Review

Course Overview

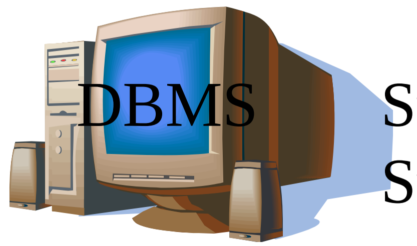
Querying
Encrypted Data

Privacy

Published data,
Statistical databases,
Differential privacy,
Location-based privacy

Encryption

Insider Threat/
Intrusion Detection/
SQL Injection
(Auditing)



Steganographic
Storage

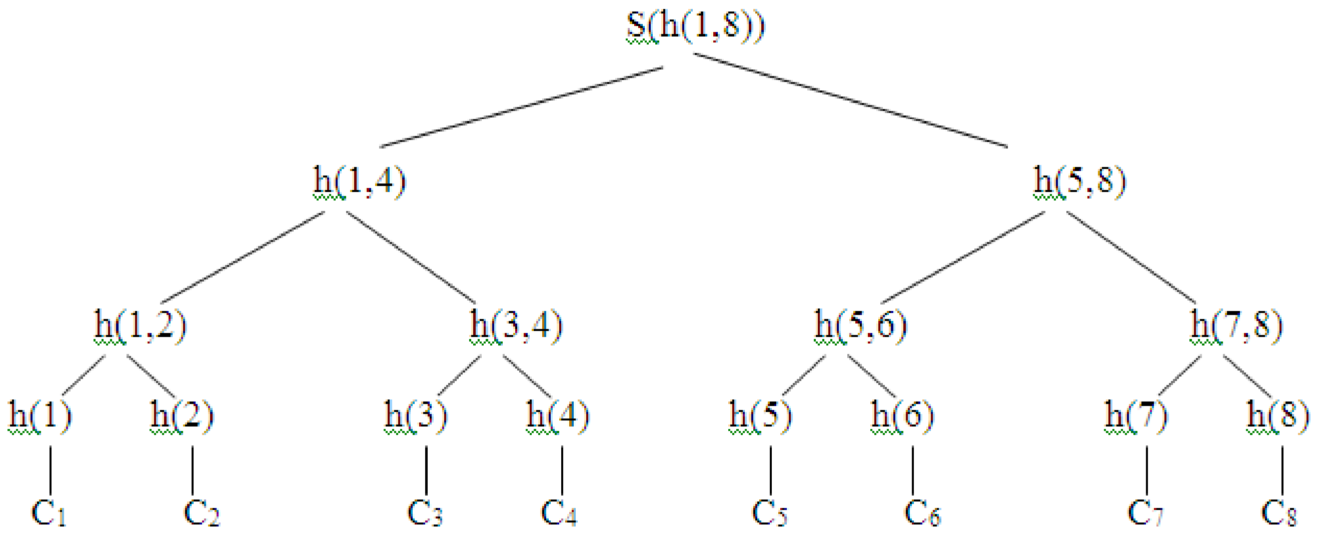
Compliance storage

~~Access Control~~
~~DAC, MAC, Role-based~~

Query
Authentication

Query Authentication

7. Consider the following Merkle hash tree.

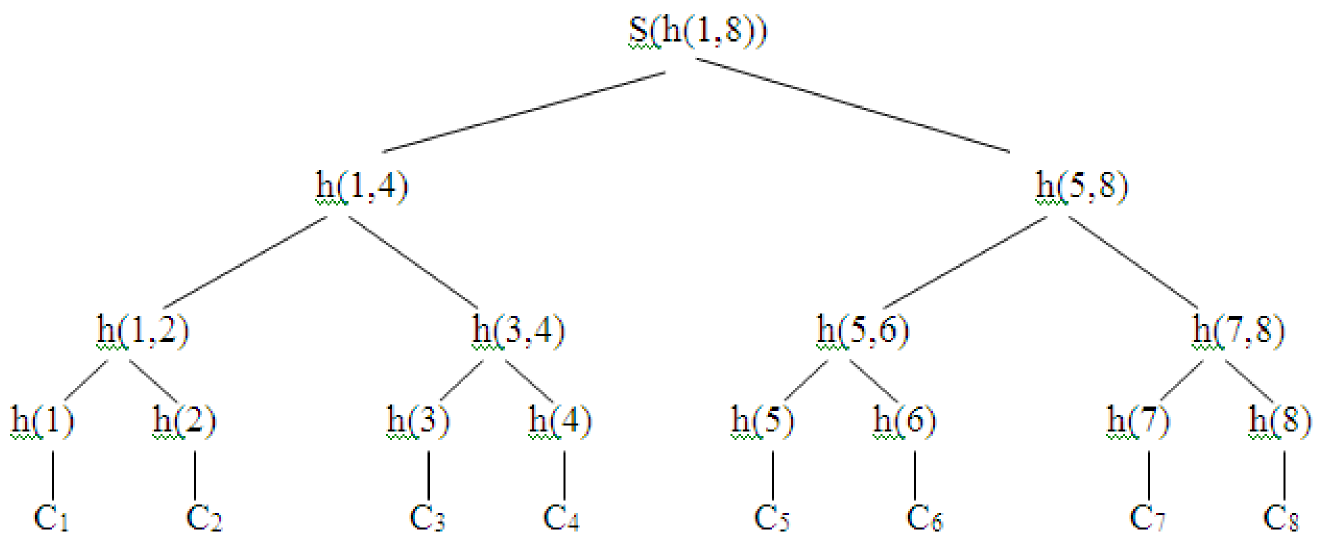


Suppose Alice issues a range query that finds all values between (and inclusive of) C_1 and C_4. What (data and hash values) should be given to her, so she can verify that the answers returned are complete and authentic?

a) C1, h(1), C2, h(2), C3, h(3), C4, h(4), h(5,8), S(h(1,8))
b) C1, h(1), C2, h(2), C3, h(3), C4, h(4), h(5,8)
c) C1, C2, C3, C4, h(5,8), S(h(1,8))
d) C1, h(1), C2, h(2), C3, h(3), C4, h(4), C5, h(5), h(6), h(7,8), S(h(1,8))
X e) C1, C2, C3, C4, C5, h(6), h(7,8), S(h(1,8))
f) C1, C2, C3, C4, C5, h(6), h(7,8)
g) None of the above.

Query Authentication

7. Consider the following Merkle hash tree.



