Privacy-Preserving Record Linkage for Big Data: Current Approaches and Research Challenges

Dinusha Vatsalan[1], Ziad Sehili[2], Peter Christen[1], and Erhard Rahm[2]

[1] Research School of Computer Science, The Australian National University,
Acton ACT 2601, Australia; {dinusha.vatsalan,peter.christen}@anu.edu.au
[2] Database Group, University of Leipzig, 04109 Leipzig, Germany;
{rahm,sehili}@informatik.uni-leipzig.de

Abstract. The growth of Big Data, especially personal data dispersed in multiple data sources, presents enormous opportunities and insights for businesses to explore and leverage the value of linked and integrated data. However, privacy concerns impede sharing or exchanging data for linkage across different organizations. Privacy-preserving record linkage (PPRL) aims to address this problem by identifying and linking records that correspond to the same real-world entity across several data sources held by different parties without revealing any sensitive information about these entities. PPRL is increasingly being required in many real-world application areas. Examples range from public health surveillance to crime and fraud detection, and national security. PPRL for Big Data poses several challenges, with the three major ones being (1) scalability to multiple large databases, due to their massive volume and the flow of data within Big Data applications, (2) achieving high quality results of the linkage in the presence of variety and veracity of Big Data, and (3) preserving privacy and confidentiality of the entities represented in Big Data collections. In this chapter, we describe the challenges of PPRL in the context of Big Data, survey existing techniques for PPRL, and provide directions for future research.

Keywords: Record linkage, Privacy, Big Data, Scalability

1 Introduction

With the Big Data revolution, many organizations collect and process datasets that contain many millions of records to analyze and mine interesting patterns and knowledge in order to empower efficient and quality decision making [28, 53]. Analyzing and mining such large datasets often require data from multiple sources to be linked and aggregated. Linking records from different data sources with the aim to improve data quality or enrich data for further analysis is occurring in an increasing number of application areas, such as in healthcare, government services, crime and fraud detection, national security, and business applications [28, 52]. Effective ways of linking data from different sources have also played an increasingly important role in generating new insights for population informatics in the health and social sciences [100].

For example, linking health databases from different organizations facilitates quality health data mining and analytics in applications such as epidemiological studies (outbreak detection of infectious diseases) or adverse drug reaction studies [20, 117]. These applications require data from several organizations to be linked, for example human health data, travel data, consumed drug data, and even animal health data [38]. Linked health databases can also be used for the development of health policies in a more efficient and effective way compared to traditionally used time-consuming survey studies [37, 89].

Record linkage techniques are also being used by national security agencies and crime investigators for effective identification of fraud, crime, or terrorism suspects [74, 125, 168]. Such applications require data from law enforcement agencies, immigration departments, Internet service providers, businesses, as well as financial institutions [125].

In recent time, record linkage is increasingly being required by social scientists in the field of population informatics to study insights into our society from 'social genome' data, the digital traces that contain person-level data about social beings [100]. The 'Beyond 2011' program by the Office for National Statistics in the UK, for example, has carried out research to study different possible approaches to producing population and socio-demographics statistics for England and Wales by linking data from several sources [56].

Record linkage within a single organization does not generally involve privacy and confidentiality concerns (assuming there are no internal threats within the organization and the linked data are not being revealed outside the organization). An example application is the deduplication of a customer database by a business using record linkage techniques for conducting effective marketing activities. However, in many countries record linkage across several organizations, as required in the above example applications, might not allow the exchange or the sharing of database records between organizations due to laws or regulations. Some example Acts that describe the legal restrictions of disclosing personal or sensitive data are: (1) the Data-Matching Program Act in Australia[3], (2) the European Union (EU) Personal Data Protection Act in Europe[4], and (3) the Health Insurance Portability and Accountability Act (HIPAA) in the USA[5].

The privacy requirements in the record linkage process have been addressed by developing 'privacy-preserving record linkage' (PPRL) techniques, which aim to identify matching records that refer to the same entities in different databases without compromising privacy and confidentiality of these entities. In a PPRL project, the database owners (or data custodians) agree to reveal only selected information about records that have been classified as matches among each other, or to an external party, such as a researcher [165]. However, record linkage requires access to the actual values of certain attributes.

[3] https://www.oaic.gov.au/privacy-law/other-legislation/government-data-matching [Accessed: 15/06/2016]

[4] http://ec.europa.eu/justice/data-protection/index_en.htm [Accessed: 15/06/2016]

[5] http://www.hhs.gov/ocr/privacy/ [Accessed: 15/06/2016]

Known as quasi-identifiers (QIDs), these attributes need to be common in all databases to be linked and represent identifying characteristics of entities to allow matching of records. Examples of QIDs are first and last names, addresses, telephone numbers, or dates of birth. Such QIDs often contain private and confidential information of entities that cannot be revealed, and therefore the linkage has to be conducted on masked (encoded) versions of the QIDs to preserve the privacy of entities. Several masking techniques have been developed (as we will describe in Sect. 3.4), using two different types of general approaches: (1) secure multi-party computation (SMC) [112] and (2) data perturbation [88].

Leveraging the tremendous opportunities that Big Data can provide for businesses comes with the challenges that PPRL poses, including scalability, quality, and privacy. Big Data implies enormous data *volume* as well as massive flows (*velocity*) of data, leading to scalability challenges even with advanced computing technology. The *variety* and *veracity* aspects of Big Data require biases, noise, variations and abnormality in data to be considered, which makes the linkage process more challenging. With Big Data containing massive amounts of personal data, linking and mining data may breach the privacy of those represented by the data. A practical PPRL solution that can be used in real-world applications should therefore address these challenges of scalability, linkage quality, and privacy. A variety of PPRL techniques has been developed over the past two decades, as surveyed in [154, 165]. However, these existing approaches for PPRL fall short in providing a sound solution in the Big Data era by not addressing all of the Big Data challenges. Therefore, more research is required to leverage the huge potential that linking databases in the era of Big Data can provide for businesses, government agencies, and research organizations.

In this chapter, we review the existing challenges and techniques, and discuss research directions of PPRL for Big Data. We provide the preliminaries in Sect. 2 and review existing privacy techniques for PPRL in Sect. 3. We then discuss the scalability challenge and existing approaches that address scalability of PPRL in Sect. 4. In Sect. 5, we describe the challenges and existing techniques of PPRL on multiple databases, which is an emerging research avenue that is being increasingly required in many Big Data applications. In Sect. 6 we discuss research directions in PPRL for Big Data, and in Sect. 7 we conclude this chapter with a brief summary of the topic covered.

2 Background

Building on the introduction to record linkage and privacy-preserving record linkage (PPRL) in Sect. 1, we now present background material that contributes to the understanding of the preliminaries. We describe the basic concepts and challenges in Sect. 2.1, and then describe the process of PPRL in Sect. 2.2.

2.1 Overview and Challenges of PPRL

Record linkage is a widely used data pre-processing and data cleaning task where the aim is to link and integrate records that refer to the same entity from two or

ID	Given_name	Surname	DOB	Gender	Address	Loan_type	Balance
6723	peter	robert	20.06.72	M	16 Main Street 2617	Mortgage	230,000
8345	smith	roberts	11.10.79	M	645 Reader Ave 2602	Personal	8,100
9241	amelia	millar	06.01.74	F	49E Applecross Rd 2415	Mortgage	320,750

Table 1. Example bank database.

PID	Last_name	First_name	Age	Address	Sex	Pressure	Stress	Reason
P1209	roberts	peter	41	16 Main St 2617	m	140/90	high	chest pain
P4204	miller	amelia	39	49 Aplecross Road 2415	f	120/80	high	headache
P4894	sieman	jeff	30	123 Norcross Blvd 2602	m	110/80	normal	checkup

Table 2. Example health database.

multiple disparate databases. The record pairs (when linking two databases) or record sets (when linking more than two databases) are compared and classified as 'matches' by a linkage model if they are assumed to refer to the same entity, or as 'non-matches' if they are assumed to refer to different entities [26, 54]. The frequent absence of unique entity identifiers across the databases to be linked makes it impossible to use a simple SQL-join [30], and therefore linkage requires sophisticated comparisons between a set of QIDs (such as names and addresses) that are commonly available in the records to be linked. However, these QIDs often contain personal information and therefore revealing or exchanging them for linkage is not possible due to privacy and confidentiality concerns.

As an example scenario, assume a demographer who aims to investigate how mortgage stress (having to pay large sums of money on a regular basis to pay off a house) is affecting people with regard to their mental and physical health. This research will require data from financial institutions as well as hospitals as shown in Tables 1 and 2. Neither of these organizations is likely willing or allowed by law to provide their databases to the researcher. The researcher only requires access to some attributes of the records (such as loan type, balance amount, blood pressure, and stress level) that are linked across these databases, but not the actual identities of the individuals that were linked. However, personal details (such as name, age or date of birth, gender, and address) are needed as QIDs to conduct the linkage due to the absence of unique identifiers across the databases.

As illustrated in the above example application (shown in Tables 1 and 2), linking records in a privacy-preserving context is important, as sharing or exchanging sensitive and confidential personal data (contained in QIDs of records) between organizations is often not feasible due to privacy concerns, legal restrictions, or commercial interests. Therefore, databases need to be linked in such ways that no sensitive information is being revealed to any of the organizations involved in a cross-organizational linkage project, and no adversary is able to learn anything about these sensitive data. This problem has been addressed by the emerging research area of PPRL [165].

The basic ideas of PPRL techniques are to mask (encode) the databases at their sources and to conduct the linkage using only these masked data. This means no sensitive data are ever exchanged between the organizations involved in a PPRL protocol, or revealed to any other party. At the end of such a PPRL process, the database owners only learn which of their own records match with a high similarity with records from the other database(s). The next steps would be exchanging the values in certain attributes of the matched records (such as loan type, balance amount, blood pressure, and stress level in the above example) between the database owners, or sending selected attribute values to a third party, such as a researcher who requires the linked data for their project [165]. Recent research outcomes and experiments conducted in real health data linkage validate that PPRL can achieve linkage quality with only small loss compared to traditional record linkage using unencoded QIDs [134, 136].

Using PPRL for Big Data involves many challenges, among them the following three key challenges need to be addressed to make PPRL viable for Big Data applications:

1. **Scalability:** The number of comparisons required for classifying record pairs or sets equals to the product of the size of the databases that are linked. This is a performance bottleneck in the record linkage process since it potentially requires comparison of all record pairs/sets using expensive comparison functions [9, 33]. Due to the increasing size of Big Data (*volume*), comparing all records is not feasible in most real-world applications. Blocking and filtering techniques have been used to overcome this challenge by eliminating as many comparisons between non-matching records as possible [9, 29, 149].
2. **Linkage quality:** The emergence of Big Data brings with it the challenge of dealing with typographical errors and other variations in data (*variety* and *veracity*) making the linkage more challenging. The exact matching of QID values, which would classify pairs or sets of records as matches if their QIDs are exactly the same and as non-matches otherwise, will likely lead to low linkage accuracy in the presence of real-world data errors. In addition, the classification models used in record linkage should be effective and accurate in classifying matches and non-matches [33]. Therefore, for practical record linkage applications, techniques are required that facilitate both approximate matching of QID values for comparison, as well as effective classification of record pairs/sets for high linkage accuracy.
3. **Privacy:** The privacy-preserving requirement in the record linkage process adds a third challenge, privacy, to the two main challenges of scalability and linkage quality [165]. Linking Big Data containing massive amounts of personal data generally involves privacy and confidentiality issues. Privacy needs to be considered in all steps of the record linkage process as only the masked (or encoded) records can be used, making the task of linking databases across organizations more challenging. Several masking techniques have been used for PPRL, as we will discuss in detail in Sect. 3.4.

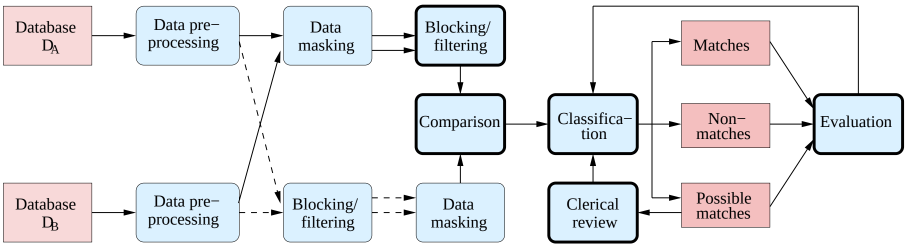

Fig. 1. Outline of the general PPRL process as discussed in Sect. 2.2. The steps shown in dark outlined boxes need to be conducted on masked database records, while dotted arrows show alternative data flows between steps.

2.2 The PPRL Process and Techniques Used

In this section we discuss the steps and the techniques used in the PPRL process, as shown in Fig. 1.

Data Pre-processing and Masking: The first important step for quality linkage outcomes is data pre-processing. Real-world data are often noisy, incomplete and inconsistent [8, 128], and they need to be cleaned in this step by filling in missing data, removing unwanted values, transforming data into well-defined and consistent forms, and resolving inconsistencies in data representations and encodings [28, 36]. In PPRL, data masking (encoding) is an additional step. Data pre-processing and masking can be conducted independently at each data source. However, it is important that all database owners (or parties) who participate in a PPRL project conduct the same data pre-processing and masking steps on the data they will use for linking. Some exchange of information between the parties about what data pre-processing and masking approaches they use, as well as which attributes to be used as QIDs, is therefore required [165].

