**A Tutorial on Privacy-Preserving Record Linkage**

Peter Christen\(^1\) and Vassilios Verykios\(^2\)

\(^1\) Research School of Computer Science, ANU College of Engineering and Computer Science, The Australian National University, Canberra, Australia

\(^2\) School of Science and Technology, Hellespont Open University, Patras, Greece

Contacts: peter.christen@anu.edu.au / verykios@eap.gr

We like to acknowledge the contributions by our PhD students Dimasha Vattalan and Alexandros Karakasis

---

**Motivation**

- Large amounts of data are being collected both by organisations in the private and public sectors, as well as by individuals
- Much of these data are about people, or they are generated by people
  - Financial, shopping, and travel transactions
  - Electronic health and financial records
  - Tax, social security, and census records
  - Emails, tweets, SMSs, blog posts, etc.
- Analysing (mining) such data can provide huge benefits to businesses and governments

---

**Motivation (continued)**

- Often data from different sources need to be integrated and linked
- Improve data quality
- Enrich data
- Allow analyses that are impossible on individual databases
- Lack of unique entity identifiers means that linking is often based on personal information
- When databases are linked across organisations, maintaining privacy and confidentiality is vital
- This is where privacy-preserving record linkage (PPRL) can help

---

**Motivating example:**

**Health surveillance (1)**

- Preventing the outbreak of epidemics requires monitoring of occurrences of unusual patterns in symptoms (in real-time)
- Data from many different sources will need to be collected (including travel and immigration records; doctors, emergency and hospital admissions; drug purchases in pharmacies; animal health data; etc.)
- Privacy concerns arise if such data are stored and linked at a central location
- Private patient data and confidential data from healthcare organisations must be kept secure, while still allowing linking and analysis

---

**Motivating example:**

**Health surveillance (2)**

- Background to record linkage and PPRL
  - Applications, history, and challenges
  - The record linkage and PPRL processes
  - Example scenarios
  - A definition and taxonomy for PPRL
- Exact and approximate PPRL techniques
  - Basic protocols for PPRL
  - Hash-encoding for exact comparison / Tea break
  - Key techniques for approximate comparison
- Selected key techniques for scalable PPRL
- Conclusions and challenges
What is record linkage?

- The process of linking records that represent the same entity in one or more databases (patient, customer, business name, etc.)
- Also known as data matching, entity resolution, data linkage, object identification, identity uncertainty, merge-purge, etc.
- Major challenge is that unique entity identifiers are often not available in the databases to be linked (or if available, they are not consistent)
  E.g., which of these records represent the same person?

<table>
<thead>
<tr>
<th>Dr. Smith, Peter</th>
<th>42 Miller Street 2602 O'Connor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pete Smith</td>
<td>42 Miller St 2600 Canberra A.C.T.</td>
</tr>
<tr>
<td>P. Smithers</td>
<td>24 MR Rd 2600 Canberra ACT</td>
</tr>
</tbody>
</table>

Applications of record linkage

- Applications of record linkage
  - Remove duplicates in a data set (internal linkage)
  - Merge new records into a larger master data set
  - Compile data for longitudinal (over time) studies
  - Clean and enrich data sets for data mining projects
  - Geocode matching (with reference address data)
- Example application areas
  - Immigration, taxation, social security, census
  - Fraud, crime, and terrorism intelligence
  - Business mailing lists, exchange of customer data
  - Social, health, and biomedical research

A short history of record linkage (1)

- Computer assisted record linkage goes back as far as the 1950s (based on ad-hoc heuristic methods)
- Basic ideas of probabilistic linkage were introduced by Newcombe & Kennedy (1962)
- Theoretical foundation by Fellegi & Sunter (1969)
- Compare common record attributes (or fields)
- Compute matching weights based on frequency ratios (global or value specific) and error estimates
- Sum of the matching weights is used to classify a pair of records as a match, non-match, or potential match
- Problems: Estimating errors and thresholds, assumption of independence, and clinical review

A short history of record linkage (2)

- Strong interest in the last decade from computer science (from many research fields, including data mining, AI, knowledge engineering, information retrieval, information systems, databases, and digital libraries)
- Many different techniques have been developed
- Major focus is on scalability to large databases, and linkage quality
  - Various indexing/blocking techniques to efficiently and effectively generate candidate record pairs
  - Various learning-based classification techniques, both supervised and unsupervised, as well as active learning based

Record linkage challenges

- No unique entity identifiers available
- Real world data is dirty (typographical errors and variations, missing and out-of-date values, different coding schemes, etc.)
- Scalability
  - Naive comparison of all record pairs is $O(m \times n)$
  - Remove likely no-matches as efficiently as possible
- No training data in many matching applications
- No record pairs with known true match status
- Privacy and confidentiality (because personal information, like names and addresses, are commonly required for matching)

The record linkage process

Database A

Data pre-processing

Indexing / Matching

Comparison

Classification

Non-matches

Matches

Evaluation

Potential Matches

Database B
**The PPRL process**

- **Example scenario (1): Public health research**
  - A research group is interested in analysing the effects of car accidents upon the health system.
  - Most common types of injuries?
  - Financial burden on the public health system?
  - General health of people after they were involved in a serious car accident?
  - They need access to data from hospitals, doctors, car and health insurers, and from the police.
  - All-identifying data have to be given to the researchers, or alternatively a trusted record linkage unit.
  - This might prevent an organisation from being able or willing to participate (insurers or police).