Consider the same set of records in Question 7. Suppose we use a signature chain scheme instead. What is the verification object returned to client?

a) C1, C2, C3, C4, C5
b) C1, C2, C3, C4, $h^{C5-C4-1}$ (C5)
c) C1, sig(C1), C2, sig(C2), C3, sig(C3), C4, sig(C4),
d) C1, sig(C1), C2, sig(C2), C3, sig(C3), C4, sig(C4), sig(C5)
e) C1, sig(C1), C2, sig(C2), C3, sig(C3), C4, sig(C4), C5, sig(C5)
f) C1, sig(C1), C2, sig(C2), C3, sig(C3), C4, sig(C4), $h^{C5-C4-1}$ (C5)
X g) None of the above.

Encrypted Domain Search

- There are two methods for building the bloom filter
  - Apply the hash functions directly on the keywords
  - Apply the hash functions on

    (id, f(keyword, secret key))-pairs
- The second method will result in lower collision (and hence false positive). True?
- False! It only helps wrt privacy – server cannot associate two documents that contain similar keywords

Data Encryption

- Consider the following two tables:

  R(A: key; B: foreignKey)   S(C: key; D: int)
- Suppose the workload contains only the following query template:

  ```
  SELECT R.A, R.B
  FROM R, S
  WHERE R.B=S.C AND S.D = variable
  ```
- How would you encrypt the tables? What work is done at the client and server?