Blocking/filtering: Blocking/filtering is the second step of the process, which is aimed at reducing the number of comparisons that need to be conducted between records by pruning as many pairs or sets of records as possible that unlikely correspond to matches [9, 29]. Blocking groups records according to a blocking criteria (blocking key) such that comparisons are limited to records in the same (or similar) blocks, while filtering prunes potential non-matches based on their properties (e.g. length differences of QIDs) [29]. The output of this step are candidate record pairs (or sets) that contain records that are potentially matching, which need to be compared in more detail. Blocking/filtering can either be conducted on masked records or locally by the database owners on unmasked records. The scalability challenge of PPRL has been addressed by several recent approaches using private blocking and filtering techniques [46, 79, 131, 133, 149, 150, 159, 164], as will be described in Sects. 4 and 5.1.

Comparison: Candidate record pairs (or sets) are compared in detail in the comparison step using comparison (or similarity) functions [31]. Various comparison functions have been used in record linkage including Levenshtein edit distance, Jaro-Winkler comparison, Soft-TFIDF string comparison, and set-based comparison using the Overlap, Dice, or Jaccard coefficient [28]. These comparison functions provide a numerical value representing the similarity of the compared QID values, often normalized into the $[0, 1]$ interval where a similarity of 1 corresponds to two values being exactly the same, and 0 means two values being completely different. Several QIDs are normally used when comparing records, resulting in one weight vector for each compared record pair that contains the numerical similarity values of all compared QIDs.

The QIDs of records often contain variations and errors, and therefore simply masking these values with a secure one-way hash-encoding function (as will be described in Sect. 3.4) and comparing the masked values will not result in high linkage quality for PPRL [35, 122]. A small variation in a pair of QID values will lead to completely different hash-encoded values [35], which enables only exactly matching QID values to be identified with such a simple masking approach. Therefore, an effective masking approach for securely and accurately calculating the approximate similarity of QID values is required. Several approximate comparison functions have been adapted into a PPRL context, including the Levenshtein edit distance [77] and the Overlap, Dice, or Jaccard coefficients [165].

Classification: In the classification step, the weight vectors of the compared candidate record pairs (or sets) are given as input to a decision model which will classify them into matches, non-matches, and possible matches [33, 54, 64], where the latter class is for cases where the classification model cannot make a clear decision. A widely used classification approach for record linkage is the probabilistic method developed by Fellegi and Sunter in the 1960s [54]. In this model, the likelihood that a pair (or set) of records corresponds to a match or a non-match is modelled based on a-priori error estimates on the data, frequency distributions of QID values, as well as their similarity calculated in the comparison step [28]. Other classification techniques include simple threshold-based and rule-based classifiers [28]. Most PPRL techniques developed so far employ a simple threshold-based classification [165].

Supervised machine learning approaches, such as support vector machines and decision trees [14, 25, 51, 52], can be used for more effective and accurate classification results. These require training data with known class labels for matches and non-matches to train the decision model. Once trained, the model can be used to classify the remaining unlabelled pairs/sets of records. Such training data, however, are often not available in real record linkage applications, especially in privacy-preserving settings [28]. Alternatively, semi-supervised techniques (such as active learning-based techniques [3, 11, 169]), that actively use examples manually labeled by experts to train and improve the decision model, need to be developed for PPRL. Recently developed advanced classification models, such as (a) collective linkage [13, 75] that considers relational similarities with other records in addition to QID similarities, (b) group linkage [123] that calculates group of records' similarities based on pair-wise similarities, and (c) graph-based linkage [59, 67, 75] that considers the structure between groups of records, can achieve high linkage quality at the cost of higher computational complexity. However, these advanced classification techniques have not been explored for PPRL so far.

Clerical review: The record pairs/sets that are classified as possible matches require a clerical review process, where these pairs are manually assessed and classified into matches or non-matches [171]. This is usually a time-consuming and error-prone process which depends upon experience of the experts who conduct the review. Active learning-based approaches can be used for clerical review [3, 11, 169]. However, clerical review in its current form is not possible in a PPRL scenario since the actual QID values of records cannot be inspected because this would reveal sensitive private information. Recent work in PPRL suggests an interactive approach with human-machine interaction to improve the quality of linkage results without compromising privacy [101].

Evaluation: The final step in the process is the evaluation of the complexity, quality, and privacy of the linkage to measure the applicability of a linkage project in an application before implementing it into an operational system. A variety of evaluation measures have been proposed [29, 33]. Given in a practical record linkage application the true match status of the compared record pairs are unlikely to be known, measuring linkage quality is difficult [6, 33]. How to evaluate the amount of privacy protection using a set of standard measures is still an immature aspect in the PPRL literature. Vatsalan et al. recently proposed a set of evaluation measures for privacy using probability of suspicion [162]. Entropy and information gain, between unmasked and masked QID values, have also been used as privacy measures [46].

Tools: Different record linkage approaches have been implemented within a number of tools. Koepcke and Rahm provided a detailed overview of eleven such tools in [96] both categories of with and without the use of learning-based (supervised) classification. The comparative evaluation study [97] benchmarks selected tools from both categories for four real-life test cases. It is found that learning-based approaches achieve generally better linkage quality especially for complex tasks requiring the combination of several attribute similarities. Current tools for *link discovery*, i.e., matching entities between sources of linked open data web, are surveyed in [120]. A web-based tool was recently developed to demonstrate several multi-party PPRL approaches (as will be described in Section 5) [132].

3 Privacy Aspects and Techniques for PPRL

Several dimensions of privacy need to be considered for PPRL, the four main ones being: (1) the number of parties and their roles, (2) adversary models, (3) privacy attacks, and (4) data masking or encoding techniques. In Sects. 3.1 - 3.4, we describe these four privacy dimensions, and in Sect. 3.5 we provide an overview of Bloom filter-based data masking, a technique widely used in PPRL.

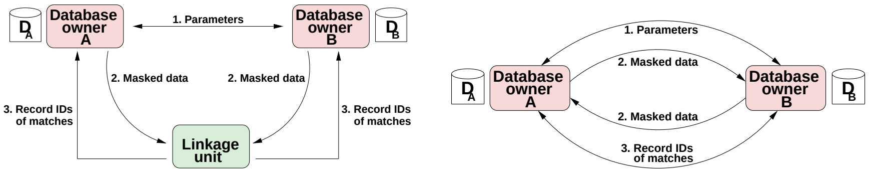

Fig. 2. Outline of PPRL protocols with (left) and without (right) a linkage unit (also known as three-party and two-party protocols, respectively).

3.1 PPRL Scenarios

PPRL techniques for linking two databases can be classified into those that require a linkage unit for performing the linkage and those that do not. The former are known as 'three-party protocols' and the latter as 'two-party protocols' [24, 27, 167]. In three-party protocols, a (trusted) third party acts as the linkage unit to conduct the linkage of masked data received from the two database owners, while in two-party protocols only the two database owners participate in the PPRL process. A conceptual illustration and the main communication steps involved in these protocols are shown in Fig. 2.

A further characterization of PPRL techniques is if they allow the linking of data from more than two data sources (multi-party) or not. Multi-party PPRL techniques identify matching record sets (instead of record pairs) from all parties (more than two) involved in a linkage, or from sub-sets of parties. Only limited work has been so far conducted on multi-party PPRL due to its increased challenges, as we will describe in Sect. 5. Similar to linking two databases, multi-party PPRL may or may not use a linkage unit to perform the linkage.

Protocols that do not require a linkage unit are more secure in terms of collusion (described below) between one of the database owners and the linkage unit. However, they generally require more complex techniques to ensure that the database owners cannot infer any sensitive information about each others' data during the linkage process.

3.2 Adversary Models

Different adversary models are assumed in PPRL techniques, including the most commonly used honest-but-curious (HBC) and malicious models [165].

1. **Honest-but-curious (HBC) or semi-honest** parties are curious in that they try to find out as much as possible about another party's input to a protocol while following the protocol steps [66, 112]. If all parties involved in a PPRL protocol have no new knowledge at the end of the protocol above what they would have learned from the output, which is generally the record pairs (certain attributes) classified as matches, then the protocol is considered to be secure in the HBC model. However, it is important to note that HBC does not prevent parties from colluding with each other with the aim to

learn about another party's sensitive information [112]. Most of the existing PPRL solutions assume the HBC adversary model.

2. **Malicious** parties can behave arbitrarily in terms of refusing to participate in a protocol, not following the protocol in the specified way, choosing arbitrary values for their data input, or aborting the protocol at any time [111]. Limited work has been done in PPRL for the malicious adversary model [58, 106, 119]. Evaluating privacy under this model is very difficult, because there exist potentially unpredictable ways for malicious parties to deviate from the protocol that cannot be identified by an observer [21, 63, 112].
3. **Covert and accountable computing** models are advanced adversary models developed to overcome the problems associated with the HBC and malicious models. The HBC model is not sufficient in many real-world applications because it is suitable only when the parties essentially trust each other. On the other hand, the solutions that can be used with malicious adversaries are generally more complex and have high computation and communication complexities, making their applications not scalable to large databases. The covert model guarantees that the honest parties can detect the misbehavior of an adversary with high probability [4], while the accountable computing model provides accountability for privacy compromises by the adversaries without excessive complexity and cost that incur with the malicious model [73]. Research is required towards transforming existing HBC or malicious PPRL protocols into these models and proving privacy of solutions under these models.

3.3 Attacks

The privacy attacks or vulnerabilities that a PPRL technique is susceptible to allow theoretical and empirical analysis of the privacy guarantees provided by a PPRL technique. The main privacy attacks of PPRL are:

1. **Dictionary attack** is possible with masking functions, where an adversary masks a large list of known values using various existing masking functions until a matching masked value is identified [165]. A keyed masking approach, such as the Hashed Message Authentication Code (HMAC) [98], can be used to prevent dictionary attacks. With HMAC the database owners exchange a secret code (string) that is added to all database values before they are masked. Without knowing the secret key, a dictionary attack is unlikely to be successful.
2. **Frequency attack** is still possible even with a keyed masking approach [165], where the frequency distribution of a set of masked values is matched with the distribution of known unmasked values in order to infer the original values of the masked values [113].
3. **Cryptanalysis attack** is a special category of frequency attack that is applicable to Bloom filter-based data masking techniques. As Kuzu et al. [102] have shown, depending upon certain parameters of Bloom filter masking, such as the number of hash functions employed and the number of bits in

a Bloom filter, using a constrained satisfaction solver allows the iterative mapping of individual masked values back to their original values.

4. **Composition attack** can be successful by combining knowledge from more than one independent masked datasets to learn sensitive values of certain records [61]. An attack on distance-preserving perturbation techniques [155], for example, allows the original values to be re-identified with high level of confidence if knowledge about mutual distances between values is available.
5. **Collusion** is another vulnerability associated with multi-party or three-party PPRL techniques, where some of the parties involved in the protocol work together to find out about another database owner's data. For example, one or several database owners collude with the linkage unit or a sub-set of database owners collude among them to learn about other parties' data.

3.4 Data Masking or Encoding

In PPRL, the linkage has to be conducted on a masked or encoded version of the QIDs to preserve the privacy of entities. Data masking (encoding) transforms original data in such a way that there exists a specific functional relationship between the original data and the masked data [55]. Several data masking functions have been used to preserve privacy while allowing the linkage. We categorize existing data masking techniques into three: (1) auxiliary, (2) blocking, and (3) matching techniques. Auxiliary techniques are the ones used as helper functions in PPRL, while blocking and matching categories are used for private blocking and matching (comparison and classification), respectively. In the following we describe key techniques in each of these three categories.

- **Auxiliary:**
  1. **Pseudo random function** (PRF) is a deterministic secure function that, when given an n-bit seed k and an n-bit argument x, returns an n-bit string $f_k(x)$ such that it is infeasible to distinguish $f_k(x)$ for different random k from a truly random function [115]. In PPRL, PRFs have been used to generate random secret values to be shared by a group of parties [58, 122, 151].
  2. **Reference values** constructed either with random faked values, or values that for example are taken from a public telephone directory, such as all unique surnames and town names, have been used in several PPRL approaches [78, 124, 141, 173]. Such lists of reference values can be used to calculate the distances or similarities between QID values in terms of the distances or similarities between QID and reference values.
  3. **Noise addition** in the form of extra records or QID values that are added to the databases to be linked is a data perturbation technique [87] that can be used to overcome the problem of frequency attacks on PPRL protocols [44, 101]. An example is shown in Fig. 3. Adding extra records, however, incurs a cost of lower linkage quality (due to false matches) and scalability (due to extra records that need to be processed and linked) [80]. Section 4.1 discusses noise addition for private blocking.

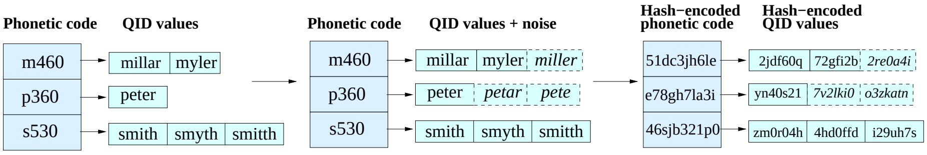

Fig. 3. An example of phonetic encoding, noise addition, and secure hash-encoding (adapted from [165]). Values represented with dotted outlines are added noise to overcome frequency attacks [162] (as will be discussed in detail in Section 4.1).

4. **Differential privacy** [50] has emerged as an alternative to random noise addition in PPRL. Only the perturbed results (with noise) of a set of statistical queries are disclosed to other parties, such that the probability of holding any property on the results is approximately the same whether or not an individual value is present in the database [50]. In recent times, differential privacy has been used in statistical (e.g. counts or frequencies) microdata publication as well as in PPRL [16, 69, 104].

