Privacy Engineering (70018)

MPC 1 - Solutions

1.1 This not really a privacy-preserving protocol at all. However it show us that we need to be careful about designing such protocols and identifying their assumptions and limitations.

PROBLEMS with the protocol include

- If anyone lies, the answer will be wrong.
- For two parties each will know the others salary, so we need at least 3 parties.
- For three parties, if 2 collude, they can work out the salary of the remaining party.
- More generally if two parties, one before and one after a middle party collude they can deduce the result of the middle party.
- Alice will know the average before others and could lie (or decide not to forward the answer to other parties).
- Anyone could increase/decrease a value being passed around.
- If all salaries are zero, they will know each other's result.
- If one party lies and enters 0, the liar will know the average of the others, but they will not know the salary of the liar.
- If parties know each other, and everyone knows that Alice has a high earning job (outlier).

1.2 Bob knows Alice is in the range `1..5  (a<6)`

Alice knows Bob is in the range `4..10 (b>3)`

1.3 Bob knows Alice is in the range `3..10 (a>=3)`

Alice knows Bob is in the range `1..6  (b<=6)`

1.4
```
Z1 = DprivA(EpubA(r)-1+1) = r (mod p)
Z2 = DprivA(EpubA(r)-1+2) ≠ r (mod p)
Z3 = DprivA(EpubA(r)-1+3) ≠ r (mod p)
Z4 = DprivA(EpubA(r)-1+4) ≠ r (mod p)
Z5 = DprivA(EpubA(r)-1+5) ≠ r (mod p)
Z6 = DprivA(EpubA(r)-1+6) ≠ r (mod p)
```

1.5 To check if two millionaires have the same wealth we can run the protocol in both directions, Alice to Bob, and Bob to Alice If the results are A>=B and B>=A then this implies that they have the same wealth. We could also do both runs in parallel if desired.

An alternative approach is that Alice could send Zk values such that positions other than the *a*th position have 1 added. For example.
If a=b=4 then this would produce the following list that is sent:
```
Z1 + 1 = DprivA(EpubA(r)-4+1) + 1
Z2 + 1 = DprivA(EpubA(r)-4+2) + 1
Z3 + 1 = DprivA(EpubA(r)-4+3) + 1
```

```
Z4     = DprivA(EpubA(r)-4+4)         Check =r implies same wealth
Z5 + 1 = DprivA(EpubA(r)-4+5) + 1
Z6 + 1 = DprivA(EpubA(r)-4+6) + 1
```

If a=4 and b=3 (i.e. different) then we have

```
Z1 + 1 = DprivA(EpubA(r)-3+1) + 1
Z2 + 1 = DprivA(EpubA(r)-3+2) + 1
Z3 + 1 = DprivA(EpubA(r)-3+3) + 1     Check ≠r implies not same
Z4     = DprivA(EpubA(r)-3+4)
Z5 + 1 = DprivA(EpubA(r)-3+5) + 1
Z6 + 1 = DprivA(EpubA(r)-3+6) + 1
```

1.6 This is to ensure that no value is sent twice in the list S that Alice sends to Bob. Otherwise if Sj=Sk (j<k) then Bob knows that j <= a < k. Recall Z is a list of distinct values, the duplicate pair would arise from the partitioning of Z into incremented and non-incremented sub-lists at position a.
e.g. if a = 3, Z = [44, 66, 77, 33, 65, 11], S = [44, 66, 77, 34, 66, 12]

1.7 Bob could find a by applying `EpubA` to values in the list. Since Bob knows the numbers `EpubA(r)-b+k`, for `k=1..6` Bob can check when the 1..6 sequence is broken. Recall that the application of keys in RSA is “reversible” i.e. the public-key (private-key) can be used for decryption (encryption) as well as encryption (decryption).

1.8 Use 1 for *interested* and 0 for *not interested*. If the result is 1 then both are interested, otherwise either one of them or neither of them is interested. If Alice lied and said she was interested when she wasn't, then Alice would be able to determine if Bob was definitely interested in her.

1.9 (i)
For Alice we have `Aa1+a2+a3 mod p`
For Bob we have `B=b1+b2+b3 mod` p

Alice sends `(a1, a3) to B`, and `(a1, a2) to C`
Bob sends `(b2, b3) to A`, and `(b1, b2) to C`

```
Alice will know   (a1, a2, a3),   (b2, b3)
Bob   will know   (b1, b2, b3),   (a1, a3)
Carol will know   (a1, a2),       (b1, b2)
```

Hard bit: Expression for multiplication:

```
AB = (a1b1+a1b2+a1b3)  +  (a2b1+a2b2+a2b3) + (a3b1+a3b2+a3b3) mod p
```

Each party can compute part of this expression e.g. **bold terms**

Alice (a1b1+**a1b2**+**a1b3**) + (a2b1+**a2b2**+**a2b3**) + (a3b1+**a3b2**+**a3b3**)
Bob (**a1b1**+**a1b2**+**a1b3**) + (a2b1+a2b2+a2b3) + (**a3b1**+**a3b2**+**a3b3**)
Carol (**a1b1**+**a1b2**+a1b3) + (**a2b1**+**a2b2**+a2b3) + (a3b1+a3b2+a3b3)

From this we can distribute the computation over the parties, each party computing a partial sum not involving their own shares e.g.

Alice: sA = a2b2 + a2b3 + a3b2 mod p

Bob: sB = a3b3 + a1b3 + a3b1 mod p

Carol: sC = a1b1 + a1b2 + a2b1 mod p

We can then use secure addition for these partial sums to get secure multiplication.

(ii) No. If Carol reveals a2 to Bob for example, he can add it to a1 and a3 which Alice sent to him to discover her number.

(iii) If Alice has all three partial sums, then she knows a1, a2, a3, b2, b3, (a1b1+a1b2+a1b3), (a2b1+a2b2+a2b3), (a3b1+a3b2+a3b3)
From this Alice may be able to determine b1 and hence B.
For example, Alice knows sB=a2b1+a2b2+a2b3 Say a1=1, a2=2, a3=3, b2=4, b3=5, sB=29 then we have
19 = 2xb1+2x4+2x5 mod p
2xb1 = 29 - 8 - 10 mod p
From this Alice can try to determine which value(s) satisfy the formula.