Data Encryption

- Since all equality operations, we can do attribute-level encryption. In this way, all processing can be done at the server!
  - E_R(EA: E_key; EB: E_foreignKey)
  - E_S(EC: E_key; ED: int)
- Assuming variable is set to 10, server query:

```
SELECT EA, EB // EA is the encrypted value for A
FROM E_R, E_S   // E_R is the encrypted table of R
WHERE E_R.EB=E_S.EC  AND E_S.ED = Encrypted(10)
```

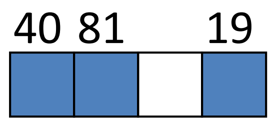
- Clients only decrypts EA and EB of each tuple

GHT

- Consider a GHT with (M,K,H) as follows:
  - m0 = m1 … = 4
  - 0 = k1 … = 2
  - h0 = key mod 4, h1 = h2 = … = key mod 8
- Insert the following keys into a GHT
  - 19, 40, 81, 121, 29, 10, 36, 80, 65, 99, 37

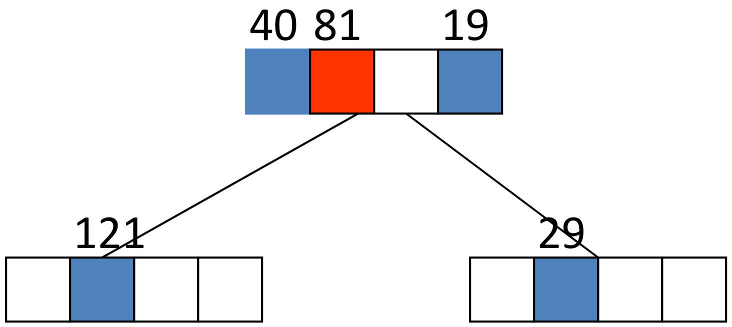
How about jumping indexes if the records were ordered?

GHT



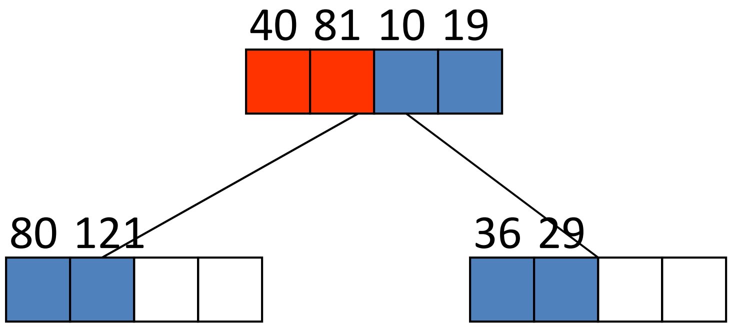
19, 40, 81, 121, 29, 10, 36, 80, 65, 99, 37

GHT



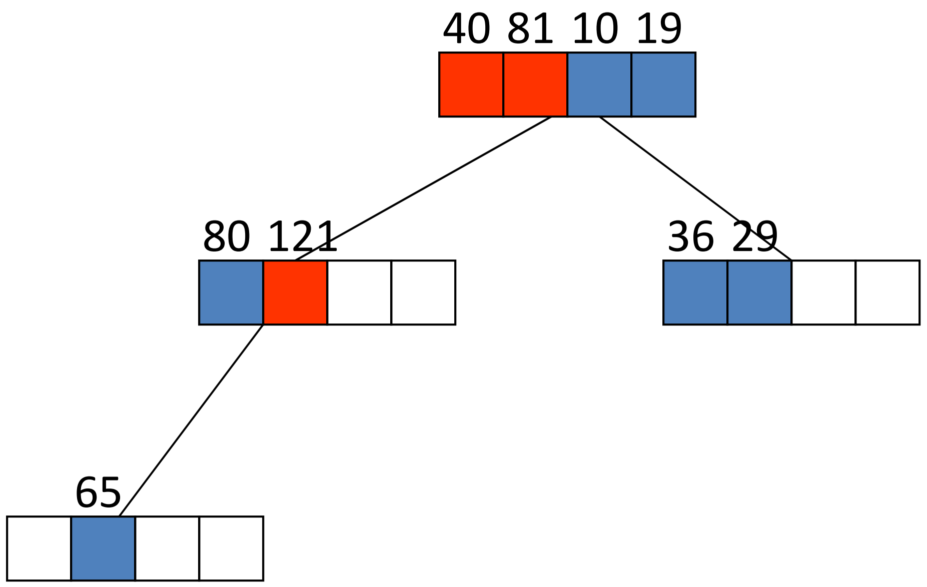
19, 40, 81, 121, 29, 10, 36, 80, 65, 99, 37