– **Blocking:**
  1. **Phonetic encoding**, such as Soundex, NYSIIS or Double-Metaphone, groups values together that have a similar pronunciation [23] in a one-to-many mapping, as shown in Fig. 3. The main advantage of using a phonetic encoding for PPRL is that it inherently provides privacy [80], reduces the number of comparisons, and thus increases scalability [23], and supports approximate matching [23, 80]. Two drawbacks of phonetic encodings are that they are language dependent [127, 146] and are vulnerable to frequency attacks [162]. Section 4.1 discusses phonetic encoding-based blocking in more details.
  2. **Generalization techniques** overcome the problem of frequency attacks on records by generalizing records in such a way that re-identification of individual records from the masked data is not feasible [107, 116, 152]. k-anonymity is a widely used generalization technique for PPRL [76, 78, 119], where a database is said to be k-anonymous if every combination of masked QID values (or blocking key values) is shared by at least k records in the database [152]. Other generalization techniques include value generalization hierarchies [68], top-down specialization [119], and binning [109, 163].

– **Matching:**
  1. **Secure hash-encoding** is one of the first techniques used for PPRL [17, 49, 126]. The widely known Message Digest (like MD5) and Secure Hash Algorithms (like SHA-1 and SHA-2) [143] are one-way hash-encoding functions [98, 143] that can be used to mask values into hash-codes (as shown in Fig. 3) such that having access to only hash-codes will make it nearly impossible with current computing technology to infer their original values. A major problem with this masking technique is, however, that only exact matches can be found [49]. Even a single character

difference between a pair of original values will result in two completely different hash-codes.

2. **Statistical linkage key (SLK)** is a derived variable generated from components of QIDs. The SLK-581 was developed by the Australian Institute of Health and Welfare (AIHW) to link records from the Home and Community Care datasets [140]. The SLK-581 consists of the second and third letters of first name, the second, third and fifth letters of surname, full date of birth, and sex. Similarly, SLK consisting of month and year of birth, sex, zipcode, and initial of first name was used for linking the Belgian national cancer registry [157]. However, as a recent study has shown these SLK-based masking provides limited privacy protection and poor sensitivity [135].
3. **Embedding space** embeds QID values into a multi-dimensional metric space (such as Euclidean [16, 141, 173] or Hamming [82]) while preserving the distances between these values using a set of pivot values that span the multi-dimensional space. A drawback with this approach is that it is often difficult to determine a good dimension of the metric space and select suitable pivot values.
4. **Encryption schemes**, such as commutative [1] and homomorphic [93] encryption, are used in PPRL techniques to allow secure multi-party computation (SMC) in such a way that at the end of the computation no party knows anything except its own input and the final results of the computation [38, 63, 112]. The secure set union, secure set intersection, and secure scalar product, are the most commonly used SMC techniques for PPRL [38, 143]. A drawback of these cryptographic encryption schemes for SMC, however, is that they are computationally expensive.
5. **Bloom filter** is a bit vector data structure into which values are mapped by using a set of hash functions. Bloom filters have been widely used in PPRL for private matching of records as they provide a means of privacy assurance [46, 47, 77, 105, 147, 170], if effectively used [103]. We will discuss Bloom filter masking in more detail in the following section.
6. **Count-min sketches** are probabilistic data structures (similar to Bloom filters) that can be used to hash-map values along with their frequencies in a sub-linear space [41]. Count-min sketches have been used in PPRL where the frequency of occurrences of a matching pair/set also needs to be identified [81, 139]. However, these approaches only support exact matching of categorical values.

Other privacy aspects in a PPRL project are public / private key pairs that are required for secure generation and exchange of keys, employee confidentiality agreements to reduce internal threats, as well as encrypted communication, secure connections, and secure servers to reduce external threats.

3.5 Bloom filters

Bloom filter encoding has been used as an efficient masking technique in a variety of PPRL approaches [46, 105, 130, 148, 149, 158, 160]. A Bloom filter b_i is

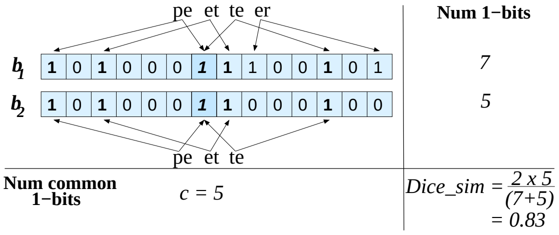
pe et te er
Num 1−bits
b_1 1 0 1 0 0 0 1 1 1 0 0 1 0 1
7
b_2 1 0 1 0 0 0 1 1 0 0 0 1 0 0
5
pe et te
Num common 1−bits
c = 5
Dice_sim = 2 x 5 / (7+5) = 0.83

Fig. 4. An example similarity (Dice-coefficient) calculation of Bloom filters for approximate matching using Schnell et al.'s approach [147] (taken from [165]).

a bit vector data structure of length l bits where all bits are initially set to 0. k independent hash functions, $h_1, h_2, \ldots, h_k$, each with range $1, \ldots l$, are used to map each of the elements s in a set S into the Bloom filter by setting the bit positions $h_1(s), h_2(s), \ldots, h_k(s)$ to 1. The Bloom filter was originally proposed by Bloom [15] for efficiently checking set membership [18]. Lai et al. [105] first adopted the concept of using Bloom filters in PPRL for identifying exactly matching records across multiple databases, as will be described in Sect. 5.2.

Schnell et al. [148] were the first to propose a method for approximate matching in PPRL using Bloom filters. In their approach, the character q-grams (substrings of length q) of QID values of each record in the databases to be linked are hash-mapped into a Bloom filter using k independent hash functions. The resulting Bloom filters are sent to a linkage unit that calculates the similarity of Bloom filters using a set-based similarity function, such as the Dice-coefficient [28]. The Dice-coefficient of two Bloom filters (b_1 and b_2) is calculated as:

$$Dice\_sim(b_1, b_2) = \frac{2 \times c}{(x_1 + x_2)}, \tag{1}$$

where c is the number of common bit positions that are set to 1 in both Bloom filters (common 1-bits), and x_i is the number of bit positions set to 1 in b_i (1-bits), $i \in \{1, 2\}$. An example similarity calculation is illustrated in Fig. 4.

Bloom filters are susceptible to cryptanalysis attacks, as shown by Kuzu et al. [102]. Using a constrained satisfaction solver, such attacks allow the iterative mapping of individual hash-encoded values back to their original values depending upon the number of hash functions employed and the length of a Bloom filter. Different Bloom filter encoding methods have been proposed in the literature to overcome such cryptanalysis attacks and improve linkage quality. Schnell et al.'s proposed method of hash-mapping all QID values of a record into one composite Bloom filter is known as Cryptographic Long-term Key (CLK) encoding [148].

Durham et al. [48] investigated composite Bloom filter encoding in detail by first hash-mapping different attributes into attribute-level Bloom filters of different lengths. These lengths depend upon the weights [54] of QID attributes that are calculated using the discriminatory power of attributes in separating matches from non-matches using a statistical approach. These attribute-level Bloom filters are then combined into one record-level Bloom filter (known as RBF) by sampling bits from each attribute-level Bloom filter. Vatsalan et al. [162] recently introduced a hybrid method of CLK and RBF (known as CLKRBF) where the Bloom filter length is kept to be the same (as in CLK) while using different numbers of hash functions to map different attributes into the Bloom filter based on their weights (to improve matching quality as in RBF).

Several non-linkage unit-based approaches have also been proposed for PPRL using Bloom filter masking, where the database owners (without a linkage unit) collaboratively (or distributively) calculate the similarity of Bloom filters [105, 158, 160]. A recent work proposed novel Bloom filter-based masking techniques that allow approximate matching of numerical data in PPRL [161]. Instead of hash-mapping *q*-grams of a string, the proposed approaches hash-map a set of neighbouring numerical values to allow approximate matching.

4 Scalability Techniques for PPRL

PPRL for Big Data needs to scale to very large data volumes of many millions of records from multiple sources. As for standard record linkage, the main techniques for high efficiency are to reduce the search space by blocking and filtering approaches and to perform record linkage in parallel on many processors.

These three kinds of optimization are largely orthogonal so that they may be combined to achieve maximal efficiency. Blocking is defined on selected attributes (blocking keys) that may be different from the QID attributes used for comparison, for example zip code. It partitions the records in a database into several blocks or clusters such that comparison can be restricted to the records of the same block, for example persons with the same zip code. Other blocking approaches like sorted neighborhood work differently but are similar in spirit. Filtering is an optimization for the particular comparison approach which optimizes the evaluation of a specific similarity measure for a predefined similarity threshold to be met by matching records. It thus utilizes different filtering or indexing techniques to eliminate pairs (or sets) of records that cannot meet the similarity threshold for the selected similarity measures [28, 43]. Such techniques can be applied for comparison within blocks, i.e., filtering could be utilized in addition to blocking.

In the next two subsections, we discuss several proposed blocking approaches and filtering approaches for PPRL. We then briefly discuss parallel PPRL which has found only limited attention so far. We will focus on PPRL with two data sources (multi-party PPRL is discussed in Sect. 5). We will furthermore mostly assume the use of a dedicated linkage unit (as shown on the left-hand side in Fig. 2) as well as the masking of records using Bloom filters (as described in Sect. 3.5). Note that a linkage unit is ideally suited for high performance as it requires minimal communication between the database owners, and the linkage unit can utilize a high performance cluster for parallel PPRL as well as blocking and filtering techniques.

4.1 Blocking Techniques

Blocking aims at reducing the search space for linkage by avoiding the comparison between every pair of records and its associated quadratic complexity. There are numerous blocking strategies [28] that mostly group records into disjoint or overlapping blocks such that only records within the same block need to be compared with each other. *Standard blocking* uses the values of a so-called blocking key to partition all records into disjoint blocks. The blocking key values (BKVs) may be the values of a selected attribute or the result of a function on one or several attribute values (e.g. the concatenation of the first two letters of last name and year of birth). Other blocking approaches include *canopy clustering* that results in overlapping clusters or blocks, and *sorted neighborhood* that sorts records according to a sorting key and only compares neighboring records within a certain window [28]. Comparing records only within the predetermined blocks may result in the loss of some matches especially if some BKVs are incorrect or missing. To improve recall a multi-pass blocking approach can be utilized, where records are blocked according to different blocking keys, at the cost of an increased number of additional comparisons.

Blocking for PPRL is based on the known approaches for regular record linkage but aims at improving privacy. A general approach with a central linkage unit is to apply a previously agreed on blocking strategy at the database owners on the original records. Then all records within the different blocks are masked (encoded), e.g. using Bloom filters, and sent to the linkage unit. The linkage unit can then compare the masked records block-wise with each other. In the following, we present selected blocking approaches for PPRL and discuss results from a comparative evaluation of different blocking schemes.

Phonetic Blocking: Blocking records based on their phonetic code is a widely used technique in record linkage [28]. The basic idea is to encode the values of a blocking key attribute (e.g. last name) with a phonetic function (as discussed in Sect. 3.4) such as Soundex or Metaphone [28]. All records with the same phonetic code, i.e. with a similar pronunciation, are then assigned to the same block. The use of phonetic blocking for PPRL has been used in several PPRL approaches, in particular in [77, 80]. Karakasidis et al. in [77] use a multi-pass approach with both Soundex and Metaphone encodings to achieve a good recall. Furthermore, they add fake records to the blocks for improved privacy.

As discussed in Sect. 3.4, adding fake records improves privacy but adds overhead in the form of extra comparisons between records and can reduce linkage quality due to the introduction of false matches. A theoretical analysis of the impact of adding fake records for Soundex-based blocking is presented in [80]. The authors study the effect of fake records on the so-called *relative information gain* which is related to the *entropy* measure. A high entropy within blocks caused by fake records introduces a high uncertainty and thus a low information gain [162]. The authors also study different methods to inject fake records to increase entropy. The most flexible of the approaches is based on the concept of k-anonymity and adds as many fake records that are required to ensure that each block has at least k records. The approach typically requires only the addition of a modest number of fake records; the choice of k also supports finding a good trade-off between privacy and quality.

Blocking with Reference Values: An alternative to adding fake records for improving the privacy of blocking is the use of reference values (as discussed in Sect. 3.4). The reference values can be used at the database owners for clustering their database records. Comparison can then be restricted to the clusters (blocks) of the same or similar reference records. Such an approach has been proposed in [78] based on *k nearest neighbor (kNN) clustering.* This approach first clusters the reference values identically at each database owner such that each cluster contains at least k reference values to ensure k-anonymity; clustering is based on the Dice-coefficient similarity between values. In the next step, each database owner assigns its records to the nearest cluster based on the Dice-coefficient between records and reference values. Finally each database owner sends its clusters (encoded reference values and records) to the linkage unit which then compares the records between corresponding clusters.

An alternate proposal utilizes a local *sorted neighborhood clustering* (*SNC-3P*) for improved performance in the blocking phase while retaining the use of reference values and support for k-anonymity [159]. Each database owner sorts a shared set of reference values and then inserts its records into the sorted list according to their sorting key. From the sorted list of reference values and records the initial Sorted Neighborhood Clusters (SNCs) are determined such that each cluster contains one reference value and a set of database records. To ensure k-anonymity, the initial clusters are merged into larger blocks containing at least k database records. This differs from kNN clustering where k reference records are needed per cluster. The merging of the initial clusters can be based on similarity or size constraints. The remaining protocol with sending the encoded records to the linkage unit for comparison works as for kNN clustering.

