THE UNIVERSITY OF MELBOURNE

# COMP20008 Elements of Data Processing

## Semester 2 2018

## Lecture 14: Data Linkage and Privacy



## Announcements




## Outline

- **Last lecture**
  - How to define similarity between records?
  - How to efficiently do linkage when matching two large databases
  - Blocking
- **Today: How to maintain privacy when doing data linkage?**
  - Why is privacy important?
  - An example method for privacy preserving linkage



## Data Linkage and Privacy

- If data matching is being conducted within a single organisation and is using databases within the organisation, privacy/confidentiality is generally not a concern.
  - Can assume individuals doing the matching are authorised, aware of policies and don't have malicious intent
  - E.g. University of Melbourne: administrator who is matching student academic results database against database of applicants for PhD study
- On the other hand, problems can arise if
  - Matched data is being passed to another organisation or being made public
  - Data matching is being conducted across databases from different organisations

## Example 1: Need for privacy in public health

- Research team investigating effects of car accidents on the public health system. Research questions
  - Most common injuries for what types of car accident?
  - When and where accidents occurred, the road and weather conditions at time of accident and health of people involved in accident, as well as two years later?
- Data needed
  - Hospital data on patients
  - Private health insurance data
  - Police
  - Road traffic authorities
- *These organisations can't share all their data with the research team.*

## Example 2: Need for privacy - business

- Two businesses wish to co-operate
  - Find how many customers and suppliers in common
  - *Don't want to share all their confidential data with another*
  - Need techniques for sharing such that
    - Only records in the two databases that are similar with each other (according to some similarity function) are identified.
    - The identities of these records and their similarities are revealed to both organisations
    - Neither of the two parties must be able to learn anything else about the other party's confidential data (the non similar records)
    - E.g. co-branded credit card, company + bank

## Example 3: Need for privacy – national security

- National crime investigation unit analysing crimes of national significance (significance to all of Australia)
- Wants to link its own database about suspicious individuals to different databases around Australia
  - Tax
  - Law enforcement
  - Financial institutions
- *Only linked records should be available to the unit*
  - It should not get access from the bank to financial data about non-suspicious individuals
  - It should not get access to tax records about non-suspicious individuals

## Example 4 – Facial recognition

- https://www.theguardian.com/technology/2018/may/01/victoria-threatens-to-pull-out-of-facial-recognition-scheme-citing-fears-of-dutton-power-grab

### Victoria threatens to pull out of facial recognition scheme citing fears of Dutton power grab

**Identity matching bill provides 'significant scope' for minister to expand powers, state warns**

- Facial matching using driver's licence, passport, visa, citizenship , (proof-of-age, firearms, marine licences)
- Concerns about access by private-sector entities (financial institutions) to facial verification services using this data
  - *Is it reasonable for them to have access?*

## Privacy Preserving Data Linkage: Problem Statement

- How can we perform data linkage for two databases, each from a different organisation
  - Without revealing any information about individuals who do not get linked across the databases (i.e. individuals who occur in one database and not in the other)
- We will need
  - Methods for computing similarity of records, without revealing the record values
    - Hashing: an important tool

## Number representation – in brief

- We can represent numbers using different bases
  - Decimal (base 10), binary (base 2), octal (base 8), hexadecimal (base 16), ..
- Example: The number 5332 (decimal: base 10) can be written as:
  - 1010011010100 (binary: base 2)
  - 12324 (octal: base 8)
  - 14d4 (hexadecimal: base 16)
- We're not concerned with the details of how the conversions are done, just that it's possible to do it

## Hashing

- A hash function H maps a data item of arbitrary size to a data item of fixed size
- Example 1
  - H(James) = 10
  - H(Kate) = 11
  - H(The quick brown fox jumped over the lazy dog) = 20
  - [take first letter of the string, 'J', 'K' or 'T']
- Example 2
  - H(32)=2
  - H(20)=2
  - H(6)=0
  - H(7)=1
  - [remainder when dividing by 3]

