



COMP20008 Elements of Data Processing

Data linkage and privacy

continued



- Phase 2B marks will be available 11 am Monday May 8th
- Detailed schedule for phase 4 oral presentations has been released
- A consultation session this week



- A privacy preserving method for data linkage using approximate matching
 - Extension of method discussed last Monday, to allow approximate matching in a private manner



- 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 trust)

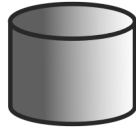


- The hash based technique using the 3rd party, can only compute exact similarity between strings in a privacy preserving manner.
- What if we wish to compute approximate similarity between two strings in a privacy preserving manner?

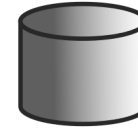


MELBOURNE

A



B



C

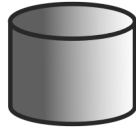


Trusted Party

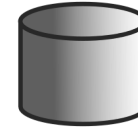


MELBOURNE

A

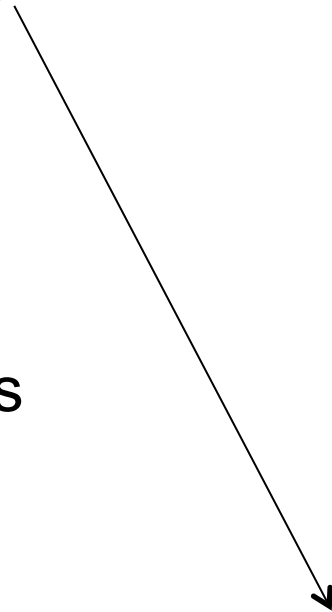


B



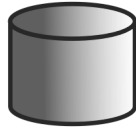
Private
Identifiers

C



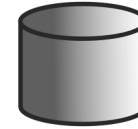
Intended Process

A



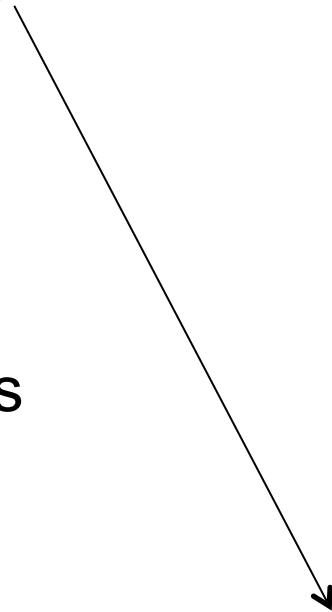
Private
Identifiers

B

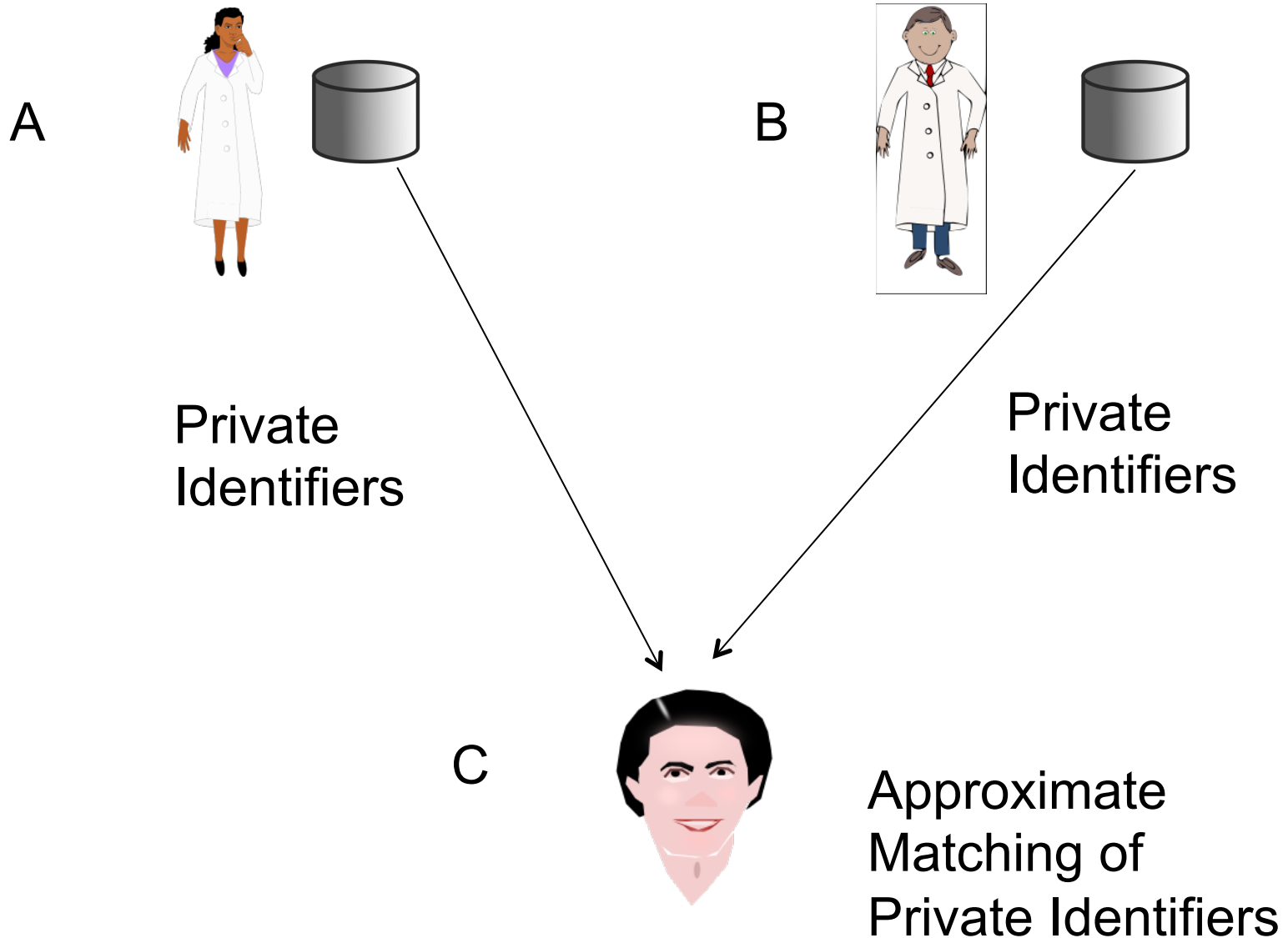


Private
Identifiers

C

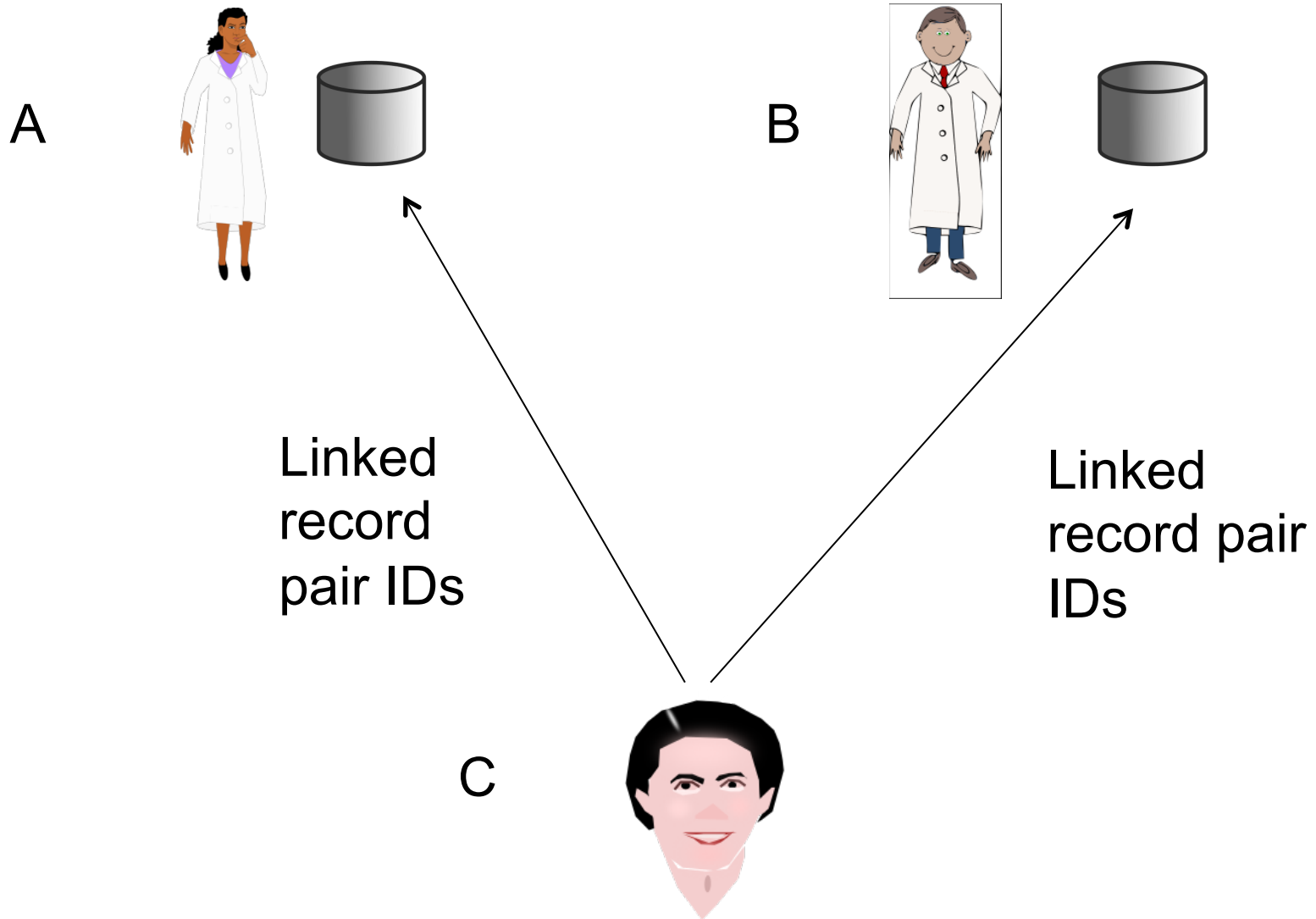


Intended Process





