

CS 458 Assignment 3
Written Response

Ruijie Zhang
20487924
r82zhang
r82zhang@uwaterloo.ca

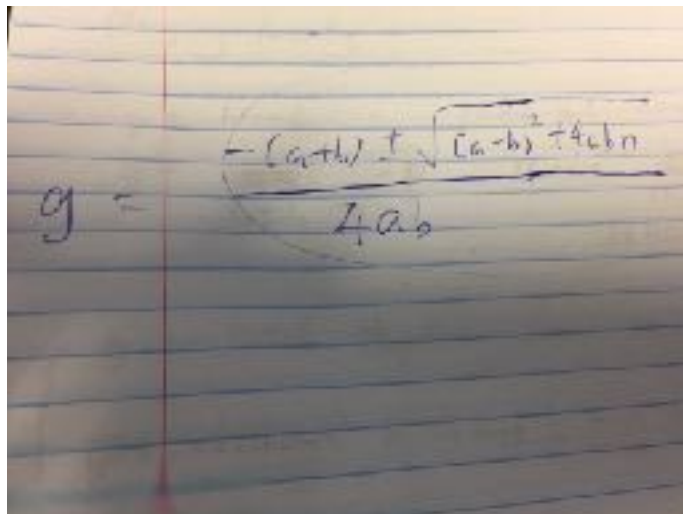
Question 1

a) $n = p * q = (2ga + 1) * (2gb + 1) = 4g^2 * ab + 1 + 2ga + 2gb = 4abg^2 + 2(a+b)g + 1$

b) $n = 4abg^2 + 2(a+b)g + 1$, it is a quadratic equation about g

$$g = \frac{-2(a+b) \pm \sqrt{4(a+b)^2 - 16ab(1-n)}}{8ab} \text{ or } \frac{-2(a+b) - (4(a+b)^2 - 16ab(1-n))^{0.5}}{8ab}$$

simplify, we get:



A photograph of a handwritten formula on lined paper. The formula is
$$g = \frac{-(a+b) \pm \sqrt{(a+b)^2 + 4abn}}{4ab}$$
 The formula is written in blue ink and is circled with a blue pen.

c) we can use the formula in b) to calculate g , and we know a and b , we can use the given formula $q = 2ga + 1$ and $p = 2gb + 1$ to determine q and p .

2.

- a) Yes, the browser will notice the phishing attack. The reason is the certificate contains the host name of CIBC, and the victim's browser will compare the host name of Mallory to the host name of CIBC, and it does not match. Then the browser will know this is not the CIBC website.
- b) The browser will NOT notice the phishing attack. Because now the domain name the user's browser sees is indeed WWW.CIBC.CA. So in this case, the hostname in the certificate does match the hostname of the user is visiting. So the browser will NOT notice this is a phishing website.
- c) The browser will NOT notice the phishing attack. Because now the domain name the browser sees is indeed WWW.CIBC.CA, and Mallory has the signing key as she can predict how the signing key is generated. She can create the signature on her own as she has everything she needed to create the signature. The browser will see the host name matches and the signature matches, so it will NOT notice the phishing attack.
- d) Because older version of IE still allows you to proceed even if they did not find the certificate. And older version of Firefox does not even warn the user about the certificate missing. Thus the attacker can still likely to succeed.

3.

d) Fingerprint of the incoming key is pretty important as it is helping us to identify another party. There is possibility that the key would be duplicate, but the fingerprint will very unlikely to be the same. Before signing the key, we must always check the key fingerprint and verify the key owner's identity to ensure that we are signing the correct key. If we do not carefully check the key we are signing, we could be talking to the wrong person and the attacker could impersonate another person and steal information from us.

4.

a) T = "SELECT SUM(Salary) FROM Employee WHERE Occupation = 'STAFF'"

Since there are half employees are STAFF, so the tracker will match N/2 records, which is between 2k and N-2K.

Let

Q1 = "SELECT SUM(Salary) FROM Employee WHERE Occupation = "STAFF" OR NAME = "Lucille"

Q2 = "SELECT SUM(Salary) FROM Employee WHERE Occupation != "STAFF" OR NAME = "Lucille"

NOT T = "SELECT SUM(Salary) FROM Employee WHERE Occupation != "STAFF""

We can get the salary of Lucille by doing:

$Q1 + Q2 - (T + \text{NOT } T) = \text{salary of Lucille}$

Because $T + \text{NOT } T$ gives the sum of all employees, while $Q1 + Q2$ gives the sum of all employees but it calculates Lucille's salary twice. So when we do the subtraction, we will get Lucille's salary.

b)

we will solve this by trial and error. We first guess a number for Rachel's salary, and do two queries and check the result to verify our guess. And then we adjust our guess. By doing this, we can get upper bound and lower bound. And we keep guessing on values between the two bounds, and because the numbers is finite, we can eventually guessed the correct number.

First we do a query like this:

`SELECT COUNT(*) FROM Employee WHERE salary >= x or NAME = "Rachel", set the result to be R1`

we do another query to be

`SELECT COUNT(*) FROM Employee WHERE salary >= x, set the result to be R2`

we compare R1 and R2. If they are the same, then we know that Rachel's salary is greater than x. Otherwise, we know that Rachel's salary is smaller than x, and then we can move x down to gradually figure out what x is. And then eventually figure out the salary of Rachel.

For example, we choose $x = 5000$,

`SELECT COUNT(*) FROM Employee WHERE salary >= 5000 or NAME = "Rachel" returns 31`

`SELECT COUNT(*) FROM Employee WHERE salary >= 5000 returns 30,`

Then we know Rachel salary must be smaller than 5000. we should try 4500 or some other number smaller than 5000 next try.

If

`SELECT COUNT(*) FROM Employee WHERE salary >= 5000 or NAME = "Rachel" returns 30`

`SELECT COUNT(*) FROM Employee WHERE salary >= 5000 returns 30`

we will know that rachel's salary is greater or equal to 5000, then we increase x to a larger number and until we know the exact bound.

However, there is one problem with this, what if Rachel's salary is too low or too high so that the query will get rejected when x gets high or low enough.

if `SELECT COUNT(*) FROM Employee WHERE salary >= 5000` returns nothing, because most people's salary are greater than 5000, then we cannot continue.

The solution to this is quite simple, we choose another number y . We first do a query like this:

`SELECT COUNT(*) FROM Employee WHERE salary <= y` and keep changing y until the result is not rejected and it is about $N/2$.

And after choosing a y , we can do a query like this:

`SELECT COUNT(*) FROM Employee WHERE salary <= y or salary >= x or NAME = "Rachel"`

`SELECT COUNT(*) FROM Employee WHERE salary <= y or salary >= x`

Now, none of the query will be rejected, we can use the same procedure to determine Rachel's salary.

5 a) $O(NbW/2)$ on average and worst case $O(NbW)$

b) The method can be used is encrypt-then-MAC to fix this vulnerability. This would provide more integrity. This would stop information leakage and thus stop the oracle attack.