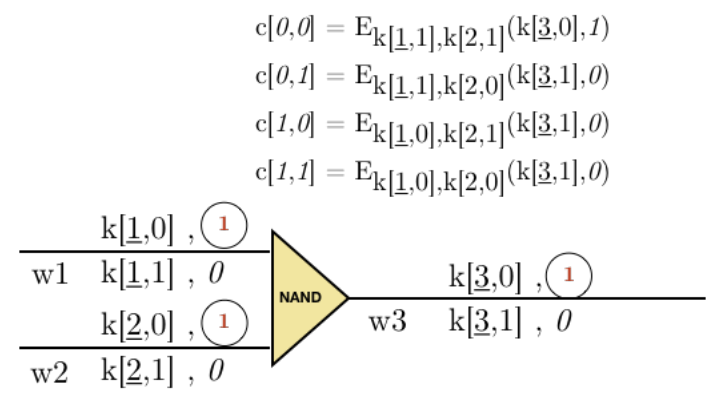
# Question 1

## Part (a)

* Alice can only work out that everyone votes yes or everyone votes no
* In many real situations at least one person will vote against the crowd and everyone will have some plausible deniability
* The goal of the protocol is also to reveal the total NUMBER of yes votes, if Alice was told this by an oracle, she would still be able to work out this information
* If Alice picks shares which add to 2 (mod p) they can disrupt the voting
* Each other party only gets 2 shares so they can’t detect the issue
* The cheating will only be exposed if the total number of votes is too large but it won’t be revealed the problem is A

## Part (b)

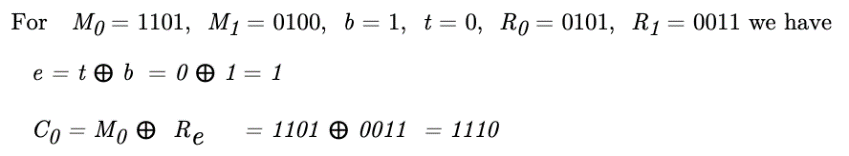


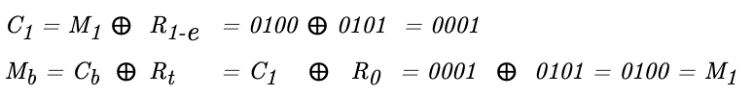
## Part (c)

* Lie in decryption mapping table (I.e., swap the results of the two rows—Bob will “think” he knows the correct answer (and hence tell Alice) --but Alice knows the correct answer is the opposite)
* Following from the above, Alice can lie in any garbled table
* Alice can even produce a circuit that can reveal Bob’s inputs. (e.g. a large XOR circuit but Alice’s inputs are all 0—essentially a passthrough)
* OT protocol violations, e.g. during OT, if Bob did not check inequality between the two public keys generated by Alice he may reveal his symmetric key k to Alice—essentially here Alice can generate two identical pub-priv keypairs.
* Send information on intermediate steps that do not reveal inputs, e.g. the final entry key that maps to the result in the decryption mapping table.

## Part (d)

### Part (i)



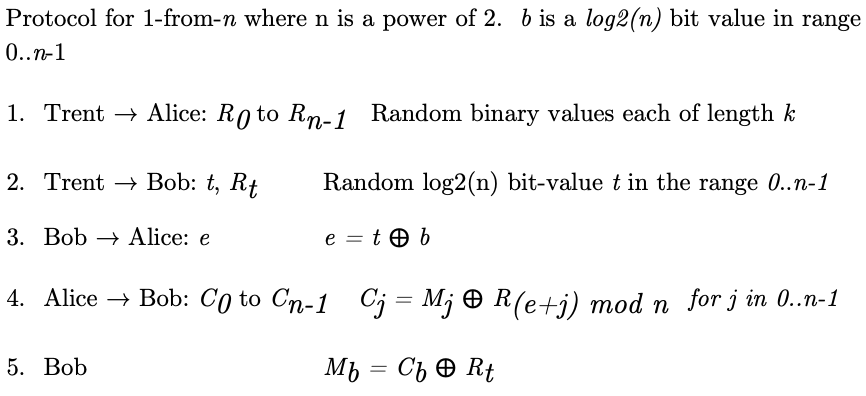


### Part (ii)

* Bob only sends Alice t xor b because t is random this is effectively sending A a random bit which doesn’t tell A anything about b
* The other message Bob receives is xor’d with a random binary string he isn’t sent. There is no way for him to find the original value.
* If Alice were malicious she could make both messages the same and learn what B ends up with.
* Alice could also make one of the C’s garbage, if the protocol finishes successfully A will learn b