An adaptation of SNC-3P for two parties without a linkage unit (*SNC-2P*) was presented in [164]. In this approach, the two database owners generate their reference values independently, so that they end up with two different sets of reference values. As for SNC-3P, each database owner sorts its reference values, inserts its records into the sorted list, builds initial SNCs (with one reference value and its associated records), and merges these clusters to guarantee k-anonymity. Afterwards the database owners exchange their reference values. These values are then merged with the local reference values and sorted. To find candidate pairs between the sources a sorted neighborhood method with a sliding window w is applied on these reference values. The window size w determines the number of reference values originating from each data source, e.g. for $w = 2$ the sliding window includes 2 reference values from each data source. In the last step, the encoded records falling into a window are exchanged for comparison.

LSH-based blocking: Locality-sensitive hashing (LSH) has been proposed to solve the problem of nearest neighbor search in high dimensional spaces [62, 70]. For blocking, LSH uses a family of hash functions to generate keys used to partition the records in a database, so that similar records are grouped into the same block [90]. Durham investigated the use of LSH for private blocking of records masked as Bloom filters [46]. She considered two families of hash functions depending on the used distance function (Jaccard or Hamming distance). For the Jaccard distance, she proposed the use of *MinHash* functions. A MinHash function h_i permutes the bits of a Bloom filter b_i and selects from the permuted Bloom filter the first index position with a set bit (1-bit). By applying ϕ MinHash functions we obtain ϕ index positions which are concatenated to generate the final MinHash key. So for Bloom filter b_i and the family of hash functions H, we determine $key(b_i)_H = concat(h_1(b_i), h_2(b_i), \cdots, h_\phi(b_i))$, where $h_i \in H$ with $1 \leq i \leq \phi$ and function $concat()$ concatenates a set of input values. For the Hamming distance, Durham proposed the use of *HLSH* hash functions that select the bit value of a Bloom filter at a random position ρ. In the same way as MinHash, ϕ HLSH functions are applied on a Bloom filter b_i and the values of the ϕ selected bits are concatenated to obtain the final hash key of b_i.

Example: Consider two Bloom filters $b_1 = 1100100011$ and $b_2 = 1100100111$, two permutations $p_1 = (10, 7, 1, 3, 6, 4, 2, 9, 5, 8)$ and $p_2 = (4, 3, 6, 7, 2, 1, 5, 10, 9, 8)$, and the MinHash family H_1 with two functions $h_1 = Min(p_1(\cdot))$ and $h_2 = Min(p_2(\cdot))$, where $Min(\cdot)$ returns the first position of a set bit in the input bit vector, and $p_i(\cdot)$ returns the input bit vector permuted using p_i. The application of h_1 and h_2 on b_1 results in $h_1(b_1) = Min(p_1(b_1)) = Min(1010001110) = 1$ and $h_2(b_1) = Min(p_2(b_1)) = Min(0000111110) = 5$. Hence the key of b_1 in H_1 is $key(b_1)_{H_1} = concat(h_1(b_1), h_2(b_1)) = (1, 5)$. In the same way we determine the key of b_2, i.e. $key(b_2)_{H_1} = (1, 5)$. Hence, for MinHash family H_1 records b_1 and b_2 are put into the same block and will be compared with each other.

Both families, MinHash and HLSH, depend on two parameters: the number of hash functions ϕ as well as the number of passes or iterations μ. Since the final hash key of a record concatenates ϕ values, using a high number ϕ leads to more homogeneous blocks and better precision (i.e., blocks containing similar records with higher probability). However a high ϕ also increases the probability of missing true matches (reduced recall). This problem is addressed by applying μ iterations, each with a different set of hash functions. Therefore each record b_i will be hashed to several blocks to allow identifying more true matches.

In [84] the authors present a theoretical analysis of the use of MinHash functions to identify good values for parameters ϕ_{opt} and μ_{opt} to efficiently achieve a good precision and recall. The naïve approach to improve recall is to increase the number of iterations μ and thus the number of blocks to which records are assigned. The drawbacks of this method are the high runtime caused by the computation of the permutations, increased number of record pairs to compare, and the large space needed to store intermediate results. This observation was experimentally confirmed in [46]. The choice of the ϕ_{opt} is also complex and depends on the expected running time (for details see [84]).

Evaluation of Private Blocking Approaches: The relatively large number of possible blocking approaches requires detailed evaluations regarding their relative scalability, blocking quality and privacy for different kinds of workloads. One of the few studies in this respect has been presented by Vatsalan et al. [162]. For scalability they considered runtime and the so-called *reduction ratio* (RR), a value indicating the number of pruned candidate pairs compared to all possible record pairs (which thus evaluates to what degree the search space is reduced). For blocking quality they considered the recall and precision metrics *pair completeness* (PC) and *pair quality* (PQ), respectively [29]. For privacy they estimated the so-called *disclosure risk* (DR) measures, which represent the probability that masked records/QID values can be linked with records or values in a publicly available dataset.

The evaluation of [162] considers six simulated blocking strategies including kNN [78], SNC-3P [159], SNC-2P [164] and LSH blocking [46]. Regarding blocking runtime, the SNC and LSH schemes performed best. All strategies except SNC-2P achieved a very high RR of almost 1. On the other hand, SNC-2P achieved the best PC. The best trade-off between RR and PC was observed for LSH. Considering the privacy aspect, SNC-2P was found to have a low DR while kNN and LSH generally expose the highest DR.

While this study provides interesting results, we see a need for additional benchmark studies given that further blocking schemes have been developed more recently and that the relative behavior of each approach depends on numerous parameter settings as well as characteristics of the chosen datasets.

4.2 Filtering Techniques

Almost all proposed PPRL schemes based on Bloom filters aim at identifying the pairs of bit vectors with a similarity above a threshold. For regular record linkage, such a threshold-based computation of comparing record pairs is known as a *similarity join* [39]. The efficient processing of such similarity joins for different kinds of similarity measures has been the focus of much research in the past, e.g. [2, 65, 138, 172]. Several approaches utilize the characteristics of the considered similarity measure and the prespecified similarity threshold to reduce the search space thereby speeding up the linkage process. This holds especially for the broad class of token-based similarity joins where the comparison of records is based on the set of tokens (e.g. q-grams) of QIDs. In this case, one can exclude all pairs of records that do not share at least one token. Further proposed optimizations for such similarity joins include the use of different kinds of filters (for example, length and prefix filters) and dynamically created inverted indexes [10]. PPJoin [172] is an efficient approach that includes these kinds of optimizations. Several filtering approaches also utilize the characteristics of similarity functions for metric spaces to reduce the search space, in particular the so-called triangle inequality (see below) [174].

For PPRL, similar filtering (similarity join) approaches are usable but need to be adapted to the comparison of masked records such as Bloom filters and the associated similarity measures. For Bloom filters it is relatively easy to apply the known token-based similarity measures by considering the set bit positions (1-bits) in the bit vectors as the “tokens”. This has already been shown in Sect. 3.5 for the Dice-coefficient similarity which is based on the degree of overlapping bit positions. This is also the case for the related Jaccard similarity. For two bit vectors b_1 and b_2 it is defined as follows:

$$Jacc\_sim(b_1, b_2) = \frac{|b_1 \wedge b_2|}{|b_1 \vee b_2|} = \frac{|b_1 \wedge b_2|}{|b_1| + |b_2| - |b_1 \wedge b_2|}, \tag{2}$$

where $|b_i|$ denotes the number of set bits in bit vector b_i which is also called its *length* or cardinality. For the example Bloom filter pair shown in Fig. 4, the Jaccard similarity is $5/7 = 0.71$. In the following, we outline several filtering approaches that have been proposed for PPRL.

Length Filter: The similarity function $Jacc\_sim$ (as well as $Dice\_sim$) allows the application of a simple *length filter* to reduce the search space. This is because the minimal similarity (overlap of set bits) can only be achieved if the lengths (number of set bits, i.e. common 1-bits) of the two input records do not deviate too much. Formally, for two records r_i and r_j with $|r_i| \leq |r_j|$, it holds that

$$Jacc\_sim(r_i, r_j) \geq s_t \Rightarrow |r_i| \geq \lceil s_t \cdot |r_j| \rceil \tag{3}$$

For example, two records cannot satisfy a (Jaccard) similarity threshold $s_t = 0.8$ if their lengths differ by more than 20%. Hence for a similarity threshold of 0.8, the length filter would already avoid the two records from the example Bloom filter pair shown in Fig. 4 without comparing in detail, since Eq. 3 ($5 \geq \lceil 0.8 \cdot 7 \rceil = 6$) does not hold. The two-party PPRL approach proposed by Vatsalan and Christen uses such a length filter for Dice-coefficient similarity [158].

PPJoin for PPRL: The privacy-preserving version of PPJoin (called P4Join) utilizes three filters to reduce the search space: the length filter as well as a prefix filter and a position filter [149]. The *prefix filter* is based on the fact that matching bit vectors need a high degree of overlap in their set bit positions in order to satisfy a predefined threshold. Pairs of records can thus be excluded from comparison if they have an insufficient overlap. This overlap test can be limited to a relatively small subset of bit positions, e.g. in the beginning (or prefix) of the vectors. To maximize this filter idea, P4Join applies a pre-processing to count for each bit position the number of records where this bit is set to 1 and by reordering the positions of all bit vectors in ascending order of these frequency counts. This way the prefixes of bit vectors contain infrequently set bit positions reducing the likelihood of an overlap with other bit vectors. The *position filter* of P4Join can avoid the comparison of two records even if their prefixes overlap depending on the prefix positions where the overlap occurs. For more details of this filter we refer to [149].

As we will see in the comparative evaluation below, the filtering approaches achieve only a relatively small improvement for PPRL since the filter tests imply already a certain overhead which is not much less than for the match tests (which are relatively cheap for Bloom filters). In addition, Bloom filter masking for PPRL should ideally have 50% of their bits set to 1 in order to make them less vulnerable to frequency attacks [118], making P4Join less effective.

Multi-bit Trees: The use of multi-bit trees was originally proposed for fast similarity search in large databases of chemical *fingerprints* (masked into Bloom filters) [99]. A query Bloom filter b_q is being searched for in a database to retrieve all elements whose similarity with b_q is above the threshold s_t. A multi-bit tree is a binary tree to iteratively assign fingerprints to its nodes based on so-called match bits. A match bit refers to a specific position of the bit vector and can be 1 or 0: it indicates that all fingerprints in the associated subtree share the specified match bit. When building up the multi-bit tree, one match bit or multiple such bits are selected in each step so that the number of unassigned fingerprints can be roughly split by half. The split is continued as long as the number of fingerprints per node does not fall under a limit ([99] recommends a limit of 6). The match bits can then be used for a query fingerprint to determine the maximal possible similarity for subtrees when traversing the tree and can thereby eliminate many fingerprints to compare.

As suggested in [5], multi-bit trees can easily be applied for PPRL using Bloom filters and Jaccard similarity. For two datasets, the larger input dataset is used to build the multi-bit trees while each record (fingerprint) of the second dataset is used for searching similar records. The multi-bit approach of [5] partitions the fingerprints according to their lengths such that all fingerprints with the same length belong to the same partition (or bucket). To apply the length filter, we can then restrict the search for similar fingerprints to the partitions meeting the length criterion of Eq. 3. Query efficiency is further improved by organizing all fingerprints of a partition within a multi-bit tree.

In evaluations of [5] and [145] the multi-bit tree approach was found to be very effective and even better or similarly effective than blocking approaches such as canopy clustering and sorted neighborhood.

PPRL for Metric Space Similarity Measures: A metric space consists of a set of data objects and a metric to compute the distance between the objects. The main property of interest that a metric or distance function d for metric spaces has to satisfy is the so-called triangle inequality. It requires that for any objects x, y and z it holds

$$d(x, z) \leq d(x, y) + d(y, z) \tag{4}$$

Distance functions for metric spaces satisfying this property include the Minkowski distances (for example, Euclidean distance), edit distance, Hamming distance and Jaccard-coefficient (but not Dice-coefficient) [174]. The triangle inequality has been used for private comparison and classification in PPRL using reference values [124, 163].

The triangle inequality has also been used to reduce the search space for similarity search and record linkage [7, 12]. In both cases we have to find for a query object q those similar objects x with a distance $d(q, x)$ lower than or equal to a maximal distance threshold (or above a minimal similarity threshold) which can be seen as a radius $rad(q)$ around q in Fig. 5. The triangle equality allows one to avoid computing the distance between two objects based on their precomputed distances to a third reference object or pivot, such as object p in Fig. 5. Utilizing the precomputed distances $d(p, q)$ and $d(p, x)$ we only have to compute the distance $d(q, x)$ for objects x that satisfy the triangle inequality

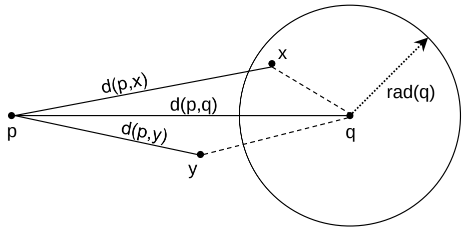

Fig. 5. Triangle inequality: Object y cannot lie within the search radius of query object q since the difference between $d(p,q)$ and $d(p,y)$ exceeds $rad(q)$ (taken from [150]).

$d(p,q) - d(p,x) \leq rad(q)$. In all other cases, comparison can be avoided such as for object y in Fig. 5.