- **Example scenario (2): Business collaboration**
  - Collaboration benefits businesses (for example in improving efficiency and reducing the costs of their supply chains).
  - They are not willing to share confidential data such as strategies and competitive knowledge.
  - Identifying which supplies and/or customers two businesses have in common must be done without revealing any other confidential knowledge.
  - Involvement of a third party to undertake the linking will be undesirable (due to the risk of collusion of the third party with either company, or potential security breaches at the third party).

- **Example scenario (3): Crime investigation**
  - A national crime investigation unit is tasked with fighting against crimes that are of national significance (such as organised crime syndicates).
  - This unit will likely manage various national databases which draw from different sources (including law enforcement and tax agencies, Internet service providers, and financial institutions).
  - These data are highly sensitive; and storage, retrieval, analysis and sharing must be tightly regulated (collecting such data in one place makes them vulnerable to outsider attacks and internal adversaries).
  - Ideally, only linked records (such as those of suspicious individuals) are available to the unit (significantly reducing the risk of privacy breaches).

- **A definition for PPRL**
  - Assume \( O_1 \cdots O_d \) are the \( d \) owners of their respective databases \( D_1 \cdots D_d \).
  - They wish to determine which of their records \( r_1^j \in D_1 \), \( r_2^j \in D_2 \), \ldots, and \( r_d^j \in D_d \). match according to a decision model \( C(r_1^j, r_2^j, \ldots, r_d^j) \) that classifies pairs (or groups) of records into one of the two classes \( M \) of matches, and \( U \) of non-matches.
  - \( O_1 \cdots O_d \) do not wish to reveal their actual records \( r_1^j \cdots r_d^j \) with any other party.
  - They are however prepared to disclose to a (maybe external) selected party the actual values of some attributes of the record pairs that are in class \( M \) to allow further analysis.

- **A taxonomy for PPRL (1)**
  - Characterise PPRL techniques along fifteen dimensions with the aim to
    - Get a clearer picture of current approaches to PPRL
    - Specify gaps between record linkage and PPRL
    - Identify directions for future research in PPRL
  - Five major topics for assessing PPRL techniques:
    1. Privacy aspects
    2. Linkage technologies
    3. Theoretical complexity
    4. Evaluation
    5. Practical aspects
A taxonomy for PPRL (2)

Taxonomy: Privacy aspects (1)
- Number of parties involved in a protocol
  - Two-party protocol: Two database owners only
  - Three-party protocol: Require a (trusted) third party, the linkage unit
  - Can a PPRL technique be extended to more than two database owners?
- Adversary model
  - Honest-but-curious behaviour: Parties follow the protocol, but they aim to learn about other party’s data
  - Malicious behaviour: Parties can refuse to participate, not follow a protocol, choose arbitrary inputs, etc.

Taxonomy: Privacy aspects (2)
- Privacy technologies
  - Secure hash encoding: `peter` → ‘4Rf#x+Y4i9le@T4o’
  - Sanitation techniques: K-anonymity and friends
  - Secure multi-party computation: Calculate a function such that parties only learn final result
- Differential privacy: Statistical database queries
  - Bloom filters: Bit-strings for set membership testing
  - Public reference values: Telephone directory
  - Phonetic encoding: Soundex, NYSII, etc.
- Extra random records: Hide sensitive real records

Taxonomy: Linkage technology (1)
- Indexing
  - Without an indexing technique, all possible $m \times n$ record pairs need to be compared
  - Indexing aims to identify candidate record pairs that likely correspond to matches
  - Different techniques employed: blocking, sampling, generalisation, clustering, binning, etc.
- Comparison
  - Record based: Compare records as a whole (long strings containing several tokens)
  - Field based: Compare values from individual fields (or attributes)

Taxonomy: Linkage technology (2)
- Matching
  - Exact: Only consider exactly matching values $\sim_{\text{exact}} = 1$, and $\sim_{\text{exact}} = 0$
  - Approximate: Also consider partial similarities $0 \leq \sim_{\text{approximate}} \leq 1$
  - Many different approximate string comparison functions
- Classification
  - Based on the similarities calculated between records
  - Either classify individual record pairs, or employ collective classification
  - Various techniques, most popular is (simple) threshold based classification

Taxonomy: Theoretical complexity
- Using ‘big $O$’ notation (linear $O(n)$, log-linear $O(n \log n)$, quadratic $O(n^2)$, etc. in number of records $n$ in the databases)
- Computation
  - Different computation requirements for database owners and linkage unit (in three-party protocols)
  - Complexity also depends upon number of attributes used
- Communication
  - Size of messages exchanged between parties
  - Number of messages is also crucial (start-up costs)
**Taxonomy: Evaluation (1)**

**Scalability**
- We can measure run-time and memory usage but these are implementation dependent.

**Reduction ratio:** Number of candidate record pairs generated compared to all possible record pairs:
$$rr = 1.0 - \frac{(B_M + B_N)}{(N_M + N_N)}$$

**Pairs completeness:** Like recall, how many true matches are in candidate record pairs:
$$pc = \frac{B_M}{N_M}$$

**Pairs quality:** How many of the candidate record pairs are true matches:
$$pq = \frac{B_M}{B_M + B_N}$$

(with $B_M$ and $B_N$ true matches and non-matches in candidate record pairs, and $N_M$ and $N_N$ all true matches and non-matches: $N_M + N_N = m \times n$).

