

CS 458 / 658: Computer Security and Privacy

Module 6 – Database Security and Privacy

Part 1 – Database Security

Fall 2022

Module outline

- 1 Introduction to database security
- 2 Access control
- 3 Integrity
- 4 Others

Relational Databases

- A (relational) database is a structured collection of data (**records**).
- Database management system (**DBMS**) provides support for queries and management of the records.
- Many popular DBMSes are based on the **relational model**.
- Stores records into one or multiple tables (**relations**)
 - Each table has rows (tuples) and named columns (attributes).
 - Tables can be related to one another.
- Structure (**schema**) set by database administrator.

Relations: example

Tables have a *primary key*: an attribute or set of attributes that is unique for each row.

Here is a table that an airline booking agency might use to store details of their customers:

Last	First	Address	City	State	Zip	Airport
ADAMS	Charles	212 Market St.	Columbus	OH	43210	CMH
ADAMS	Edward	212 Market St.	Columbus	OH	43210	CMH
BENCHLY	Zeke	501 Union St.	Chicago	IL	60603	ORD
CARTER	Marlene	411 Elm St.	Columbus	OH	43210	CMH
CARTER	Beth	411 Elm St.	Columbus	OH	43210	CMH
CARTER	Ben	411 Elm St.	Columbus	OH	43210	CMH

Q: What is the issue with storing data in a flattened table like this?

A: Lots of repeated parameters. This affects the storage cost, query speed, etc.

Relations: normalization

Normalization eliminates redundant storage of data, which

- optimizes the storage costs,
- improves query speed, and
- reduces future maintenance costs.

Table: FamilyInfo

Last	Address	City	State	Zip
ADAMS	212 Market St.	Columbus	OH	43210
BENCHLY	501 Union St.	Chicago	IL	60603
CARTER	411 Elm St.	Columbus	OH	43210

Last	First
ADAMS	Charles
ADAMS	Edward
BENCHLY	Zeke
CARTER	Marlene
CARTER	Beth
CARTER	Ben

Zip	Airport
43210	CMH
60603	ORD

Table: AirportInfo

Table: NameInfo

Database queries

The most popular language for query and manipulation of a relational database is SQL.

- A single table query

```
SELECT Address FROM FamilyInfo
WHERE (Zip = "43210") AND (Last ="ADAMS")
```

- A join query across multiple tables

```
SELECT Last, Airport
FROM FamilyInfo JOIN AirportInfo
ON FamilyInfo.Zip = AirportInfo.Zip
```

- An aggregation

```
SELECT COUNT(Last) FROM FamilyInfo
WHERE City = "Columbus"
```

- A change of record content

```
UPDATE FamilyInfo SET Address =
"1 Town St." WHERE Last = "ADAMS"
```

Last	Address	City	State	Zip
ADAMS	212 Market St.	Columbus	OH	43210
BENCHLY	501 Union St.	Chicago	IL	60603
CARTER	411 Elm St.	Columbus	OH	43210

Table: FamilyInfo

Zip	Airport
43210	CMH
60603	ORD

Table: AirportInfo

Security requirements for a database

- **Access control** Access control
 - who can read? who can write?
- Authentication
 - how do we know if a DB client (or server) is not masquerading as someone else
- Confidentiality
 - what if the DB server is compromised? what about network tapping?
- **Integrity** Integrity
 - how do we guarantee that the data is in an intact and sensible state
- Availability
 - redundancy, failover
- Auditability
 - a.k.a. provenance, proving how we ended up with a specific state

Module outline

- 1 Introduction to database security
- 2 Access control**
- 3 Integrity
- 4 Others

Access control - Recall OS module

Recall some *types* of access control

- Discretionary Access Control (DAC)
 - owners can delegate (grant/revoke) privileges to others
 - *If you own the data, you can do anything with it.*
- Role-based Access Control (RBAC)
 - ties in users' privileges to their position or roles in the organization
 - *Assign labels to users and assign privileges to labels.*
- Mandatory Access Control (MAC)
 - users and objects are assigned labels based on their 'security level'
 - *You don't own the data even if you create it. The data has labels too and may deny access from its creator.*

Access control for databases

All three types of access control (DAC, RBAC, MAC) apply to databases (with various forms of implementations).

- Most commercial DBs have native support for DAC and RBAC
- Multi-level security database is an implementation of MAC

Things to consider when designing an access control scheme:

- Granularity: Access control on *relations* (tables), *records*, *attributes*
- Supporting different operations: SELECT, INSERT, UPDATE, DELETE

DAC for databases

DAC is built-in in the SQL language.

- Use the GRANT keyword to assign a privilege to a user
- Use the REVOKE keyword to withdraw a privilege.

Different types of privileges have built-in support:

- Account-level privileges:
 - DBMS functionalities (e.g. shutdown server),
 - creating or modifying tables,
 - routines (database functions),
 - users and roles.
- Relation-level privileges:
 - SELECT,
 - UPDATE,
 - REFERENCES privileges on a relation

DAC example: account-level privilege

Accounts A1, A2, A3

Relations: nil

Account-level privilege

```
> Admin: GRANT CREATE USER TO A1;
```

Sysadmin grants user A1 privilege to create users (and roles).

Account-level privilege

```
> A1: CREATE USER A3;
```

User A1 now uses her privilege to create another user.

DAC example: account-level privilege

Accounts A1, A2, A3

Relations: nilEmployee

Account-level privilege

```
> Admin: GRANT CREATE TABLE TO A2;
```

Sysadmin grants user A2 privilege to create new tables.

Account-level privilege

```
> A2: CREATE TABLE Employee (...);
```

User A2 now uses her privilege to create the Employee table.

DAC example: relation-level privilege

Accounts A1, A2, A3

Relations: Employee

Relation-level privilege

```
> A2: GRANT SELECT ON Employee TO A3;
```

The table owner (A2) grants user A3 the privilege to run SELECT queries on the Employee table.

Relation-level privilege

```
> A2: GRANT SELECT ON Employee TO A3 WITH GRANT OPTION;
```

The table owner (A2) grants user A3 the privilege to run SELECT queries on the Employee table **and to further delegate that privilege to other users.**

DAC example: relation-level privilege

Accounts A1, A2, A3

Relations: Employee

Relation-level privilege

```
> A3: GRANT SELECT ON Employee TO A1;
```

A3 now can exercise her delegation rights

Relation-level privilege

```
> A2: REVOKE SELECT ON Employee FROM A1;
```

The table owner (A2) however, reserves the rights to revoke any privilege she considers as improper.

Fine-grained DAC

Something is missing in the DAC scheme we've seen so far:

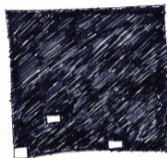


Fig. 74. "Privacy means my life is a black box, except for the items I choose to share with others." By Lauren, age 32

The solution is SQL **views**:

- For an SQL query, we can generate a view that represents the result of that query.
- Views can be used to only reveal certain columns (attributes after `SELECT`) and rows (defined by the `WHERE` clause) for access control.

Fine-grained DAC using SQL views

Accounts A1, A2, A3

Relations: Employee(Name, SIN, DOB, Address, Salary, Dpt)

Create a view

```
> A2: CREATE VIEW CSEmployeePublicInfo
      SELECT Name, DOB, Address FROM Employee
      WHERE Dpt = "CS";
```

The table owner (A2) creates a view that only exposes the (Name, DOB, Address) information for Employees in the CS department.

Relation-level privilege via views

```
> A2: GRANT SELECT ON CSEmployeePublicInfo TO A3;
```

The table owner (A2) grants user A3 the privilege to run SELECT queries on the restrict view instead of the whole Employee table.

Fine-grained DAC: what about write operations?

Accounts A1, A2, A3

Relations: Employee(Name, SIN, DOB, Address, Salary, Dpt)

Column-specific update privilege

```
> A2: GRANT UPDATE ON Employee (Address) TO A3;
```

The table owner (A2) grants user A3 the privilege to UPDATE the Employee table but only on the Address attribute.

We can also add additional restrictions with *triggers* (we will see these later)

From DAC to RBAC

Q: If we have DAC in the SQL language, why do we need RBAC?

- Need to manually change privileges for multiple users who want to perform the same task, or when a user changes positions in an organization (i.e., **roles**).

RBAC for databases

Creating and using roles

```
> Admin: CREATE ROLE "DptAdmin", "CompanyHR";

> Admin: GRANT "DptAdmin" TO A1;
> Admin: GRANT "CompanyHR" TO A3;

> A2: GRANT SELECT ON CSEmployeePublicInfo TO "DptAdmin";
> A2: GRANT UPDATE ON Employee(Address) TO "CompanyHR";
```

What about MAC?

We show a case study that aims to implement MAC for a database: multi-level security (MLS).

The theory behind MLS is the Bell-LaPadula **confidentiality** model:

- There are security classifications or security levels applied to
 - *Subjects*: i.e., database users — security clearances
 - *Objects*: i.e., each cell in a table — security classifications
- An example of security levels:
Top Secret > Secret > Classified > Unclassified
- Security goal: ensures that information does not flow to those not cleared for that level.
- Principles (simplified view):
 - *The simple security property*: S can read O iff $L(S) \geq L(O)$ (no read up)
 - *The star property*: S can write O iff $L(S) \leq L(O)$ (no write down)













Recall: Bell-LaPadula

Principles:

- *The simple security property:* S can read O iff $L(S) \geq L(O)$ (no read up)
- *The star property:* S can write O iff $L(S) \leq L(O)$ (no write down)

Q: Who can read what? Who can write what?

 Alice: Secret
 Trent: Top secret
A:  Bob: Classified

Object	Sec. Class	Can read?	Can write?
1	Top secret		  
2	Secret	 	 
3	Classified	  	

MLS table example

- Users with different clearances see different versions of reality

Name		Salary		Performance		TC
Smith	U	40000	C	Fair	S	S
Brown	C	80000	S	Good	C	S

- Assume **Name** is the primary key
- Each attribute has a classification label and a value at that label.
- TC label (Tuple Classification) = *Highest* clearance for any of its attributes.
- **Primary key label \leq Lowest clearance for any of its attributes.**

Q: Why having this requirement?

A: Otherwise a user may see a partial record without knowing what that record is about.

MLS read-down by filtering

What is the output of `SELECT * FROM Employee` for different users?

Name		Salary		Perf		TC
Smith	U	40000	C	Fair	S	S
Brown	C	80000	S	Good	C	S

Filtering the table for users having **classified** clearance:

Name		Salary		Perf		TC
Smith	U	40000	C	-	C	C
Brown	C	-	C	Good	C	C

Filtering the table for users having **unclassified** clearance:

Name		Salary		Perf		TC
Smith	U	-	U	-	U	U

More examples: MLS read-down

Levels are: $U < C < S$

Name		Salary		Perf		TC
Alice	U	40000	U	Fair	C	C
Bob	C	80000	C	Good	C	C
Carol	C	80000	S	Good	C	S
Dave	S	80000	S	Fair	S	S

Q: How do we filter the table for users with clearance levels S, C, and U?

A: S gets the full table.

More examples: MLS read-down

Levels are: $U < C < S$

Name		Salary		Perf		TC
Alice	U	40000	U	Fair	C	C
Bob	C	80000	C	Good	C	C
Carol	C	80000	S	Good	C	S
Dave	S	80000	S	Fair	S	S

Q: How do we filter the table for users with clearance levels S, C, and U?

A: C gets

Name		Salary		Perf		TC
Alice	U	40000	U	Fair	C	C
Bob	C	80000	C	Good	C	C
Carol	C	-	C	Good	C	C

More examples: MLS read-down

Levels are: $U < C < S$

Name		Salary		Perf		TC
Alice	U	40000	U	Fair	C	C
Bob	C	80000	C	Good	C	C
Carol	C	80000	S	Good	C	S
Dave	S	80000	S	Fair	S	S

Q: How do we filter the table for users with clearance levels S, C, and U?

A: U gets

Name		Salary		Perf		TC
Alice	U	40000	U	-	U	U

MLS invisible polyinstantiation

- A user with **low clearance** attempts to insert data in a field that already contains **higher classification** data.
- Rejecting an update could leak information downwards.

Name		Salary		Perf		TC
Smith	U	40000	C	Fair	S	S
Brown	C	80000	S	Good	C	S

A user with **classified** clearance issues a write-up:

```
UPDATE Employee SET Perf = "Great" WHERE Name = "Smith";
```

Name		Salary		Perf		TC
Smith	U	40000	C	Fair	S	S
Smith	U	40000	C	Great	C	C
Brown	C	80000	S	Good	C	S

We do not merge automatically! Why?

MLS visible polyinstantiation

- A user with **high clearance** attempts to insert data in a field that already contains **lower classification** data.
- Overwriting the low data would result in leaking information downwards.

Name		Salary		Perf		TC
Smith	U	40000	C	Fair	S	S
Brown	C	80000	S	Good	C	S

A user with **secret** clearance issues a write-down:

```
UPDATE Employee SET Perf = "Bad" WHERE Name = "Brown";
```

Name		Salary		Perf		TC
Smith	U	40000	C	Fair	S	S
Brown	C	80000	S	Good	C	S
Brown	C	80000	S	Bad	S	S

An explicit declassification is needed to merge the instantiations. Or maybe you'd like to keep some information private...

Recap: DB Access Control

SQL Basics

```
> SELECT <col> FROM <object> WHERE <condition>;  
> UPDATE <object> SET <col=val> WHERE <condition>;
```

DAC:

```
> GRANT <privilege> ON <object> TO <user>;  
> GRANT <privilege> ON <object> TO <user> WITH GRANT OPTION;  
> GRANT UPDATE ON <object> (<col>) TO <user>  
> REVOKE <privilege> ON <object> FROM <user>;  
> CREATE VIEW <view> SELECT (...); SELECT <col> FROM <view>;
```

RBAC:

```
> CREATE ROLE <role>; GRANT <role> TO <user>;
```

Recap: DB Access Control

MAC through MLS:

- Attributes have classification labels, you can only see those that have classification equal or lower to your clearance (you won't see a row if the primary key has higher classification).
- TC label is the highest classification of the row; the primary key label has the lowest classification of the row.
- *Invisible polyinstantiation*: a user with **low clearance** inserts data in a field that already has **high classification** data.
- *Visible polyinstantiation*: a user with **high clearance** inserts data in a field that already has **low classification** data.

Module outline

- 1 Introduction to database security
- 2 Access control
- 3 Integrity**
- 4 Others

Security requirements for a database

- **Access control** Access control
 - who can read? who can write?
- Authentication
 - how do we know if a DB client (or server) is not masquerading as someone else
- Confidentiality
 - what if the DB server is compromised? what about network tapping?
- **Integrity** Integrity
 - how do we guarantee that the data is in an intact and sensible state
- Availability
 - redundancy, failover
- Auditability
 - a.k.a. provenance, proving how we ended up with a specific state

Isn't integrity covered in crypto-protocols?

We are talking about a different type of integrity here.

- In cryptography: integrity means that data cannot be changed without being detected
- In databases: integrity means that the data records are in a sensible/correct state

We will cover the following types of integrity properties:

- Element integrity
- Referential integrity
- All-or-nothing/Atomicity

The goal of ensuring integrity is to prevent users from making changes that will result in an invalid database state. These changes can be **either intentional or unintentional**.

Element integrity

Example on element integrity violations

```
CREATE TABLE Employee (Name VARCHAR(255), Age INTEGER);  
INSERT INTO Employee VALUES ("SMITH", 400);
```

Q: What is the problem here? Developer mistake?

A: The type system is not expressive enough. There is no way to restrict that Age must be in a proper range (e.g., 0-150).

And there are even more tricky situations, for example:

- At all times, there is at most one employee can have the Position attribute set to "CEO".
- A salary increase cannot exceed 100% of the current salary.