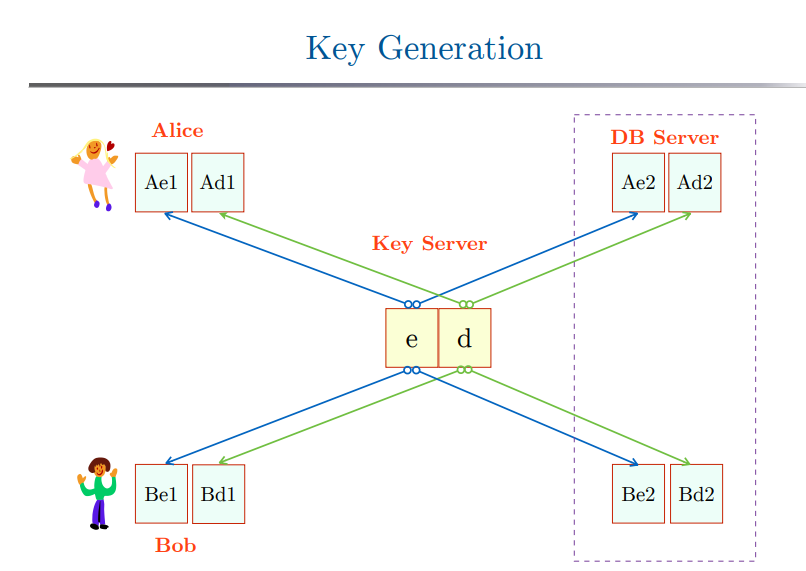
### Part (iii)

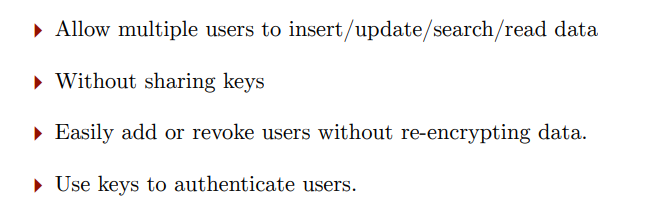
* In the tutorial answer, e+j should be e XOR j