**Taxonomy: Evaluation (2)**

**Matching quality**
- Classifying record pairs into matches (same entity) and non-matches (different entities) is a binary classification problem.
- Use any classification quality measure, such as accuracy, precision, recall, f-measure, ROC, etc.

**However:** High class imbalance (many more non-matches compared to matches, $N_M << N_N$ and $B_M << B_N$) means accuracy is not suitable.
- Better to use precision, recall, f-measure, etc.

**Taxonomy: Evaluation (3)**

**Privacy**
- Least ‘standardised’ area of evaluation, with various measures used.

**Information entropy (and relative information gain):**
- How much information an attacker can learn.

**Secure multi-party computation simulation proof:**
- Simulate a solution under different adversary models, proof adversary can learn nothing except the expected output.

**Probability of re-identification:** How likely an adversary can correctly guess a sensitive attribute value.

---

**Taxonomy: Practical aspects**

**Implementation**
- Programming language used (if implemented), or only theoretical proof-of-concept.
- Sometimes no details are published.

**Data sets**
- Real-world data sets or synthetic data sets.
- Public data (like UCI Repository) or confidential data.

**Targeted application areas**
- Include health care, census, business, web, finance, etc.
- Sometimes not specified.

---

**Tutorial Outline**

**Background to record linkage and PPRL**
- Applications, history, and challenges.
- The record linkage and PPRL processes.
- Example scenarios.
- A definition and taxonomy for PPRL.

**Exact and approximate PPRL techniques**
- Basic protocols for PPRL.
- Hash-encoding for exact comparison.
- Key techniques for approximate comparison.
- Selected key techniques for scalable PPRL.
- Conclusions and challenges.

---

**Basic protocols for PPRL**

- Two basic types of protocols.
- Two-party protocol: Only the two database owners who wish to link their data.
- Three-party protocols: Use a (trusted) third party (linkage unit) to conduct the linkage.

- Generally, three main communication steps:
  1. Exchange of which attributes to use in a linkage, pre-processing methods, encoding functions, parameters, secret keys, etc.
  2. Exchange of the somehow encoded database records.
  3. Exchange of records classified as matches (or their identifiers only).
Two-party protocol

- More challenging than three-party protocols, but more secure (no third party involved, so no collusion possible)
- Main challenge: How to hide sensitive data from the other database owner
- Step 2 (exchange of the encoded database records) is generally done over several iterations of communication

Three-party protocol

- Easier than two-party protocols, as third party (Carol) prevents database owners from directly seeing each other’s sensitive data
- Linkage unit never sees unencoded data
- Collusion is possible: One database owner gets access to data from the other database owner via the linkage unit

Hash-encoding for PPRL (1)

- A basic building block of many PPRL protocols
- Idea: Use a one-way hash-encoding function to encode values, then compare these hash-codes
  - One-way hash functions like MD5 (message digest) or SHA (secure hash algorithm)
  - Convert a string into a hash-code (MD5 128 bits, SHA-1 160 bits, SHA-2 224–512 bits)
  - For example: ‘peter’ → ‘101010…10101’ or ‘4R#x+YI9@e@t4o’
  - ‘pete’ → ‘011101…011101’ or ‘Z5%o7TqJ@??Jie’
- Single character difference in input values results in completely different hash codes

Hash-encoding for PPRL (2)

- Having only access to hash-codes will make it nearly impossible with current computing technology to learn their original input values
  - Brute force dictionary attack (try all known possible input values and all known hash-encoding functions)
  - Can be overcome by adding a secret key (known only to database owners) to input values before hash-encoding
  - For example, with secret key: ‘42-rocks’:
    - ‘peter’ → ‘42-rocks-peter’ → 19=ale@Qt8?4#87B’
- Frequency attack still possible (compare frequency of hash-values to frequency of known attribute values)

Frequency attack example

- If frequency distribution of hash-encoded values closely matches the distribution of values in a (public) database, then “re-identification” of values might be possible

Problems with hash-encoding

- Simple hash-encoding only allows for exact matching of attribute values
  - Can to some degree be overcome by pre-processing, such as phonetic encoding (Soundex, NYSIS, etc.)
  - Database owners clean their values, convert name variations into standard values, etc.
- Frequency attacks are possible
  - Can be overcome by adding random records to distort frequencies
- First PPRL approaches based on hash-encoding were developed in the mid 1990s by French health researchers (Dussere, Quentin, Bouzelat, et al.)
**Tutorial Outline**

- Background to record linkage and PPRL
  - Applications, history, and challenges
  - The record linkage and PPRL processes
  - Example scenarios
  - A definition and taxonomy for PPRL
- Exact and approximate PPRL techniques
  - Basic protocols for PPRL
  - Hash-encoding for exact comparison
  - Key techniques for approximate comparison
  - Selected key techniques for scalable PPRL
- Conclusions and challenges