MELBOURNE





1. Computing approximate similarity of two record fields (strings)
2. Representing a record field (string) in a privacy preserving manner (so that it is difficult to reverse engineer its value)
3. Computing approximate similarity of two record fields (strings) that have been represented in privacy preserving manner.



- We wish to compute the (approximate) similarity between two strings X and Y , where $0 \leq \text{sim}(X,Y) \leq 1$
- Convert each string into 2-grams. Convert to lower case and list all substrings of length 2. E.g.
 - James \rightarrow ['ja', 'am', 'me', 'es']
 - Jamie \rightarrow ['ja', 'am', 'mi', 'ie']

$$\text{sim}(X, Y) = \frac{2h}{x + y}$$

- Where
 - h = number of common 2-grams
 - x = number of 2 grams in string X
 - y = number of 2 grams in string Y
 - Known as the “Dice coefficient”
- $\text{Sim}(\text{James}, \text{Jamie}) = (2*2) / (4+4) = 0.5$



- *Using 2-grams, what is the similarity between strings x and y , where*
 - $x = \text{Melbourne}$
 - $y = \text{Bournemouth}$



- Similarity measure can easily be extended to 3-grams, 4-grams,(q-grams)
- *Why not 1-grams?*
- *Why not 10-grams?*



- A bloom filter is an array of l bits, with all bits initially set to zero. E.g. A bloom filter with $l=25$ bits will initially contain 25 zeros

