Some Reminders for a Seamless Online Class…

- Please turn on your video
- Mute yourself (press and hold spacebar when you’d like to talk)
- Don’t do anything you wouldn’t do in an in-person class
- I will occasionally check the chat for messages if you’d like to share there instead
- Please say your name before you speak

Announcements

- Please complete the course evaluation!
- Doris received a # of requests for an extension, so we're doing a blanket extension of 3 days for the project to 05/09
- Grading questions…
  - Bottomline: all of you have done really well!
  - I see no reason to curve if the absolute scores are as high as 95+
    - This means you're learning the content (at least from my perspective)
    - Of course, this depends on the project

Recap

- Data-savviness is the future!
- “Classical” relational databases
    - Notion of a DBMS
    - The relational data model and algebra: bags and sets
    - SQL Queries, Modifications, DDL
    - Database Design
    - Views, constraints, triggers, and indexes
    - Query processing & optimization
    - Transactions
- Non-classical data systems
    - Data preparation:
        - Semi-structured data and document stores
        - Unstructured data and search engines
    - Data Exploration:
        - Cell-structured data and spreadsheets
        - Dataframes and dataframe systems
        - OLAP, summarization, and visual analytics
    - Batch Analytics:
        - Compression and column stores
        - Parallel data processing and map-reduce
        - Streaming, sketching, approximation
    - Special Topics:
        - Graph processing systems
        - **Security and Privacy**

Today's Lecture

- Let's start by talking about database security
  - Access control
  - Authentication
  - SQL injection attacks

Access Control

- Relational databases support the ability to give users certain privileges to do certain types of activities on tables or columns within tables
- DBMS keeps track of which users can do what
- The privileges so granted can be revoked as well

- These privileges can be granted to specific *roles* instead of specific users
  - Roles can be, for example, sales_employee, data scientist, manager, …
  - Syntax for roles or individual users is similar

Access Control Syntax

- GRANT privileges ON object TO users [WITH GRANT OPTION]

- Object: table or view
- Privileges:
  - SELECT
    - The right to read all columns of the object
  - INSERT/UPDATE [(column name)]
    - The right to insert/update rows for the named column names
    - Can omit column name if right is for all columns
  - DELETE
    - The right to delete rows from the object
  - REFERENCES [(column name)]
    - The right to define foreign keys (in other tables) that refer to the specified column of object, or to all columns

Access Control Syntax

- GRANT privileges ON object TO users [WITH GRANT OPTION]

- Object: table or view
- Privileges: SELECT/ INSERT / UPDATE / DELETE / REFERENCES
- GRANT OPTION allows the user to pass the privileges onto other users
- Only the original creator of the object (table or view) has the option of doing CREATE, ALTER, or DROP on the object
  - The creator of a view has automatic SELECT privileges on the view: this is because they had to have SELECT privileges on the underlying tables/views to be able to even define the view
  - And so they have grant option only if they had grant option on the underlying tables/views that the new view was defined
  - Similarly, if the view is updatable, and the user holds INSERT, DELETE, UPDATE on the underlying table the user similarly has same privileges on the view

Let's take an example…

- Sailors (sid, sname, rating, age)
- Boats (bid, bname, color)
- Reserves (sid, bid, day)

- CREATE VIEW ActiveSailors (name, age, day) AS SELECT S.sname, S.age, R.day FROM Sailors AS S, Reserves AS R WHERE S.sid = R.sid AND S.Rating > 6

- A user who can access ActiveSailors but not Sailors or Reserves knows the names of sailors who have reservations, but not the bids of boats reserved

Let's take an example…

- Sailors (sid, sname, rating, age)
- Boats (bid, bname, color)
- Reserves (sid, bid, day)