**Approximate string matching (1)**

- **Aim:** Calculate a normalised similarity between two strings \(0 \leq \text{sim}_{\text{approx-match}} \leq 1\)
- **Q-gram based approximate comparisons**
  - Convert a string into q-grams (sub-strings of length q)
    - For example, for q = 2: ‘peter’ → ‘pe’, ‘et’, ‘te’, ‘er’
  - Find q-grams that occur in two strings, for example using the Dice coefficient:
    \[ \text{sim}_{\text{Dice}} = 2 \times \frac{c_1 \cap c_2}{c_1 + c_2} \]
    - \(c_1\) = number of q-grams in string \(s_1\), \(c_2\) = number of q-grams in string \(s_2\), \(c_1 \cap c_2\) = number of common q-grams
  - With \(s_1 = \text{‘peter’}\) and \(s_2 = \text{‘peter’}\):
    - \(c_1 = 4\), \(c_2 = 3\), \(c_1 \cap c_2 = 3\) (‘pe’, ‘et’, ‘te’).
    - \(\text{sim}_{\text{Dice}} = 2 \times 3/(4+3) = 6/7 = 0.86\)
  - Variations based on Overlap or Jaccard coefficients

**Approximate string matching (2)**

- **Edit-distance based approximate comparisons**
  - The number of basic character edits (insert, delete, substitute) needed to convert one string into another
  - Can be calculated using a dynamic programming algorithm (of quadratic complexity in length of strings)
  - Convert distance into a similarity as
    \[ \text{sim}_{\text{EDTT}} = 1 - \frac{\text{dist}_{\text{EDTT}}}{\max(l_1, l_2)} \]
    - \(l_1\) = length of string \(s_1\), \(l_2\) = length of \(s_2\)
  - With \(s_1 = \text{‘peter’}\) and \(s_2 = \text{‘peter’}\), \(l_1 = 5, l_2 = 4\),
    \(\text{sim}_{\text{EDTT}} = 1 - 1/5 = 4/5 = 0.8\)
  - Variations consider transposition of two adjacent characters, allow different edit costs, or allow for gaps

**Q-gram based PPRL: Blindfolded record linkage (1)**

- **Proposed by Churches and Christen**
  (Biomed Central, 2004 and PAKDD, 2004)
- **Basic idea:** Securely calculate Dice coefficient using a trusted third party
- **Assumptions**
  - Alice has database A, with attributes A.a, A.b, etc.
  - Bob has database B, with attributes B.a, B.b, etc.
  - They wish to determine whether any of the values in A.a (A.b, etc.) match any of the values in B.a (B.b, etc.), without revealing the actual values to anybody
- **Protocol consists of 4 main steps** (illustrated in detail, as this is the first PPRL approach presented)

**Q-gram based PPRL: Blindfolded record linkage (2)**

- **Protocol step 1:**
  - Alice and Bob agree on q, a secret random key, and a one-way hash encoding function
  - They also agree on a standard of preprocessing strings
- **Protocol step 2 (Alice)**
  - Alice converts each value in A.a into a q-gram list
  - Next she calculates non-empty q-gram sub-lists for each q-gram list
    - For example: ‘peter’ → [‘pe’, ‘et’, ‘te’, ‘er’],
      [‘pe’], [‘et’], [‘te’], [‘er’]
Q-gram based PPRL: Blindfolded record linkage (3)
- Protocol step 3 (Alice)
  - Alice transmits each sub-list into a secure hash code and stores them into a table $A. a. hash. bigr. comb$
  - Alice computes an encrypted version of the record identifier and stores it in $A. a. encrypt. rec. id$
  - Next she places the number of q-grams of each $A. a. hash. bigr. comb$ into $A. a. hash. bigr. comb. len$
  - She then places the length (total number of q-grams) of each original string into $A. a. len$
  - Alice then sends the quadruplet $(A. a. encrypt. rec. id, A. a. hash. bigr. comb, A. a. hash. bigr. comb. len, A. a. len)$ to the linkage unit, Carol

Q-gram based PPRL: Blindfolded record linkage (4)
- Bob conducts steps 2 and 3 on his values in $B. a$ and sends his quadruplet to Carol
- Protocol step 4 (Carol)
  - For each value of $a. hash. bigr. comb$ shared by $A$ and $B$, for each unique pairing of $(A. a. encrypt. rec. id, B. a. encrypt. rec. id)$, Carol calculates the Dice coefficient:
    $$sim_{Dice} = \frac{A. a. hash. bigr. comb. len}{A. a. len + B. a. len}$$
  - Carol then selects the maximum $sim_{Dice}$ for each pair $(A. a. encrypt. rec. id, B. a. encrypt. rec. id)$ and sends these results to Alice and Bob

Q-gram based PPRL: Blindfolded record linkage (5)
- Simple example: Alice has (‘ra1’, ‘peter’) and Bob has (‘rb2’, ‘pete’) (and assume $q = 2$)
  - Alice’s quadruplets (shown unencoded):
  - Bob’s quadruplets:
    - (‘rb2’, [‘pe’, ‘et’, ‘te’], 3, 3),
    - (‘rb2’, [‘et’, ‘te’], 2, 3),
    - (‘rb2’, [‘pe’, ‘te’], 2, 3),
    - (‘rb2’, [‘pe’, ‘et’], 2, 3), etc.