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

- We may store strings in the bloom filter by using hash functions $H_1 \dots H_k$ to turn on certain bits.
 - Each hash function H_i ($1 \leq i \leq k$) maps an input string X to a value in the range $[0, l-1]$.
 - To store a string X in the bloom filter, set array index $H_i(X)$ in the bloom filter to value '1', for each $H_i(X)$



- **To store the string “ha”**
 - Suppose $H_1(\text{ha})=22$, $H_2(\text{ha})=15$
 - » We therefore set bits 22 and 15 to have value ‘1’ in the bloom filter
- **To store the string “ap”**
 - Suppose $H_1(\text{ap})=3$, $H_2(\text{ap})=11$
 - » We therefore sets bits 3 and 11 to have value ‘1’ in the bloom filter
- **To store the string “pp”**
 - Suppose $H_1(\text{pp})=1$, $H_2(\text{pp})=7$
 - » We therefore sets bits 1 and 7 to have value ‘1’ in the bloom filter
- **To store the string “py”**
 - Suppose $H_1(\text{py})=13$, $H_2(\text{py})=15$
 - » We therefore set bits 13 and 15 to have value ‘1’ in the bloom filter



- To store a set of strings $\{X_1, X_2, \dots, X_n\}$ in the bloom filter, each element X_j is hash coded using the k hash functions and all bits having indices $H_i(X_j)$ are set to 1 (for $1 \leq i \leq k$ and for $1 \leq j \leq n$).
- If a bit was set to 1 before, no change is made.