Check element integrity with triggers

A typical way to enforce element integrity is to use **triggers**, i.e., procedures that are automatically executed after each write operation, including INSERT, UPDATE, DELETE, ... queries

An example on SQL trigger

```
CREATE TRIGGER AgeCheck ON Employee
  AFTER INSERT, UPDATE
  FOR EACH ROW
  BEGIN
    IF NEW.Age >= 150
      BEGIN
        RAISERROR ("Invalid age")
      END
  END
END;
```

Foreign key

Table: FamilyInfo

Last (PK)	Address	City	State	Zip (FK)
ADAMS	212 Market St.	Columbus	OH	43210
BENCHLY	501 Union St.	Chicago	IL	60603
CARTER	411 Elm St.	Columbus	OH	43210

Last (FK)	First
ADAMS	Charles
ADAMS	Edward
BENCHLY	Zeke
CARTER	Marlene
CARTER	Beth
CARTER	Ben

Table: NameInfo

Zip (PK)	Airport
43210	CMH
60603	ORD

Table: AirportInfo

Foreign key

The foreign key in a table points at a primary key in another table.

Foreign key in table creation

```
CREATE TABLE FamilyInfo (  
  Last VARCHAR(255) NOT NULL,  
  Address VARCHAR(1024),  
  City VARCHAR(128),  
  State VARCHAR(128),  
  Zip VARCHAR(128),  
  PRIMARY KEY (Last),  
  FOREIGN KEY (Zip) REFERENCES AirportInfo(Zip),  
);
```

Referential integrity

Referential integrity ensures that each value of a foreign key *refers* to a valid primary key value, i.e. **there are no dangling foreign keys**.

One use case: to prevent accidental or intentional deletion of records that are still being used.

Last (PK)	Address	City	State	Zip (FK)
ADAMS	212 Market St.	Columbus	OH	43210
BENCHLY	501 Union St.	Chicago	IL	60603
CARTER	411 Elm St.	Columbus	OH	43210

Table: FamilyInfo

Zip (PK)	Airport
43210	CMH
60603	ORD

Table: AirportInfo

For example: here we cannot delete a tuple in `AirportInfo` if its primary key is being used (as a foreign key) in `FamilyInfo`

Inconsistent state

Recall that integrity is about ensuring the data records are in a sensible/correct state **at all times**.

But what if a transaction requires two or more write operations?

For example: transfer money from Alice to Bob requires two UPDATE:

- UPDATE Ledger SET Balance = Balance - 100 WHERE Name = "Alice";
- UPDATE Ledger SET Balance = Balance + 100 WHERE Name = "Bob";

Q: What happens if the database fails after the first UPDATE?

A: The money would be lost!

Transaction as an all-or-nothing mechanism

Transaction (abort)

```
BEGIN TRANSACTION;
```

```
UPDATE Ledger SET Balance = Balance - 100 WHERE Name = "Alice";
```

```
UPDATE Ledger SET Balance = Balance + 100 WHERE Name = "Bob";
```

```
COMMIT TRANSACTION;
```

Transaction as an all-or-nothing mechanism

Transaction (commit or rollback)

```
BEGIN TRANSACTION;
UPDATE Ledger SET Balance = Balance - 100 WHERE Name = "Alice";
SELECT @balance = Balance FROM Ledger WHERE Name = "Alice";
IF @balance < 0
  BEGIN
    ROLLBACK TRANSACTION;
  END
ELSE
  BEGIN
    UPDATE Ledger SET Balance = Balance + 100 WHERE Name = "Bob";
    COMMIT TRANSACTION;
  END
END
```

Data race

Notice that in the prior example, we used an unusual syntax to update the balance:

Atomic update (implicit)

```
UPDATE Ledger SET Balance = Balance - 100 WHERE Name = "Alice";
```

If used on its own (i.e., not in a transaction context), this is implicitly translated into a transaction:

Atomic update (explicit)

```
BEGIN TRANSACTION;  
  SELECT @balance = Balance FROM Ledger WHERE Name = "Alice";  
  UPDATE Ledger SET Balance = @balance - 100 WHERE Name = "Alice";  
COMMIT TRANSACTION;
```

Why must we enclose it within a transaction? (see next slide)

Data race

If two clients send the request concurrently, what will be the result?

Client 1

```
SELECT @balance = Balance
  FROM Ledger WHERE Name = "Alice";

UPDATE Ledger SET Balance =
  @balance - 100 WHERE Name = "Alice";
```

Client 2

```
SELECT @balance = Balance
  FROM Ledger WHERE Name = "Alice";

UPDATE Ledger SET Balance =
  @balance - 100 WHERE Name = "Alice";
```

One possible interleaving:

Transaction interleavings

```
SELECT @balance = Balance FROM Ledger WHERE Name = "Alice";
SELECT @balance = Balance FROM Ledger WHERE Name = "Alice";
UPDATE Ledger SET Balance = @balance - 100 WHERE Name = "Alice";
UPDATE Ledger SET Balance = @balance - 100 WHERE Name = "Alice";
```

Q: How much is deducted from Alice's balance?

Transaction as a serialization mechanism

Transaction interleavings

```
BEGIN TRANSACTION;  
  SELECT @balance = Balance FROM Ledger WHERE Name = "Alice";  
  UPDATE Ledger SET Balance = @balance - 100 WHERE Name = "Alice";  
COMMIT TRANSACTION;  
BEGIN TRANSACTION;  
  SELECT @balance = Balance FROM Ledger WHERE Name = "Alice";  
  UPDATE Ledger SET Balance = @balance - 100 WHERE Name = "Alice";  
COMMIT TRANSACTION;
```

Recap: Integrity

- Integrity in DBs refers to ensuring the records are in a sensible/correct state
- **Triggers** (for **element integrity**): some code that runs after some instructions (e.g., INSERT, UPDATE) to check the (element) integrity.
- **Referential integrity**: we can define foreign keys, which point at a primary key in another table; referential integrity ensures there are no dangling foreign keys.
- **Atomicity**: if a transaction requires more than one operation, we might want to enclose it with BEGIN TRANSACTION (then we can rollback if something goes wrong, or commit the transaction if everything went fine).
- Atomicity also prevents data race issues.

Module outline

- 1 Introduction to database security
- 2 Access control
- 3 Integrity
- 4 Others**

Security requirements for a database

- ① **Access control**: who can read/write
- ② Authentication: how do we know if a DB client/server is not masquerading as someone else
- ③ Confidentiality: how to protect the data (at rest and in transit)
- ④ **Integrity**: how to guarantee that the data is in an intact and sensible state
- ⑤ Availability: redundancy, Failover
- ⑥ Auditability: keeping logs, proving how we ended up with a specific state

Authentication

This is a recap of what we learned from last module...

Q: How does a client authenticate a DBMS server?

A: Certificates

Q: How does a DBMS server authenticate a client?

A: Some possibilities:

- Passwords
- Certificates
- LDAP (Lightweight Directory Access Protocol) server

Confidentiality

Q: What does confidentiality in databases mean?

A: Protect the content of the database

The DBMS is simply an application that runs on some OS, alongside with other applications.

- Perhaps that machine itself is stolen and an attacker then removes the hard-drive, and attempts to read off the database contents from the hard-drive.
- Perhaps that other applications are compromised and attackers simply scan over your file system and extract all files related to the database content.
- Perhaps that storage provider itself is malicious, especially in the cloud computing setting, and are curious about what you store in your database.

Confidentiality

Solution? If trust is an issue, check if cryptography can be helpful.

- File-level encryption
- Column-level encryption

Q: Obviously the key cannot be stored alongside the data, then in this case, how do you supply the key to the DBMS?

A: Many possible solutions, e.g., establish a secure channel with the DBMS via TLS and send the key, etc.

Availability

Availability is about recognizing the fact that:

- Transactions can fail due to physical problems.
 - System crashes. Disk failures.
 - Physical problems/catastrophes: power failures, floods, fire, thefts.
- Contingency plans are needed to *recover* from these events

High availability in enterprise settings

- Redundancy: reduce risk that service is affected from some component failure transparently transfer operations to another functioning component.
 - Uninterrupted power supplies.
 - Multiple hard-drives in RAID configurations (with error-detection codes or error-correction codes).
- Database clusters: Redundancy by more machines. Load-balancing among clustered machines.
- Failover: deal with catastrophes etc., when machines are down.
 - Clustered machines are in the same physical location, so all machines may be down.
 - Primary system handles traffic regularly WHILE secondary system takes over in case of failures.

Auditability

Expecting the DBMS will never fail in access control or integrity is a dangerous thought!

In the event of a data breach, we want to be able to:

- **retroactively** identify who has run these queries without authorization.
- hold users **accountable** and **deter** such accesses.
- comply with relevant legislation, e.g. HIPAA for health data.

Auditability

- Set an audit policy (or policies) to observe queries received by the DBMS.
- DBMS generates an audit trail or log of events that comply with the audit policy. This log can be processed later into DB tables.
- Archive the audit log periodically to ensure *availability* of the logs for future.

Recap: Security requirements for a database

- 1 Access control: who can read/write
- 2 Authentication: how do we know if a DB client/server is not masquerading as someone else
- 3 Confidentiality: how to protect the data (at rest and in transit)
- 4 Integrity: how to guarantee that the data is in an intact and sensible state
- 5 Availability: redundancy, Failover
- 6 Auditability: keeping logs, proving how we ended up with a specific state

CS 458 / 658: Computer Security and Privacy

Module 6 – Database Security and Privacy

Part 2 – Database Privacy: Inference and Syntactic Notions of Privacy

Fall 2022

Module outline

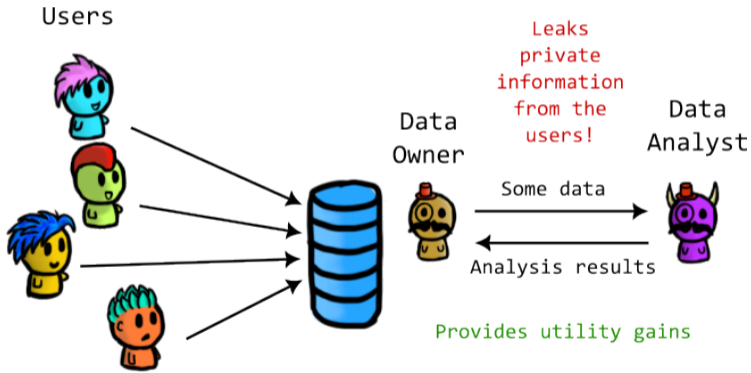
- 5 Introduction: privacy and utility

- 6 Inference attacks
 - SQL attacks
 - Census attacks
 - Linking attacks

- 7 Syntactic Notions of Privacy
 - k -anonymity
 - ℓ -diversity
 - t -closeness
 - Limitations of syntactic privacy notions

System Model

Possible scenario: some users provide their data to a data owner, the owners shares some of this data with a data analyst. This has privacy and utility implications.

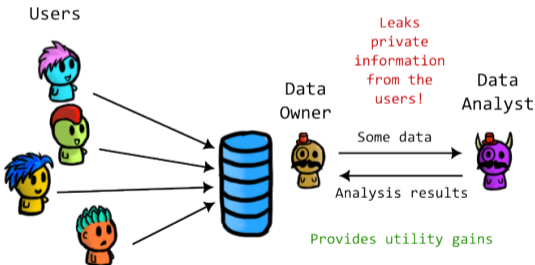


There are variations of this model, of course... (e.g., maybe the data owner/collector is a service provider that does the analysis itself)

System Model

Q: Let's name at least three possible scenarios that fit this model, together with their privacy leakage, and the utility gains

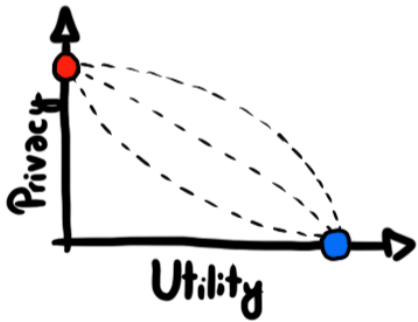
Some examples filled during the lecture:



Scenario	Privacy risks	Utility gain
Social media	We publish lots of sensitive info, pictures, etc.	We use social media apps for free
Virtual assistants	They hear what we say	They help us; also the recordings help improve them
Census	Personal info in census	Helps in determining how to allocate resources

A conflict of privacy and utility

Regardless of how we quantify privacy and utility, they always go against each other:



Q: What's an easy approach to be in the red and blue points here?

A: Red point: do not provide/release/publish any data.
Blue point: release all data without protecting it.

Finding data release mechanisms to be somewhere in between and enjoy a good **privacy-utility trade-off** is hard!

Roadmap

First, we will see different examples of **inference attacks** that extract private information from the released data.

- ① SQL-based attacks
- ② Census attacks
- ③ Linking attacks

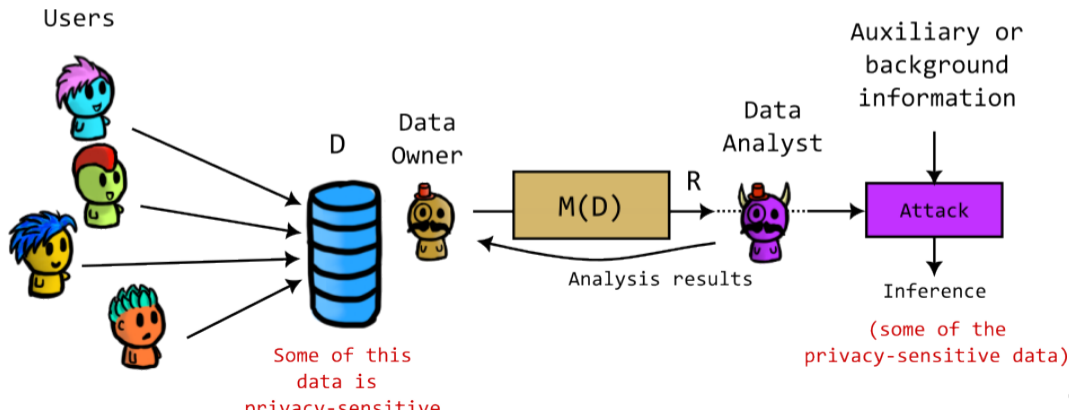
Then, we will see **syntactic notions** of privacy (and attacks against them)

- ① k -anonymity
- ② ℓ -diversity
- ③ t -closeness

Finally, we will see a **semantic notion** of privacy: **differential privacy**

Inference Attacks

The adversary (e.g., the system provider, a data analyst, an eavesdropper, etc.) gains access to some (sanitized) data. The adversary could have auxiliary/background information. An inference attack **infers** privacy-sensitive information from this information.



Attacks that use SQL queries: setup

Consider a setting where we have a large relational database (a table) with some sensitive attributes.

- **Utility** — we want to allow certain SQL queries, as data analysts want to learn interesting properties of the data.
 - e.g., get the average salary of everyone in this company
- **Privacy** — We also want to protect the privacy of the users whose data is in the database.
 - e.g., without revealing each individual's salary

We saw that privacy and utility are conflicting goals.

Attacks that use SQL queries: releasing aggregates

- We could think of restricting “dangerous” queries, and allowing “harmless” others.
- You’re forbidden to issue queries that fetch a particular attribute
 - e.g., `SELECT Salary FROM Employee ...`
- But using aggregates (e.g., `SUM`, `AVG`, or `COUNT`) are allowed
 - e.g., `SELECT AVG(Salary) FROM Employee ...`

You will learn attacks that use aggregate SQL queries. You need to **understand** these attacks and know how to **perform** them.

Aggregate queries that we will use

```
SELECT SUM(<Attribute>) FROM <Table> WHERE <Condition>
SELECT COUNT(*) FROM <Table> WHERE <Condition>
```

Do aggregates protect against inference?