Q-gram based PPRL: Blindfolded record linkage (6)
- Several attributes a, b, c, etc. can be compared independently (by different linkage unit)
- These linkage units send their results to another party (David), who forms a (sparse) matrix by joining the results
- The final matching weight for a record pair is calculated by summing individual $sim_{Dice}$
- David arrives at a set of blindly linked records (triplets of $(A. a. encrypt. rec. id, B. a. encrypt. rec. id, sim_{Dice})$)
- Drawbacks: large communication overheads, Carol can mount a frequency attack (count how often certain hashed q-gram values appear)

Q-gram based PPRL: Using Bloom-filters (1)
- Proposed by Schnell et al. (Biomed Central, 2009)
- A Bloom filter is a bit-array, where a bit is set to 1 if a hash-function $H_x(x)$ maps an element $x$ of a set into this bit (elements in our case are q-grams)
  - $0 \leq H_x(x) \leq I$, with $I$ the number of bits in Bloom filter
  - Many hash functions can be used (Schnell: $k = 30$)
  - Number of bits can be large (Schnell: $I = 1000$ bits)
- Basic idea: Map q-grams into Bloom filters using hash functions only known to database owners, send Bloom filters to a third party which calculates Dice coefficient (number of 1-bits in Bloom filters)

Q-gram based PPRL: Using Bloom-filters (2)
- 1-bits for string ‘peter’: 7, 1-bits for ‘pete’: 5, common 1-bits: 5, therefore $sim_{Dice} = 2 \times 5/(7+5) = 10/12 = 0.83$
- Collisions will effect the calculated similarity values
- Number of hash functions and length of Bloom filter need to be carefully chosen
Q-gram based PPRL: Using Bloom-filters (3)

- Map all attributes into one Bloom filter, or build individual attribute Bloom filters
- Frequency attacks are possible
  - Frequency of q-bits reveals frequency of q-grams (especially problematic for short strings)
  - Using more hash functions can improve security
- Add random (dummy) string values to hide real values
- Kuzu et al. (2011) proposed a constraint satisfaction cryptanalysis attack (certain number of hash functions and Bloom filter length are vulnerable)
- Durham (2012) improved security by random bit-sampling from attribute Bloom filters

Secure edit-distance for PPRL (1)

- Proposed by Atallah et al. (WPES, 2003)
- Calculate edit distance between two strings such that parties only learn final result (two party protocol)
- Basic idea: The dynamic programming matrix is split across the two parties: $M = M_A + M_B$

\[
\begin{array}{cccccccc}
0 & 2 & 1 & 1 & 1 & 1 & 1 & 1 \\
2 & 0 & 1 & 2 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 1 & 2 & 1 & 1 & 1 \\
1 & 1 & 1 & 0 & 1 & 2 & 1 & 1 \\
1 & 1 & 1 & 1 & 0 & 1 & 2 & 1 \\
1 & 1 & 1 & 1 & 1 & 0 & 1 & 2 \\
1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
\end{array}
\]

- ‘gail’ → substitute ‘i’ with ‘y’ and insert ‘e’ → ‘gaye’

Secure edit-distance for PPRL (2)

- Matrix $M$ is built row-wise
- Element $M[i,j]$ is the number of edits needed to convert $s_1[i]$ into $s_2[j]$
- Calculated as:
  - $M[i,j] = \min\{M[i-1,j-1] + S(s_1[i], s_2[j]), $ $M[i-1,j] + D(s_1[i]), $ $M[i,j-1] + I(s_2[j])\}$
- (a substitute) $M[i-1,j-1]$ (a delete) $M[i,j]$ (an insert)
- (often the ‘different’ costs are set to 1)
- At each step of the protocol, Alice and Bob need to determine the minimum of three values, without learning at which position the minimum occurred

Secure edit-distance for PPRL (3)

- Protocol requires a secure function to calculate the minimum value in a shared vector, $\hat{c} = \hat{a} + \hat{b}$
- without knowing the position of the minimum (and a variation to calculate the maximum of values)
- To check if $c_i > c_j$, use: $c_i > c_j = (a_i + b_i) > (a_j + b_j) \iff (a_i - a_j) > (b_i - b_j)$
- To ‘hide’ position of minimum value, use a ‘blind and permute’ protocol based on homomorphic encryption (first Alice blinds Bob, then Bob blinds Alice)
- Homomorphic encryption: $E(a) + E(b) = E(a + b)$
- For substitution cost, check if $\min(s_1[i], s_2[j])$ is different from $\max(s_1[i], s_2[j])$

Secure edit-distance for PPRL (4)