- Say Tarique created Boats, Sailors, Reserves
- Examples of GRANT commands issued by Tarique:
  - GRANT INSERT, DELETE ON Reserves TO Janice WITH GRANT OPTION
  - GRANT SELECT ON Reserves TO Amy
  - GRANT SELECT ON Sailors TO Amy WITH GRANT OPTION
  - GRANT UPDATE (rating) ON Sailors TO Carlos
  - GRANT REFERENCES (bid) ON Boats TO Bob
- Amy tries to declare the view ActiveSailors via the command:
  - CREATE VIEW ActiveSailors (name, age, day) AS SELECT S.sname, S.age, R.day FROM Sailors AS S, Reserves AS R WHERE S.sid = R.sid AND S.Rating > 6
  - Q: Can Amy do this?
  - Yes. She has the SELECT privileges on underlying relations Sailors and Reserves
  - Q: Can she now give SELECT privileges on ActiveSailors to Bob via:
    - GRANT SELECT ON ActiveSailors TO Bob
  - No. She doesn't have GRANT OPTION on Reserves, and therefore not on ActiveSailors

Let's take an example…

- Sailors (sid, sname, rating, age)
- Boats (bid, bname, color)
- Reserves (sid, bid, day)

- Say Tarique created Boats, Sailors, Reserves
- Examples of GRANT commands issued by Tarique:
  - GRANT INSERT, DELETE ON Reserves TO Janice WITH GRANT OPTION
  - GRANT SELECT ON Reserves TO Amy
  - GRANT SELECT ON Sailors TO Amy WITH GRANT OPTION
  - GRANT UPDATE (rating) ON Sailors TO Carlos
  - GRANT REFERENCES (bid) ON Boats TO Bob
- Amy declares the view ActiveSailors via the command:
  - CREATE VIEW ActiveSailors (name, age, day) AS SELECT S.sname, S.age, R.day FROM Sailors AS S, Reserves AS R WHERE S.sid = R.sid AND S.Rating > 6
- Next Amy declares the view YoungSailors via:
  - CREATE VIEW YoungSailors (sid, age, rating) AS SELECT * FROM Sailors WHERE age<18
- She can then give privileges on the view to others:
  - GRANT SELECT ON YoungSailors TO Ben, Martha
  - Ben and Martha can execute queries on YoungSailors but not on Sailors directly
- Carlos can run the following command:
  - UPDATE Sailors SET rating = 8
  - But cannot run UPDATE Sailors SET rating = rating - 1, since this involves reading it

Revoking Privileges

- Syntax:
  - REVOKE [GRANT OPTION FOR] privileges ON object FROM users {RESTRICT | CASCADE}
  - CASCADE:
    - Withdraw the privileges not just from the specified users, but also all other users who hold these privileges thanks *solely* to the specified users
    - So those users would have go get their privileges "another way"
    - RESTRICT only does so for the specified users

Example of Revocations

- Focusing on Sailors
- GRANT SELECT ON Sailors TO Amy WITH GRANT OPTION (Tarique)
- GRANT SELECT ON Sailors TO Bin WITH GRANT OPTION (Amy)
- REVOKE SELECT ON Sailors FROM Amy CASCADE (Tarique)

- Q: What will happen?
- Both Amy and Bin will lose their privileges on Sailors

Example of Revocations

- Focusing on Sailors; new sequence
- GRANT SELECT ON Sailors TO Amy WITH GRANT OPTION (Tarique)
- GRANT SELECT ON Sailors TO Bin WITH GRANT OPTION (Tarique)
- GRANT SELECT ON Sailors TO Bin WITH GRANT OPTION (Amy)
- REVOKE SELECT ON Sailors FROM Amy CASCADE (Tarique)