GHT



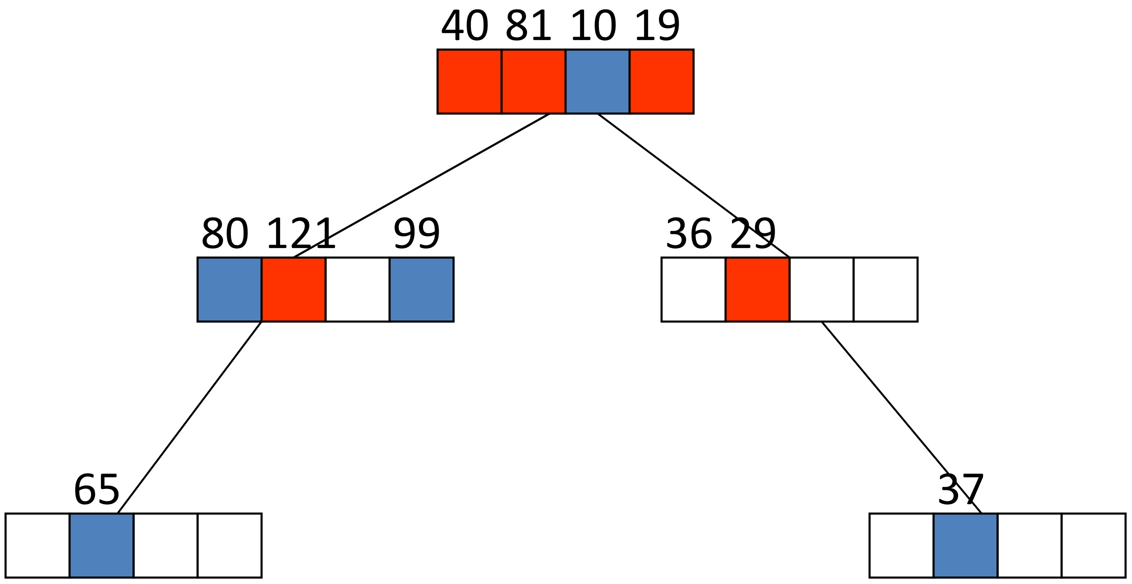
19, 40, 81, 121, 29, 10, 36, 80, 65, 99, 37

GHT



19, 40, 81, 121, 29, 10, 36, 80, 65, 99, 37

GHT



19, 40, 81, 121, 29, 10, 36, 80, 65, 99, 37

Data Privacy

- Let *M*(Qa, Qb, C, D) be the table that stores the original microdata, where (Qa, Qb) is the quasi-identifier. Consider the following k-anonymization algorithm:

  Algorithm EasyK

  Step 1: SELECT * FROM *M* ORDER BY Qa, Qb

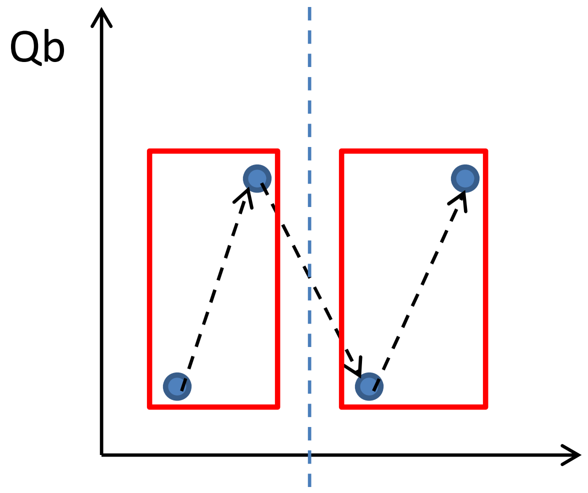
  Step 2: Split the output of **Step 1** into groups of *k* continuous tuples. For example, group 1 contains tuples 0...*k*-1, group 2 contains tuples *k*...2*k*-1, etc. Obviously, the last group may contain between *k* and 2*k*-1 tuples.