Several alternatives to utilize the triangle inequality to reduce the search space for PPRL have been studied in [150], in particular for the Hamming distance which has been shown to be equivalent to the Jaccard similarity [172]. The best performance was achieved for a pivot-based approach that selects a certain number of data objects from a sample of the first dataset as pivots and assigns each other object of the first dataset to its closest pivot. For each pivot, the maximal distance (radius) for its objects is also recorded. Pivots are iteratively determined from the sample set of objects such that the object with the greatest distance to all previously determined pivots becomes the next pivot. The rational behind this selection strategy is to have a relatively large distance between pivots so that searching for similar objects can be restricted to the objects of relatively few pivots. Determining the pivots from a sample rather than from all objects limits the overhead of pivot selection. The search for similar (matching) objects can be restricted to the pivots for which there is a possible overlap with the radius of the query objects. For the objects of the relevant pivots the triangle inequality is further used to prune objects from the comparison.

Comparative Evaluation: The performance of pivot-based approaches for metric similarity measures has been evaluated in [150] and compared with the use of P4Join and multi-bit trees. The evaluation has been done for synthetically generated datasets of 100,000 to 500,000 records such that 80% of the records are in the first dataset and 20% in the second. Bloom filters of length 1,000 bits are used to mask the QIDs of records, and the comparison is based on a Jaccard similarity threshold of 0.8 or the corresponding Hamming distance for the metric-space approach.

Table 3 summarizes the runtimes of the different approaches as well as for a naïve nested loop approach without any filtering (all implemented using Java) on a standard PC (Intel i7-4770, 3.4GHz CPU with 16 GB main memory). The results show that both multi-bit trees and P4Join perform similarly but achieve only modest improvements (less than a factor 2) compared to the naïve nested loop scheme. By contrast the pivot-based metric space approach achieves order-of-magnitude improvements. For the largest dataset it only needs 1.7 minutes

Table 3. PPRL runtime in minutes for different dataset sizes and filtering approaches (taken from [150]).

Algorithms	Datasets				
	100,000	200,000	300,000	400,000	500,000
NestedLoop	3.8	20.8	52.1	96.8	152.6
Multi-bit Tree	2.6	11.3	26.5	50.0	75.9
P4Join	1.4	7.4	24.1	52.3	87.9
Pivots (Metric Space)	0.2	0.4	0.9	1.3	1.7

and is 40 times faster than using multi-bit trees. A general observation for all approaches is that the runtimes increase more than linearly (almost quadratically) with the size of datasets, indicating a potential scalability problem despite the reduction of the search space. Substantially larger datasets would thus create performance problems even for the best filtering approach indicating that additional runtime improvements are necessary, e.g. by the use of parallel PPRL.

4.3 Parallel PPRL

PPRL in Big Data applications involves the comparison of a large number of masked records as the main part of the overall execution pipeline. Parallel linkage on multiple processors aims at improving the execution time proportionally to the number of processors [42, 91, 92]. This can be achieved by partitioning the set of all record pairs to be compared, and conducting the comparison of the different partitions in parallel on different processors. A special case would be to utilize a blocking approach to compare the records in different blocks in parallel. In the following we discuss two approaches for parallel PPRL that have been proposed: one utilizes graphics processors or GPUs for parallel processing within a single machine, and the other one is based on Hadoop and its MapReduce framework. Both approaches have also been used for general record linkage.

Parallel PPRL with GPUs: The utilization of Graphical Processing Units (GPUs) to speed-up similarity computations is a comparatively new approach [57, 121]. Modern GPUs provide thousands of cores that allow for a massively-parallel application of the same instruction set to disjoint data partitions. The availability of frameworks like OpenCL and CUDA simplify the utilization of GPUs to parallelize general purpose algorithms. The GPU programs (called *kernels*) are typically written in a dialect of the general programming language C. Kernel execution requires the input and output data to be transferred between the main memory of the host system and the memory of the GPU, and it is important to minimize the amount of data to be transferred. Further limitations are that there is no dynamic memory allocation on GPUs (all resources required by a program need to be allocated a priori) and that only basic data types (e.g., int, long, float) and fixed-length data structures (e.g., arrays) can be used.

Despite such limitations, the utilization of GPUs is a promising approach to speed-up PPRL. This is especially the case for Bloom filter masking where

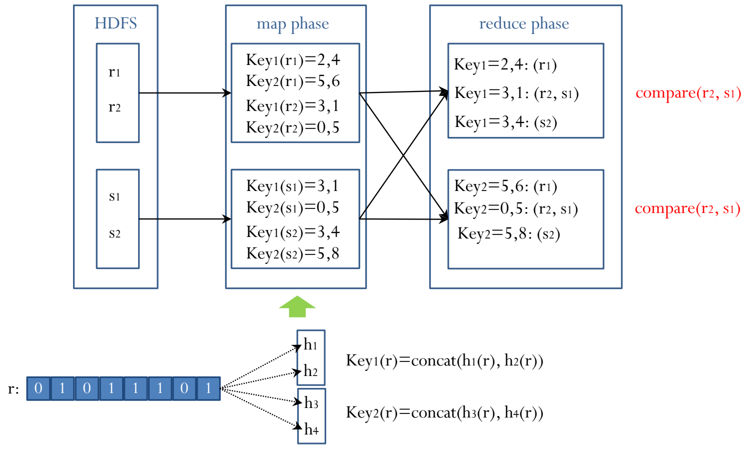

Fig. 6. Parallel PPRL with MapReduce using LSH blocking [83], where the MinHash keys for Bloom filters are computed in the Map phase and records with the same MinHash signature will be sent to the same Reduce task for matching.

all records are represented as bit vectors of equal length. These vectors can easily be stored in array data structures on the GPU. Furthermore, similarity computations can be broken down into simple bit operations which are easily processed by GPUs.

A GPU-based implementation for PPRL using the P4Join filtering is described in [149]. It sorts the bit vectors of the two input datasets initially according to their number of set bits (1-bits) and partitions the set of bit vectors into equal-sized blocks such that multiple of such blocks fit into the GPU memory. Pairs of blocks are then continuously loaded into the GPU for parallel comparison. To limit unnecessary data transfers, the length filter (described in Sect. 4.2) is applied to avoid transferring pairs of blocks that do not meet the length filter restriction. The kernel programs also apply the prefix filter to save comparisons.

The evaluation in [149] showed that the GPU implementation is highly efficient and improves runtimes by a factor of 20, even for a low-profile graphics card (Nvidia GeForce GT 540M with 96 CUDA cores@672MHz, 1GB memory). It would be interesting to realize GPU versions of other PPRL approaches and to utilize more powerful graphics cards with thousands of cores for improved performance.

Hadoop-based Parallel PPRL: Many Big Data applications are based on local Shared Nothing clusters driven by software from the open-source Hadoop ecosystem for parallel processing. Depending on the data volume and needed degree of parallelism up-to thousands of multi-processor nodes are utilized. A main reason for the success of Hadoop is that its programming frameworks, in particular MapReduce and newer platforms such as Apache Spark[6] or Apache

[6] http://spark.apache.org [Accessed: 15/06/2016]

Flink[7], make it easy to develop programs that can be automatically executed in parallel on Hadoop clusters.

Several approaches and implementations have utilized MapReduce for parallel record linkage [94, 166]. In its simplest form, the Map tasks read the input data in parallel and apply a blocking key to assign each record to a block. Then the data records are dynamically redistributed among the Reduce tasks such that all records with the same blocking key are sent to the same Reduce task. Comparison is then performed block-wise and in parallel by the Reduce tasks. For highly skewed block sizes this simple approach can result in load balancing problems for the Reduce tasks; approaches to solve this data skew or load balancing problem are proposed in [95].

The sketched approach can in principle also be applied for parallelizing PPRL, e.g., if the linkage unit utilizes a Hadoop cluster. One such approach for using MapReduce to speed-up PPRL has been proposed in [83]. The authors apply a LSH-based blocking method using the MinHash approach (see Section 4.1). The use of MapReduce is rather straightforward and illustrated in Figure 6. The Bloom filters of both sources are initially stored in the distributed file system (HDFS) as chunks. In the Map phase, records are read sequentially and for each Bloom Filter r a set of j MinHash keys are computed. The records are then redistributed so that records with the same key are sent to the same Reduce task for comparison. The main drawback of this strategy is that records may be compared several times by different Reduce tasks because they could share many keys (as shown in Fig. 6 for records r_2 and s_1). To overcome this problem the authors proposed another strategy by chaining two MapReduce jobs, where the first one is similar to the described method except that the Reduce phase only outputs the pairs of records' identifiers instead of comparing the records. In the second MapReduce job, duplicate records pairs are grouped at the same Reducer to be compared only once. In this process, the Bloom filters are not redistributed (but only their identifiers) by storing the Bloom filters in a relational database from where they are read when needed. The evaluation of this parallel LSH approach in [83] was limited to only 2 and 4 nodes and small datasets (about $300,000$ records) so that the overall scalability of the approach remains unclear.

For future work, it would be valuable to investigate and compare different parallel PPRL approaches utilizing the Hadoop ecosystem. The approaches could also utilize the Spark or Flink frameworks which support more operators than only Map and Reduce, and support efficient distributed in-memory processing.

5 Multi-Party PPRL

While there have been many different approaches proposed for PPRL [165], most work thus far has concentrated on linking records from only two databases (or parties). Only some approaches have investigated linking records from three

[7] https://flink.apache.org/ [Accessed: 15/06/2016]

or more databases [76, 105, 119, 122, 126, 130, 131, 160], with most of these being limited to exact matching or matching of categorical data only, as will be discussed below. However, as the example applications described in Sect. 1 have shown, linking data from multiple databases is increasingly being required for several Big Data applications. In the following, we describe existing techniques of multi-party private blocking and private comparison and classification for multi-party PPRL (MP-PPRL).

5.1 Multi-Party Private Blocking Techniques

Private blocking for MP-PPRL is crucial due to the exponential growth of the comparison space with the number of databases linked. However, this has not been studied until recently, making MP-PPRL not practical in real applications.

Tree-based approaches: The first approach [130] is based on a single-bit tree (adapted from multi-bit tree [99]) data structure, which is constructed iteratively to arrange records (masked into Bloom filters) such that similar records are placed into the same tree leaf while non-similar records are placed into different leaf nodes in the tree. At each iteration, the set of Bloom filters in a tree node is recursively split based on selected (according to a privacy criteria) bit positions which are agreed upon by all parties. A drawback with this approach, however, is that it might miss true matches due to the recursive splitting of Bloom filters. Furthermore, a communication step is required among all parties for each iteration.

This limitation of missing true matches in the single-bit tree-based approach [130] has been addressed in [131] using a multi-bit tree [99] data structure (as we discussed in Sect. 4.2) that is combined with canopy clustering. Multi-bit tree-based filtering for PPRL of two databases was first introduced by Schnell [144]. In [131] the concept of multi-bit trees was used to split the databases (masked into Bloom filters) individually by the parties into small mini-blocks, which are then merged into larger blocks according to privacy and computational requirements based on their similarity using a canopy clustering technique [40].

Linkage unit-based approaches: A communication-efficient approach for multi-party private blocking by using a linkage unit was recently proposed [133], as illustrated in Fig. 7. In the first step of this approach, local blocks are generated individually by each party using a private blocking technique (which is considered to be a black box). For example, the private blocking approach based on multi-bit tree and canopy clustering [131] (described above) can be used for local blocking. A block representative in the form of a min-hash signature [19] is then generated for each block and sent to a linkage unit. The linkage unit applies global blocking using locality sensitive hashing (LSH) to identify the candidate block sets from all parties or from sub-sets of parties based on the similarity between block representatives. Local blocking provides the database owners with more flexibility and control over their blocks while eliminating all communications among them. This approach outperforms existing multi-party

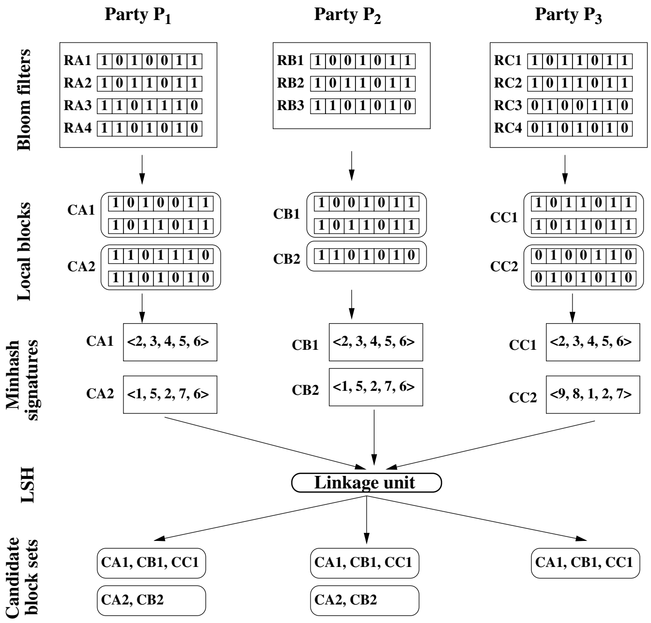

Fig. 7. Multi-party private blocking approach as proposed by Ranbaduge et al. [133] (adapted from [133]). Candidate block sets from all three parties ($CA1, CB1, CC1$) and sub-set of two parties ($CA2, CB2$) are identified to be compared and classified.

private blocking approaches in terms of scalability, privacy, and blocking quality, as validated by a set of experiments conducted by the authors [133].