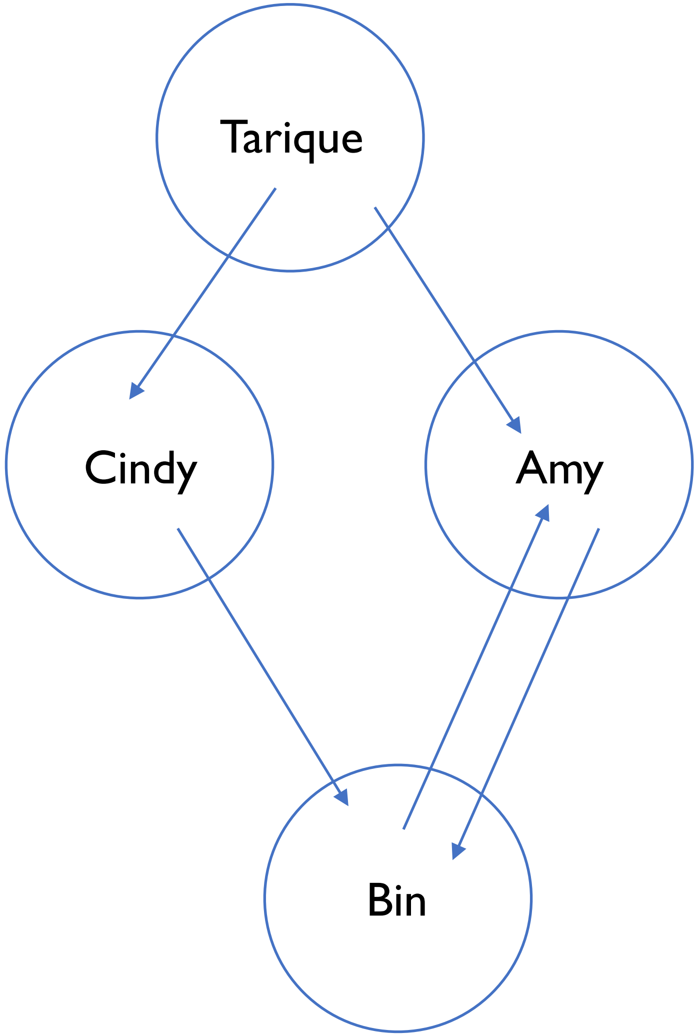
- Q: What will happen?
- Only Amy will lose her privileges

Even more complicated example

- Focusing on Sailors; new sequence
- GRANT SELECT ON Sailors TO Amy WITH GRANT OPTION (Tarique)
- GRANT SELECT ON Sailors TO Bin WITH GRANT OPTION (Amy)
- GRANT SELECT ON Sailors TO Amy WITH GRANT OPTION (Bin)
- GRANT SELECT ON Sailors TO Cindy WITH GRANT OPTION (Tarique)
- GRANT SELECT ON Sailors TO Bin WITH GRANT OPTION (Cindy)
- REVOKE SELECT ON Sailors FROM Amy CASCADE (Tarique)
- Q: What will happen?
- No real changes: everyone continues to hold the same privileges
- Q: What will happen if Tarique removes the privileges from Cindy as well?
- Everyone loses privileges

Even more complicated example

- Focusing on Sailors; new sequence
- GRANT SELECT ON Sailors TO Amy WITH GRANT OPTION (Tarique)
- GRANT SELECT ON Sailors TO Bin WITH GRANT OPTION (Amy)
- GRANT SELECT ON Sailors TO Amy WITH GRANT OPTION (Bin)
- GRANT SELECT ON Sailors TO Cindy WITH GRANT OPTION (Tarique)
- GRANT SELECT ON Sailors TO Bin WITH GRANT OPTION (Cindy)
- REVOKE SELECT ON Sailors FROM Amy CASCADE (Tarique)
- Q: What will happen?
- No real changes: everyone continues to hold the same privileges
- Q: What will happen if Tarique removes the privileges from Cindy as well?
- Everyone loses privileges



OK, so now what

- We can handle access control via granting and revoking privileges
- Amy may be accessing the database via an internet application. How do we ensure that Amy is not deceived by a scammy website?
- Likewise, how do we ensure that Amy is actually Amy and not Bin?

- Enter authentication. A key ingredient of authentication is encryption.
- We'll cover encryption very briefly…

Encryption/Decryption

- Encryption takes a message and an encryption key , and encrypts the message:
  - encrypt: (message, key) —> encrypted_message
- Decryption takes the encrypted message and a decryption key, and decrypts the message:
  - decrypt: (encrypted_message, key) —> message
- Two types of encryption/decryption:
  - Symmetric: encryption and decryption keys are the same and hidden
    - These schemes are often cheaper
    - AES (Advanced Encryption Standard), DES (Data E. S.) are examples
  - Asymmetric: the keys are different
    - Popular example: public-key encryption
    - These schemes are often more expensive