- Atallah et al. describe several variations of their protocol for different cases of costs $S(\cdot, \cdot)$, $D(\cdot, \cdot)$, and $I(\cdot)$
- Certain applications might only allow inserts and deletions, others have substitution costs depending upon the ‘distance’ from $s_1[i]$ to $s_2[j]$
- Major drawback of this protocol: For each element in $M$ one communication step is required (number of communication steps is quadratic in the length of the two strings)
- Not scalable to linking large databases
**Secure TF-IDF and Euclidean distance for PPRL (1)**
- Proposed by Ravikumar et al. (PSDM, 2004)
- Use a secure dot product protocol to calculate distance metrics (two party protocol)
- TF-IDF (term-frequency, inverse document frequency)
  - Weighting scheme used to calculate Cosine similarity between text documents based on their term vectors
- Soft TF-IDF (Cohen et al., KDD 2003) combines an approximate string comparison function with TF-IDF, leading to improved matching results
- Basic idea: Calculate stochastic dot product by sampling vector elements and use secure set intersection protocol to calculate similarity

**Secure TF-IDF and Euclidean distance for PPRL (2)**
- Calculate the secure dot product of two vectors $\hat{a}$ (held by Alice), and $\hat{b}$ (held by Bob) (vector elements are TF-IDF weights for tokens in records)
  1. Alice calculates normalisation $z_A = \sum_{i=1}^{n} a_i$, with $n$ being the dimension of vector $\hat{a}$ (Bob calculates $z_B$ on his vector, also assumed to be of length $n$)
  2. They each sample $k < n$ elements, $i \in \{1, \ldots, n\}$ with probability $a_i/z_A$ into set $T_A$, or $b_i/z_B$ into set $T_B$
  3. Use secure set intersection cardinality protocol (Vaidya and Clifton, 2005) to find $v = |T_A \cap T_B|$, then average $v' = v / k$
  4. Calculate dot product as: $v'' = v' + z_A \times z_B$

**Secure TF-IDF and Euclidean distance for PPRL (3)**
- Experiments on bibliographic database Cora (records containing author names, article titles, dates, and venues of conferences and workshops)
- After around $k = 1,000$ samples (with $n = 10,000$), the secure stochastic scalar product achieved results comparable to the scalar product using the full vectors.
- Major drawback of this protocol: Requires $k$ messages between Alice and Bob to calculate secure set intersection
- Not scalable to linking large databases

**Approximate matching in PPRL using a public reference table (1)**
- Proposed by Pang et al. (IPM, 2009)
- Basic idea: Use large public list of reference (string) values available to both Alice and Bob, and calculate distance estimates based on triangular inequality
- Assume reference value $r$ and private values $s_A$ held by Alice and $s_B$ held by Bob, and edit-distance function $ED(s_A, s_B)$:
  - $ED(s_A, s_B) \leq ED(s_A, r) + ED(s_B, r)$
- A trusted third party calculates these distances based on encoded string and reference values

**Approximate matching in PPRL using a public reference table (2)**
- Major drawback: Security issues, as third party can conduct analysis of string distances and size of cluster neighbourhoods (assuming the reference table is available to the third party)
- The size of clusters and the distribution of distances in a cluster can allow identification of rare names (for each reference value, there will be a specific distribution of how many other reference values have a distance of 1, 2, 3, etc. edits)
  - For example: ‘sydney’ [ed1=5, ed2=15, ed3=154, ed4=4371, …] ‘wollongong’ [ed1=0, ed2=0, ed3=4, ed4=5, …]

**Approximate matching in PPRL using a public reference table (3)**
- If $s_A$ and $s_B$ are compared with several reference values, the mean of distance estimates is used
- This approach can be employed with different (string) distance measures
- A scalable approach if private values are only compared with ‘similar’ reference values (neighbourhood clustering)
Approximate matching in PPRL using a public reference table (4)

- Security issues can be overcome by
  - aiming to have all clusters being the same size
  - use relative distances (add or subtract constant to all distances sent to the linkage unit)
- Recent, Vatsalan et al. proposed a two-party protocol based on reference values (AusDM, 2011)
  - Basic idea is to use binning of similarity values to hide actual values between the two database owners
  - Use of the reverse triangular inequality for similarities rather than distances
  - Scalability is achieved through the use of phonetic encoding to generate blocks (clusters)

Phonetic encoding based PPRL (1)

- Proposed by Karakasidis and Verykios (BCI, 2009)
- Use phonetic encoding functions (like Soundex, NYSIS, Double-Metaphone, etc.) to generalise and obfuscate sensitive values
  - Soundex('peter') = 'p360'
  - Soundex('gail') = 'g400'
  - Soundex('pedro') = 'p360'
  - Soundex('gaye') = 'g400'
- Basic idea: Two database owners phonetically encode (and hash-encode) their values, add 'faked' phonetic values, and send these to a third party to conduct the linking
- The use of computationally fast phonetic algorithms make this an efficient approach

Phonetic encoding based PPRL (2)

- A five-step protocol
  1. The two database owners convert their alphanumeric attribute values into phonetic codes
  2. Fake phonetic codes are injected into the encoded data set, and the data are sent to a third party
  3. Phonetic codes are joined at the third party, which then returns the matching codes to the two database owners
  4. Each database owner asks from the other the data for their true phonetic codes
  5. Data (records or record identifiers only) for real matched phonetic codes are exchanged

Phonetic encoding based PPRL (3)

- The quantitative measuring of privacy by means of Relative Information Gain (RIG) is used (Karakasidis et al., DPM 2011)
  - Low RIG means no information can be gained from encoded phonetic values only
  - It is shown that phonetic codes do provide privacy
- Privacy is achieved by three ways:
  1. Generalisation properties of phonetic encoding (converting similar values into the same codes)
  2. Injection of fake codes (obfuscation), to maximise privacy in terms of RIG
  3. Secure hash encoding of all values communicated