Data analysts could **infer** sensitive data, through **output of allowed aggregate queries**.

Inference does not have to be a full and accurate recovery of the sensitive data (although sometimes it can be).

- e.g., the employee's salary is \$12,345.67

Instead, even a partial revealing of the data is considered as a successful inference and hence a privacy leak.

- e.g., the salary is within the range of \$10,000 and \$20,000

Our goal is to minimize (unintentional) leaks of sensitive data to the data analysts through the allowed queries.

SQL inference attack: single query

One single query that directly outputs the sensitive data

Direct attack

```
SELECT SUM(Salary) FROM Employee
  WHERE Name = "Adams"
  AND (Gender = "M" OR Gender = "F" OR Gender = "NB");
```

Countermeasure

If the SELECT clause output includes less than k results, then drop the query.

k is usually application-specific.

Inference attack: multiple queries

Name (PK)	Age	Zip	Salary
Alice	32	N2L 0G7	55 000 CAD
Bob	34	N2L 3E4	65 000 CAD
Carol	26	N2L 0E1	35 000 CAD
...			

Table: Employee (example only)

Countermeasure

If the SELECT clause output includes less than k results, then drop the query (e.g., $k = N/10$)

Q: Only issuing “SELECT SUM(Salary) FROM Employee WHERE (...)” queries, how can we get Alice’s salary?

A: We need two queries:

Q_1 : SELECT SUM(Salary) FROM Employee;

Q_2 : SELECT SUM(Salary) FROM Employee WHERE Name != "Alice";

Return $Q_1 - Q_2$.

Inference attack: multiple queries

The countermeasure was not enough! We need a better one:

Countermeasure

Suppose the database has a total of N records. If the `SELECT` clause output includes less than k results, or more than $N - k$ results (but less than N results), then drop the query.

NOTE: a query that includes N records (i.e., all records) is OK.

Inference attack: multiple queries (v2)

Name (PK)	Age	Zip	Salary
Alice	32	N2L 0G7	55 000 CAD
Bob	34	N2L 3E4	65 000 CAD
Carol	26	N2L 0E1	35 000 CAD
...			

Table: Employee (example only)

Countermeasure

Suppose the database has a total of N records. If the SELECT clause output includes less than k results, or more than $N - k$ results (but less than N results), then drop the query (e.g., $k = N/10$).

Q: Only issuing “SELECT SUM(Salary) FROM Employee WHERE (...)” queries, how can we get Alice’s salary?

Assumptions:

- “Alice” is in the dataset, but you don’t know anything else about them.
- The median age in the company is 30.

Name (PK)	Age	Zip	Salary
Alice	?	?	?
⋮	⋮	⋮	

Countermeasure

Suppose the database has a total of N records. If the SELECT clause output includes less than k results, or more than $N - k$ results (but less than N results), then drop the query (e.g., $k = N/10$).

The median age in the company is 30

Q₁: SELECT SUM(Salary) FROM Employee WHERE

Q₂: SELECT SUM(Salary) FROM Employee WHERE

Q₃: SELECT SUM(Salary) FROM Employee WHERE

Inference attack: multiple queries (v2)

Name (PK)	Age	Zip	Salary
Alice	32	N2L 0G7	55 000 CAD
Bob	34	N2L 3E4	65 000 CAD
Carol	26	N2L 0E1	35 000 CAD
...			

Table: Employee (example only)

Countermeasure

Suppose the database has a total of N records. If the SELECT clause output includes less than k results, or more than $N - k$ results (but less than N results), then drop the query (e.g., $k = N/10$).

A: We need three queries: let $C = (\text{Age} < 30)$

Q_1 : SELECT SUM(Salary) FROM Employee;

Q_2 : SELECT SUM(Salary) FROM Employee WHERE Name = "Alice" OR C;

Q_3 : SELECT SUM(Salary) FROM Employee WHERE Name = "Alice" OR NOT C;

Return $Q_2 + Q_3 - Q_1$.

Inference attack: multiple queries (v2) (Bonus)

Name (PK)	Age	Zip	Salary
Alice	32	N2L 0G7	55 000 CAD
Bob	34	N2L 3E4	65 000 CAD
Carol	26	N2L 0E1	35 000 CAD
...			

Table: Employee (example only)

Countermeasure

Suppose the database has a total of N records. If the SELECT clause output includes less than k results, or more than $N - k$ results (but less than N results), then drop the query (e.g., $k = N/10$).

Another valid solution proposed in the classroom:

A: We need three queries: let $C = (\text{Age} < 30)$

Q_1 : SELECT SUM(Salary) FROM Employee;

Q_2 : SELECT SUM(Salary) FROM Employee WHERE Name != "Alice" AND C ;

Q_3 : SELECT SUM(Salary) FROM Employee WHERE Name != "Alice" AND NOT C ;

Return $Q_1 - Q_2 - Q_3$.

Inference attack: tracker attack

This is also called the tracker attack.

We find a **tracker** query T that satisfies the restriction (i.e., that more than k but less than $N - k$ records satisfy):

- e.g., `SELECT SUM(Salary) FROM Employee WHERE Age < 30;`
- Note that this tracker will depend on the problem

Let C be the constraint $C = (\text{Age} < 30)$.

Tracker attack

```
Q1: SELECT SUM(Salary) FROM Employee WHERE Name = "Alice" OR C;
```

```
Q2: SELECT SUM(Salary) FROM Employee WHERE Name = "Alice" OR NOT C;
```

```
Q3: SELECT SUM(Salary) FROM Employee;
```

$Q_1 + Q_2 - Q_3$ reveals the secret salary.

More practice!

Countermeasure

Database has N records. A query is dropped if it includes less than k results, or more than $N - k$ results (but less than N results) ($k = N/4$). Only $\text{SUM}(\langle \text{Att} \rangle)$ queries are allowed.

Q: The `Employee` table has columns (`Name` (**PK**), `ZIP`, `DOB`, `Salary`). The `ZIP` codes are all in Waterloo. `DOB` are just the years, between 1980 and 2000, with approximately the same number of records per year.

How do we get Alice's salary with three queries max.?

A: e.g., choose $C = \text{DOB} > 1990$

Then run the three queries in the previous slide

More practice!

Countermeasure

Database has N records. A query is dropped if it includes less than k results, or more than $N - k$ results (but less than N results) ($k = N/4$).

Q: The Employee table has columns (Name (PK), ZIP, DOB, Salary). The ZIP codes are all in Waterloo. DOB are just the years, between 1980 and 2000, with approximately the same number of records per year.

How do we get Alice's ZIP with `COUNT(*)` queries only?

One solution

There are many ways of doing this, since we do not have a restriction on the number of queries. This is one possible solution:

A: We can run the following three queries for each ZIP z :

N : `SELECT COUNT(*) From Employee`

Q_1 : `SELECT COUNT(*) From Employee WHERE $DOB > 1990$ OR (Name = Alice AND ZIP = z)`

Q_2 : `SELECT COUNT(*) From Employee WHERE $DOB \leq 1990$ OR (Name = Alice AND ZIP = z)`

If Alice does not have $ZIP = z$, then $Q_1 + Q_2 = N$.

If Alice does have $ZIP = z$, then $Q_1 + Q_2 = N + 1$

This is very inefficient in the sense that it requires too many queries. In A3 you have to be a bit more creative to get higher marks.

SQL-based inference attacks: conclusions

Having controls on the **type** and **shape** of queries is unlikely to be sufficient. We need better (and more systematic) solutions to protect data privacy.

You need to know how to design these attacks (you need them for Assignment 3 written Q4!).

- There is not a general way of doing these attacks, you have to understand them and then be creative with the actual problem at hand.
- You cannot assume things that are not explicitly stated in the problem (e.g., Alice's gender)

The census reconstruction attack

Suppose that we have some statistical data about a Census block:

- 1 There are four people in total.
- 2 Two of these people have age 17.
- 3 Two of these people self-identify as White.
- 4 Two of these people self-identify as Asian.
- 5 The average age of people who self-identify as White is 30.
- 6 The average age of people who self-identify as Asian is 32.

Q: Can you guess the age of everyone in the dataset?

Inference across multiple sources

- What we have seen so far uses information in a single database only.
- The inference problem is **more severe** when the adversary has access to multiple data sources as long as they can link and aggregate the information from different sources.
- It is more severe because access controls rarely apply across data sources.
- How does the adversary get **external data sources**?
 - Use publicly available data, e.g. census data, regional records.
 - Purchase data records from a data broker
 - Governments might also share their dossiers with each other.
 - Large companies may collect information about their customers.

Data linking

Now, what can we learn from combining these datasets that we didn't learn before?

If these datasets include identifiers that are verinyms, or persistent pseudonyms, one can *link* data records across these datasets to learn more information about an individual or an entity.

We will see a series of inference attacks on public data releases that are supposed to protect the privacy of the data suppliers but failed.

Anonymity failure: AOL Search Data Set

- August 6, 2006: AOL released 20 million search queries from 658,000 users over a 3-month period in 2006.
- AOL assigned a random number to each user:
 - 4417749 “numb fingers”
 - 4417749 “60 single men”
 - 4417749 “landscapers in Lilburn, GA”
 - 4417749 “dog that urinates on everything”
 - 711391 “life in Alaska”
- August 9: New York Times article re-identified user 4417749
 - Thelma Arnold, 62-year old widow from Lilburn, GA

Takeaway: simply attaching a random number to each users' record is insufficient to get a high level of nymity.

Anonymity failure: NYC Taxi dataset release

- NYC Taxi Commission released 173 million “anonymized” NYC Taxi trip logs due to a FOIA request
- Each trip log includes information about the trip as well as persistent pseudonyms for each taxi itself.
 - pick-up location (latitude, longitude) and time
 - drop-off location (latitude, longitude) and time
 - MD5 hash of the taxi medallion number
 - MD5 hash of the driver license number
- These parameters were collected in order to learn about taxi usage and traffic patterns.

Anonymity failure: NYC Taxi dataset release

Anonymity problem 1 with this data release: Pick-up / drop-off times and locations can be correlated with celebrities' travels (background knowledge from other news sources).

Example:

You know that a celebrity was spotted leaving the JFK airport at 6pm. \implies You look for pick-up records near JFK around 6pm and see where they drop-off. \implies After filter out infeasible locations, you might be able to identify the taxi that they took and deduce where they lived or visited.

Takeaway: Perhaps these drop-offs/pick-ups could be published at a lower granularity, at the cost of lower utility for statistical analysis of traffic etc?

Anonymity failure: NYC Taxi dataset release

Anonymity problem 2 with this data release: Does hashing help with hiding identities of the drivers and taxicabs?

Background: These two identifiers have the following structures:

- License numbers are 6 or 7 digit numbers
- Medallion numbers are either
 - [0-9] [A-Z] [0-9] [0-9]
 - [A-Z] [A-Z] [0-9] [0-9] [0-9]
 - [A-Z] [A-Z] [A-Z] [0-9] [0-9] [0-9]

Q: How would you uncover their identities?

A: brute-force! There are only 1 million license numbers at most, and 17 million medallion numbers.

Takeaway: Hashing identifiers does not provide anonymity. With a small input space, a dictionary attack can be conducted efficiently.

Anonymity failure: Massachusetts Insurance Health Records

Massachusetts released “anonymized” health records:

- ZIP code
- Gender
- Date of birth
- **Health information**

Massachusetts’ voter registration lists contains:

- ZIP code
- Gender
- Date of birth
- **Name**

Fun fact: 87% of U.S. population can be uniquely identified using ZIP code, gender, and date of birth!

Lessons learned from linking attacks

- Datasets included data that was useful for research (primary data), as well as some identifiers (“quasi-identifiers”).
- *“Quasi-identifiers”* can be used to link data across multiple records in the same dataset (NYC Taxi dataset or AOL search data) or across different datasets (Massachusetts case).
- *Background knowledge* relating to the primary data, can be used to further de-anonymize records.

Privacy vs utility trade-off

What can be done about each type of data in these data releases?

For **quasi-identifiers**:

- Reduce granularity to *deter* linking: e.g. year instead of DOB, only first couple digits of zip code. \implies Increases anonymity set.
- Remove attribute(s) to *prevent* linking altogether: e.g. no random number in AOL dataset or no medallion/license number in NYC taxi dataset. Will reduce utility of the dataset.

For **primary data**:

- Reduce granularity.
- Remove sensitive attributes.
- Publish aggregate statistics.
- Change values slightly (add randomness).

Module outline

- 5 Introduction: privacy and utility
- 6 Inference attacks
 - SQL attacks
 - Census attacks
 - Linking attacks
- 7 Syntactic Notions of Privacy**
 - k -anonymity
 - ℓ -diversity
 - t -closeness
 - Limitations of syntactic privacy notions

Syntactic Notions of Privacy

- Syntactic notions of privacy ensure that the **released data** satisfies a certain property.
- The data to be protected is typically a **table**, and the set of attributes can be classified into:
 - Identifiers: uniquely identify a participant
 - Quasi-identifiers: in combination with external information, can identify a participant (ZIP, DOB, Gender, etc.)
 - Confidential attributes: attributes (columns) that contains privacy-sensitive information.
 - Non-confidential attributes: are not considered sensitive
- We are going to see three syntactic notions of privacy:
 - k -anonymity
 - ℓ -diversity
 - t -closeness

Syntactic Notions of Privacy

- We are going to see three syntactic notions of privacy:
 - *k*-anonymity
 - *l*-diversity
 - *t*-closeness

- For each syntactic notion of privacy, you will learn (and need to know):
 - What it **is**
 - Why it provides **privacy**
 - How to **compute** it
 - How to **provide** it (e.g., by publishing data in a privacy-preserving way by following certain – given – utility rules)

k-anonymity example, with a single quasi-identifier

A simple dataset, where the quasi-identifier is ZIP.

ZIP	Party affiliation
N1CFFA	Green Party
G0ANFA	Liberal Party
N1C5YN	Green Party
N2J0HJ	Conservative Party
N1C4KH	Green Party
G0A3G4	Conservative Party
G0A3GN	Liberal Party
N2JWBV	New Democratic Party
N2JWBV	Liberal Party

k-anonymity example, with multiple quasi-identifiers

A simple dataset table (quasi-identifiers are ZIP and DOB)

ZIP	DOB	Party affiliation
N1CFF	1962-01-24	Green Party
G0ANF	1975-12-30	Liberal Party
N1C5YN	1966-10-17	Green Party
N2J0HJ	1996-08-14	Conservative Party
N1C4KH	1963-04-06	Green Party
G0A3G4	1977-07-09	Conservative Party
G0A3GN	1973-08-14	Liberal Party
N2JWBV	1990-11-02	New Democratic Party
N2JWBV	1990-01-25	Liberal Party

k-anonymity example, with multiple quasi-identifiers

Q: What is the k -anonymity level of this table? (ZIP and DOB are QI)

ZIP	DOB	Party affiliation
N1C***	196*_**_**	Green Party
G0A***	197*_**_**	Liberal Party
N1C***	196*_**_**	Green Party
N2J***	199*_**_**	Conservative Party
N1C***	196*_**_**	Green Party
G0A***	197*_**_**	Conservative Party
G0A***	197*_**_**	Liberal Party
N2J***	199*_**_**	New Democratic Party
N2J***	199*_**_**	Liberal Party

A: The table is 3-anonymous

ZIP	DOB	Party affiliation
N1C***	196*_**_**	Green Party
N1C***	196*_**_**	Green Party
N1C***	196*_**_**	Green Party

G0A***	197*_**_**	Liberal Party
G0A***	197*_**_**	Liberal Party
G0A***	197*_**_**	Conservative Party

N2J***	199*_**_**	Conservative Party
N2J***	199*_**_**	New Democratic Party
N2J***	199*_**_**	Liberal Party

k-anonymity practice!