Public-Key Encryption: Key Ideas

- Each user holds two types of keys:
  - A private key and a public key each: k1 and k2
  - You can imagine these keys to be "inverses" of each other
- So how does this work: if Alice wants to send a message m to Bob,
  - then she can simply encrypt the message with Bob's public key k1, knowing that only Bob has the private key k2
  - Bob will receive the packet and then invert it using their private key to get back the message
- How can Bob be sure the message is from Alice?
  - Alice herself has a public key k1' and a private key k2'
  - So Alice can lock her outgoing message with both her private key k2' and Bob's public key k1
  - Bob simply needs to unlock using his private key k2 and Alice's public key k1'

Usual Procedure

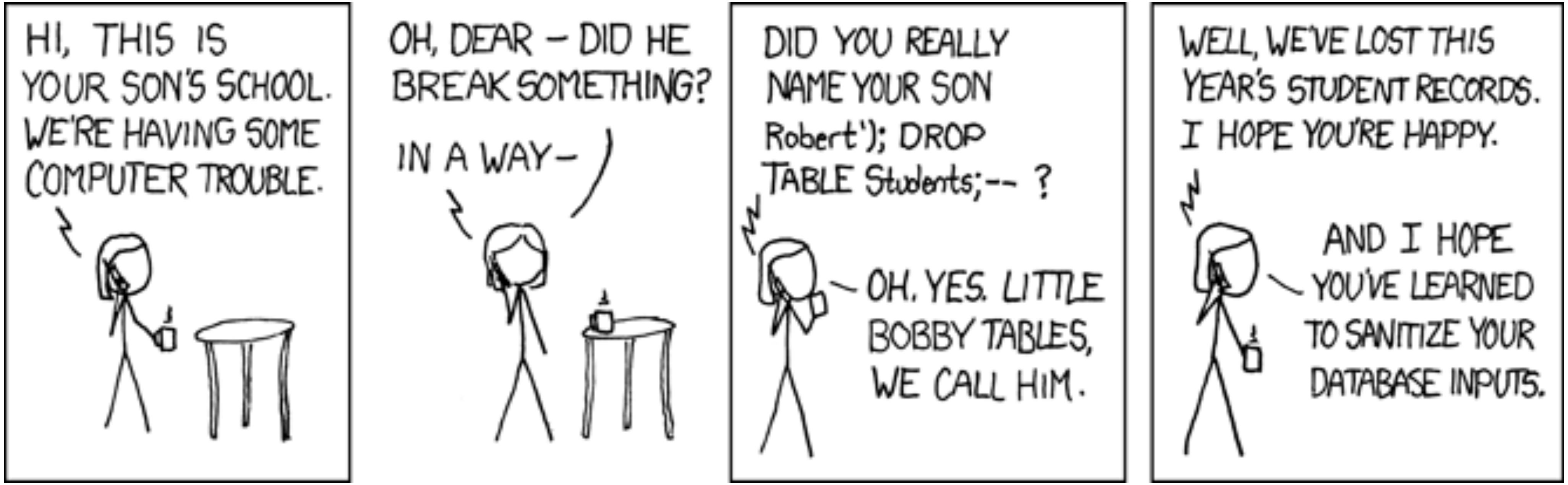
- Use asymmetric public-key encryption to exchange a secure shared key
    - Expensive but more secure
- Then apply symmetric encryption with the shared key
    - Less secure but you know that via public-key encryption only the two parties have the shared key

- Lots more details here! Number theory is your friend

SQL Injection: Very Brief Primer

- Even with authentication and access control, sometimes you only want certain queries to be run on certain subsets of dat
  - E.g., a student is allowed to only view their data but not anyone else's
  - This is hard to enforce with the access control policies defined; and would not be prevented via authentication.
- One way to constrain the space of queries is to only allow queries to be issued via forms on webpages.
  - These forms will accept arguments as free text fields or dropdowns
- For example, a program may accept a string $A from a user form, and use it as an argument to a SQL query issued to a database
  - SQL Query:
    - "SELECT balance FROM Accounts WHERE Customer =" + $A
  - But if we're not careful and we let the user enter any value for A, they can do "evil"
  - For example, if they set A = "Alice; SELECT * FROM Accounts;" they can learn about all account IDs
- Simple approach — sanitize inputs.
  - For example, don't allow ";" in your input fields. Or first check if there are any special keywords "SELECT", "FROM" in the input fields.