Tutorial Outline

- Background to record linkage and PPRL
  - Applications, history, and challenges
  - The record linkage and PPRL processes
  - Example scenarios
  - A definition and taxonomy for PPRL
- Exact and approximate PPRL techniques
  - Basic protocols for PPRL
  - Hash-encoding for exact comparison
  - Key techniques for approximate comparison
- Selected key techniques for scalable PPRL
- Conclusions and challenges

Blocking aware private record linkage (1)

- Proposed by Al-Lawati et al. (IQIS, 2005)
- A three party protocol featuring the first attempt for private blocking to make PPRL scalable
- Basic idea: Private record linkage is achieved by using hash signatures based on TF-IDF vectors
  - These vectors are built on tokens (unigrams) extracted from attribute values
  - Three blocking approaches were presented, they provide a tradeoff between performance and privacy achieved
**Blocking aware private record linkage (2)**

<table>
<thead>
<tr>
<th>Database A</th>
<th>Database B</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Value</td>
</tr>
<tr>
<td>a1</td>
<td>(x', b)</td>
</tr>
<tr>
<td>a2</td>
<td>(v', b)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>HS(a1)</td>
<td>TF-IDF(a1, b)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>HS(a2)</td>
<td>0</td>
<td>0</td>
<td>TF-IDF(a2, v')</td>
</tr>
<tr>
<td>HS(b1)</td>
<td>TF-IDF(b1, v')</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>HS(b2)</td>
<td>TF-IDF(b2, v')</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(F is an array of floating-point numbers)

- Database owners can independently generate their TF-IDF weight vectors, and encode them into hash signatures (HS)
- Sent to a third party, which can calculate Cosine similarity

---

**Blocking aware private record linkage (3)**

- Three blocking approaches based on token intersection (Jaccard similarity): Records are only compared if their token intersection is non-empty
  - **Simple blocking**: a separate block is generated for each token in a record
  - **Record-aware blocking**: combines the hash signature of each record with a record ID so that duplicates appearing in simple blocking are eliminated
  - **Frugal third party blocking**: the database owners do a secure set intersection to identify common tokens

- All three blocking approaches are vulnerable to frequency attacks (database, block and vocabulary sizes, and record length)

---

**Privacy-preserving schema and data matching (1)**

- Proposed by Scannapieco et al. (SIGMOD, 2007)
- Schema matching is achieved by using an intermediate 'global' schema sent by the linkage unit to the database owners
  - The database owners assign each of their linkage attributes to the global schema
  - They send their hash-encoded attribute names to the linkage unit
- Basic idea of record linkage is to map attribute values into a multi-dimensional space such that distances are preserved (using the SparseMap algorithm)

---

**Privacy-preserving schema and data matching (2)**

- Three phases involving three parties
- Phase 1: Setting the embedding space
  - Database owners agree upon a set of (random) reference strings (known to both)
  - Each reference string is represented by a vector in the embedding space
- Phase 2: Embedding of database records into space using SparseMap
  - Essentially, vectors of the distances between reference and database values are calculated
  - Resulting vectors are sent to the third party

---

**Privacy-preserving schema and data matching (3)**

- Phase 3: Third party stores vectors in a multi-dimensional index and conducts a nearest-neighbour search (von nods to each other are classified as matches)
- Major drawbacks:
  - Matching accuracy depends upon parameters used for the embedding (dimensionality and distance function)
  - Certain parameter settings give very low matching precision results
  - Multi-dimensional indexing becomes less efficient with higher dimensionality
  - Susceptible to frequency attacks (closeness of nearest neighbours in multi-dimensional index)

---

**Efficient private record linkage**

- Proposed by Yakout et al. (ICDE, 2009)
- Convert the three-party protocol by Scannapieco et al. into a two-party protocol
- Basic idea:
  - Embed records into a multi-dimensional space, then map them into complex numbers
  - Exchange these complex numbers between the database owners
  - Possible matching record pairs are those which have complex numbers within a certain maximum distance
  - Calculate actual distances between records using a secure scalar product based on random numbers
**A hybrid approach to PPRL (1)**

- Proposed by Inan et al. (ICDE, 2008)
- Use k-anonymity to generalise (sanitise) databases and find 'blocks' of possible matching record pairs
- Basic idea: In a first step, generate value generalisation hierarchies (VGH); in a second step calculate distances between records with same generalised values using a secure multi-party (SMC) approach (based on homomorphic encryption)
- VGHs are hierarchical tree-like structures where a node at each level is a generalisation of its descendants

**A hybrid approach to PPRL (2)**

<table>
<thead>
<tr>
<th>ID</th>
<th>Education</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Junior Sec</td>
<td>22</td>
</tr>
<tr>
<td>2</td>
<td>Senior Sec</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>Junior Sec</td>
<td>27</td>
</tr>
<tr>
<td>4</td>
<td>Bachelor</td>
<td>33</td>
</tr>
<tr>
<td>5</td>
<td>Bachelor</td>
<td>39</td>
</tr>
<tr>
<td>6</td>
<td>Grad School</td>
<td>34</td>
</tr>
</tbody>
</table>