Age	Gender	...
23	F	
25	F	
33	F	
35	F	
27	M	
30	M	
32	M	
21	NB	
25	NB	

Age and Gender are the quasi-identifiers

Q: What is the *k*-anonymity in the following cases?

- ① We hide the Age
- ② We hide the Gender (but not the Age)
- ③ We report the most significant digit of Age, plus the Gender
- ④ We only report the most significant digit of Age, but not the Gender

A: The table is:

- ① 2-anonymous
- ② 1-anonymous (or just not anonymous)
- ③ 1-anonymous (or just not anonymous)
- ④ 4-anonymous

More k -anonymity practice!

k -anonymity

For each published record, there exists at least $k - 1$ other records with the same quasi-identifier.

Gender	DOB	Party affiliation
M	1968-**-**	Green Party
F	1975-**-**	Liberal Party
O	1966-**-**	Green Party
M	1962-**-**	Green Party
M	1962-**-**	Conservative Party
O	1966-**-**	Conservative Party
F	1973-**-**	Liberal Party
F	1973-**-**	Liberal Party
O	1968-**-**	Green Party
F	1975-**-**	Liberal Party

Q: What is the largest k for which this table is k -anonymous? (quasi-identifiers are Gender and DOB)

- 1 As in the left
- 2 If we hide the least-significant digit of year
- 3 If we hide the gender column
- 4 Both 2 and 3

A: 1, 3, 2, 4

More k -anonymity practice: how to provide it

Age	Province	...
21	ON	
23	ON	
26	ON	
32	ON	
33	ON	
35	ON	
36	ON	
43	ON	
45	ON	
<hr/>		
22	BC	
24	BC	
26	BC	
27	BC	
32	BC	
33	BC	
43	BC	
45	BC	
49	BC	

Age and Province are the quasi-identifiers. We must reduce the granularity of Age to provide some k -anonymity.

Q: If we replace the Age with age ranges [20-29], [30-39], [40-49], what is the k -anonymity level?

A: The table would be 2-anonymous

Q: Can you **design** ranges that provide a higher level of k -anonymity, with the constraints that 1) ranges must cover all ages from 20 to 49, 2) you must create 3 age ranges, 3) each range must contain at least one record

A: [20-26], [27-35], [36-49] makes the table 3-anonymous

k-anonymity and privacy

A 3-anonymized table (organized by equi-class)

ZIP	DOB	Party affiliation
N1C***	196*_*_*_*	Green Party
N1C***	196*_*_*_*	Green Party
N1C***	196*_*_*_*	Green Party
G0A***	197*_*_*_*	Liberal Party
G0A***	197*_*_*_*	Liberal Party
G0A***	197*_*_*_*	Conservative Party
N2J***	199*_*_*_*	Conservative Party
N2J***	199*_*_*_*	New Democratic Party
N2J***	199*_*_*_*	Liberal Party

You know what *k*-anonymity is, **how to compute it**, and **how to provide it**

Q: Why does it provide privacy?

A: We cannot identify the actual record of a user (that provided a record) based on their quasi-identifiers. This can make it hard to guess the user's confidential attributes

Q: Is this good enough?

Homogeneity attack

ZIP	DOB	Party affiliation
N1C***	196*-*-**	Green Party
N1C***	196*-*-**	Green Party
N1C***	196*-*-**	Green Party
G0A***	197*-*-**	Liberal Party
G0A***	197*-*-**	Liberal Party
G0A***	197*-*-**	Conservative Party
N2J***	199*-*-**	Conservative Party
N2J***	199*-*-**	New Democratic Party
N2J***	199*-*-**	Liberal Party

Q: If you know Alice (N1C***, 196*-*-**) is in this table, what will you learn?

A: Alice's party affiliation is the Green Party

Homogeneity attack

It happens when sensitive values lack diversity. In the worst case, for a given quasi-identifier, all sensitive data values are identical.

Background knowledge attack

ZIP	DOB	Party affiliation
N1C***	196*_*_*_**	Green Party
N1C***	196*_*_*_**	Green Party
N1C***	196*_*_*_**	Green Party
G0A***	197*_*_*_**	Liberal Party
G0A***	197*_*_*_**	Liberal Party
G0A***	197*_*_*_**	Conservative Party
N2J***	199*_*_*_**	Conservative Party
N2J***	199*_*_*_**	New Democratic Party
N2J***	199*_*_*_**	Liberal Party

Q: If you know Bob (G0A***, 197*_*_*_**) is in this table, and Bob does not like Liberal Party, what will you learn?

A: Bob's party affiliation is the Conservative Party

Background knowledge attack

It filters out infeasible values and, in the worst case, narrows the inference down to a single value.

l-diversity example

A 3-anonymized 3-diversified table

ZIP	DOB	Salary
N3P***	199*_*_*_*	20K
N3P***	199*_*_*_*	15K
N3P***	199*_*_*_*	25K
H1A***	196*_*_*_*	100K
H1A***	196*_*_*_*	90K
H1A***	196*_*_*_*	120K
S4N***	197*_*_*_*	50K
S4N***	197*_*_*_*	60K
S4N***	197*_*_*_*	65K

You know what l -diversity **is**, **how to compute it**, and (potentially) **how to provide it**.

Q: Why does it provide privacy?

A: It alleviates the issues of k -anonymity that we saw above. Given someone's quasi-identifiers, and access to the published database, l -diversity makes it harder to guess that individual's sensitive values

Q: Is this good enough?

Similarity attack

ZIP	DOB	Salary	Disease
N3P***	199*_**_**	20K	gastric ulcer
N3P***	199*_**_**	15K	gastritis
N3P***	199*_**_**	25K	stomach cancer
H1A***	196*_**_**	100K	heart attack
H1A***	196*_**_**	90K	flu
H1A***	196*_**_**	120K	bronchitis
S4N***	197*_**_**	50K	COVID
S4N***	197*_**_**	60K	kidney stone
S4N***	197*_**_**	65K	pneumonia

Q: If you know Charles, who earns a low salary is in this table, what will you learn?

A: Charles has a stomach disease

Similarity attack

If the sensitive values of an equi-class are different but have the same (or similar) semantic meaning, ℓ -diversity does not prevent the adversary from learning this.

What went wrong?

ZIP	DOB	Virus X Test
N3P***	199*_**_**	Positive
N3P***	199*_**_**	Positive
N3P***	199*_**_**	Positive
... 45 more positive cases ...		
N3P***	199*_**_**	Negative
<hr/>		
H1A***	196*_**_**	Negative
H1A***	196*_**_**	Negative
H1A***	196*_**_**	Negative
... 945 more negative cases ...		
H1A***	196*_**_**	Positive

- The data in each equi-class (i.e., records that share the same quasi-identifier) is **unexpectedly skewed**.
- The “unexpected” feeling comes from the distribution of sensitive values of the whole dataset being different than the distribution of the sensitive values per class

t-closeness

t-closeness

The distribution of sensitive values in each equi-class is no further than a threshold t from the overall distribution of the sensitive values in the table.

Equi-class: each set of identical quasi-identifiers is an equi-class.

We have to define a notion of distance between distributions. For example, see the [original paper](#) that proposes *t*-closeness on ICDE'07.

Variational distance (or EMD Categorical Distance – using Equal Distance)

For two distributions over m values, $\mathbf{P} = (p_1, \dots, p_m)$, $\mathbf{Q} = (q_1, \dots, q_m)$:

$$D[\mathbf{P}, \mathbf{Q}] = \frac{1}{2} \sum_{i=1}^m |p_i - q_i|$$

t-closeness example

ZIP	Virus	
N3P***	Pos	x15
N3P***	Neg	x25
H1A***	Pos	x15
H1A***	Neg	x45



$$D[\mathbf{P}_{N3P}, \mathbf{Q}] = \frac{1}{2} \left(\left| \frac{15}{40} - \frac{30}{100} \right| + \left| \frac{25}{40} - \frac{70}{100} \right| \right) = 0.075$$

$$D[\mathbf{P}_{H1A}, \mathbf{Q}] = \frac{1}{2} \left(\left| \frac{15}{60} - \frac{30}{100} \right| + \left| \frac{45}{60} - \frac{70}{100} \right| \right) = 0.05$$

t-close with $t = 0.075$ (the **maximum** of these values)

If you had more equi-classes, that would be more **P**'s, so you would have to compute more distances $D[\mathbf{P}, \mathbf{Q}]$, and then pick the **maximum** distance as t .

Some notes on how to compute t -closeness

- If you had k equi-classes, you would have to compute k distances, and take the maximum of those distances as the value of t .
- If you had m distinct sensitive values, the histograms would have m bars.
- If you had more than one sensitive attribute (column), you can compute t -closeness for *each* sensitive value (e.g., a table is t_1 -close w.r.t. Salary and t_2 -close w.r.t. Virus).

Note: the examples from the [original paper](#) by Li et al. use more complicated distance metrics.

Recap

- Privacy vs. utility conflict
- SQL inference attacks (single query, multiple queries, tracker attacks)
 - You need to understand how to attack using aggregate queries when there are simple countermeasures
- Census reconstruction attacks, linking attacks
 - You need to understand the intuition behind these attacks, and how to do simple examples.
- Syntactic notions of privacy (k -anonymity, ℓ -diversity, t -closeness)
 - Homogeneity attack, background knowledge attack, similarity attack, skewness attack (know how to answer questions similar to the ones in the slides)
 - Know what they are, why they provide privacy, how to compute k , ℓ , and t (no need to know anything other than EMD Categorical for t -closeness), and how to provide them (only k -anonymity)

CS 458 / 658: Computer Security and Privacy

Module 6 – Database Security and Privacy

Part 3 – Differential Privacy

Fall 2022

Module outline

- 8 The Dinur-Nissim reconstruction attack
- 9 Introduction to Differential Privacy
- 10 Properties of Differential Privacy
- 11 Differentially Private Mechanisms
 - Laplace mechanism
 - Randomized Response
 - Discrete mechanisms
- 12 Recap and Practice

We are being too honest...

In all the cases covered in the inference attacks in Part 2, we always gave *faithful* results:

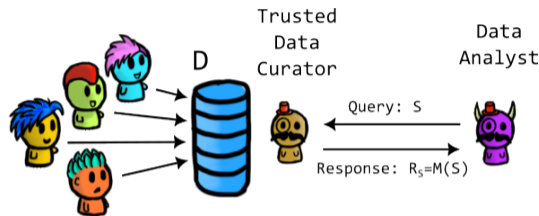
For example:

- The SUM of the salaries
- The census data

The data release was deterministic. If we instead added some random noise to the answers, some of these attacks would be much harder (e.g., the SQL attacks we saw)

The **Dinur-Nissim reconstruction attack** illustrates why, when a mechanism adds too little noise when responding to aggregated queries, an adversary can still reconstruct the database *with high accuracy and efficiency*.

Let's formalize the setup



- There is a database, D , which potentially contains sensitive information about individuals.
- The **database curator** has access to the full database. **We assume the curator is trusted.**
- The **data analyst** consumes the data by asking a series of **queries** to the curator. Each query is denoted as S and the curator provides a **response** to query S with R_S . **The analyst may be honest or malicious.**
- The way in which the curator responds to queries is called the **mechanism**. Formally, $M : S \rightarrow R_S$. We'd like a mechanism that
 - gives statistically useful responses but
 - avoids leaking sensitive information about individuals.

Threat model

- We assume the adversary **knows all but one** attributes. The unknown attribute is binary (e.g., COVID)
- The adversary can ask aggregated queries (e.g., COUNT(*))

Name	ZIP	DOB	COVID
Alice	K8V 7R6	5/2/1984	1
Bob	V5K 5J9	2/8/2001	0
Charlie	V1C 7J2	10/10/1979	1
David	R4K 5T1	4/4/1944	0
Eve	G7N 8Y3	1/1/1954	1

For example, the adversary could ask for how many rows satisfying condition (Name = "Charlie" OR DOB > 1980) have COVID = 1.

Since the adversary knows all other attributes, they pick **arbitrary rows**, and ask for how many of those rows have the COVID bit set to 1.

Threat model: more general

Representing the database as binary, where each record contains k bits, the adversary knows all a 's, and wants to know the b 's.

The adversary can query for counts of how many rows from a given set have $b = 1$.

For simplicity, we represent each query as the *set of rows* where the adversary wants to get the sum of the bits b .

Basically, there is a secret binary vector $B = [b_1, b_2, \dots, b_n]$ and the adversary can query for sums of bits in arbitrary positions.

$$D = \left[\begin{array}{cccc|c} a_{\{1,1\}} & a_{\{1,2\}} & \dots & a_{\{1,k-1\}} & b_1 \\ a_{\{2,1\}} & a_{\{2,2\}} & \dots & a_{\{2,k-1\}} & b_2 \\ \vdots & \vdots & \dots & \vdots & \vdots \\ a_{\{n,1\}} & a_{\{n,2\}} & \dots & a_{\{n,k-1\}} & b_n \end{array} \right]$$

For example, if $n = 5$, then the query $S = [1, 0, 1, 0, 0]$ will be asking for the sum $b_1 + b_3$, and query $S = [0, 1, 0, 1, 1]$ asks for $b_2 + b_4 + b_5$.

But the data curator adds noise to the summation!

Curator mechanism

Given a secret bit vector $B = [b_1, b_2, \dots, b_n]$.

Upon receiving a query $S = [s_1, s_2, \dots, s_n]$, the curator first calculates the true answer:

$$\langle S, B \rangle = \sum_{i=1}^n s_i \cdot b_i$$

Then, the curator adds some random noise N :

$$R_S = \langle S, B \rangle + N$$

Example: for $B = [1, 1, 1, 0, 0]$ and $S = [1, 0, 1, 0, 1]$, then the true answer is $\langle S, B \rangle = 2$.

Curator mechanism: noise

Upon receiving a query $S = [s_1, s_2, \dots, s_n]$, the curator computes

$$R_S = \langle S, B \rangle + N$$

Let's consider a noise that is upper-bounded by:

$$|N| < E$$

Q: What are the pros/cons of using a noise with large E ?

A: More noise (larger E) provides more privacy, but less utility

The inefficient attack

Theorem: If the analyst is allowed to ask 2^n subset queries to a dataset of n users (n rows), and the curator adds noise with some bound E , then based on the results, the adversary can reconstruct the database in (at least) all but $4E$ positions.

e.g., $E = \frac{n}{400} \implies$ reconstruction of 99% entries in the database.

Algorithm:

- For an attacker, there are 2^n candidate databases (all possible values of B).
 - e.g., if the true database has 3 users, we have $2^3 = 8$ candidate databases
- For each candidate database $C \in \{0, 1\}^n$, if there exists a query S such that $|\langle S, C \rangle - R_S| > E$, rule out C .
- Any database candidate not ruled out (C) differs with the actual database (D) by $4E$ at max.

The inefficient attack - Example

Let's see an example.

In the example, we have a database with $n = 3$ users (rows).

$$D = \left[\begin{array}{cccc|c} a_{\{1,1\}} & a_{\{1,2\}} & \cdots & a_{\{1,k-1\}} & b_1 \\ a_{\{2,1\}} & a_{\{2,2\}} & \cdots & a_{\{2,k-1\}} & b_2 \\ a_{\{3,1\}} & a_{\{3,2\}} & \cdots & a_{\{3,k-1\}} & b_3 \end{array} \right]$$

The adversary queries for all 2^n combinations $\{0, 1\}^n$, i.e.,
 $S \in \{[0, 0, 0], [0, 0, 1], [0, 1, 0], \dots, [1, 1, 1]\}$

The curator uses noise N sampled randomly from $\{-0.5, +0.5\}$.

You need to know **how** to perform this reconstruction attack, and understand **why** the reconstruction attack is possible.