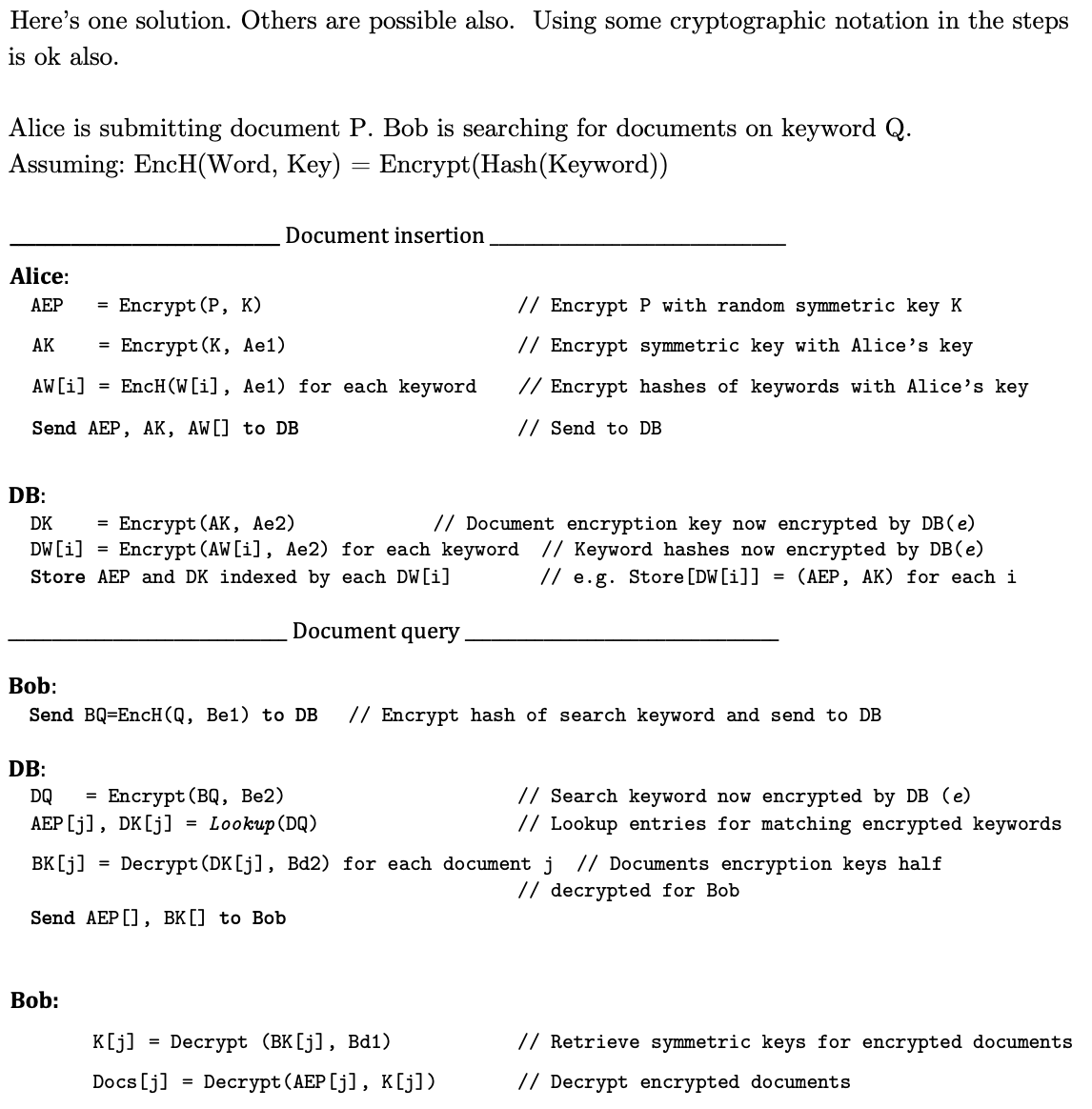
# Question 2

## Part (a)

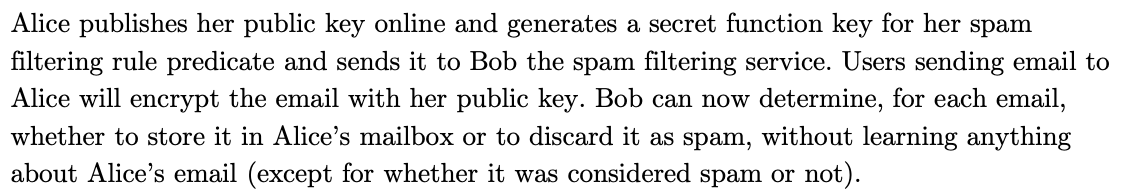




## Part (b)



## Part ©



# Question 3

3

A

I)

An attribute is some piece of information stored about an individual in a dataset

A quasi-identifier is an attribute that alone doesn’t uniquely identify an individual in a dataset but could in combination with other information

A uniqueness attack is where an attacker collects sufficient quasi-identifier about a victim so that every record in the dataset, bar one, can be ruled out as not being them.

II)

Where any equivalence class formed using a set of quasi-identifiers contains at least k people

III) Homogeneity attack (Set of quasi-identifiers all contain the same sensitive data, therefore you can deduct what sensitive attribute that person will have anyways)

|  |  |  |
| --- | --- | --- |
| Id | Blood Type | Illness |
| 0 | A | Covid 19 |
| 1 | B+ | Halitosis |
| 2 | B- | Feels like a pair of curtains |
| 3 | B+ | Lung cancer |
| 4 | A | Covid 19 |
| 5 | A | Covid 19 |
| 6 | B- | Heart Disease |
| 7 | A | Covid 19 |
| 8 | B- | Asthma |
| 9 | B- | Hemophilia |
| 10 | B+ | Lung Cancer |

Everyone with blood type A has Covid

B)

I)

Uniqueness attacks and unicity attacks are applied to different kinds of data. In a small data dataset an attacker can find some unsensitive information about a target, quasi-identifiers, that when combined together uniquely identifies them in the dataset and allows them to perform a ‘uniqueness attack’ to discover sensitive information about the target.

A unicity attack is relevant in a big data scenario where all data about an individual is potentially sensitive. If an attacker is able to obtain a portion of a user's data, they are able to perform a unicity attack if they can uniquely identify their targets trace in the dataset.