## Non invertible (one way) hash function

- Non invertible hash function. Given the output H(X), extremely hard to reconstruct X. Examples
  - MD5 hash function (produces a 32 digit hexadecimal number, equal to a string of 128 bits)
    - H(James)= d52e32f3a96a64786814ae9b5279fbe5
    - H(I love data wrangling)= 614416fa9d994aa8225ebd7c50f22060
    - H(12345678)= 25d55ad283aa400af464c76d713c07ad
  - SHA-3-512 hash function (produces a 128 digit hexadecimal number, equal to a string of 512 bits)
    - H(James)=02c56351888fa73ff825ffd65526b264ebefe7916fa5d8d5c58e766bfdd1de8e85b68bf12599b9d21eca6683d4abfa8616acfa6834e7c478e394374a7b015898
    - H(12345678)=8a56bac869374c669443a1626ff0967af258123f83faf6b55e31dd541e6bbd90308a3385713294bf2e8861bc8cf8f8feda41f9c4db19d5811a6b5de85eac9870

## Hash function Demo

- http://emn178.github.io/online-tools/sha3_512.html

## Hash encoding for exact matching: 2 party protocol

- Each organisation
  - Applies a (one way) hash function to the attribute used to join the databases
  - Shares its hashed values with the other organisation. Each checks which ones match. These are the linked records.

Org. A

| Name | H(Name) |
|---|---|
| Jill | 8347 |
| Jane | 6992 |

Org. B

| Name | H(Name) |
|---|---|
| Bob | 2332 |
| Jane | 6992 |

## Small changes in input, large change in output

- Disadvantage 1: What about single character differences in the original value? E.g. MD5 hash function
  - H(James)= d52e32f3a96a64786814ae9b5279fbe5
  - H(Jamex)= c3bfa7fa6ad2b987619bb4c932e65b4a
  - Single character difference results in a completely different output. This is generally true for one way hash functions such as MD5, SHA ....

## Dictionary attack

- Disadvantage 2: An organisation could mount a dictionary attack to "invert" the hash function. E.g. Organisation A generates a hash dictionary by computing hashes for all words of length 4
  - H(aaaa)=…
  - H(aaab)= …
  - H(aaac)= …
  - H(aaad)= …
  - …..
  - H(zzzz)= …
- Organisation A then scans the hashed values received from Organisation B. Checks if any match its hash dictionary. If yes, privacy is lost for those items.
- Could also generate dictionary for all known words, pairs of words, …. [up to some limit of feasibility]
- d077f244def8a70e5ea758bd8352fcd8 example

## Hash encoding for exact matching: 3 party protocol

- Possible solution
  - Involve a trusted 3rd party (Organisation C)
  - Organisations A and B send their hashed values to Organisation C, who then checks for matches.
  - What if Organisation C is malicious?
    - Organisation C could mount a dictionary attack and guess the hashed values
    - Solution: A and B perform "dictionary attack resistant" hashing

## 3rd Party Protocol using salt

A

| Name | H(Name) |
|---|---|
| Jill+SECRET_KEY | 1112 |
| Jane+SECRET Key | 9341 |

B

| Name | H(Name) |
|---|---|
| Bob+SECRET_KEY | 2996 |
| Jane+SECRET_KEY | 9341 |

- Organisations A and B concatenate a secret word to every name field in their data before hashing (known as a *salt*). Organisation C does not know what this word is and thus can't perform a dictionary attack to "reverse" the hashed values it receives.

## Hashing and salting

- In June 2012 dating site eHarmony was hacked
  - 1.5 million password hashes publicly released
- In June 2012 social networking site LinkedIn was hacked
  - 6.5 million hashed password stolen and publicly released

- *Neither company used a salt when hashing the passwords*
  - Many passwords were thus susceptible to a brute force dictionary attack on the hashed values

## Question

- The two party protocol isn't robust to a dictionary attack.
  - Why doesn't adding salt to the hash function help here?

## Question 2017 exam

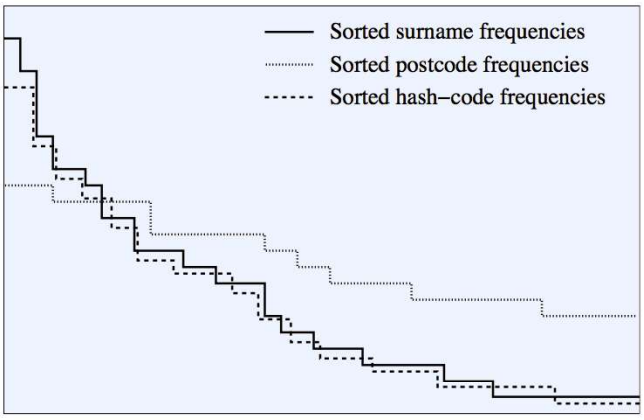
- Suppose we are using a 3 party exact matching scheme for privacy preserving data linkage. Alice and Bob each have their own dataset and they are trying to link the datasets together on people's names. They agree on a pre-processing strategy to standardise their data, which converts all upper case letters to lower case letters and converts all nick names to full names (e.g. "jim" to "james", "bob" to "robert", "liz" to "elizabeth").