SQL Injection



Today's Lecture

- Let's start by talking about database security
  - Access control
  - Authentication
  - SQL injection attacks
- Next: database privacy

Some content drawn from Ashwin M's Privacy Class, Duke U.

Data Privacy: A Brief Primer

- Decisions are being made using data
    - Both via aggregate statistics
    - Or via models that build on the aggregate statistics
- However, the privacy of individuals is often not respected in such decision making

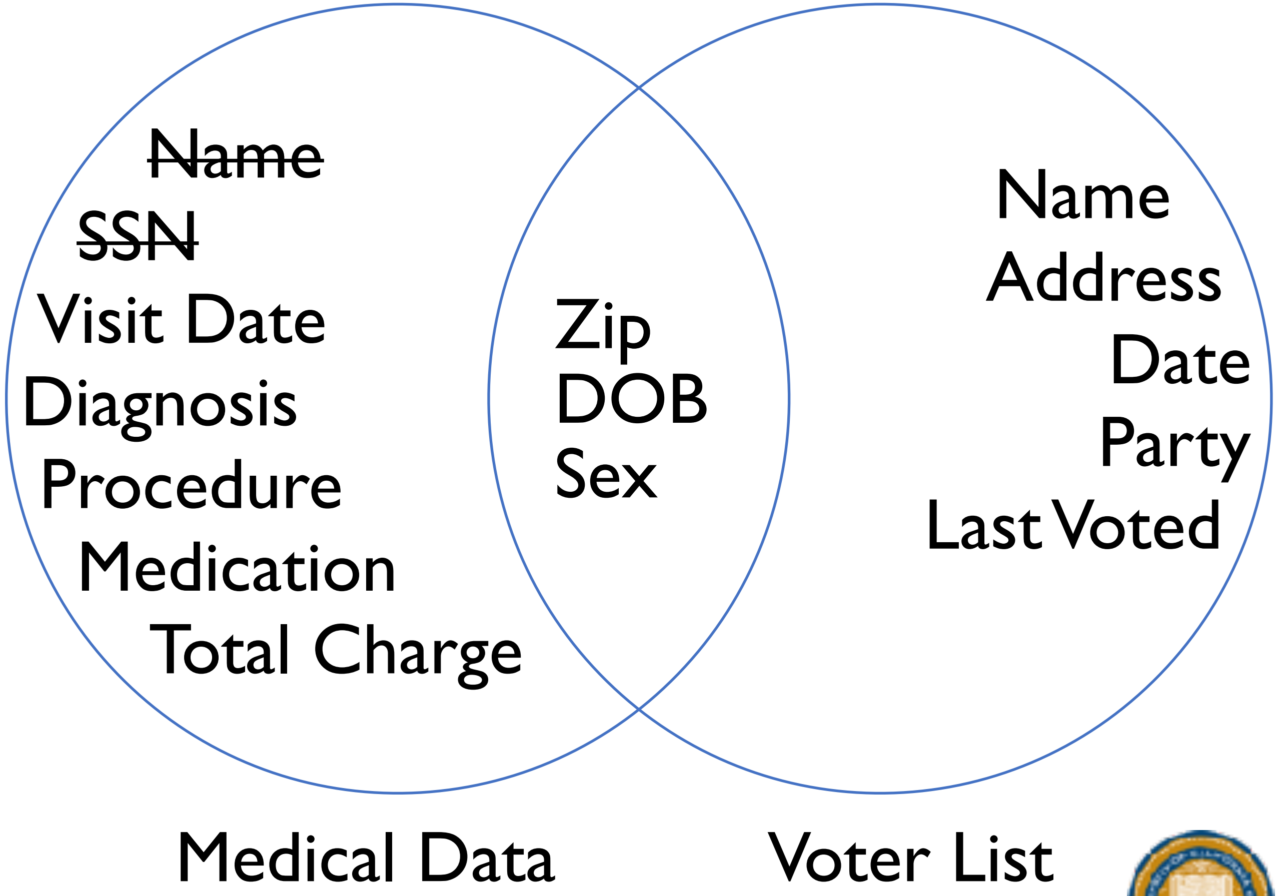
- Example: say I am building a contact tracing app for COVID-19
    - Say we "ask" everyone to install an app that tracks everything that the person does in terms of where they go and what they do
    - I keep all of this data in my database
    - Then, for every person who tests positive, I decide to publish their names and their entire list of locations
    - OK — this is bad: not everyone may want to know that they have COVID-19.
    - So, instead of publishing their names, I anonymize their names
    - Q: Why does not suffice?