  Step 3: For each group from step 2, generalize the quasi-identifier by using the **Minimum Bounding Rectangle** of all tuples in the group

- How good is this anonymization scheme?

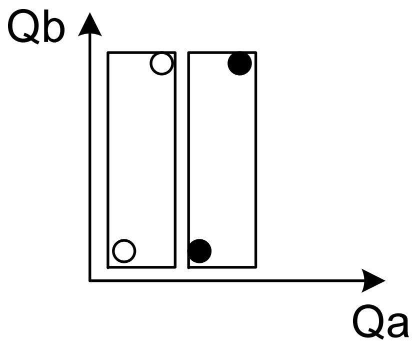
Data Privacy

Let K = 2.

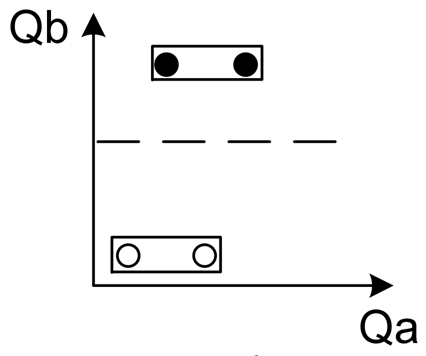


EasyK

Data Privacy



EasyK

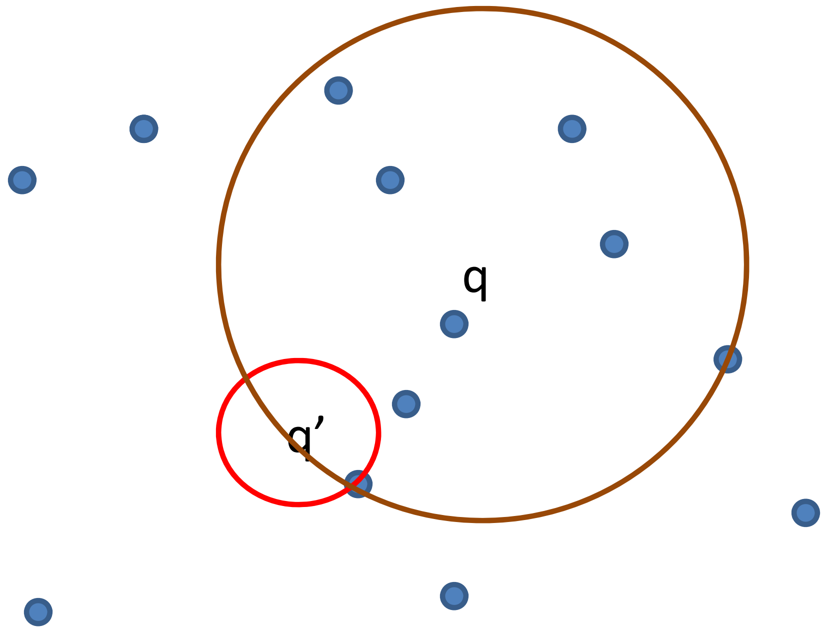


Mondrian

- Assume k=2. Mondrian (another scheme) splits across Qb. After generating the MBRs of the resulting groups, the extents of the groups are much smaller in Mondrian. Given that both methods generate groups with the same number of objects, the information loss for Mondrian is smaller.