3-anonymous generalisation

**A hybrid approach to PPRL (3)**

- Generalised and hash-encoded attribute values are sent to the third party, which can classify record pairs as matches, non-matches or possible matches (depending upon how many generalised attribute values two records have in common)
- SMC approach is used to calculate similarities of possible matches (computationally more expensive)
- User can set threshold to tune between precision and recall of the resulting matched record pairs
- Main drawback: Cannot be applied on alphanumeric values (names and addresses) that do not have a VGH

**Private record matching using differential privacy (1)**

- Proposed by Inan et al. (EDBT, 2010)
- A modification of their k-anonymity generalisation approach (improved security, and no third party required)
- Use a differential privacy based approach for blocking (differential privacy boils down to adding noise to aggregate queries in statistical database to avoid disclosure by combining results)
- Basic idea: the database owners disclose only the perturbed results of a set of statistical queries, and use special indexing techniques that are compliant with differential privacy

**Private record matching using differential privacy (2)**

- Database owners partition their data into sub-sets, and exchange their size and extend
  - Spatial indexing techniques (BSP-Tree, KD-Tree, or R-Tree) are used to form sub-sets (hyper-rectangles)
  - Blocking phase filters out pairs of sub-sets that cannot contain matches
  - Construct transcripts that satisfy differential privacy (add output perturbation)
  - The way queries for the transcripts are generated is a crucial aspect of this approach
  - SMC approach based on homomorphic encryption is used to calculate similarities for record pairs not removed by blocking

**Reference table based k-anonymous private blocking (1)**

- Proposed by Karakasidis and Verykios (SAC, 2012)
- The only private blocking approach up to the moment suitable for blocking any type of data
- The first private blocking method which assures k-anonymity for each of the blocked elements
- May be combined with any private matching method
- No information leaked and not susceptible to frequency attack
- Basic idea: Based on the intuition that if two data elements are similar to a third one, they are very likely to be similar with each other
**Reference table based k-anonymous private blocking (2)**

- The method consists of the following steps
  - Data holders agree on a common publicly available corpus of data, called reference table
  - They cluster the reference table data using the nearest neighbour density clustering algorithm (with cluster size more than k for assuring k-anonymity)
  - The most similar cluster for each attribute value is found, and values in the same cluster form a block
  - The number of blocks formed is equal to the number of reference table clusters
  - The blocks are sent to a third party and records from corresponding blocks are privately matched using any private approximate matching algorithm

---

**Tutorial Outline**

- Background to record linkage and PPRL
  - Applications, history, and challenges
  - The record linkage and PPRL processes
  - Example scenarios
  - A definition and taxonomy for PPRL
  - Exact and approximate PPRL techniques
    - Basic protocols for PPRL
    - Hash-encoding for exact comparison
    - Key techniques for approximate comparison
  - Selected key techniques for scalable PPRL
  - Conclusions and challenges

---

**Conclusions**

- Significant advances to achieving the goal of PPRL have been developed in recent years
  - Various approaches based on different techniques
  - Can link records securely, approximately, and in a (somewhat) scalable fashion
  - So far, most PPRL techniques concentrated on approximate matching techniques, and on making PPRL more scalable to large databases
  - However, no large-scale comparative evaluations of PPRL techniques have been published
  - Only limited investigation of classification and linking assessment in PPRL

---

**Challenges and future work (1)**

- Improved classification for PPRL
  - Mostly simple threshold based classification is used
  - No investigation into advanced methods, such as collective entity resolution techniques
  - Supervised classification is difficult – no training data in most situations
- Assessing linkage quality and completeness
  - How to assess linkage quality (precision and recall)?
    - How many classified matches are true matches?
    - How many true matches have we found?
  - Evaluating actual record values is not possible (as this would reveal sensitive information)

---

**Advertisment: Book ‘Data Matching’ by P Christen (Springer)**

- Book series: Data-Centric Systems and Applications (http://www.springer.com/series/5258)
- Publication in mid August 2012
- Content:
  1. Introduction and (2) The Data Matching Process
  3. Data Pre-Processing, (4) Indexing, (5) Field and Record Comparison, (6) Classification, and (7) Evaluation of Matching Quality and Complexity
  8. Privacy Aspects of Data Matching, (9) Further Topics and Research Directions, and (10) Data Matching Systems

---

**Challenges and future work (2)**

- A framework for PPRL is needed
  - To facilitate comparative experimental evaluation of PPRL techniques
  - Needs to allow researchers to plug-in their techniques
  - Benchmark data sets are required (biggest challenge, as such data is sensitive)
- PPRL on multiple databases
  - Most work so far is limited to linking two databases (in reality often databases from several organisations)
  - Pair-wise linking does not scale up
  - Preventing collusion between (sub-groups of) parties becomes more difficult
Thank you for attending our tutorial!

Enjoy the rest of PAKDD and your stay in Malaysia.

For questions please contact:

peter.christen@anu.edu.au
verykios@eap.gr

References (1)


References (2)


Bouzid M., Quaint C., and Dussere L.: Extraction and anonymity protocol of medical file. ASIA Fall Symposium, 1996.


References (3)


References (4)


References (5)