- For any string X , we can thus check if it is stored in the bloom filter by hashing X using the k hash functions.
 - If at least one of the bits having value $H_i(X)$ is set to 0, then X is definitely not a member of the bloom filter.
 - If all of the the bits having value $H_i(X)$ are set to 1, then X appears to be a member of the bloom filter. However, it might not really be a member (a false positive). Probability of false positive is
$$\left(1 - e^{-kn/l}\right)^k$$
 - *Question: Why isn't X a member of the bloom filter with certainty, if all bits having value $H_i(X)$ are set to 1?*



- Google Chrome Browser used to use bloom filters to store malicious URLs.
 - When you visited a website the browser checked if that URL is in the filter. If the answer was (possibly) yes, then the browser would query Google's servers for more detailed information. Otherwise, browsing would continue as normal.
 - The bloom filter was a compressed data structure of all the malicious URLs Google knew about, stored by the browser.
 - Much less space than maintaining a full database of URLs



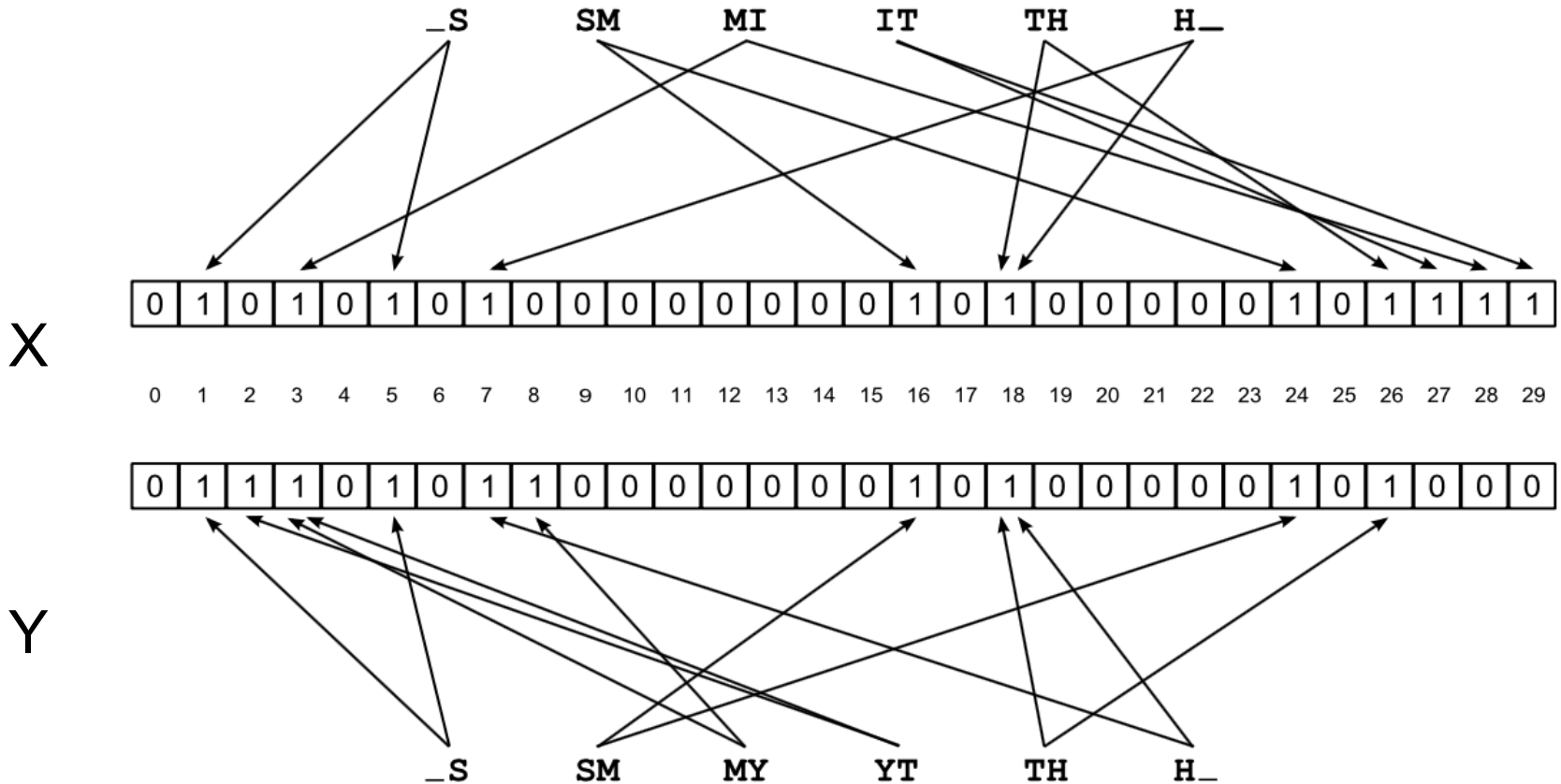
- Given two strings
 - `_SMITH_`
 - 2-Grams: `_S`, `SM`, `MI`, `IT`, `TH`, `H_`
 - `_SMYTH_`
 - 2-Grams: `_S`, `SM`, `MY`, `YT`, `TH`, `H_`
- The 2-grams of the first string are stored in bloom filter B1, the 2-grams of the second string are stored in bloom filter B2. Both bloom filters are the same length and use the same hash functions.
- If the two strings have a lot of 2-grams in common, then their bloom filters will have a large number of identical bit positions set to 1.