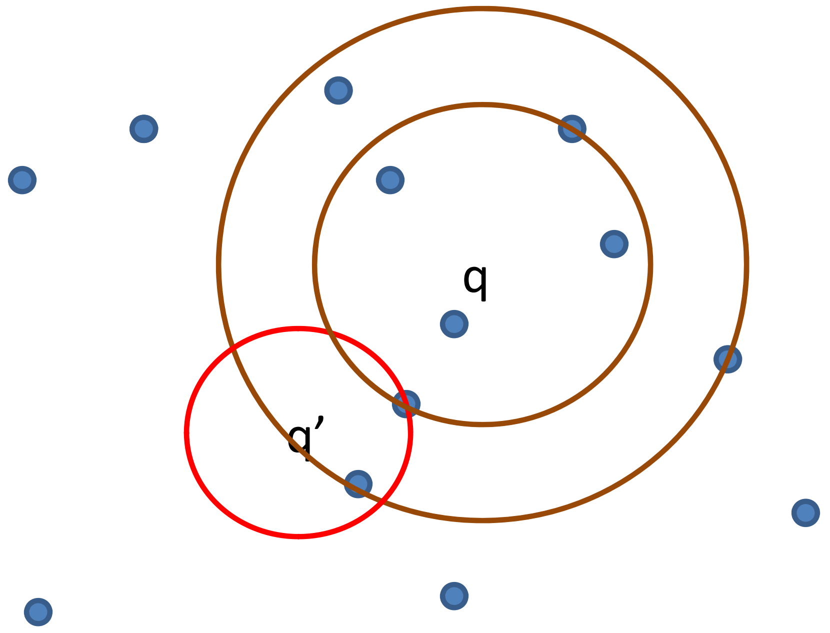
Location-based Privacy

- Consider the following set of points. Let q' be the fake query point of q. What are the sets of data points returned?



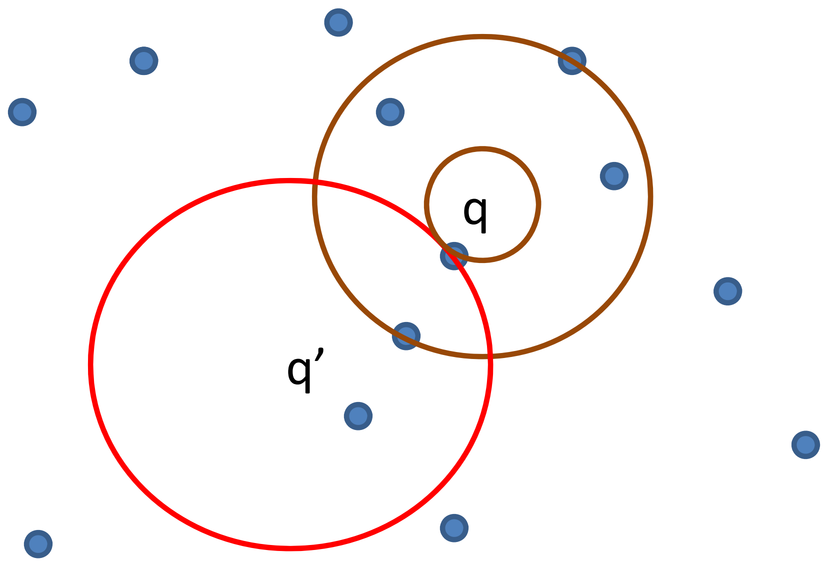
Location-based Privacy

- Consider the following set of points. Let q’ be the fake query point of q. What are the sets of data points returned?



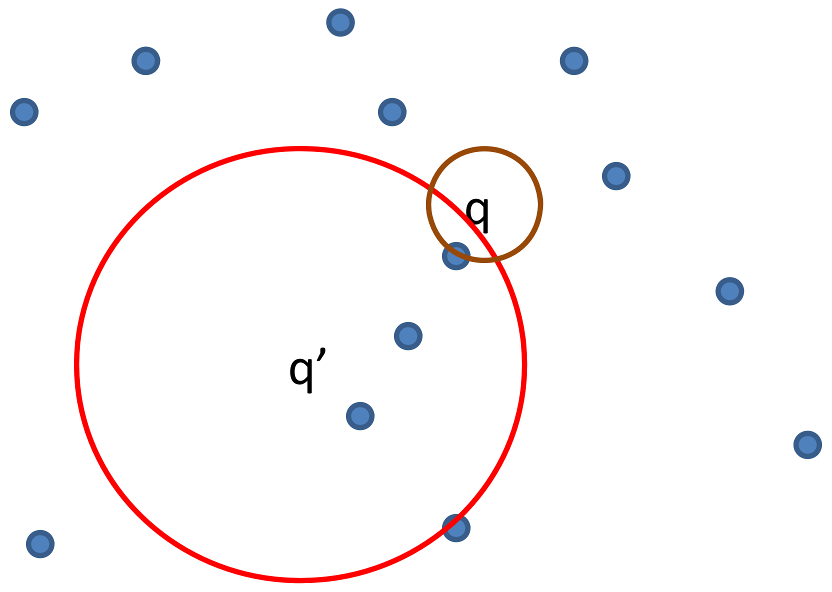
Location-based Privacy

- Consider the following set of points. Let q’ be the fake query point of q. What are the sets of data points returned?