A lot of different types of data are very sensitive

- Census surveys
- IRS Records
- Medical records
- Insurance records
- Search logs
- Shopping histories
- Photos
- Videos
- Smart phone Sensors
- Mobility trajectories

History of Data Privacy: Sweeney 2002

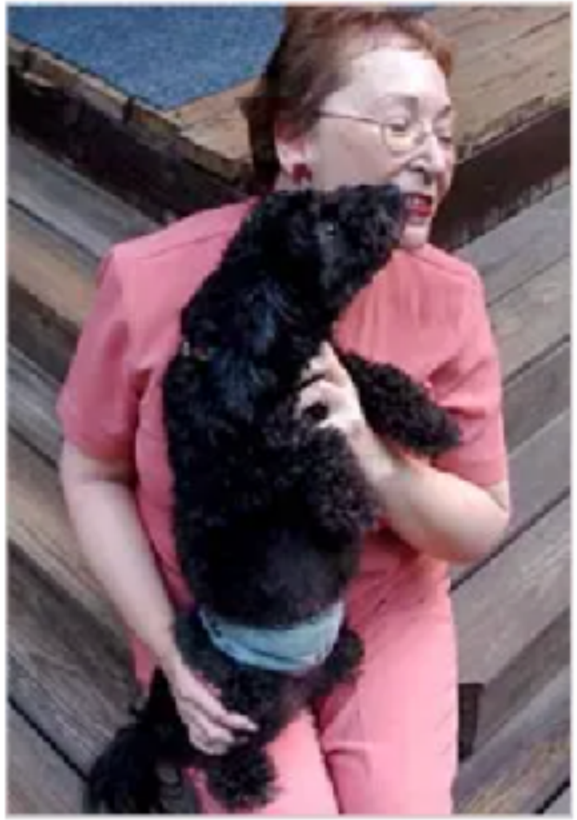
- Sweeney [2002]
- Governor of MA's medical record uniquely identified by the Zipcode, DOB, and Sex.
- Name then linked to diagnoses
- The triple ends up being a quasi-identifier



History of Data Privacy: AOL Logs 2006

- AOL published a subset of their search logs in 2006
  - Anonymized to remove the user identifiable info
- Even without them, search queries can serve as a pretty good quasi-identifier for individuals
  - Good representation for one's interests
  - Easy if you do "vanity searches"

A Face Is Exposed for AOL Searcher



Thelma Arnold's identity was betrayed by AOL records of her Web searches, like ones for her dog, Dudley, who clearly has a problem.
Erik S. Lesser for The New York Times

Since then…

- Researchers have reverse-engineered private data via even more sophisticated mechanisms
  - Ranging from ML algorithms doing microtargeting of ads
  - … to identifying individuals in a genome mixture
    - e.g., did Alice participate in the study?