Explain a possible disadvantage of their data pre-processing strategy with respect to the effectiveness of the privacy preserving data linkage.

## Frequency attack

- This third party scheme prevents a dictionary attack, but may still be susceptible to a frequency attack.
  - 3rd party compares the distribution of hashed values to some known distribution
    - E.g. distribution of surname frequencies in a public database versus distribution of hash values
    - May be able to guess some of the hashed values!
- Organisations A and B could prevent this by adding random records to manipulate the frequency distribution

## Frequency attack [slide from Peter Christen]




## Summary/ Privacy preserving linkage

- Organisations A and B can determine which records in the two databases are an exact match in a privacy preserving manner by
  - using a trusted third party C, and
  - using one way hash functions with a salt, and
  - adding random records
- A reasonably private scheme (depending on how much the third party is trusted)

## Challenge 1

- The hash based technique using the 3rd party, can only compute exact similarity between strings in a privacy preserving manner.
- What if we wish to compute approximate similarity between two strings in a privacy preserving manner?
  - To be covered in the next lecture

## Challenge 2: Public release

- Suppose organisation wishes to make one of its internal datasets public, for social good purposes
  - E.g. NASA releasing images of Mars
  - City of San Francisco, crime data
  - CERN, particle physics data
  - Bank, data on credit scoring and people who experiences financial distress
- Can be very, very difficult to prevent data linkage attacks or reverse engineering of people's identities
  - America Online search logs
  - Medicare Benefits Schedule data (https://pursuit.unimelb.edu.au/articles/understanding-the-maths-is-crucial-for-protecting-privacy)
  - https://arxiv.org/pdf/1712.05627.pdf

## America Online Search Logs

- In 2006, America Online released a file with 3 months of ``anonymized'' search queries of 658k users.
  - After a public outcry, data quickly taken down, but couldn't be removed completely from the Web
  - Ranked 58 out of the 101 dumbest moments in business by CNNMoney.com
  - http://www.nytimes.com/2006/08/09/technology/09aol.html?_r=0

| User id | Time | Search Query |
|---|---|---|
| 1 | .. | .. |
| 1 | … | … |
| 1 | … | … |
| 2 | … | . |
| 2 | … | … |
| 2 | .. | … |
| 3 | … | … |

## Public release: Solutions

- Don't release the data at all!, or
- Release an obfuscated version of the data (e.g. with noise added to all the records)
  - This is the basis of methods such as k-anonymity and differential privacy (we will likely look at in a couple of weeks)

## Summary and plan for coming lectures

- Summary
  - 2 party protocol
    - Exact matching
    - Dictionary attack
  - 3 party protocol
    - Dictionary attack
      - Dictionary attack resistant: adding salt
    - Frequency attack
  - Challenges:
    - Approximate matching
    - Public release

- Next lecture:
  - Wednesday: Privacy preserving data linkage, allowing approximate matching (rather than just exact matching)

## Acknowledgements


- Material in this lecture partly adapted from
  - Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution and Duplicate Detection, Peter Christen, Springer, 2012. Available as an e-book for download by University Library
    - Read Sections 1,2, 4.1,4.2,5.4, 8.1, 8.2


## What you need to know from this lecture

- be able to explain in what circumstances privacy is an important issue for data linkage -understand the objective of privacy preserving data linkage
- understand the use of one way hashing for exact matching in a 2 party privacy preserving data linkage protocol
- understand the vulnerabilities of 2 party privacy preserving data linkage protocol to i) small differences in input, ii) dictionary attack
- understand the operation of the 3 party protocol for privacy preserving linkage, using hash encoding with salt for exact matching. Understand the disadvantages of this protocol