Karapiperis and Verykios recently proposed a multi-party private blocking approach based on LSH [85]. This approach uses L independent hash tables (or blocking groups), each of which consists of key-bucket pairs where keys represent the blocking keys and buckets host a linked list aimed at grouping similar records that were previously masked into Bloom filters. Each hash table is assigned with a set of K hash functions which is generated by a linkage unit and sent to all the database owners to populate their set of blocks accordingly. The same authors extended this approach by proposing a frequent pairs scheme (FPS) [86] for further reducing the number of comparisons while maintaining a high level of recall. This approach achieves high blocking quality by identifying similar record pairs that exhibit a number of LSH collisions above a given threshold, and then performs distance calculations only for those similar pairs. Empirical results showed significant improvement in running time due to a drastic reduction of candidate pairs by the FPS, while achieving high blocking quality [86].

A major drawback of these multi-party private blocking techniques is that they still result in an exponential comparison space with an increasing number of databases to be linked, especially when the databases are large. Therefore, efficient communication patterns, such as ring-based or tree-based [114, 142], as

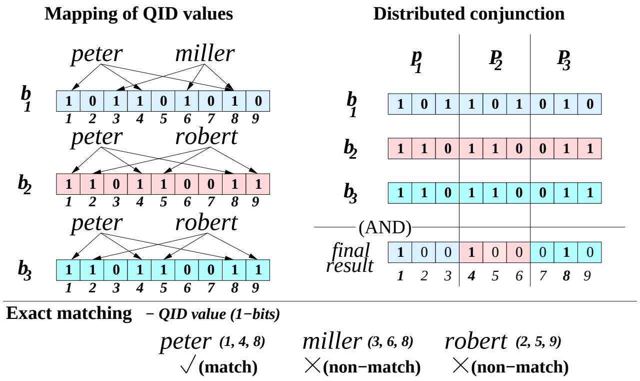

Fig. 8. Bloom filter masking-based exact matching approach for MP-PPRL as proposed by Lai et al. [105] (adapted from [165]).

well as advanced filtering techniques, such as those discussed in Sect. 4.2, need to be investigated for multi-party PPRL in order to make PPRL scalable and viable in Big Data applications.

5.2 Multi-Party Private Comparison and Classification Techniques

Several private comparison and classification techniques for MP-PPRL have been developed in the literature. However, they fall short in providing a practical solution either because they allow exact matching only or they are computationally not feasible with the size and number of multiple databases. In the following we describe these approaches and their drawbacks.

Secure Multi-party Computation (SMC)-based approach: An approach based on SMC using an oblivious transfer protocol was proposed in [122] for multi-party private comparison and classification. While provably secure, the approach only performs exact matching of masked records and it is computationally expensive compared to efficient perturbation-based privacy techniques such as Bloom filters and k-anonymity [165].

Generalization-based approaches: A multi-party private comparison and classification approach was introduced in [76] to perform secure equi-join of masked records from multiple k-anonymous databases by using a linkage unit. The database records are k-anonymised by the database owners and sent to a linkage unit. The linkage unit then compares and classifies records by applying secure equi-join, which allows exact matching only.

Another multi-party private comparison and classification approach based on k-anonymity for categorical values was proposed in [119]. In this approach, a top-down generalization is performed on the QIDs to provide k-anonymous privacy (as discussed in Sect. 3.4) and the generalized blocks are then classified into matches and non-matches using the C4.5 decision tree classifier.

Probabilistic data structure-based approaches: An efficient multi-party private comparison and classification approach for exact matching of masked records using Bloom filters was introduced by Lai et al. [105], as illustrated in Fig. 8. Each party hash-maps their record values into a single Bloom filter and then partitions its Bloom filter into segments according to the number of parties involved in the linkage. The segments are exchanged among the parties such that each party receives a corresponding Bloom filter segment from all other parties. The segments received by a party are combined using a conjunction (logical AND) operation. The resulting conjuncted Bloom filter segments are then exchanged among the parties to generate the full conjuncted Bloom filter. Each party compares its Bloom filter of each record with the final conjuncted Bloom filter. If the membership test of a record's Bloom filter is successful then the record is considered to be a match across all databases. Though the computation cost of this approach is low since the computation is completely distributed among the parties without a linkage unit and the creation and processing of Bloom filters are very fast, the approach can only perform exact matching.

Another efficient multi-party approach for private comparison and classification of categorical data was recently proposed [81] using a Count-Min sketch data structure (as described in Sect. 3.4). Sketches are used to summarize records individually by each database owner, followed by a secure intersection of these sketches to provide a global synopsis that contains the common records across parties and their frequencies. The approach uses homomorphic operations, secure summation, and symmetric noise addition privacy techniques.

Developing privacy-preserving approximate string comparison functions for multiple (more than two) values has only recently been considered [160]. This MP-PPRL approach adapts Lai et al.'s Bloom filter-based exact matching approach [105] (as described above) for approximate matching to distributively calculate the Dice-coefficient similarity of a set of Bloom filters from different parties using a secure summation protocol. This approach is illustrated in Fig. 9. The Dice-coefficient of P Bloom filters $(b_1, \cdots, b_P)$ is calculated as:

$$Dice\_sim(b_1, \cdots, b_P) = \frac{P \times c}{\sum_{i=1}^{P} x_i} = \frac{P \times \sum_{i=1}^{P} c_i}{\sum_{i=1}^{P} x_i}, \qquad (5)$$

where c_i is the number of common bit positions that are set to 1 in ith Bloom filter segment from all P parties such that $c = \sum_{i=1}^{P} c_i$, and x_i is the number of bit positions set to 1 in b_i (1-bits), where $x = \sum_{i=1}^{P} x_i$ and $1 \leq i \leq P$.

Similar to Lai et al.'s approach [105], the Bloom filters are split into segments such that each party receives a certain segment of the Bloom filters from all other parties. A logical conjunction is applied to calculate c_i individually by each party P_i (with $1 \leq i \leq P$) which are then summed to calculate c using a secure summation protocol. A secure summation of x_i is also performed to calculate x. These two sums are then used to calculate the Dice-coefficient similarity of the Bloom filters using Eq. 5. A limitation of this approach is that it can only be used to link a small number of databases due to its large number of logical conjunction calculations (even when a private blocking technique is used).

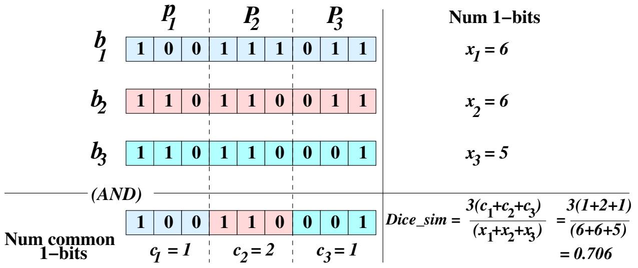
P_1
P_2
P_3
Num 1−bits
b_1 1 0 0 1 1 1 0 1 1 $x_1 = 6$
b_2 1 1 0 1 1 0 0 1 1 $x_2 = 6$
b_3 1 1 0 1 1 0 0 0 1 $x_3 = 5$
(AND)
1 0 0 1 1 0 0 0 1
Num common 1−bits
$c_1 = 1$
$c_2 = 2$
$c_3 = 1$
$Dice\_sim = \frac{3(c_1+c_2+c_3)}{(x_1+x_2+x_3)} = \frac{3(1+2+1)}{(6+6+5)}$
$= 0.706$

Fig. 9. Bloom filter masking-based approximate matching approach for MP-PPRL proposed by Vatsalan and Christen [160] (adapted from [160].

Therefore, more work needs to be done in multi-party private comparison and classification to enable efficient and effective PPRL on multiple large databases including sub-set matching (i.e. identifying matching records across subset of parties).

6 Open Challenges

In this section we first describe the various open challenges of PPRL, and then discuss these challenges in the context of the four V's *volume*, *variety*, *velocity*, and *veracity* of Big Data.

6.1 Improving Scalability

The trend of Big Data growth dispersed in multiple sources challenges PPRL in terms of complexity (*volume*), which increases exponentially with multiple large databases. Much research in recent years has focused on improving the scalability of the PPRL process, both with regard to the sizes of the databases to be linked, as well as with the number of databases to be linked. While significant progress has been made in both these directions, further efforts are required to make all aspects of the PPRL process scalable. Both directions are highly relevant for Big Data applications.

Even small blocks can still lead to a large number of record pair (or set) comparisons that are required in the comparison step, especially when databases from multiple (more than two) sources are to be linked. For each set of blocks across several parties, potentially all combinations of record sets need to be compared. For a block that contains B records from each of P parties, B^P comparisons are required. Crucial are efficient adaptive comparison techniques that stop the comparison of records across parties once a pair of records has been classified to be a non-match between two parties. For example, assume the record set $\langle r_A, r_B, r_C, r_D \rangle$, where r_A is from party A, r_B is from party B, and so on. Once the pair r_A and r_B are compared and classified as a non-match, there is no need to compare all other possible record pairs (r_A with r_C, r_A with r_D,

r_B with r_C, and so on) if the aim of the linkage is to identify sets of records that match across all parties involved in a PPRL.

A very challenging aspect is the task of identifying sub-sets of records that match across only a sub-set of parties. An example is to find all patients that have medical records in the databases of any three out of a group of five hospitals. In this situation, all potential sub-sets of records need to be compared and classified. This is a challenging problem with regard to the number of comparisons required and has not been studied in the literature so far.

6.2 Improving Linkage Quality

The *veracity* and *variety* aspects (errors and variations) of Big Data need to be addressed in PPRL by developing accurate and effective comparison and classification techniques for high linkage quality. How to efficiently calculate the similarity of more than two values using approximate comparison functions in PPRL is an important challenge with multi-source linking. Most existing PPRL solutions for multiple parties only support exact matching [81, 105] or they are applicable to QIDs of only categorical data [76, 119]. Thus far only one recent approach supports approximate matching of string data for PPRL on multiple databases [160] (as described in Sect. 5.2).

In the area of non-PPRL, advanced collective [13] and graph-based [59, 75] classification techniques have been developed in recent times. These techniques are able to achieve high linkage quality compared to the basic pair-wise comparison and threshold-based classification approach that is often employed in most PPRL techniques. Group linkage [123] is the only advanced classification technique that has so far been considered for PPRL [106].

For classification techniques that require training data (i.e. supervised classifiers), a major challenge in PPRL is how such training data can be generated. Because of privacy and confidentiality concerns, in PPRL it is generally not possible to gain access to the actual sensitive QID values (to decide if they refer to a true match or a true non-match). The advantage of certain collective and graph-based approaches [13, 75] is that they are unsupervised and therefore do not require training data. However, their disadvantage is their high computational complexities (quadratic or even higher) [137]. Investigating and adapting advanced classification techniques for PPRL will be a crucial step towards making PPRL useful for practical Big Data applications, where training data are commonly not available, or are expensive to generate.

6.3 Dynamic Data and Real-time Matching

All PPRL techniques developed so far, in line with most non-PPRL techniques, only consider the batch linkage of static databases. However, a major aspect of Big Data is the dynamic nature of data (*velocity*) that requires adaptive systems to link data as they arrive at an organization, ideally in (near) real-time. Limited work has so far investigated temporal data [32, 108] and real-time [31, 71, 129] matching in the context of record linkage. Temporal aspects can be considered by adapting the similarities between records depending upon the time difference between them, while real-time matching can be achieved using sophisticated adaptive indexing techniques. Several works have been done on dynamic privacy-preserving data publishing on the cloud by developing an efficient and adaptive QID index-based approach over incremental datasets [175, 176].

Linking dynamic databases in a PPRL context opens various challenging research questions. Existing masking (encoding) methods used in PPRL assume static databases that allow parameter settings to be calculated a-priori that lead to secure masking of QID values. For example, Bloom filters in average should have 50% of their bits set to 1, making frequency attacks more difficult [118]. Such masking might not stay secure as the characteristics of data are changing over time. Dynamic databases also require novel comparison functions that can adapt to changing data as well as adaptive masking techniques.

6.4 Improving Security and Privacy

In addition to the four V's of Big Data, another challenging aspect that needs to be considered for Big Data applications is *security and privacy.* As we discussed in Sect. 3.2, most work in PPRL assumes the *honest-but-curious* (HBC) adversary model [66, 112]. Most PPRL protocols also assume that the parties do not collude with each other (i.e. a sub-set of two or more parties do not collaborate with the aim to learn sensitive information of another party) [112]. However, in a commercial environment and in PPRL scenarios where many parties are involved, such as is likely in Big Data applications, collusion is a real possibility that needs to be prevented. Only few PPRL techniques consider the *malicious* adversary model [165]. The techniques developed based on this security model commonly have high computational complexities and are therefore currently not practical for the linkage of large databases. Therefore, because the HBC model might not be strong enough while the malicious model is computationally too expensive, novel security models that lie between those two need to be investigated for PPRL. Two of these are the *covert* adversary model [4] and *accountable computing* [72], which have been discussed in Sect. 3.2. Research directions are required to develop new protocols that are practical and at the same time more secure than protocols based on the HBC model.

With regard to privacy, most PPRL techniques are known to leak some information during the exchange of data between the parties (such as the number and sizes of blocks, or the similarities between compared records). How sensitive such revealed information is for a certain dataset heavily depends upon the parameter settings used by a protocol. Sophisticated attack methods [102] have been developed that exploit the subtle pieces of information revealed by certain PPRL protocols to iteratively gather information about sensitive values. Therefore, there is a need to harden existing PPRL techniques to ensure they are not vulnerable to such attacks. Preserving privacy of individual entities is more challenging with multi-party PPRL due to the increasing risk of collusion between a sub-set of parties which aim to learn about another (sub-set of) party's private data. Distributing computations among pairs or groups of parties can reduce the likelihood of collusion between parties if individual pairs or groups can use different secret keys (known only to them) for masking their values.