The inefficient attack - Example (I)

True database has:
 $B=[1,0,1]$

Query



- $S=[0,0,0]$
- $S=[0,0,1]$
- $S=[0,1,0]$
- $S=[0,1,1]$
- $S=[1,0,0]$
- $S=[1,0,1]$
- $S=[1,1,0]$
- $S=[1,1,1]$

Candidate Binary Vectors

- $C=[0,0,0]$
- $C=[0,0,1]$
- $C=[0,1,0]$
- $C=[0,1,1]$
- $C=[1,0,0]$
- $C=[1,0,1]$
- $C=[1,1,0]$
- $C=[1,1,1]$

0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	1
0	0	1	1	0	0	1	1
0	1	1	2	0	1	1	2
0	0	0	0	1	1	1	1
0	1	0	1	1	2	1	2
0	0	1	1	1	1	2	2
0	1	1	2	1	2	2	3

Expected answers
 (without noise)
 for each candidate



Noise sampled
 at random from
 $\{-0.5, +0.5\}$

The inefficient attack - Example (II)

True database has:

$B=[1,0,1]$

Query True answer Noise (secret) Reported answer

$S=[0,0,0]$

$R_S=0+(+0.5)=0.5$

$S=[0,0,1]$

$R_S=1+(-0.5)=0.5$

$S=[0,1,0]$

$S=[0,1,1]$

$S=[1,0,0]$

$S=[1,0,1]$

$S=[1,1,0]$

$S=[1,1,1]$

Candidate Binary Vectors

$C=[0,0,0]$ $C=[0,0,1]$ $C=[0,1,0]$ $C=[0,1,1]$ $C=[1,0,0]$ $C=[1,0,1]$ $C=[1,1,0]$ $C=[1,1,1]$

0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	1
0	0	1	1	0	0	1	1
0	1	1	2	0	1	1	2
0	0	0	0	1	1	1	1
0	1	0	1	1	2	1	2
0	0	1	1	1	1	2	2
0	1	1	2	1	2	2	3

Expected answers (without noise) for each candidate

Noise sampled at random from $\{-0.5, +0.5\}$

The inefficient attack - Example (III)

True database has:

$B=[1,0,1]$

Query True answer Noise (secret) Reported answer

$S=[0,0,0]$ $R_S=0+(+0.5)=0.5$
 $S=[0,0,1]$ $R_S=1+(-0.5)=0.5$
 $S=[0,1,0]$ $R_S=0+(+0.5)=0.5$
 $S=[0,1,1]$ $R_S=1+(+0.5)=1.5$
 $S=[1,0,0]$ $R_S=1+(-0.5)=0.5$
 $S=[1,0,1]$ $R_S=2+(-0.5)=1.5$
 $S=[1,1,0]$ $R_S=1+(-0.5)=0.5$
 $S=[1,1,1]$ $R_S=2+(-0.5)=1.5$

Candidate Binary Vectors

$C=[0,0,0]$ $C=[0,0,1]$ $C=[0,1,0]$ $C=[0,1,1]$ $C=[1,0,0]$ $C=[1,0,1]$ $C=[1,1,0]$ $C=[1,1,1]$

0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	1
0	0	1	1	0	0	1	1
0	1	1	2	0	1	1	2
0	0	0	0	1	1	1	1
0	1	0	1	1	2	1	2
0	0	1	1	1	1	2	2
0	1	1	2	1	2	2	3

Expected answers (without noise) for each candidate



Noise sampled at random from $\{-0.5, +0.5\}$

The inefficient attack - Example (IV)

True database has:

$B=[1,0,1]$

Query

True answer

Noise (secret)

Reported answer

$S=[0,0,0]$

$R_S=0+(+0.5)=0.5$

$S=[0,0,1]$

$R_S=1+(-0.5)=0.5$

$S=[0,1,0]$

$R_S=0+(+0.5)=0.5$

$S=[0,1,1]$

$R_S=1+(+0.5)=1.5$

$S=[1,0,0]$

$R_S=1+(-0.5)=0.5$

$S=[1,0,1]$

$R_S=2+(-0.5)=1.5$

$S=[1,1,0]$

$R_S=1+(-0.5)=0.5$

$S=[1,1,1]$

$R_S=2+(-0.5)=1.5$

Candidate Binary Vectors

$C=[0,0,0]$	$C=[0,0,1]$	$C=[0,1,0]$	$C=[0,1,1]$	$C=[1,0,0]$	$C=[1,0,1]$	$C=[1,1,0]$	$C=[1,1,1]$
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	1
0	0	1	1	0	0	1	1
1	1	1	2	1	1	1	2
0	0	0	0	1	1	1	1
1	1	1	1	1	2	1	2
0	0	1	1	1	1	1	1
1	1	1	2	1	2	2	2

Expected answers (without noise) for each candidate

Noise sampled at random from $\{-0.5, +0.5\}$

The inefficient attack - Practice

True database has:
 $B = [?, ?, ?]$

Query

$S = [0, 0, 0]$

$S = [0, 0, 1]$

$S = [0, 1, 0]$

$S = [0, 1, 1]$

$S = [1, 0, 0]$

$S = [1, 0, 1]$

$S = [1, 1, 0]$

$S = [1, 1, 1]$

Reported answer

$R_S = 0.5$

$R_S = 0.5$

$R_S = 1.0$

$R_S = 1.0$

$R_S = 0.5$

$R_S = 1.5$

$R_S = 1.5$

$R_S = 2.5$

Candidate Binary Vectors

$C = [0, 0, 0]$

$C = [0, 0, 1]$

$C = [0, 1, 0]$

$C = [0, 1, 1]$

$C = [1, 0, 0]$

$C = [1, 0, 1]$

$C = [1, 1, 0]$

$C = [1, 1, 1]$

0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	1
0	0	1	1	0	0	1	1
0	1	1	2	0	1	1	2
0	0	0	0	1	1	1	1
0	1	0	1	1	2	1	2
0	0	1	1	1	1	2	2
0	1	1	2	1	2	2	3

Expected answers
 (without noise)
 for each candidate



Noise sampled
 at random from
 $\{-0.5, 0, +0.5\}$

Q: Can you guess the privacy-sensitive column B (or a list of candidate B 's)

A: There is only one candidate: $B = [1, 1, 0]$

The inefficient attack

- **Note:** If an adversary is allowed to ask a lot of queries, it does not matter how much (linear) noise is added to the database.
 - The adversary will be able to reconstruct a large fraction of the data!
- But again, for this attack to work, you need to send a large number of queries.
 - That's why it is inefficient / impractical!
- There is a more efficient attack:
 - **Theorem:** If the analyst is allowed to ask $O(n)$ queries to a dataset of n users, and the curator adds noise with some bound $E = O(\alpha\sqrt{n})$, then based on the results, a computationally efficient adversary can reconstruct the database in all but $O(\alpha^2 n)$ positions.

Module outline

- 8 The Dinur-Nissim reconstruction attack
- 9 Introduction to Differential Privacy**
- 10 Properties of Differential Privacy
- 11 Differentially Private Mechanisms
 - Laplace mechanism
 - Randomized Response
 - Discrete mechanisms
- 12 Recap and Practice

So far...

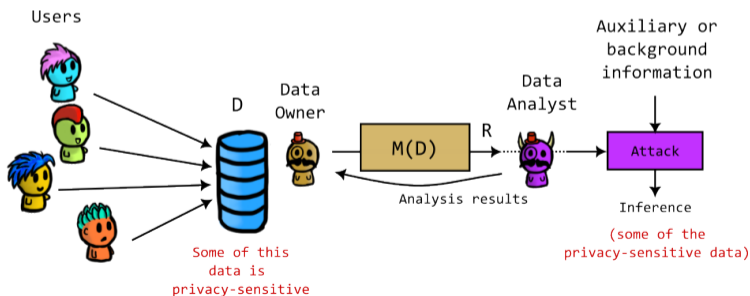
So far, we have seen some defenses and attacks:

- Restrict type and shape of SQL queries
 - Attacks are still possible (we saw SQL attacks that used SUM and COUNT queries)
- Syntactic notions of privacy (k -anonymity, ℓ -diversity, ...)
 - Attacks are still possible (skewness, background information...)
- Naive noise addition
 - Attacks are still possible (we saw the Dinur-Nissim reconstruction attack)

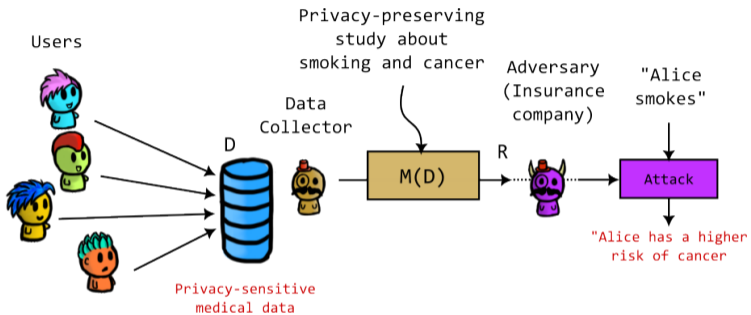
It seems like nothing works... we need a “notion” of privacy that provides *formal guarantees* against attacks.

How do we provide privacy against auxiliary information?

- We have a dataset D , with each user contributing to one entry (row) of the database.
- We have a release mechanism M that publishes some data $R = M(D)$.
- Can we provide privacy when the adversary has arbitrary aux. information?



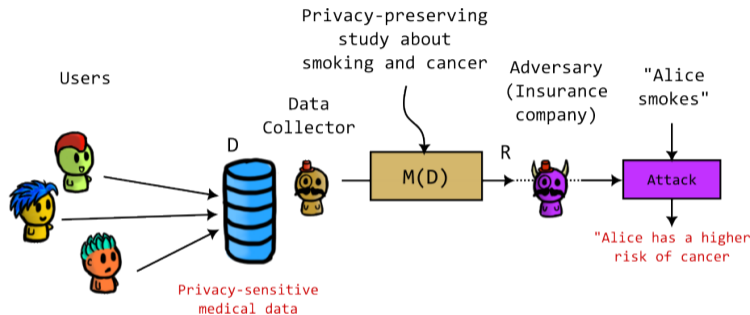
Example: strong auxiliary information



Q: Thanks to the study, the adversary learns that Alice has higher risk of cancer. Is this a violation of Alice's privacy? Is this the study's fault? Should we design an M to prevent this?

A: The adversary would've reached the same conclusion even if Alice hadn't participated in the study! We cannot prevent this (without completely destroying utility, i.e., not doing the study)

Example: strong auxiliary information

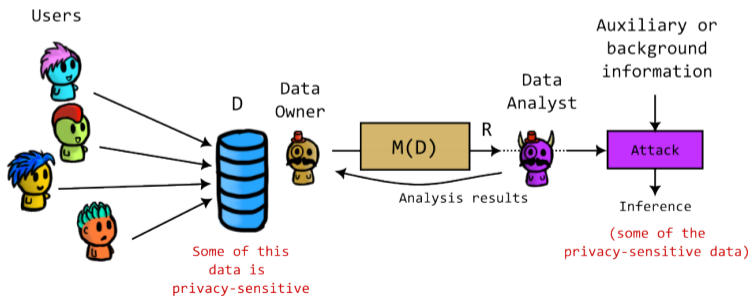


A: The adversary can still learn the same sensitive information about Alice, even though she wasn't even in the database

We cannot guarantee *absolute* privacy.

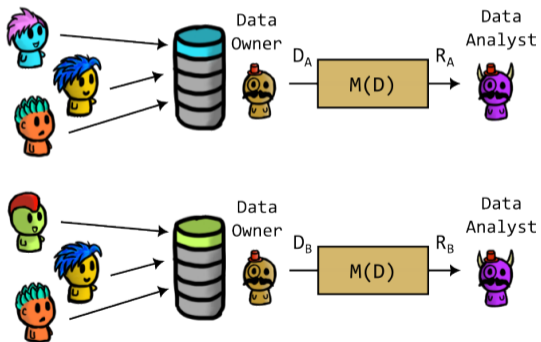
What if we are less ambitious...

- If the adversary has *strong background information*, there is nothing M can do about it!
- We should instead ensure that the adversary cannot gain significant **new** information from R (i.e., we want a “differential” and not an “absolute” privacy)



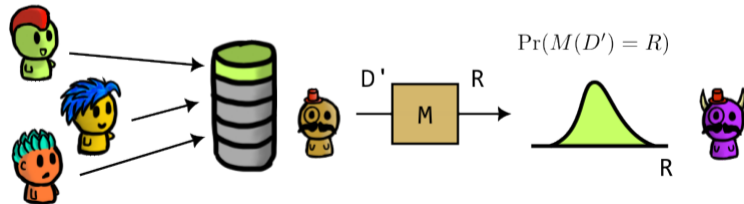
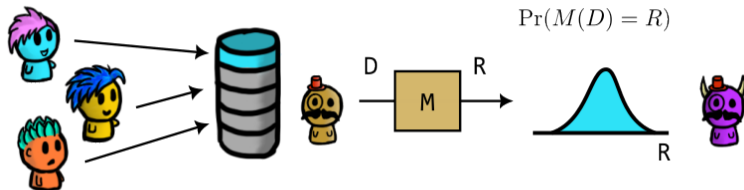
Possible privacy goal

What if we try to make these cases similar?



- We want the R_A and R_B to be “similar” ($R_A \approx R_B$).
- This would ensure M does not depend “too much” on any single user.
- However, note that M is randomized (e.g., adds noise).
- Thus, instead of ensuring $R_A \approx R_B$, we ensure their probability distributions are “similar”.

We're almost there...



- These datasets are usually called **neighboring** datasets (and usually denoted by D and D')

- We want $\Pr(M(D) = R)$ and $\Pr(M(D') = R)$ to be close (for all R)

- How do we define “close”?

How do we define “close” distributions?

This is a possible definition of “closeness”:

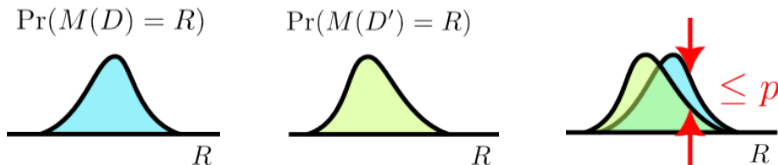
Tentative privacy definition (this is not an actual definition)

A mechanism M is private (with some privacy parameter p) if the following holds for all possible outputs R and all pairs of neighboring datasets (D, D') :

$$\Pr(M(D') = R) - p \leq \Pr(M(D) = R) \leq \Pr(M(D') = R) + p$$

Note that here smaller p means more privacy.

This would mean that:



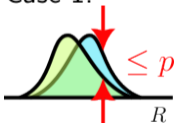
Does this really work?

Tentative privacy definition (this is not an actual definition)

A mechanism M is private (with some privacy parameter p) if the following holds for all possible outputs R and all pairs of neighboring datasets (D, D') :

$$\Pr(M(D') = R) - p \leq \Pr(M(D) = R) \leq \Pr(M(D') = R) + p$$

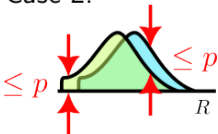
Case 1:



Case 1 seems fine...

Q: ... but do you see an issue with case 2?

Case 2:



A: There are some outputs R that can only happen if Bob was in the dataset (i.e., on D'). These rule out D . This is what allowed us to do Dinur-Nissim!

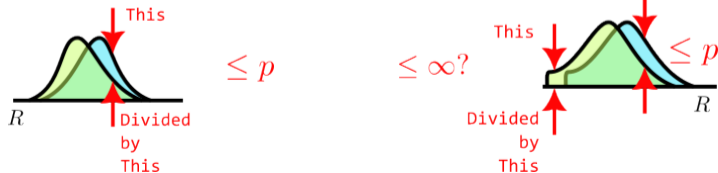
What if we make the distance multiplicative?