II)

Each point must be from a different time stamp but each point can be at the same location

This is only the case if T >= K otherwise we can’t select K unique points

M^K \* T \* (T - 1) \* … \* (T – K + 1)

M^K \* T! / (T – K)! ~~-> This should be M^K\*T!/K!~~

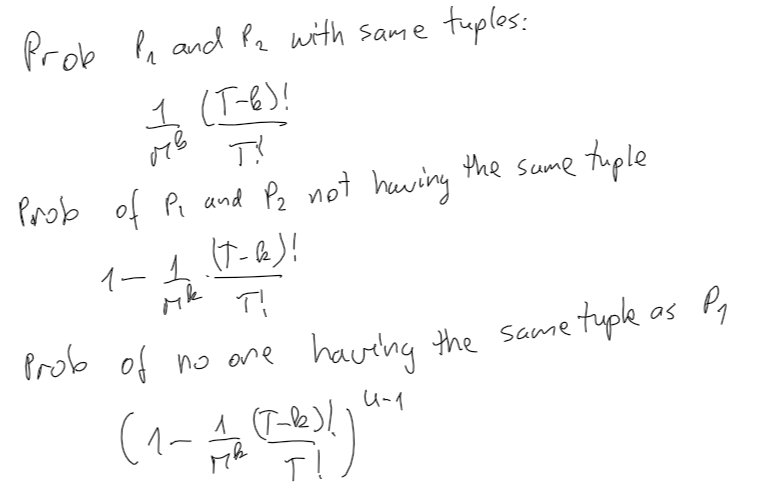
M^K \* P(T, K) <- We could argue this isn’t right as e.g. we pick from M (1, 2) and (2,1) and then from T (3, 4) and (4, 3), that would be the same P

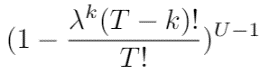
By Chris’ point we would have M^K \* C(T, K) (because C(T,K) = P(T,K) / k!

iii)

P(user has same tuple of points) = 1 / (number of possible values for P) = (using answer from ii)

Suggested better answer: 1 / M^K as we match the times, and then by random chance it would be M possibility for every point to match.

iv) By the alternative answers above, we have (1 - 1/M^K)^(U-1) ?

v) P(user has same tuple of points) can now be written as  so unicity can be written as 

I got $ (1 - \frac{(T – k)!}{(M\cdot \lambda)^k \cdot T!})^{(U – 1)})$

(Another answer would be, p(user == other user) = 1\lambda^k, then unicity = (1-1\lambda^K)^(U-1),

As lambda increases, the unicity increases. But the effect will become miniscule as k increases, as lambda^k decreases exponentially.

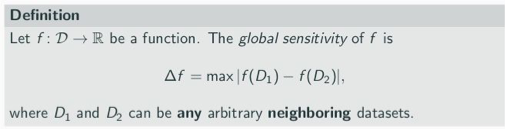
# Question 4

QBS and DP

A)

QSR is a technique that can be applied in a QBS where the system will refuse to give the answer to count queries if it is under a certain value.

This prevents uniqueness attacks but can be worked around by intersection attacks

B) The global sensitivity of a function **f** captures the magnitude by which a single individual’s data can change the function f in the worst case, and therefore, the uncertainty in the response that we must introduce in order to hide anyone’s participation.

C) I)

Think about D1 = {1, 0} and D2 = {0} (I.e., the smallest possible dataset that is nonempty), delta f = ½

Ii)

It is e-differentially private by definition (see the week 4 exercise 2.1)

It is noted that after adding the noise, the possible range exceeds [0, 1], meaning that sometimes it might return a positive number that is larger than 1 or smaller than 0.

(iii)

Use Local DP.

(iv)

Let D:{0} D’:{0,1}

M2(D)=Lap(1/epsilon)

M2(D’)=(1+Lap(1/epsilon))/2

Find S (x, \inf)

Let Pr(M(D)>x) > Pr(M(D’)>x)

* Pr(Lap(1/epsilon)>x) > exp(epsilon)Pr((1+Lap(1/epsilon))/2>x)
* 1/2 exp(-x\*epsilon) > 1/2 exp(-(2x-1)\*epsilon)\* exp(epsilon)
* -x\*epsilon > (-2x+2) \*epsilon
* -x > -2x+2
* x > 2

so if we choose D:{0} D’:{0,1} and S (2, \inf), show that this mechanism is not DP. Note that although f(D) is bounded, M(D), therefore, in theory M(D) can -> inf.