Most PPRL techniques have mainly been focusing on the privacy of the individual records that are to be linked [162]. However, besides individual record privacy, the privacy of a group of individuals also needs to be considered. Often the outcomes of a PPRL project are sets of linked records that represent people with certain characteristics (such as certain illnesses, or particular financial circumstances). While the names, addresses and other personal details of these people are not revealed during or after the PPRL process, their overall characteristics as a group could potentially lead to the discrimination of individuals in this group if these characteristics are being revealed. The research areas of privacy-preserving data publishing [60] and statistical confidentiality [45] have been addressing these issues from different directions.

PPRL is only one component in the management and analysis of sensitive, person-related information by linking different datasets in a privacy-preserving manner. However, achieving an effective overall privacy preservation needs a comprehensive strategy regarding the whole data life cycle including collection, management, publishing, exchange and analysis of data to be protected ('privacy-by-design') [22]. Hence, it is necessary to better understand the role of PPRL in the life cycle for sensitive data to ensure that it can be applied and that the match results are both useful and privacy-preserving.

In research, the different technical aspects to preserve privacy have partially been addressed by different communities with little interaction. For example, there is a large body of research on privacy-preserving data publishing [60] and on privacy-preserving data mining [110, 156] that have been largely decoupled from the research on PPRL. It is well known that data analysis may identify individuals despite the masking of QID values [152]. Hence, there is similar risk that the combined information of matched records together with some background information could lead to the identification of individuals (known as *re-identification*). Such risks must be evaluated and addressed within a comprehensive privacy strategy including a closely aligned PPRL and privacy-preserving data analysis/mining approach.

6.5 Evaluation, Frameworks, and Benchmarks

How to assess the quality (how many classified matches are true matches) and completeness (how many true matches have been classified as matches) of the records linked in a PPRL project is very challenging because it is generally not possible to inspect linked records due to privacy concerns. Manual assessment of individual records would reveal sensitive information which is in contradiction to the objective of PPRL. Not knowing how accurate and complete linked data are is however a major issue that will render any PPRL protocol impractical in applications where linkage completeness and quality are crucial, as is the case in many Big Data applications such as in the health or security domains.

Recent initial work has proposed ideas and concepts for interactive PPRL [101] where parts of sensitive values are revealed for manual assessment. How to actually implement such approaches in real applications, while ensuring the revealed information is limited to a certain level of detail (for example providing k-anonymous privacy for a certain value of $k > 1$ [152]) is an open research question that must be solved. Interactive manual evaluation might also not be feasible in Big Data applications where the size and dynamic nature of data, as well as real-time processing requirements, prohibit any manual inspection.

With regard to evaluating the privacy protection that a given PPRL technique provides, unlike for measuring linkage quality and completeness (where standard measurements such as run-time, reduction ratio, pairs completeness, pairs quality, precision, recall, or accuracy are available [28]), there are currently no standard measurements for assessing privacy in PPRL. Different measurements have been proposed and used [46, 162, 165], making the comparison of different PPRL techniques difficult. How to assess linkage quality and completeness, as well as privacy, are must-solve problems as otherwise it will not be possible to evaluate the efficiency, effectiveness, and privacy protection of PPRL techniques in real-world applications, leaving these techniques non-practical.

An important direction of future work for PPRL is the development of frameworks that allow the experimental comparison of different PPRL techniques with regard to their scalability, linkage quality, and privacy preservation. No such framework currently exists. Ideally, such frameworks allow researchers to easily ‘plug-in’ their own algorithms such that over time a collection of PPRL algorithms is compiled that can be tested and evaluated by researchers, as well as by practitioners to allow them to identify the best technique to use for their application scenario.

An issue related to frameworks is the availability of publicly available benchmark datasets for PPRL. While this is not a challenge limited to PPRL but to record linkage research in general [28, 96], it is particularly prominent for PPRL as it deals with sensitive and confidential data. While for record linkage techniques publicly available data from bibliographic or consumer product databases might be used [96], such data are less useful for PPRL research as they have different characteristics compared to personal data. The nature of the datasets to be linked using PPRL techniques is obviously in strong contradiction to them being made public. Ideally researchers working in PPRL are able to collaborate with practitioners that do have access to real sensitive and confidential databases to allow them to evaluate their techniques on such data.

A possible alternative to using benchmark datasets is the use of synthetic data that are generated based on the characteristics of real data using data generators [34, 153]. Such generators must be able to generate data with similar distribution of values, variations, and errors as would be expected in real datasets from the same domain. Several such data generators have been developed and are used by researchers working in PPRL as well as record linkage in general.

6.6 Discussion

As we have discussed in this section, there are various challenges that need to be addressed in order to make PPRL practical for applications in a variety of domains. Some of these challenges are general and not just affect PPRL for Big Data, others are specific to certain types of applications, including those in the Big Data space.

The challenge of scalability of PPRL towards very large databases is highly relevant to the *volume* of Big Data, while the challenge of linkage quality of PPRL is highly relevant to the *veracity* and *variety* of Big Data. The dynamic nature of data in many Big Data applications, and the requirement of being able to link data in real-time, are challenging all aspects of PPRL, as well as record linkage in general [129]. This challenge corresponds to the *velocity* of Big Data and it requires the development of novel techniques that are adaptive to changing data characteristics, and that are highly efficient with regard to fast linking of streams of query records. While the *volume*, *variety*, and *veracity* aspects of Big Data have been studied for PPRL to some extent, the *velocity* aspect has so far not been addressed in a PPRL context.

Making PPRL more secure and more private is challenged by all four V's of Big Data. Larger data *volume* likely means that only encoding techniques that require little computational efforts per record can be employed, while dynamic data (*velocity*) means such techniques have to be adaptable to changing data characteristics. *Variety* means PPRL techniques have to be made more secure and private for various types of data, while *veracity* requires them to also take data uncertainties into account. The challenge of integrating PPRL into an overall privacy-preserving approach has also not seen any work so far. All four V's of Big Data will affect the overall efficiency and effectiveness of systems that enable the management and analysis of sensitive and confidential information in a privacy-preserving manner. The more basic challenges of improving scalability, linkage quality, privacy and evaluation need to solved first before this more complex challenge of an overall privacy-preserving system can be addressed.

The final challenge of evaluation is affected by all aspects of Big Data. Improved evaluation of PPRL systems requires that databases that are large, heterogeneous, dynamic, and that contain uncertain data, can be handled and evaluated efficiently and accurately. So far no research in PPRL has investigated evaluation specifically for Big Data. While the lack of general benchmarks and frameworks is already a gap in PPRL and record linkage research in general, Big Data will make this challenge even more pronounced. Compared to frameworks that can handle small and medium sized static datasets only, it is even more difficult to develop frameworks that enable privacy-preserving linking of very large and dynamic databases, as is making such datasets publicly available. No work addressing this challenge in the context of Big Data has been published.

7 Conclusions

Privacy-preserving record linkage (PPRL) is an emerging research field that is being required by many different applications to enable effective and efficient linkage of databases across different organizations without compromising privacy and confidentiality of the entities in these databases. In the Big Data era, tremendous opportunities can be realized by linking data at the cost of additional challenges. In this chapter, we have provided background material required to understand the applications, process, and challenges of PPRL, and we have reviewed existing PPRL approaches to understand the literature. Based on the analysis of existing techniques, we have discussed several interesting and challenging directions for future work in PPRL for Big Data.

With the increasing trend of Big Data in organizations, more research is required towards the development of techniques that allow for multiple large databases to be linked in privacy-preserving, effective, and efficient ways, thereby facilitating novel ways of data analysis and mining that currently are not feasible due to scalability, quality, and privacy-preserving challenges.

Acknowledgments. This work was partially funded by the Australian Research Council under Discovery Project DP130101801, the German Academic Exchange Service (DAAD) and Universities Australia (UA) under the Joint Research Co-operation Scheme, and also funded by the German Federal Ministry of Education and Research within the project Competence Center for Scalable Data Services and Solutions (ScaDS) Dresden/Leipzig (BMBF 01IS14014B).

References

1. R. Agrawal, A. Evfimievski, and R. Srikant. Information sharing across private databases. In *ACM SIGMOD*, pages 86–97, 2003.
2. A. Arasu, V. Ganti, and R. Kaushik. Efficient Exact Set-Similarity Joins. In *Proceedings of VLDB*, pages 918–929, 2006.
3. A. Arasu, M. Götz, and R. Kaushik. On active learning of record matching packages. In *ACM SIGMOD*, pages 783–794, 2010.
4. Y. Aumann and Y. Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. *Journal of Cryptology*, 23(2), 2010.
5. T. Bachteler, J. Reiher, and R. Schnell. Similarity Filtering with Multibit Trees for Record Linkage. Technical Report WP-GRLC-2013-01, German Record Linkage Center, 2013.
6. D. Barone, A. Maurino, F. Stella, and C. Batini. A privacy-preserving framework for accuracy and completeness quality assessment. *Emerging Paradigms in Informatics, Systems and Communication*, 2009.
7. J. E. Barros, J. C. French, W. N. Martin, P. M. Kelly, and T. M. Cannon. Using the triangle inequality to reduce the number of comparisons required for similarity-based retrieval. In *Electronic Imaging: Science & Technology*, 1996.
8. C. Batini and M. Scannapieca. *Data quality: Concepts, methodologies and techniques.* Data-Centric Systems and Appl. Springer, 2006.
9. R. Baxter, P. Christen, and T. Churches. A comparison of fast blocking methods for record linkage. In *SIGKDD Workshop on Data Cleaning, Record Linkage and Object Consolidation*, 2003.
10. R. J. Bayardo, Y. Ma, and R. Srikant. Scaling Up All Pairs Similarity Search. In *WWW*, pages 131–140, 2007.
11. K. Bellare, S. Iyengar, A. G. Parameswaran, and V. Rastogi. Active sampling for entity matching. In *ACM SIGKDD*, 2012.

12. A. Berman and L. G. Shapiro. Selecting good keys for triangle-inequality-based pruning algorithms. In *IEEE Workshop on Content-Based Access of Image and Video Database*, pages 12–19, 1998.
13. I. Bhattacharya and L. Getoor. Collective entity resolution in relational data. *ACM TKDD*, 1(1), 2007.
14. M. Bilenko and R. J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *ACM SIGKDD*, pages 39–48, 2003.
15. B. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.
16. L. Bonomi, L. Xiong, R. Chen, and B. Fung. Frequent grams based embedding for privacy preserving record linkage. In *ACM CIKM*, pages 1597–1601, 2012.
17. H. Bouzelat, C. Quantin, and L. Dusserre. Extraction and anonymity protocol of medical file. In *AMIA Fall Symposium*, pages 323–327, 1996.
18. A. Broder, M. Mitzenmacher, and A. Mitzenmacher. Network applications of Bloom filters: A survey. In *Internet Mathematics*, 2002.
19. A. Z. Broder. On the resemblance and containment of documents. In *Compression and Complexity of Sequences*, pages 21–29. IEEE, 1997.
20. E. Brook, D. Rosman, and C. Holman. Public good through data linkage: measuring research outputs from the Western Australian Data Linkage System. *Aust NZ Journal of Public Health*, 32:19–23, 2008.
21. R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.
22. A. Cavoukian and J. Jonas. Privacy by design in the age of Big Data. Technical report, TR Information and privacy commissioner, Ontario, 2012.
23. P. Christen. A comparison of personal name matching: Techniques and practical issues. In *IEEE ICDM Workshop on Mining Complex Data*, pages 290–294, 2006.
24. P. Christen. Privacy-preserving data linkage and geocoding: Current approaches and research directions. In *IEEE ICDM Workshop on Privacy Aspects of Data Mining*, pages 497–501, 2006.
25. P. Christen. Automatic record linkage using seeded nearest neighbour and support vector machine classification. In *ACM SIGKDD*, pages 151–159, 2008.
26. P. Christen. Febrl: An open source data cleaning, deduplication and record linkage system with a graphical user interface. In *ACM SIGKDD*, pages 1065–1068, 2008.
27. P. Christen. Geocode matching and privacy preservation. In *Workshop on Privacy, Security, and Trust in KDD*, pages 7–24. Springer, 2009.
28. P. Christen. *Data Matching - Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection.* Springer, 2012.
29. P. Christen. A survey of indexing techniques for scalable record linkage and deduplication. *IEEE TKDE*, 24(9):1537–1555, 2012.
30. P. Christen, T. Churches, and M. Hegland. Febrl – A parallel open source data linkage system. In *Springer PAKDD*, pages 638–647, 2004.
31. P. Christen, R. Gayler, and D. Hawking. Similarity-aware indexing for real-time entity resolution. In *ACM CIKM*, pages 1565–1568, 2009.
32. P. Christen and R. W. Gayler. Adaptive temporal entity resolution on dynamic databases. In *PAKDD*, pages 558–569, 2013.
33. P. Christen and K. Goiser. Quality and complexity measures for data linkage and deduplication. In *Quality Measures in Data Mining*, volume 43 of *Studies in Computational Intelligence*, pages 127–151. Springer, 2007.
34. P. Christen and D. Vatsalan. Flexible and extensible generation and corruption of personal data. In *ACM CIKM*, pages 1165–1168, 2013.