Tentative privacy definition II (this is not an actual definition)

A mechanism M is private (with some privacy parameter p) if the following holds for all possible outputs R and all pairs of neighboring datasets (D, D') :

$$\Pr(M(D') = R) \cdot \frac{1}{p} \leq \Pr(M(D) = R) \leq \Pr(M(D') = R) \cdot p$$

Again, smaller p (but $p \in [1, \infty)$) means more privacy. This would mean that:



This makes more sense, since $p = \infty$ means “no privacy”.

Finally: Differential Privacy

Instead of a “ p ”, we use e^ϵ , with privacy parameter ϵ :

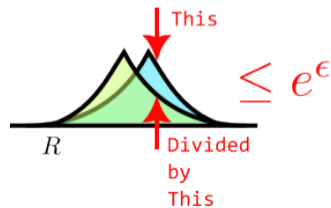
Differential Privacy

A mechanism $M : \mathcal{D} \rightarrow \mathcal{R}$ is ϵ -differentially private (ϵ -DP) if, for all possible outputs $R \in \mathcal{R}$ and all pairs of neighboring datasets (D, D') ($\in \mathcal{D}$):

$$\Pr(M(D) = R) \leq \Pr(M(D') = R) \cdot e^\epsilon$$

Some notes:

- We do not need the $e^{-\epsilon}$ on the left, since this must hold for all (D, D') ; for example, it also holds for (D', D) .
- $\epsilon \in [0, \infty)$; this ensures that $e^\epsilon \in [1, \infty)$



Finally: Differential Privacy

Differential Privacy

A mechanism $M : \mathcal{D} \rightarrow \mathcal{R}$ is ϵ -differentially private (ϵ -DP) if, for all possible outputs $R \in \mathcal{R}$ and all pairs of neighboring datasets $(D, D') (\in \mathcal{D})$:

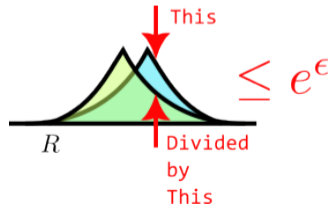
$$\Pr(M(D) = R) \leq \Pr(M(D') = R) \cdot e^\epsilon$$

Q: Which is “more private”: $\epsilon = 1$ or $\epsilon = 2$?

A: Smaller ϵ means more privacy; larger ϵ means less privacy

Q: What does $\epsilon = 0$ mean?

A: Perfect privacy! (independent of the database!)



Some notes

Differential Privacy

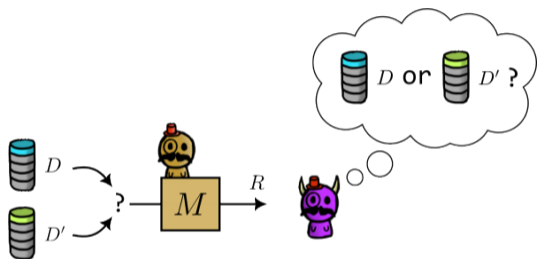
A mechanism $M : \mathcal{D} \rightarrow \mathcal{R}$ is ϵ -differentially private (ϵ -DP) if, for all possible outputs $R \in \mathcal{R}$ and all pairs of neighboring datasets $(D, D') (\in \mathcal{D})$:

$$\Pr(M(D) = R) \leq \Pr(M(D') = R) \cdot e^\epsilon$$

- DP was proposed in 2006 Cynthia Dwork et al. [\[DMNS06\]](#)
- They won the Gödel Price in 2017 and the Test-of-Time Award in 2016.
- Adopted by Apple, Google, Microsoft, and the US Census Bureau for the 2020 US Census, etc.
- There is no consensus on how small ϵ should be. “Roughly”:
 - $\epsilon < 0.1$ is high privacy ($e^{0.1} \approx 1.1$)
 - $0.1 < \epsilon < 1$ is good privacy ($e^1 \approx 2.7$)
 - $\epsilon > 5$ starts getting too big ($e^5 \approx 148$)
 - $\epsilon > 100\,000$ is crazy... yet some works use this

DP interpretation as a game

What does $\Pr(M(D) = R) \leq \Pr(M(D') = R) \cdot e^\epsilon$ even mean?

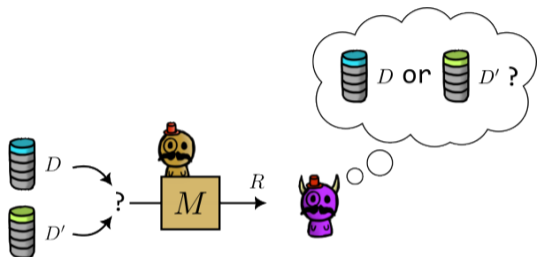


- We choose the input to be D or D' (at random)
- The adversary sees R , and we assume it knows M and knows that the input was **either D or D'** .

- These assumptions are many times unrealistic, but we want privacy even in this **worst-case scenario**
- The adversary computes $p_D = \Pr(M(D) = R)$ and $p_{D'} = \Pr(M(D') = R)$
- Optimal guess: the input was D if $p_D \geq p_{D'}$
- If M is ϵ -DP, the adversary's probability of error is

$$\frac{1}{e^\epsilon + 1} \leq p_{\text{error}} \leq 0.5$$


DP interpretation as a game



- If M is ϵ -DP, the adversary's probability of error is

$$\frac{1}{e^\epsilon + 1} \leq p_{\text{error}} \leq 0.5$$

What does this mean?

ϵ	p_{err} range	Privacy
0	$0.5 \leq p_{\text{err}} \leq 0.5$	Perfect!
0.1	$0.47 \leq p_{\text{err}} \leq 0.5$	Very high
1	$0.26 \leq p_{\text{err}} \leq 0.5$	OK?
5	$0.006 \leq p_{\text{err}} \leq 0.5$	Bad
10	$0.00004 \leq p_{\text{err}} \leq 0.5$	Meaningless?
100 000	$10^{-43430} \leq p_{\text{err}} \leq 0.5$	

About DP and empirical attack performance

DP ensures protection even against a strong adversary that knows that the input dataset was **either D or D'** (and it provides the guarantee for all possible outputs R , even those that are very unlikely to happen).

In practice, an algorithm that provides $\epsilon = 10$ **might** provide high *empirical* protection against existing attacks.

But why use DP as a defense if you are going to configure it to $\epsilon = 10$? The theoretical worst-case guarantee is meaningless at that point, you might as well use something that does not provide DP.

Module outline

- 8 The Dinur-Nissim reconstruction attack
- 9 Introduction to Differential Privacy
- 10 Properties of Differential Privacy**
- 11 Differentially Private Mechanisms
 - Laplace mechanism
 - Randomized Response
 - Discrete mechanisms
- 12 Recap and Practice

Properties of Differential Privacy

Depending on how we define **neighboring datasets**, we can distinguish between:

- Bounded Differential Privacy
- Unbounded Differential Privacy

Depending on who **runs the DP mechanism**, we have two broad settings:

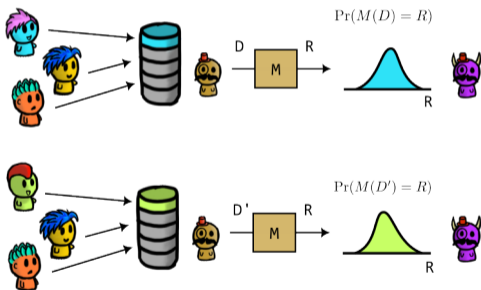
- Central Differential Privacy
- Local Differential Privacy

Differentially private mechanisms have some basic **properties**:

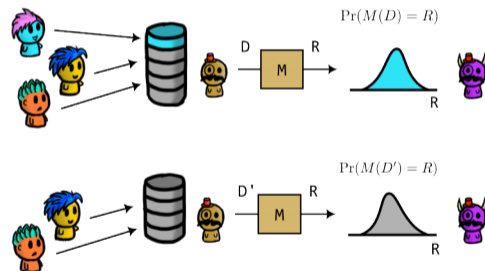
- Post-processing
- Group privacy
- Sequential composition
- Parallel composition

Bounded vs. Unbounded DP

When D and D' have the same number of entries (e.g., n) but differ in the value of one, we have **bounded DP**.



When D and D' are such that you get one by deleting an entry from the other one, we have **unbounded DP**.

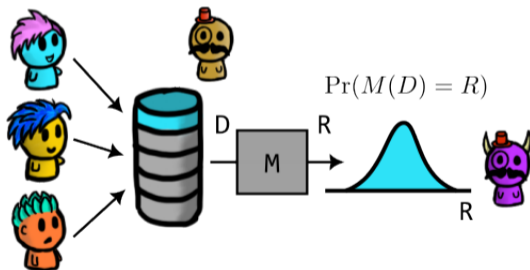


These are just slightly different guarantees of privacy. It is important to know which one your DP algorithm is providing. In practice, there is not a big difference.

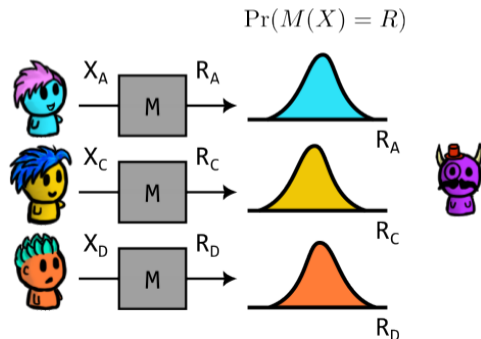
Central DP vs. Local DP

Depending on who runs the DP mechanism, we have two settings:

When there is a centralized (trusted) aggregator, we call it **central DP**:

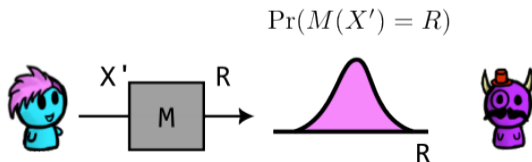
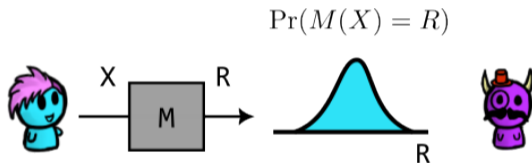


When each user runs the mechanism themselves, and reports it directly to the analyst (adversary), we call it **local DP**:



Local DP: what are the “neighboring datasets” here?

In the local setting, usually the user has a value X , and providing ϵ -DP means hiding whether the value was X or another value X' :



Local DP

M provides ϵ -DP (or ϵ -LDP – for Local Differential Privacy), if the following holds for all X, X' and outputs R :

$$\Pr(M(X) = R) \leq \Pr(M(X') = R) \cdot e^\epsilon$$

Sometimes this is defined for a notion of “neighboring inputs”, e.g., for all (X, X') such that $|X - X'| < 1$.

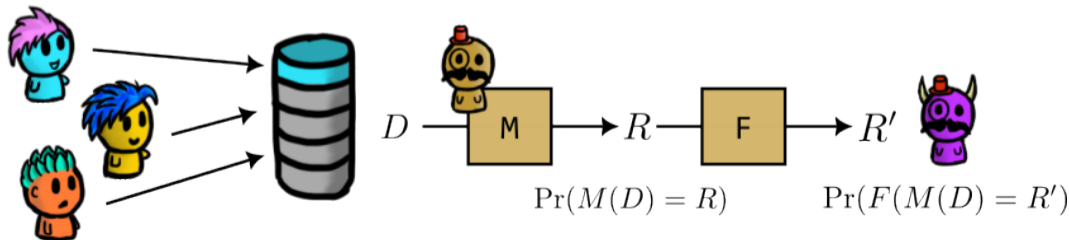
Properties: robustness to post-processing

If M is differentially private, given any deterministic or randomized function F , $F(M)$ is also differentially private.

In other words:

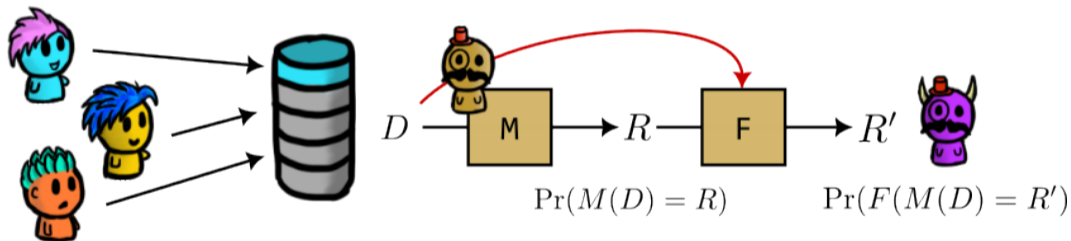
$$\Pr(M(D) = R) \leq \Pr(M(D') = R) \cdot e^\epsilon \Rightarrow \Pr(F(M(D)) = R) \leq \Pr(F(M(D')) = R) \cdot e^\epsilon$$

With an image: R' does not leak more than R (actually, R could leak more!).



This is not post-processing!

If F uses D again, this is of course **NOT** post-processing (it's more like "re-processing"?)



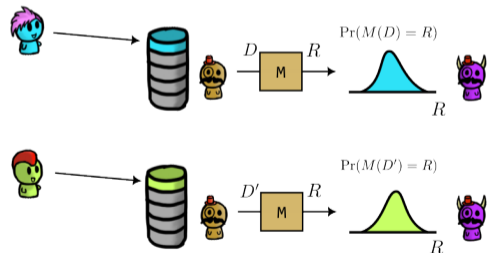
For example, if $F(D, R) = D$ you just leaked the whole database, and $\epsilon \rightarrow \infty$ (no privacy).

Properties: group privacy

If M provides ϵ -DP for all (D, D') that differ in 1 entry, it provides $n\epsilon$ -DP for all (D, D') that differ in n entries

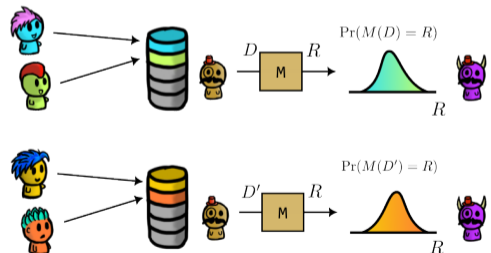
If this is ϵ -DP:

$$\Pr(M(D) = R) \leq \Pr(M(D') = R) \cdot e^\epsilon$$



Then this is 2ϵ -DP:

$$\Pr(M(D) = R) \leq \Pr(M(D') = R) \cdot e^{2\epsilon}$$

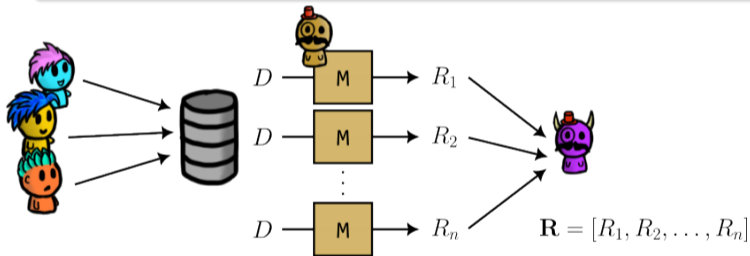


Proof:

$$\Pr(M(D_{A,B}) = R) \leq \Pr(M(D_{A,D}) = R) \cdot e^\epsilon \leq \Pr(M(D_{C,D}) = R) \cdot e^\epsilon \cdot e^\epsilon = \Pr(M(D_{C,D} = R)) \cdot e^{2\epsilon}$$

Properties: sequential composition

If we run an ϵ -DP mechanism independently n times on the same dataset D , publishing all results provides $n\epsilon$ -DP.