Privacy Violations Using Microtargeted Ads: A Case Study

PDF

Aleksandra Korolova
Department of Computer Science, Stanford University

Resolving Individuals Contributing Trace Amounts of DNA to Highly Complex Mixtures Using High-Density SNP Genotyping Microarrays

Nils Homer, Szabolcs Szelinger, Margot Redman, David Duggan, Waibhav Tembe, Jill Muehling, John V. Pearson, Dietrich A. Stephan, Stanley F. Nelson, David W. Craig

Published: August 29, 2008 • https://doi.org/10.1371/journal.pgen.1000167

Privacy-Preserving Data Publishing

- We want to publish a dataset D containing information about individuals, by transforming it to D', where:
  - The individuals' information is protected
  - The dataset D' is still useful for analysis

- A few solutions
  - K-Anonymity
  - L-diversity
  - Differential Privacy

K-anonymity

- Take the identifiable information and “generalize” it to ensure that there is at least k records that potentially match each individual’s record

	Non-Sensitive			Sensitive
	Zip Code	Age	Nationality	Condition
1	13053	28	Russian	Heart Disease
2	13068	29	American	Heart Disease
3	13068	21	Japanese	Viral Infection
4	13053	23	American	Viral Infection
5	14853	50	Indian	Cancer
6	14853	55	Russian	Heart Disease
7	14850	47	American	Viral Infection
8	14850	49	American	Viral Infection
9	13053	31	American	Cancer
10	13053	37	Indian	Cancer
11	13068	36	Japanese	Cancer
12	13068	35	American	Cancer

	Non-Sensitive			Sensitive
	Zip Code	Age	Nationality	Condition
1	130**	< 30	*	Heart Disease
2	130**	< 30	*	Heart Disease
3	130**	< 30	*	Viral Infection
4	130**	< 30	*	Viral Infection
5	1485*	≥ 40	*	Cancer
6	1485*	≥ 40	*	Heart Disease
7	1485*	≥ 40	*	Viral Infection
8	1485*	≥ 40	*	Viral Infection
9	130**	3*	*	Cancer
10	130**	3*	*	Cancer
11	130**	3*	*	Cancer
12	130**	3*	*	Cancer

K-anonymity

	Non-Sensitive			Sensitive
	Zip Code	Age	Nationality	Condition
1	13053	28	Russian	Heart Disease
2	13068	29	American	Heart Disease
3	13068	21	Japanese	Viral Infection
4	13053	23	American	Viral Infection
5	14853	50	Indian	Cancer
6	14853	55	Russian	Heart Disease
7	14850	47	American	Viral Infection
8	14850	49	American	Viral Infection
9	13053	31	American	Cancer
10	13053	37	Indian	Cancer
11	13068	36	Japanese	Cancer
12	13068	35	American	Cancer

	Non-Sensitive			Sensitive
	Zip Code	Age	Nationality	Condition
1	130**	< 30	*	Heart Disease
2	130**	< 30	*	Heart Disease
3	130**	< 30	*	Viral Infection
4	130**	< 30	*	Viral Infection
5	1485*	≥ 40	*	Cancer
6	1485*	≥ 40	*	Heart Disease
7	1485*	≥ 40	*	Viral Infection
8	1485*	≥ 40	*	Viral Infection
9	130**	3*	*	Cancer
10	130**	3*	*	Cancer
11	130**	3*	*	Cancer
12	130**	3*	*	Cancer

- Downsides: If we know, for example, that the person in question — a neighbor — is older than 30 and lives in a 13053 zipcode, then we know they have cancer
- So generalization only works so well
  - "Hiding" in a group of k doesn't work unless there is diversity in the sensitive values
- One approach to fix this: l-diversity
  - Ensures that each group of k also has diversity in the sensitive values
  - Still not sufficient

Differential Privacy

- Stronger notion of privacy
- High level idea: the presence or absence of any given individual's record should not affect the outcome of the perturbed D'
  - So noise is injected an appropriate amount
- The amount is controlled by a knob
  - Knob governs how much a single data point can impact the probability of any outcome
  - If the knob is set to 2, for example, it says that no outcome is more than twice as likely with the individuals data included, than if it is not included
- Complicated math! Gödel prize!

Takeaways

- Security and Privacy are both hugely important!
- Data System Security is supported via access control, authentication (enforced via encryption) and avoiding SQL injection attacks
- Data Privacy is something to be worried about when publishing artifacts: models, data, …
  - Need to ensure that no individual is identifiable, especially in conjunction with other external information
  - K-anonymity as a simple notion with flaws: differential privacy is a stronger notion but harder to follow
- Still very much an area in flux!