- Use the formula

$$\text{sim}(B1, B2) = \frac{2h}{b1 + b2}$$

- Where h is the number of bits set to 1 in both bloom filters
- $b1$ is the number of bits set to 1 in bloom filter $B1$
- $b2$ is the number of bits set to 1 in bloom filter $B2$



String X has 11 bits set to 1. String Y has 10 bits set to 1.
 There are 8 common bits set to 1.
 $\text{Sim}(X,Y) = (2 \cdot 8) / (10 + 11) = 0.762$



- Given two strings X and Y and a similarity value $\text{SIM}(X,Y)$
 - Need to choose a threshold value for deciding whether to report that X matches Y (whether X is approximately similar to Y)



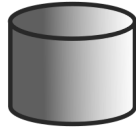
- Following work in “Privacy Preserving Record Linkage Using Bloom Filters”, *BMC Medical Informatics and Decision Making*, Schnell et al, 2009
 - Two databases, 15,000 people each
 - Bloom filter with 2-grams
 - Bloom filter length $l=500$ bits
 - Number of hash functions $k=15$
 - Performance of bloom filter approach
 - Was almost exactly the same as approximate match using original string values (no bloom filter).
 - Identified almost the same matched pairs and non-matched pairs
- Parameter setting is important for performance and security



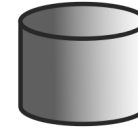
Privacy preserving (approximate) linkage process

