Name: Jianan Luo
Student #: 20523403
UW UserId: j43luo

Written Response Questions

1.

a)

Known: p=97, g=5, a=36, b=58

Alice has the secret a and sends A = (g^a mod p) to Bob

A = 5^36 mod 97
= 50

Bob has the secret b and sends B = (g^b mod p) to Alice

B = 5^58 mod 97
= 44

Alice and Bob both compute shared key k = (g^(ab) mod p) = (A^b mod p) = (B^a mod p)

k = 5^(36*58) mod 97
= 75

b)

Eve might recover original secret values. This is a discrete logarithm problem. There's a wikipedia page, basically The discrete logarithm is a general algorithm for computing logb a in finite groups G is to raise b to larger and larger powers k until the desired a is found. This algorithm is sometimes called trial multiplication. It requires running time linear in the size of the group G and thus exponential in the number of digits in the size of the group. Therefore it is an exponential-time algorithm, practical only for small groups G. However, the discrete logarithm problem is considered to be computationally intractable. That is, no efficient classical algorithm is known for computing discrete logarithms in general.

c)

If Mallory behaves as active Man-In-The-Middle (MITM) attacker, she would get Alice's public key and respond Alice back; she could also get Bob's public key and respond Bob back. This would make Alice & Bob think they are talking to each other, but in fact they are talking to Mallory. So Mallory can read their messages. This Attack can be prevent by Alice & Bob use CA to exchange their public keys by using digital certificates.

2.

d)

Fingerprints can help receiver make sure nothing is being changed and make sure it is the message sent from the sender not anyone else. In this case, let's say if Alice receives something, she can verify it's from Bob not Mallory by checking the fingerprint with it. If user do not follow this procedure properly, man-in-the-middle attacks could possibly happen.

3.

a)

Tracker attacks:

q(C) = q(C or T) + q(C or not T) - q(S)

Let's say C is Name == 'Leonardo' T is Occupation == 'Staff'

Tracker:

```
SELECT SUM(Salary)
FROM Employee
WHERE Occupation == 'Staff'
```

q(C ot T):

```
SELECT SUM(Salary)
FROM Employee
WHERE Occupation == 'Staff' OR Name == 'Leonardo'
```

q(C or not T):

```
SELECT SUM(Salary)
FROM Employee
WHERE Occupation != 'Staff' OR Name == 'Leonardo'
```

q(s):

```
SELECT SUM(Salary)
FROM Employee
```

By using q(C or T) + q(C or not T) - q(S) we can calculate q(C) the salary for Leonardo

b)

Are they not correct. The following is the new table with value l: 2

| Name | Birthdate | Occupation | Allegiance |
|---|---|---|---|
| * | 7** | Specialist | Quendor |
| * | 8** | Specialist | Quendor |
| * | 7** | Staff | Quendor |
| * | 7** | Specialist | Antharia |
| * | 7** | Staff | Antharia |
| * | 7** | Specialist | Quendor |
| * | 8** | Specialist | Antharia |
| * | 8** | Staff | Antharia |
| * | 7** | Staff | Kovalli |
| * | 7** | Staff | Kovalli |

Programming Questions

1.

3.1:

5: (b) if O(r|y) = 0 then stop and output (r(b-n+1) xor 0) * (r(b-n+2) xor (n-1)) * (r(b-n+3) xor (n-2)) *.....* r(b)

6: output r(b) xor 0

3.2:

1: take r(j) = a(j) xor (b-j+1)

5: r(j-1) xor i xor 0

2.

Average case: (# of blocks) * （size of block）* (# of possible value for each char) / 2

Worse case: (# of blocks) * （size of block）* (# of possible value for each char)

3.

Using MAC can fix this vulnerability. It provides Integrity. The reason why it will fix the vulnerability is that the attacker will never know if the message padding is right or not by MAC will always return padding error when authentication fails.