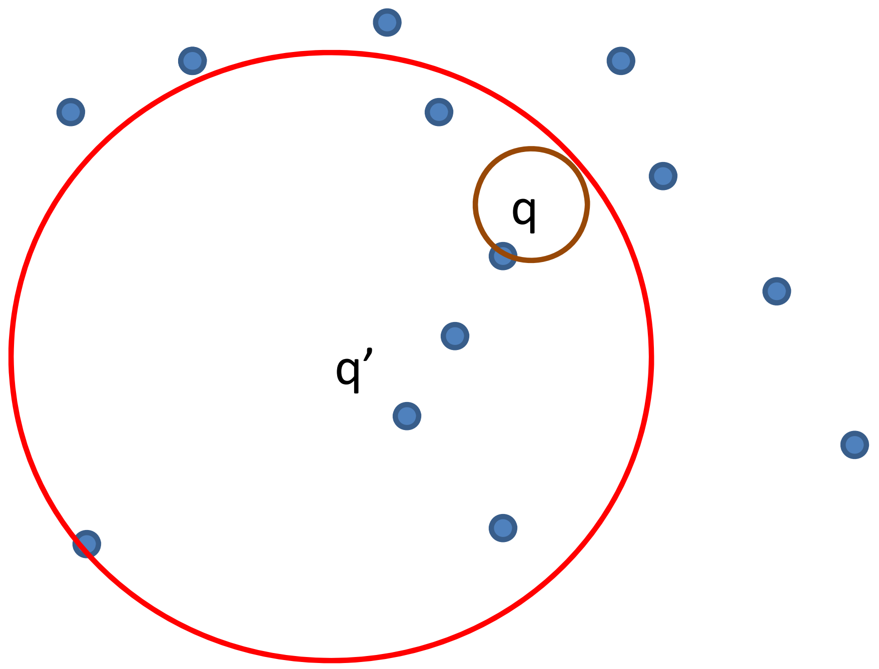
Location-based Privacy

- Consider the following set of points. Let q' be the fake query point of q. What are the sets of data points returned?



Location-based Privacy

- Consider the following set of points. Let q’ be the fake query point of q. What are the sets of data points returned?



StegFS

- In handling traffic analysis, we can store the various keys (dummy, encryption) at the trusted agent or distribute them to the users. What is the tradeoff?

- Stored at trusted agent
  - Risk of compromise if trusted agent is attacked
  - Stronger in terms of plausible deniability

- Distribute keys
  - Only the users that are log on will be compromised
  - If number of users log on is small, it is easier to detect existence of hidden files for these users (e.g., fewer dummy files)

Insider Threats

Suppose Q1 is the “normal” queries of users. Is Q2 anomalous?
Is this a false positive/negative?

```
Q1: SELECT p.type FROM PRODUCT p
      WHERE p.cost < 1000;
```

```
Q2: SELECT p.type FROM PRODUCT p
      WHERE p.cost < 1000 AND p.type IN
            (SELECT q.type FROM PRODUCT q);
```

```
Q2’: SELECT p.type FROM PRODUCT p WHERE true;
```

Insider Threats

Suppose Q1 is the "normal" queries of users. Is Q2 anomalous? Is this a false positive/negative?

Same query but is treated as anomalous!

```
Q1: SELECT p.type FROM PRODUCT p
      WHERE p.cost < 1000;
```

```
Q2: SELECT p.type FROM PRODUCT p
      WHERE p.cost < 1000 AND p.type IN
            (SELECT q.type FROM PRODUCT q);
```

```
Q2': SELECT p.type FROM PRODUCT p WHERE true;
```

Different selection attributes
- can be detected