MELBOURNE

A



B



C

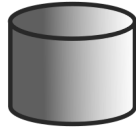


Trusted Party

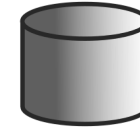


MELBOURNE

A



B



Bloom filters,
one for each
record

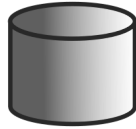
C





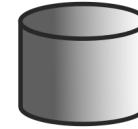
MELBOURNE

A



Bloom filters,
one for each
record

B

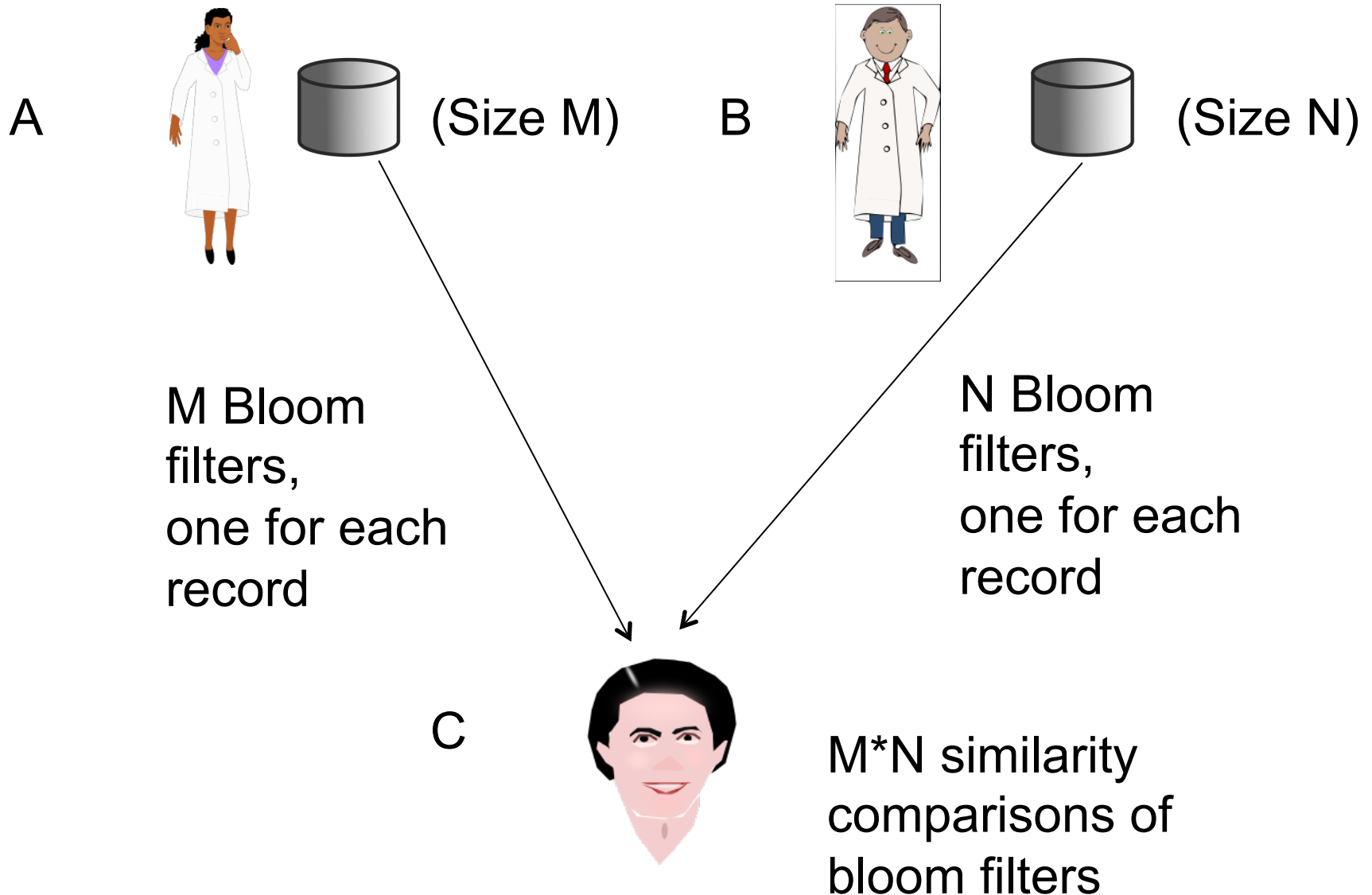


Bloom filters,
one for each
record

C

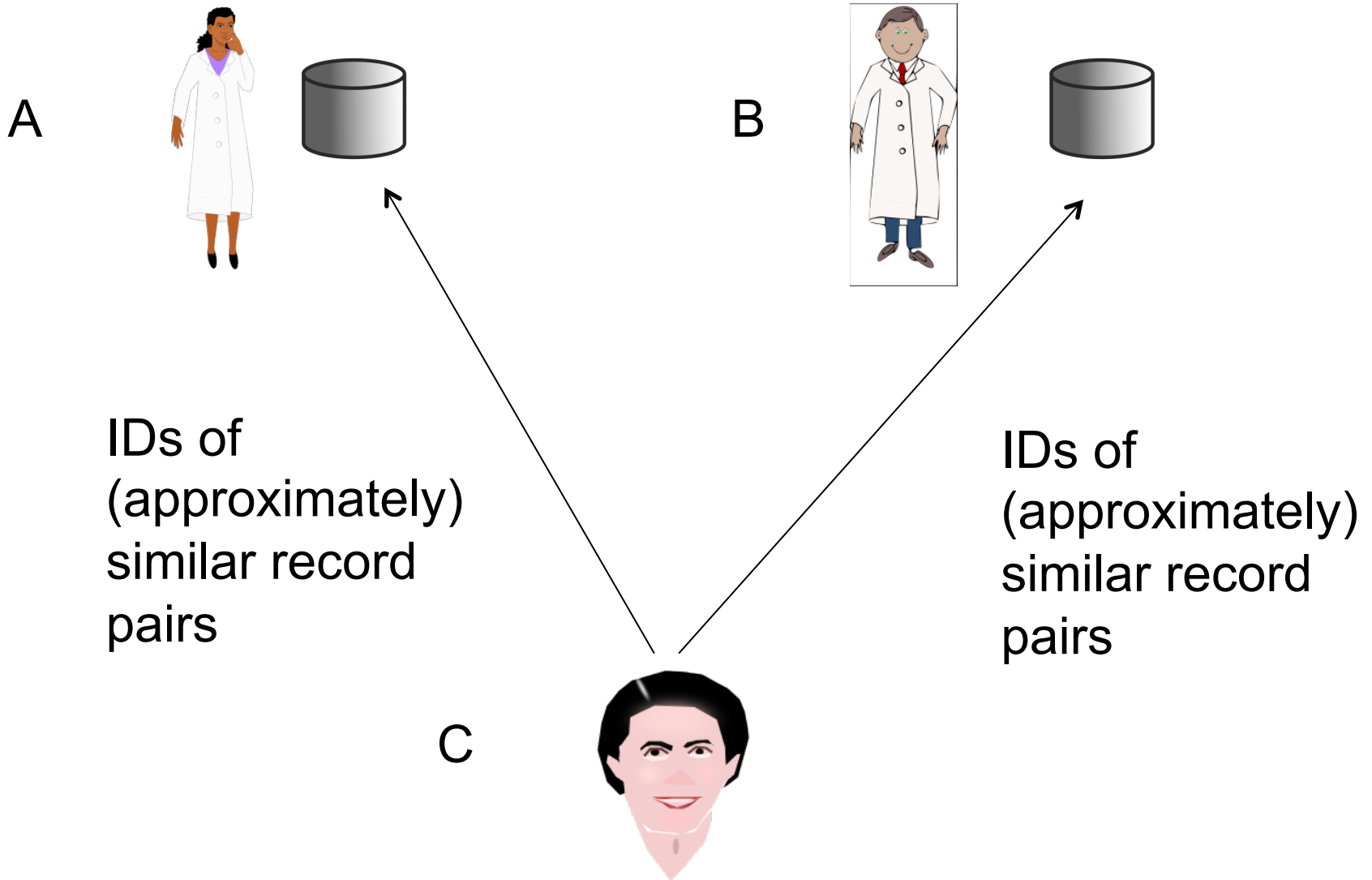


MELBOURNE



Privacy preserving (approximate) linkage process

MELBOURNE





- A and B agree on a bit array length l and k hash functions
 - A and B also agree on a salt and use it in their hash functions
- A and B may also add dummy records (and their corresponding bloom filters) to lessen the chance of C mounting a frequency attack
- May be problems with using bloom filters to represent short strings
 - A and B might add random 2-grams or random bits to the bloom filters, for extra security



- 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>)



- 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



- Relevant work on data linkage, including privacy
 - 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
- This presentation closely follows the following paper
 - Privacy Preserving Record Linkage Using Bloom Filters, *BMC Medical Informatics and Decision Making*, Schnell et al, 2009
 - <http://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/1472-6947-9-41>



- understand the steps of the 3 party protocol for privacy preserving data linkage with approximate matching (using bloom filters)
- understand how to compute similarity of two strings based on 2-grams and why this method is useful
- understand how a bloom filter works (how it is created, how strings are inserted, how strings are checked for membership)
- understand how a bloom filter provides a private representation for strings
 - understand how bloom filters can be used to compare two strings for approximate similarity and the formula for doing this (Dice similarity coefficient)



- Friday 12th : Blockchain
- Monday 15th: Privacy-preserving data publication