**Solution:**

We have:

Our goal is to find *x* and *y* s. t. because we would have decrypted the message without knowing anything else about it as:

In order to find such number, we need to find the Bézout identity and a way is to use the extended Euclidian algorithm as follows:

|  |  |
| --- | --- |
| **Euclidean Algorithm** | **Highlighting the reminders** |
|  |  |

After having highlighted the reminders we can collapse all the equation inside the last one obtaining, as we’d imagine the gcd(17,23) expressed in terms of those two numbers:

Now we only have to just compute:

= 37

# Exercise 4

**4: verifiable Secret Sharing**

Let p=83 and q=(p-1)/2 = 41.

Assume a (2,n) Verifiable Secret Sharing scheme with Modulus 41 which uses the Feldmann scheme with g=10 and modulus 83. The commitments are:

C0 = 23

C1 = 4

Party P7 (x=7) receives share s7 = 29: verify that this is a valid share.

**Solution:**

Having the initial commitments, computing it for x = 7 is immediate:

We can now verify it against the supposed shares received:

# Exercise 5

**5: Find elliptic Curve Points**

Consider the Elliptic curve

defined over the modular integer field Z5:

find all the points EC(Z5)

**Solution**

Given any point that lies in an EC it induce a subgroup on the elliptic curve, we can compute all elliptic curve points as sums of a given point.

Let P = (0,1) clearly it lies on the EC, we can now sum it to itself to obtain [2]P that would be another point.

* [2]P =
* [3]P = [2]P + P = (1,3) + (0,1 ) =
* [4]P =
* [5]P =
* [6]P =

After this point futher sums leads to point that doesn’t lies on the EC, or restart the cycle. So all the point found so far, lies on the EC and are the only points that the EC generates.

Another possible method would be to check if all the point (= the pairs given by [0,4]X[0,4] ) does or does not lies on the EC

**Notes on the Quadratic Residues**

It does exist another method to calculate the cardinality of the EC and all of its point that relies on the quadratic residues mod n as follows:

*In number theory, an integer q is called a* ***quadratic residue modulo n*** *if it is congruent to a perfect square modulo n; i.e., if there exists an integer x such that:*

*Otherwise, q is called a quadratic* ***nonresidue*** *modulo n.*

In order to calculate all points we can do as follows:

Let E be the curve defined as above: mod 5  
Let’s write a simple table were we populate their quadratic residues mod 5

|  |  |
| --- | --- |
|  |  |
| 0 | 0 |
| 1 | 1 |
| 2 | 4 |
| 3 | 4 |
| 4 | 1 |

Now, solving the above equation for means find   
But having now filled the table of possible quadratic residues has to be equal to   
one or more terms in the x column.

In order to be more precise and clear we now create a table summing up all the information so far gathered, and we will see that it would be much simpler now to find all possible points.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | **Points** |
| 0 | 0 | 1 | (0,1),(04) |
| 1 | 1 | 4 | (1,2),(1,3) |
| 2 | 4 | 3 | - |
| 3 | 4 | 4 | (3,2),(3,3) |
| 4 | 1 | 3 | - |

The last column is populated like this: ( ) indeed when there is no s. t.

# Exercise 6

**6: Pairing Based crypto**

Being e: GxG🡪Gt a bilinear map, and g a generator of G, simplify the expression:

e(gx x gy, gz)w / e(gwz, gx)

**Solution:**

Given all the hypothesis we can rewrite it like this: