Privacy Engineering (408)

Secure Multiparty Computation

Exercises with solutions

1. For the Millionaires' problem if Alice's wealth is £3M, Bob's is £6M and the range of their wealth is £1M to £10M, what do Alice and Bob know about the range of the other's wealth.

   Bob knows Alice is in range 1..5 (a<6)
   Alice knows Bob is in the range 4..10 (3<b)

2. For the Millionaires' problem if Alice's wealth is £6M, Bob's is £3M and the range of their wealth is £1M to £10M, what do Alice and Bob know about the range of the other's wealth.

   Bob knows Alice is in range 3..10 (a>=3)
   Alice knows Bob is in the range 1..6 (6>=b)

3. How can the Millionaires' problem be adapted for equality?

   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 they have the same wealth. We could also do both runs in parallel if desired.

   A suggestion made in class was that Alice should send a Zk such that positions other than the ath positions had 1 added. This would give for Alice=4 the following List:

```
Z1 = DA(EA(R)-3) + 1
Z2 = DA(EA(R)-2) + 1
Z3 = DA(EA(R)-1) + 1
Z4 = DA(EA(R))
Z5 = DA(EA(R)+1) + 1
Z6 = DA(EA(R)+2) + 1
```

   If b=4 that Z4=R, for other values of Zk≠R

4. For the Millionaires' problem in the slides with Alice = £4M, Bob=£3M show that (i) Z_3= R mod p, (ii) Z_4≠R mod p

   For b=3, k=3

```
Z1 = DA(EA(R)-2)
Z2 = DA(EA(R)-1)
Z3 = DA(EA(R))    = R mod p
```

```
Z4 = DA(EA(R)+1) ≠ R mod p
Z5 = DA(EA(R)+2) + 1
Z6 = DA(EA(R)+3) + 1
```

5. Consider the following 1-from-*n* oblivious transfer protocol in an honest-but-curious (semi-honest) model.

   1. Alice generates *n* random public-private key pairs

      $(pub_1, priv_1), \ldots (pub_n, priv_n)$

      Alice sends the public keys $pub_1, ..., pub_n$ to Bob.

   2. Bob generates *n* random symmetric keys $k_1, ..., k_n$ and computes

      $G_b = E_{pub_b}(k_b)$ and $G_z = k_z$ for all $z \in \{1..n\}$ and $z \neq b$

      Bob sends G_1 to G_n to Alice

   3. Alice computes

      $H_z = D_{priv_z}(G_z)$,

      $C_z = E_{H_z}(M_z)$ for all $z \in \{1..n\}$

      Alice sends C_1 to C_n to Bob

   4. Bob computes $M_b = D_{k_b}(C_b)$

   For this protocol:

   i) Show that Bob's output equals M_b.

   Bob's output is Dkb . Ex (Mb) where x = Dprivb . Epubb (kb) = kb
   giving Dkb . Ekb (Mb) = Mb

   ii) Explain why Alice learns nothing about *b*. What assumptions do you have to make about the two cryptosystems for this to be true?

   Alice is told G1=k1, G2=k2, .. Gb=Epubb (kb), ... Gn=kn However these are just binary values, she doesn't know which one Bob will use for the final decryption. We need to assume that the G values are indistinguishable e.g. are padded to the same length, otherwise Alice could assume the symmetric key lengths from the public-key encryption.

   iii) Explain why Bob learns nothing about M_z for $z \neq b$. What assumptions do you have to make about the two cryptosystems for this to be true?

   If Bob attempts to decrypt using z!=b, for example, if b!=1 Bob will get:
   Dk1 . Ex (M1) = rubbishMessage because x = Dpriv1 (k1) = rubbish key

This assumes that x is an acceptable key for the Symmetric Encryption function, i.e. securely padded/truncated to length.

iv) If Alice were dishonest, is there anything she could do to learn b? If so, describe how. If not, explain why not.

If Alice is dishonest, she does not need to run the protocol correctly e.g. not use random numbers, could re-use values from previous run, could use different encryption function etc. Alice can determine b using the difference in the size of G elements. Alice could also encrypt the same message in step 3, essentially controlling which secret Bob gets, i.e. b is irrelevant

v) If Bob were dishonest, is there anything he could do to learn messages other than M_b? If so describe how. If not, explain why not.

Bob can be dishonest in step 2 he could do $G_z = E_{pub_z}(k_z)$ for all keys, then he can learn all secrets M1 to Mz . Even he can set key k1=k2=k3=...=kn.

6. Consider the following *1*-from-*2* oblivious transfer protocol based on the well-known Diffie-Hellman key-exchange protocol in an honest-but-curious setting.

1. Alice generates a random number a (from $\mathbb{Z}p$). Similarly, Bob generates a random number b. Bob's message selection bit is m.

2. Alice sends $A=g^a$ to Bob. g is a suitable generator for the group.

3. If $m=0$ Bob sends $B=g^b$ to Alice.
   If $m=1$ Bob sends $B= Ag^b$ to Alice.

4. Alice computes $k_0=Hash(B^a)$, $k_1=Hash((B/A)^a)$,
   $C_0 = E_{k_0}(M_0)$, $C_1 = E_{k_1}(M_1)$

   Alice sends C_0 and C_1 to Bob.

5. Bob computes $k = Hash(A^b)$, $M_m = D_m(C_m)$,

For this protocol:

i) Explain why Bob's output equals M_m.

In step 3, B is either $B = g^b$ or $B = Ag^b = g^a g^b = g^{(a+b)}$

In step 4, keys k_0 and k_1 are either hashed from $B = g^b$ if Bob's selection bit $m=0$ i.e.

k_0 from $(g^b)^a = g^{ba} = g^{ab}$

k_1 from $(g^b / g^a)^a = g^{ba} / g^{a^2} = g^{ab-a^2} = g^{a(b-a)}$

or k_0 and k_1 are hashed from $B = g^{(a+b)}$ if Bob's selection bit $m=1$ i.e.

k_0 from $(g^{(a+b)})^a = g^{(a+b)a} = g^{a^2+ba} = g^{a^2+ab}$

k_1 from $(g^{(a+b)} / g^a)^a = g^{(a+b)a} / g^{a^2} = g^{a^2+ab-a^2} = g^{ab}$

In both cases one of the keys is hashed from g^{ab}, Bob is also able to generate this key from the hash $A^b = g^{ab}$.

If $m=0$ Bob will correctly decrypt M_0. If $m=1$ Bob will correctly decrypt M_1.

ii) Explain why Alice learns nothing about m and why Bob learns nothing about M_z for $z \neq m$. What assumptions do you have to make about the two cryptosystems for this to be true?

Alice is told either g^b or $g^{(a+b)}$ which are just two random values. The keys produced by Alice are distinct, Bob is only able to decrypt 1 message.
We assume that both the DH crypto-setup, hash function and message encryption scheme are secure. Bob must not be able to compute key g^{ab} from either $g^{a(a+b)}$ or g^{a+b} otherwise he will know both keys.

iii) Explain what, if any, issues arise if Alice sets a to 0. What if Bob sets b to 0 (with Alice generating a random number a as normal)?

If Alice sets a=0. This will give $B= g^b$ for both m=0 and m=1. Alice doesn't learn m from this. Alice will generate $k_0= g^0$=1 and $k_1= g^0$=1. Bob will learn both messages.

If Bob sets b=0. This will give $B= g^0$=1 for m=0 and $B= g^a$ for m=1. Assuming the protocol is known to Alice then she will learn m and know which message Bob chose.

7. Three privacy rights campaigners are having dinner around a table at a restaurant in South Kensington. The restaurant owner tells them that their dinner has been paid for, anonymously, either by one of them, or by Bookface. The three campaigners respect each other's right to make an anonymous payment but they would like to know if Bookface is paying. They decide to find out whether it is Bookface who has paid, or one of them, without exposing which one of them it is. They devise the following protocol:

1. Each campaigner flips a coin and shows it to their left neighbour, i.e. each campaigner will see the outcome of two coin flips: their own and that of the right neighbour.

2. Each campaigner then announces whether the outcomes of the two coin flips that they have seen are the "*Same*" or "*Different*". If the campaigner is the payer, the campaigner says the opposite (i.e. lies).

For this protocol show that:

i) An odd number of "*Same*" announcements means that Bookface is paying, while an even number means that one of them is paying.

Here's one approach.

Let's assume that BookFace is paying, in this case all campaigners will announce truthfully. We have eight cases to consider:

For (A=Head, B=Head, C=Head) and (A=Tail, B=Tail, C=Tail) each campaigner will respond with "*Same*", i.e. we have 3 "*Same*" announcements (odd) or 0 "*Different*" announcements (even).

For each of the other six cases, we will have either (2 Heads and 1 Tail), or (1 Head and 2 Tails), this leads to 1 "*Same*" 2 "*Different*" announcements (even again)

We can present a similar argument for the case when BookFace is not-paying leading to an odd number of "*Different*" announcements for all cases.

A nicer approach is to reason using binary values for same/different and XOR operations. A more laborious approach is to enumerate all cases.

ii) A non-paying campaigner cannot tell which of the other two is the payer, if Bookface is not paying.

1) the two coins that the non-payer sees are the same, one of the other campaigners said "*Different*," and the other one said "*Same*." If the hidden outcome was the same as the two outcomes she sees, the campaigner who said "*Different*" is the payer; if the outcome was different, the one who said "*Same*" is the payer. But since the coin should be fair, both possibilities are equally likely.

2) the coins that the non-payer sees are different; if both other campaigners said "*Different*," then the payer is closest to the coin that is the same as the hidden coin; if both said "*Same*" then the payer is closest to the coin that differs from the hidden coin. Thus, in each sub-case, a non-paying campaigner learns nothing about which of the other two is paying.

8. Devise an alternative protocol to Q7 based on the message passing scheme used for average salary.

Just use the same protocol with each party using 1 if paid, 0 if not paid. The sum at the end is either 0 for no one has paid or 1 if one of them has paid.

9. Alice has a number A, Bob a number B. Devise a protocol to securely multiply the two numbers. Use Carol to help with the computation - she has no number (provides no input).

   Hint: split A into shares a1, a2, a3, and B into shares b1, b2, b3. Distribute them like in secure voting, then devise sub-expressions for Alice, Bob and Carol and combine them to get the final answer.

   For Alice we have A=a1+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 work 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.

10. How could you use the secure multiplication protocol to determine if Alice and Bob where interested in dating each other? What if Alice lied and said she was interested when she wasn't?

    Use 1 for interested and 0 for not interested. If result is 1 then both are interested, otherwise either one of them or neither of them is interested.

11. Is secure multiplication still secure if Carol colludes with Alice or Bob?

    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.

12. In secure multiplication why is secure addition used, could we not just add the sub-expressions computed by Alice, Bob, Carol. Hint: consider the case when Alice

knows sB.

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

$$19 = 2xb1+2x4+2x5 \bmod p$$
$$2xb1 = 29 - 8 - 10 \bmod p$$

From this Alice can then try to determine which value(s) satisfy the formula.