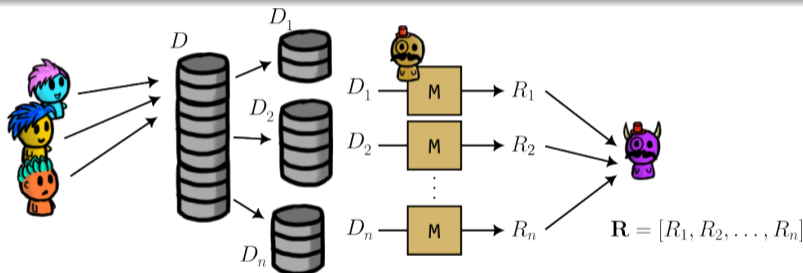
This means that releasing DP information that depends on the same data (or even the same user) decreases privacy **a lot**.

Here, we have $\Pr([M(D), \dots, M(D)] = \mathbf{R}) \leq \Pr([M(D'), \dots, M(D')] = \mathbf{R}) \cdot e^{n\epsilon}$

Note: if we run different mechanisms with $\epsilon_1, \epsilon_2, \dots, \epsilon_n$, publishing all results provides $(\sum_{i=1}^n \epsilon_i)$ -DP.

Properties: parallel composition

If we run an ϵ -DP mechanism independently over disjoint subsets of a dataset D , publishing all results provides ϵ -DP still.



Changing a row in $D = [D_1, D_2, \dots, D_n]$ would only change a row in D' , e.g., $D' = [D'_1, D_2, \dots, D_n]$, so $\Pr([M(D_1), M(D_2), \dots, M(D_n)] = \mathbf{R}) \leq \Pr([M(D'_1), M(D_2), \dots, M(D_n)] = \mathbf{R}) \cdot e^\epsilon$

Note: if we run different mechanisms with $\epsilon_1, \epsilon_2, \dots, \epsilon_n$ over disjoint subsets, publishing all results provides $(\max_i \epsilon_i)$ -DP.

Module outline

- 8 The Dinur-Nissim reconstruction attack
- 9 Introduction to Differential Privacy
- 10 Properties of Differential Privacy
- 11 Differentially Private Mechanisms**
 - Laplace mechanism
 - Randomized Response
 - Discrete mechanisms
- 12 Recap and Practice

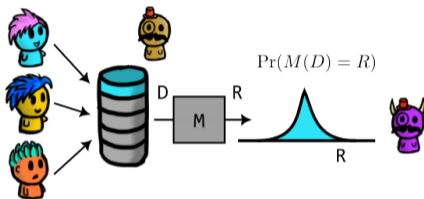
Roadmap

We are going to see examples of mechanisms M that provide differential privacy.

We will see:

- The **Laplace mechanism**:
 - A general mechanism to provide DP in continuous domains.
 - You need to know how to compute the sensitivity of the query and ϵ .
- The **Randomized Response** mechanism:
 - A mechanism to provide DP when the inputs and outputs are binary (and typically in the local model).
 - You need to know how to compute ϵ for variants of this.
 - You need to know how to do basic analysis to compute aggregate statistics
- For general **discrete mechanisms**:
 - You need to know how to compute ϵ .
- We will also see how to use the **properties** of DP we saw before.

The Laplace mechanism



- We are in the central model, and the analyst/adversary wants the COUNT or SUM of some attribute in the dataset.
- We denote the query by f (e.g., $f(D)$ would be true output of the query)

ℓ_1 -sensitivity of a query f

Is the maximum change (in ℓ_1 -norm, i.e., absolute value) in the output of the query when we run it over D vs. D' (for any D, D'):

$$\Delta_f = \max_{D, D'} |f(D) - f(D')|$$

Example: for count queries, $\Delta_f = 1$.

Example: for SUM(Salary) queries, $\Delta_f =$ “maximum salary difference possible”.

Laplace mechanism

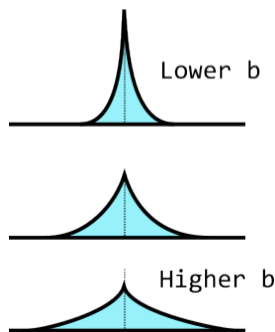
We reply to query f as $M(D) = f(D) + Y$ where Y is noise from a Laplacian distribution with parameter $b = \Delta_f/\epsilon$. This provides ϵ -DP.

Laplacian noise

Laplace distribution The Laplace probability density function (pdf) is:

$$p_Y(y) = \frac{1}{2b} \cdot e^{-\frac{|y|}{b}}$$

- (Disclaimer: I drew these, they are not *actually* Laplace pdfs).
- Higher b means higher noise.
- The variance of the Laplacian is $2b^2$.
- The distribution of $R \equiv M(D) = f(D) + Y$ is therefore a Laplace distribution centered at $f(D)$
- Mathematically, the Laplacian centered at $f(D)$:
 $p_Y(f(D) - R)$

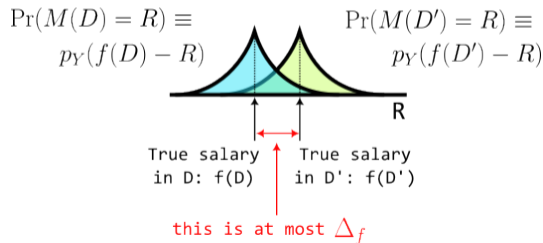


Back to the Laplacian mechanism

Laplacian mechanism

We reply to query f as $M(D) = f(D) + Y$ where Y is noise from a Laplacian distribution with parameter $b = \Delta_f/\epsilon$. This provides ϵ -DP.

Comparing $\Pr(M(D) = R) = p_Y(f(D) - R)$ vs. $\Pr(M(D') = R) = p_Y(f(D') - R)$.



- Remember, we set $b = \Delta_f/\epsilon$.
- This makes sense, because:
 - For a given query (fixed Δ_f), if we want more privacy (lower ϵ), we need more noise (higher b).
 - For a given privacy level (fixed ϵ), if a query has high sensitivity (higher Δ_f), we need more noise (higher b).

⁰For simplicity, we also use $\Pr(M(D) = R)$ for pdfs in the case R is not discrete

Proof: why does the Laplacian noise provide DP?

This slide is here for completeness. You won't be asked to prove things like this. But reading and understanding this proof will help you understand why the Laplacian distribution is "ideal" for DP.

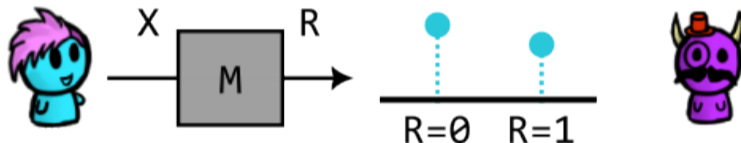
$$\begin{aligned} \frac{\Pr(M(D) = R)}{\Pr(M(D') = R)} &= \frac{p_Y(f(D) - R)}{p_Y(f(D') - R)} = \frac{\frac{1}{2b} \cdot \exp\left(-\frac{|f(D) - R|}{b}\right)}{\frac{1}{2b} \cdot \exp\left(-\frac{|f(D') - R|}{b}\right)} = \exp\left(-\frac{|f(D) - R| - |f(D') - R|}{b}\right) \\ &= \exp\left(\frac{|f(D') - R| - |f(D) - R|}{b}\right) \leq \exp\left(\frac{|f(D) - f(D')|}{b}\right) \leq \exp\left(\frac{\Delta_f}{b}\right) \end{aligned}$$

If we set $b = \Delta_f/\epsilon$, then we get $\frac{\Pr(M(D) = R)}{\Pr(M(D') = R)} \leq \exp(\epsilon)$, which is what we wanted for DP!

Local DP setting with a binary secret X and response R

We are in the local model. Alice has a secret bit X , and reports another bit R .

$$\Pr(M(X) = R)$$



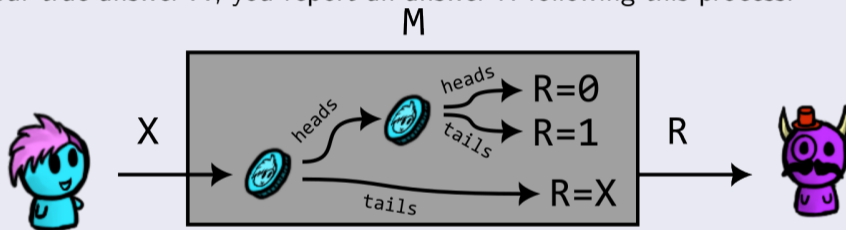
This can be used to answer to binary questions, e.g.,

- “Have you voted for party Y ?”
- “Have you tested positive for virus Z ?”
- “Have you cheated in exam W ?”

Randomized Response

Randomized Response

Given your true answer X , you report an answer R following this process:



Q: What are these probabilities?

(with an unbiased coin)

$$\Pr(R = 0 | X = 0)$$

$$\Pr(R = 1 | X = 0)$$

$$\Pr(R = 0 | X = 1)$$

$$\Pr(R = 1 | X = 1)$$

A: The probabilities are:

$$\Pr(R = 0 | X = 0) = 0.75$$

$$\Pr(R = 1 | X = 0) = 0.25$$

$$\Pr(R = 0 | X = 1) = 0.25$$

$$\Pr(R = 1 | X = 1) = 0.75$$

Randomized Response: computing ϵ

Here we write $\Pr(M(X) = R)$ as $\Pr(R|X)$ (it's the same)

Differential Privacy (local model)

A mechanism M is ϵ -DP if, for all possible input pairs (X, X') and all possible outputs R ,

$$\Pr(R|X) \leq \Pr(R|X') \cdot e^\epsilon \quad \Rightarrow \quad \frac{\Pr(R|X)}{\Pr(R|X')} \leq e^\epsilon$$

A: We have this:

$$\Pr(R = 0|X = 0) = 0.75$$

$$\Pr(R = 1|X = 0) = 0.25$$

$$\Pr(R = 0|X = 1) = 0.25$$

$$\Pr(R = 1|X = 1) = 0.75$$

For output $R = 0$:

$$\frac{\Pr(R = 0|X = 0)}{\Pr(R = 0|X = 1)} = 3$$

$$\frac{\Pr(R = 0|X = 1)}{\Pr(R = 0|X = 0)} = 1/3$$

Similar for $R = 1$

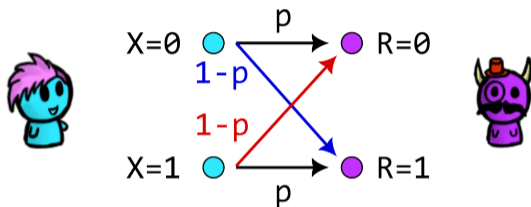
Going through all the values of $R, X, X' \in \{0, 1\}$, the largest ratio of probabilities is 3.

Therefore, the ratios are $\leq 3 = e^{\log 3}$.

Which means $\epsilon = \log 3 \approx 1.10$.

Randomized Response: doing some statistical analysis

We can draw randomized response as:



where $p = 0.75$.

Assume there are n users reporting values, and a fraction p_0 have $X = 0$, while a fraction $p_1 = 1 - p_0$ have $X = 1$.

Maybe you see the math more clearly this way:

$$\mathbb{E}\{R\} = \Pr(R = 1) = \Pr(R = 1|X = 0) \Pr(X = 0) + \Pr(R = 1|X = 1) \Pr(X = 1)$$

Now we are the analyst/adversary. We collect responses from all n users.

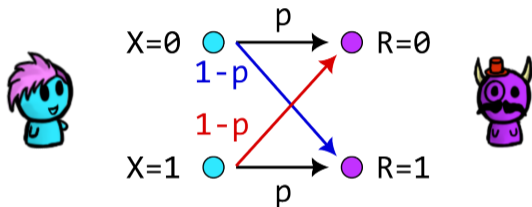
Q: Given p_0 and p_1 , what is the probability that a response is $R = 1$ (or $\mathbb{E}\{R\}$)?

A: From the users that had $X = 0$, a fraction $1 - p$ of them will report $R = 1$. From the users that had $X = 1$, a fraction p will report $R = 1$. Therefore,

$$\mathbb{E}\{R\} = p_0 \cdot (1 - p) + p_1 \cdot p$$

Randomized Response: doing some statistical analysis

We can draw randomized response as:



where $p = 0.75$.

Assume there are n users reporting values, and a fraction p_0 have $X = 0$, while a fraction $p_1 = 1 - p_0$ have $X = 1$.

$$\mathbf{A:} \mathbb{E}\{R\} = p_0 \cdot (1 - p) + p_1 \cdot p$$

We have collected n values of R , so we have an empirical estimate of $\mathbb{E}\{R\}$. Let \bar{R} be this empirical estimate (the average of all received R 's).

Q: How would you estimate the percentage of users that had $X = 1$?

$$\mathbf{A:} \bar{R} \approx p_0 \cdot (1 - p) + p_1 \cdot p$$

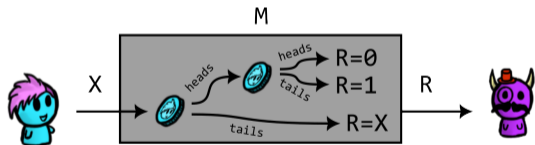
$$\Rightarrow \hat{p}_1 = \frac{\bar{R} - (1 - p)}{2 \cdot p - 1}$$

Let's do an example!

Disclaimer

You have $\epsilon = 1.1$ (high-ish privacy); no matter what you report in this exercise, you can always claim that was not your true answer!

Let's do an example of the statistical analysis. We want to learn how many of you cheated in an exam *before covid*.

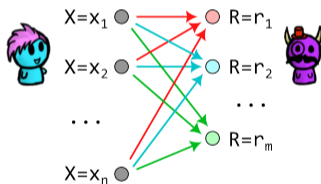


Q: $X = 1$ is "I have cheated". Flip two coins, get your response, and let's count the number of $R = 1$

p	
n	
$\#R = 0$	
$\#R = 1$	
\bar{R}	
$\hat{p}_1 = \frac{\bar{R} - (1 - p)}{2 \cdot p - 1}$	

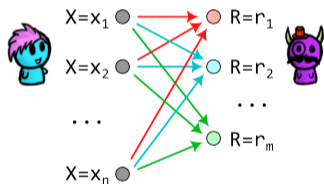
Computing ϵ for discrete mechanisms

Given a mechanism M with a discrete input space $\mathcal{X} = \{x_1, x_2, \dots, x_n\}$ and discrete output space $\mathcal{R} = \{r_1, r_2, \dots, r_m\}$, you can compute ϵ this way:



Computing ϵ for discrete mechanisms

Given a mechanism M with a discrete input space $\mathcal{X} = \{x_1, x_2, \dots, x_n\}$ and discrete output space $\mathcal{R} = \{r_1, r_2, \dots, r_m\}$, you can compute ϵ this way:



1 List inputs vs outputs

$X=x_1$

$X=x_2$

$X=x_n$

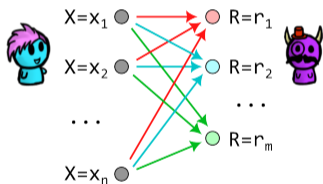
$R=r_1$

$R=r_2$

$R=r_m$

Computing ϵ for discrete mechanisms

Given a mechanism M with a discrete input space $\mathcal{X} = \{x_1, x_2, \dots, x_n\}$ and discrete output space $\mathcal{R} = \{r_1, r_2, \dots, r_m\}$, you can compute ϵ this way:

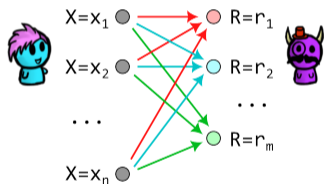


- 1 List inputs vs outputs
- 2 Compute the probabilities

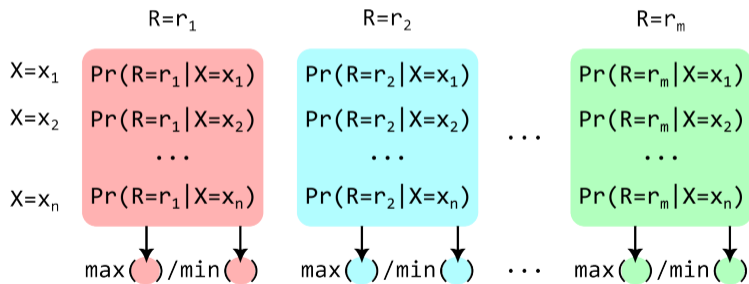
	$R=r_1$	$R=r_2$...	$R=r_m$
$X=x_1$	$\Pr(R=r_1 X=x_1)$	$\Pr(R=r_2 X=x_1)$		$\Pr(R=r_m X=x_1)$
$X=x_2$	$\Pr(R=r_1 X=x_2)$	$\Pr(R=r_2 X=x_2)$...	$\Pr(R=r_m X=x_2)$
...
$X=x_n$	$\Pr(R=r_1 X=x_n)$	$\Pr(R=r_2 X=x_n)$		$\Pr(R=r_m X=x_n)$

Computing ϵ for discrete mechanisms

Given a mechanism M with a discrete input space $\mathcal{X} = \{x_1, x_2, \dots, x_n\}$ and discrete output space $\mathcal{R} = \{r_1, r_2, \dots, r_m\}$, you can compute ϵ this way:

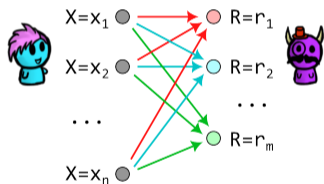


- 1 List inputs vs outputs
- 2 Compute the probabilities
- 3 Compute the max. ratio per output

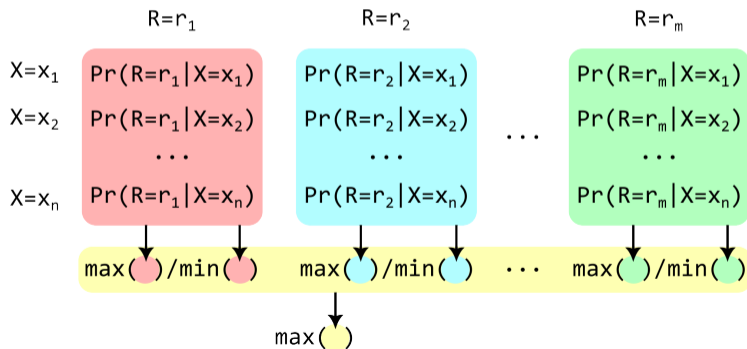


Computing ϵ for discrete mechanisms

Given a mechanism M with a discrete input space $\mathcal{X} = \{x_1, x_2, \dots, x_n\}$ and discrete output space $\mathcal{R} = \{r_1, r_2, \dots, r_m\}$, you can compute ϵ this way:

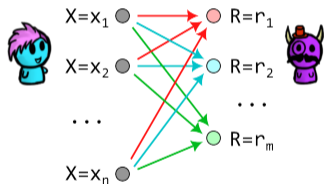


- 1 List inputs vs outputs
- 2 Compute the probabilities
- 3 Compute the max. ratio per output
- 4 Take the maximum of the ratios

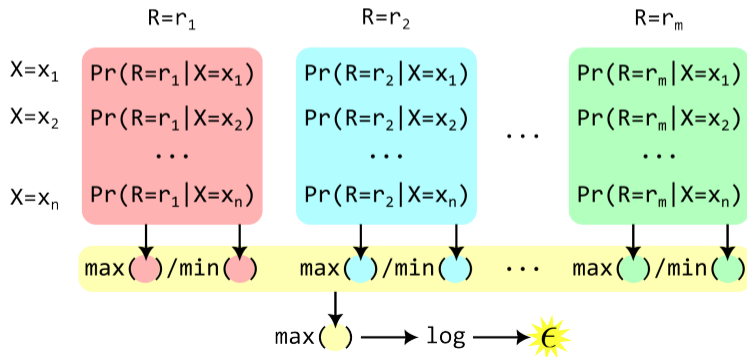


Computing ϵ for discrete mechanisms

Given a mechanism M with a discrete input space $\mathcal{X} = \{x_1, x_2, \dots, x_n\}$ and discrete output space $\mathcal{R} = \{r_1, r_2, \dots, r_m\}$, you can compute ϵ this way:



- 1 List inputs vs outputs
- 2 Compute the probabilities
- 3 Compute the max. ratio per output
- 4 Take the maximum of the ratios
- 5 ϵ is the (natural) log of that maximum.



Module outline

- 8 The Dinur-Nissim reconstruction attack
- 9 Introduction to Differential Privacy
- 10 Properties of Differential Privacy
- 11 Differentially Private Mechanisms
 - Laplace mechanism
 - Randomized Response
 - Discrete mechanisms
- 12 Recap and Practice

Recap: Differential Privacy

- Dinur-Nissim: shows why naive noise addition does not protect against database reconstruction attacks.
 - You need to know how to run the attack
- Differential privacy: ensures the adversary doesn't gain a lot of *new* information about a user.
 - Definition: ensures the output probability distribution does not change much when changing the input a bit.
 - DP interpretation as a game: shows that DP assumes a worst-case strong adversary, and gives insight into the privacy that different values of ϵ provide.
- Properties:
 - Robustness to post-processing. (!)
 - Group privacy.
 - Sequential composition. (!)
 - Parallel composition.
- DP mechanisms:
 - Laplacian (compute sensitivities Δ_f , know that $b = \Delta_f/\epsilon$ gives ϵ -DP)
 - Randomized response (how to compute ϵ and do basic statistical analyses)
 - Discrete mechanisms (how to compute ϵ)

Problem 1: computing b in the Laplace mechanism

Q: Find the (smallest) parameter b of the Laplace mechanism that you need to choose to provide ϵ -DP (assume the “bounded DP” setting) in the following settings. Explain your choices.

- 1 The dataset has n users, and the analyst wants to learn how many users have tested positive in a test for VIRUS X.
- 2 The dataset has n users, and the adversary wants to learn the *sum* of all the salaries. Salaries range from \$10 000 to \$100 000.
- 3 The dataset has n users, and the adversary wants to learn the *average* of all the salaries. Salaries range from \$10 000 to \$100 000.

Remember, for a query with sensitivity $\Delta_f = \max_{D, D'} |f(D) - f(D')|$, if we add Laplacian noise with $b = \Delta_f / \epsilon$, we provide ϵ -DP.

Problem 2: using the properties of DP

Q: We have a dataset with attributes Name and VIRUS. The attribute VIRUS is either Pos (the user tested positive for the virus) or Neg (negative) The dataset has n users (one row per user).

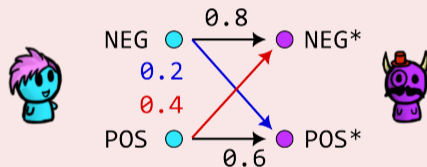
The data curator returns answers to the analyst by adding Laplacian noise with $b = 1$ to the true answer.

- 1 The data analyst asks for the total number of users that tested positive for the virus. What is the value of ϵ that the mechanism (adding Laplacian noise with $b = 1$) provides?
- 2 The data analyst asks for the total number of users that tested positive for the virus. Then, they ask *again* for the total number of users that positive for the virus. What is the ϵ after observing these two responses?
- 3 The data analyst asks for the total number of users that tested positive for the virus. Then, they ask for the total number of users that tested *negative* for the virus. What is the ϵ after observing these two responses?

Remember, for a query with sensitivity $\Delta_f = \max_{D, D'} |f(D) - f(D')|$, if we add Laplacian noise with $b = \Delta_f / \epsilon$, we provide ϵ -DP.

Problem 3: modified randomized response

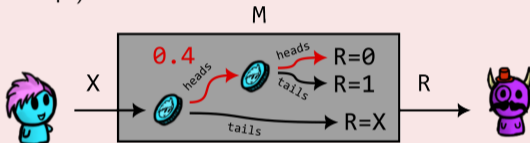
Q: To report their VIRUS X test results in a differentially private way, citizens of a small town (5 000 residents) use the following modified version of randomized response.



- Does this provide ϵ -DP? If so, compute the (smallest) ϵ it provides. If not, explain why.
- A total of 1 500 citizens have reported a POS* result. Give an estimate of the true number of citizens that have tested positive using the average-based estimator seen in the classroom.

Problem 4: randomized response with a biased coin

Q: Alice runs randomized response but her coin is biased, and has a slightly lower probability of heads ($\Pr(\text{heads}) = 0.4$, in both coin flips).



- 1 Does this provide ϵ -DP? If so, compute the (smallest) ϵ it provides. If not, explain why.
- 2 Does this provide more or less differential privacy than the version of randomized response with an unbiased coin? Explain why.
- 3 Alice reports her noisy value to the analyst. The analyst complains that the mechanism is too noisy, and requests Alice to re-run the mechanism again. Alice does so, providing the analyst with a second output. Is still Alice protected with the same level of ϵ -DP as in point 1?

Problem 5: 3-to-2 mechanism

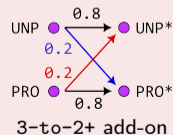
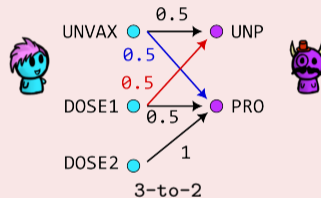
Q:

Trent has designed the following mechanism, that he calls 3-to-2, to report an individual's vaccine status in a privacy-preserving way:

- 1 Does this provide ϵ -DP? If so, compute the (smallest) ϵ it provides. If not, explain why.

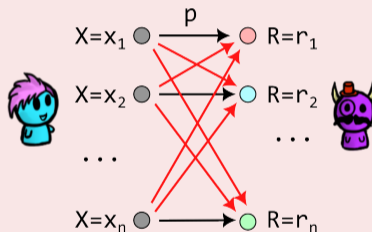
Alice complains that this mechanism is not private enough. She proposes to re-randomize the output, by performing the following randomized response *after* the 3-to-2 mechanism, and calls the mechanism 3-to-2+:

- 3 Without doing any math: can 3-to-2+ provide a worse DP level than 3-to-2?
- 4 Does 3-to-2+ provide ϵ -DP? If so, compute the (smallest) ϵ it provides. If not, explain why.



Problem 6: generalized randomized response

Q: The following mechanism is an extension of randomized response to a scenario where the input and output spaces are discrete and contain n elements. In this example, $\mathcal{X} = \mathcal{R}$ are points of interest in a town. When the user is in location x_i , they report location r_i with probability p , and otherwise report any other location in the map (with the same probability).



- 1 Consider the case where $n = 2$. What the ϵ -DP level that this mechanism provide (you can assume $p > 0.5$, and need to give a general expression of ϵ as a function of p).
- 2 Now consider that general case, where $n > 2$. You can assume that $p > 1/n$. Give a general expression of ϵ in this case (as a function of p and n).
- 3 Check that, when $n = 2$ and $p = 0.75$, we should get the ϵ seen in the classroom for the unbiased-coin randomized response.

Solutions

The following is a list of solutions for these problems.

If you have any questions about this, please ask on Piazza or come to office hours.

Problem 1: Solution

A: We just use $b = \Delta_f/\epsilon$.

- 1 The sensitivity of a count query is 1, so $b = 1/\epsilon$.
- 2 The sensitivity of the sum salary query is the maximum difference between salaries, so $b = 90\,000/\epsilon$.
- 3 The sensitivity of the average query (in the bounded DP notion) is $90\,000/n$, so $b = 90\,000/n\epsilon$.

Problem 2: Solution

A: We just use $b = \Delta_f/\epsilon \Rightarrow \epsilon = \Delta_f/b$.

- 1 The sensitivity of a count query is 1, and $b = 1$, so $\epsilon = 1$.
- 2 This is sequential composition: they are asking two queries that depend on the same data, so the new ϵ is the sum of the ϵ of both queries, i.e., 2.
- 3 This is again sequential composition: both queries depend on the whole dataset. ϵ is also 2.

Problem 3: Solution

A:

- 1 It provides DP. The highest ratio in an output is $0.6/0.2 = 3$, and therefore $\epsilon = \log 3$.
- 2 We use the formula:

$$\mathbb{E}\{\text{POS}^*\} = \Pr(\text{POS}) \cdot 0.6 + \Pr(\text{NEG}) \cdot 0.2$$

Now, since $\Pr(\text{NEG}) = 1 - \Pr(\text{POS})$, and using the approximation $\mathbb{E}\{\text{POS}^*\} \approx 1\,500/5\,000 = 0.3$, we can solve for $\Pr(\text{POS})$ and we get 0.25.

Problem 4: Solution

A:

- 1 It does provide DP. We first compute the probabilities $\Pr(R|X)$:

$$\Pr(0|0) = 0.76, \Pr(1|0) = 0.24, \quad \Pr(0|1) = 0.16, \Pr(1|1) = 0.84$$

Then,

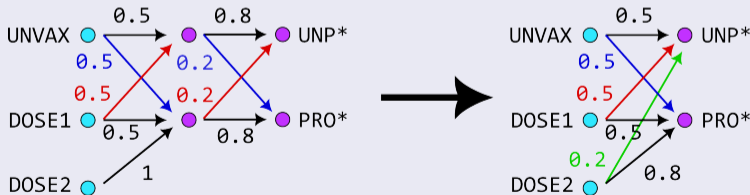
$$\epsilon_{bRR} = \log \left(\max \left\{ \frac{0.84}{0.24}, \frac{0.76}{0.16} \right\} \right) \approx 1.56.$$

- 2 For standard randomized response, we got $\epsilon_{RR} \approx 1.10$, so this biased-RR provides *less* privacy (higher ϵ).
- 3 No, Alice has answered two queries using an ϵ_{bRR} -DP mechanism, so reporting those answers provides $\epsilon = 2\epsilon_{bRR} \approx 3.12$ DP.

Problem 5: Solution

A:

- 1 The mechanism does not provide DP! Note that $\Pr(\text{UNP}|\text{DOSE2}) = 0$ but, for example, $\Pr(\text{UNP}|\text{UNVAX}) = 0.5$. When computing probability ratios for this output, we will get a $0.5/0$, so $\epsilon \rightarrow \infty$, which means it does not provide DP.
- 2 It cannot provide a worse DP level, because 3-to-2 is already non-DP ($\epsilon \rightarrow \infty$). However, even if it was ϵ -DP, adding an extra randomized function would not make it less private (robustness to post-processing)
- 3 We need to compute the new probabilities of this mechanism. Doing some basic math, we get that the 3-to-2+ can be written as follows:



This means that $\epsilon = \log(0.5/0.2) \approx 0.92$.

Problem 6: Solution

A:

- ① When $n = 2$, then we have that

$$\epsilon = \log \max \left(\frac{p}{1-p}, \frac{1-p}{p} \right)$$

Since $p > 0.5$, then $p > (1-p)$, so $\epsilon = \log(p/(1-p))$.

- ② $\Pr(R = r_i | X = x_i) = p$, and since all the red arrows have the same probability, $\Pr(R = r_i | X \neq x_i) = (1-p)/(n-1)$.

Since $p > 1/n$, then $\Pr(R = r_i | X = x_i) > \Pr(R = r_i | X \neq x_i)$, and we get

$$\epsilon = \log \left(\frac{\Pr(R = r_i | X = x_i)}{\Pr(R = r_i | X \neq x_i)} \right) = \log \left(\frac{p(n-1)}{(1-p)} \right).$$

- ③ When $n = 2$ and $p = 0.75$, we get $\epsilon = \log(0.75/0.25) = \log 3$, which is the result we saw with unbiased RR.