35. T. Churches and P. Christen. Some methods for blindfolded record linkage. *BioMed Central Medical Informatics and Decision Making, 2004*, 4(9), 2004.
36. T. Churches, P. Christen, K. Lim, and J. X. Zhu. Preparation of name and address data for record linkage using hidden Markov models. *BioMed Central Medical Informatics and Decision Making, 2002*, 2(9), 2002.
37. D. E. Clark. Practical introduction to record linkage for injury research. *Injury Prevention*, 10:186–191, 2004.
38. C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Zhu. Tools for privacy preserving distributed data mining. *SIGKDD Explorations*, 4(2):28–34, 2002.
39. W. W. Cohen. Data Integration Using Similarity Joins and a Word-Based Information Representation Language. *ACM TOIS*, 18(3):288–321, 2000.
40. W. W. Cohen and J. Richman. Learning to match and cluster large high-dimensional data sets for data integration. In *ACM SIGKDD*, 2002.
41. G. Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.
42. G. Dal Bianco, R. Galante, and C. A. Heuser. A fast approach for parallel deduplication on multicore processors. In *ACM Symposium on Applied Computing*, pages 1027–1032, 2011.
43. D. Dey, V. Mookerjee, and D. Liu. Efficient techniques for online record linkage. *IEEE Transactions on Knowledge and Data Engineering*, 23(3):373–387, 2010.
44. W. Du and M. Atallah. Protocols for secure remote database access with approximate matching. In *Proceedings of the 1st ACM Workshop on Security and Privacy in E-Commerce*, pages 87–111. Springer, 2000.
45. G. T. Duncan, M. Elliot, and J.-J. Salazar-González. *Statistical Confidentiality: principles and practice.* Springer New York, 2011.
46. E. Durham. *A framework for accurate, efficient private record linkage.* PhD thesis, Faculty of the Graduate School of Vanderbilt University, Nashville, TN, 2012.
47. E. Durham, Y. Xue, M. Kantarcioglu, and B. Malin. Private medical record linkage with approximate matching. In *AMIA Annual Symposium*, 2010.
48. E. A. Durham, C. Toth, M. Kuzu, M. Kantarcioglu, Y. Xue, and B. Malin. Composite Bloom filters for secure record linkage. *IEEE TKDE*, 1(99), 2013.
49. L. Dusserre, C. Quantin, and H. Bouzelat. A one way public key cryptosystem for the linkage of nominal files in epidemiological studies. *Medinfo*, 8, 1995.
50. C. Dwork. Differential privacy. *International Colloquium on Automata, Languages and Programming*, pages 1–12, 2006.
51. M. G. Elfeky, V. S. Verykios, and A. K. Elmagarmid. TAILOR: A record linkage toolbox. In *IEEE ICDE*, 2002.
52. A. Elmagarmid, P. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE TKDE*, 19(1):1–16, 2007.
53. U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy. *Advances in knowledge discovery and data mining.* the MIT Press, 1996.
54. I. P. Fellegi and A. B. Sunter. A theory for record linkage. *Journal of the American Statistical Society*, 64(328):1183–1210, 1969.
55. S. E. Fienberg. Confidentiality and disclosure limitation. *Encyclopedia of Social Measurement*, 1:463–69, 2005.
56. O. for National Statistics. Matching anonymous data. In *Beyond 2011*, 2013.
57. B. Forchhammer, T. Papenbrock, T. Stening, S. Viehmeier, U. Draisbach, and F. Naumann. Duplicate Detection on GPUs. In *Database Systems for Business, Technology, and Web*, pages 165–184, 2013.
58. M. Freedman, Y. Ishai, B. Pinkas, and O. Reingold. Keyword search and oblivious pseudorandom functions. *Theory of Cryptography*, 2005.

59. Z. Fu, J. Zhou, P. Christen, and M. Boot. Multiple instance learning for group record linkage. In *PAKDD, Springer LNAI*, 2012.
60. B. Fung, K. Wang, R. Chen, and P. S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Surveys*, 42(4):14, 2010.
61. S. R. Ganta, S. P. Kasiviswanathan, and A. Smith. Composition attacks and auxiliary information in data privacy. In *ACM SIGKDD*, pages 265–273, 2008.
62. A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB*, pages 518–529, 1999.
63. O. Goldreich. *Foundations of cryptography: Basic applications*, volume 2. Cambridge Univ Press, 2004.
64. L. Gu and R. Baxter. Decision models for record linkage. In *Selected Papers from AusDM, Springer LNCS 3755*, 2006.
65. M. Hadjieleftheriou, A. Chandel, N. Koudas, and D. Srivastava. Fast Indexes and Algorithms for Set Similarity Selection Queries. In *IEEE ICDE*, 2008.
66. R. Hall and S. Fienberg. Privacy-preserving record linkage. In *PSD*, pages 269–283, 2010.
67. M. Herschel, F. Naumann, S. Szott, and M. Taubert. Scalable iterative graph duplicate detection. *IEEE TKDE*, 24(11):2094–2108, 2012.
68. A. Inan, M. Kantarcioglu, E. Bertino, and M. Scannapieco. A hybrid approach to private record linkage. In *IEEE ICDE*, pages 496–505, 2008.
69. A. Inan, M. Kantarcioglu, G. Ghinita, and E. Bertino. Private record matching using differential privacy. In *EDBT*, 2010.
70. P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *ACM Symp. on the Theory of Computing*, pages 604–613, 1998.
71. E. Ioannou, W. Nejdl, C. Niederée, and Y. Velegrakis. On-the-fly entity-aware query processing in the presence of linkage. *PVLDB*, 3(1-2):429–438, 2010.
72. W. Jiang and C. Clifton. Ac-framework for privacy-preserving collaboration. In *SDM*, pages 47–56. SIAM, 2007.
73. W. Jiang, C. Clifton, and M. Kantarcıoğlu. Transforming semi-honest protocols to ensure accountability. *Elsevier DKE*, 65(1):57–74, 2008.
74. J. Jonas and J. Harper. Effective counterterrorism and the limited role of predictive data mining. *Policy Analysis*, 584, 2006.
75. D. Kalashnikov and S. Mehrotra. Domain-independent data cleaning via analysis of entity-relationship graph. *ACM TODS*, 31(2):716–767, 2006.
76. M. Kantarcioglu, W. Jiang, and B. Malin. A privacy-preserving framework for integrating person-specific databases. In *PSD*, pages 298–314, 2008.
77. A. Karakasidis and V. S. Verykios. Secure blocking+secure matching = secure record linkage. *JCSE*, 5:223–235, 2011.
78. A. Karakasidis and V. S. Verykios. Reference table based k-anonymous private blocking. In *ACM SAC*, pages 859–864, 2012.
79. A. Karakasidis and V. S. Verykios. A sorted neighborhood approach to multidimensional privacy preserving blocking. In *IEEE ICDMW*, pages 937–944, 2012.
80. A. Karakasidis, V. S. Verykios, and P. Christen. Fake injection strategies for private phonetic matching. In *DPM, Springer*, volume 7122, pages 9–24, 2012.
81. D. Karapiperis, D. Vatsalan, V. S. Verykios, and P. Christen. Large-scale multi-party counting set intersection using a space efficient global synopsis. In *DASFAA*, 2015.
82. D. Karapiperis, D. Vatsalan, V. S. Verykios, and P. Christen. Efficient record linkage using a compact hamming space. In *EDBT*, 2016.

83. D. Karapiperis and V. S. Verykios. A distributed framework for scaling up lsh-based computations in privacy preserving record linkage. In *ACM BCI*, 2013.
84. D. Karapiperis and V. S. Verykios. A distributed near-optimal LSH-based framework for privacy-preserving record linkage. *ComSIS*, 11(2):745–763, 2014.
85. D. Karapiperis and V. S. Verykios. An LSH-based blocking approach with a homomorphic matching technique for privacy-preserving record linkage. *IEEE TKDE*, 27(4):909–921, 2015.
86. D. Karapiperis and V. S. Verykios. A fast and efficient hamming LSH-based scheme for accurate linkage. *Springer KAIS*, pages 1–24, 2016.
87. H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *IEEE ICDM*, 2003.
88. H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. Random-data perturbation techniques and privacy-preserving data mining. *Springer KAIS*, 7(4), 2005.
89. C. W. Kelman, J. Bass, and D. Holman. Research use of linked health data – A best practice protocol. *Aust NZ Journal of Public Health*, 26:251–255, 2002.
90. H. Kim and D. Lee. Harra: fast iterative hashed record linkage for large-scale data collections. In *EDBT*, pages 525–536, 2010.
91. H.-s. Kim and D. Lee. Parallel linkage. In *ACM CIKM*, pages 283–292, 2007.
92. T. Kirsten, L. Kolb, M. Hartung, A. Groß, H. Köpcke, and E. Rahm. Data partitioning for parallel entity matching. In *Proc. 8th Intl. Workshop on Quality in Databases (QDB)*, 2010.
93. L. Kissner and D. Song. Private and threshold set-intersection. In *Technical Report.* Carnegie Mellon University, 2004.
94. L. Kolb, A. Thor, and E. Rahm. Dedoop: Efficient Deduplication with Hadoop. *PVLDB*, 5(12):1878–1881, 2012.
95. L. Kolb, A. Thor, and E. Rahm. Load balancing for mapreduce-based entity resolution. In *IEEE ICDE*, pages 618–629, 2012.
96. H. Köpcke and E. Rahm. Frameworks for entity matching: A comparison. *Elsevier DKE*, 69(2):197–210, 2010.
97. H. Köpcke, A. Thor, and E. Rahm. Evaluation of entity resolution approaches on real-world match problems. *PVLDB*, 3(1):484–493, 2010.
98. H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-hashing for message authentication. In *Internet RFCs*, 1997.
99. T. G. Kristensen, J. Nielsen, and C. N. Pedersen. A tree-based method for the rapid screening of chemical fingerprints. *Algorithms for Molecular Biology*, 5(1):9, 2010.
100. H. Kum, A. Krishnamurthy, A. Machanavajjhala, and S. Ahalt. Population informatics: Tapping the social genome to advance society: A vision for putting "big data" to work for population informatics. *Computer*, 2013.
101. H.-C. Kum, A. Krishnamurthy, A. Machanavajjhala, M. K. Reiter, and S. Ahalt. Privacy preserving interactive record linkage. *JAMIA*, 21(2):212–220, 2014.
102. M. Kuzu, M. Kantarcioglu, E. Durham, and B. Malin. A constraint satisfaction cryptanalysis of Bloom filters in private record linkage. In *PETS, Springer LNCS*, volume 6794, pages 226–245, 2011.
103. M. Kuzu, M. Kantarcioglu, E. A. Durham, C. Toth, and B. Malin. A practical approach to achieve private medical record linkage in light of public resources. *JAMIA*, 20(2):285–292, 2013.
104. M. Kuzu, M. Kantarcioglu, A. Inan, E. Bertino, E. Durham, and B. Malin. Efficient privacy-aware record integration. In *ACM EDBT*, 2013.
105. P. Lai, S. Yiu, K. Chow, C. Chong, and L. Hui. An Efficient Bloom filter based Solution for Multiparty Private Matching. In *SAM*, 2006.

106. F. Li, Y. Chen, B. Luo, D. Lee, and P. Liu. Privacy preserving group linkage. In *Scientific and Statistical Database Management*, pages 432–450. Springer, 2011.
107. N. Li, T. Li, and S. Venkatasubramanian. T-closeness: Privacy beyond k-anonymity and l-diversity. In *IEEE ICDE*, pages 106–115, 2007.
108. P. Li, X. Dong, A. Maurino, and D. Srivastava. Linking temporal records. *Proceedings of VLDB*, 4(11):956–967, 2011.
109. Z. Lin, M. Hewett, and R. B. Altman. Using binning to maintain confidentiality of medical data. In *AMIA Symposium*, page 454, 2002.
110. Y. Lindell and B. Pinkas. Privacy preserving data mining. In *CRYPTO*, pages 36–54. Springer, 2000.
111. Y. Lindell and B. Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. *EUROCRYPT*, pages 52–78, 2007.
112. Y. Lindell and B. Pinkas. Secure multiparty computation for privacy-preserving data mining. *JPC*, 1(1):5, 2009.
113. H. Liu, H. Wang, and Y. Chen. Ensuring data storage security against frequency-based attacks in wireless networks. In *DCOSS, Springer LNCS*, volume 6131, pages 201–215, 2010.
114. H. Lu, M.-C. Shan, and K.-L. Tan. Optimization of multi-way join queries for parallel execution. In *VLDB*, pages 549–560, 1991.
115. M. Luby and C. Rackoff. How to construct pseudo-random permutations from pseudo-random functions. In *CRYPTO*, volume 85, page 447, 1986.
116. A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM TKDD*, 1(1):3, 2007.
117. B. A. Malin, K. El Emam, and C. M. O'Keefe. Biomedical data privacy: problems, perspectives, and recent advances. *JAMIA*, 20(1):2–6, 2013.
118. M. Mitzenmacher and E. Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis.* Cambridge University Press, 2005.
119. N. Mohammed, B. Fung, and M. Debbabi. Anonymity meets game theory: secure data integration with malicious participants. *PVLDB*, 20(4):567–588, 2011.
120. M. Nentwig, M. Hartung, A.-C. Ngonga Ngomo, and E. Rahm. A survey of current link discovery frameworks. *Semantic Web Journal*, 2016.
121. A. N. Ngomo, L. Kolb, N. Heino, M. Hartung, S. Auer, and E. Rahm. When to reach for the cloud: Using parallel hardware for link discovery. In *ESWC*, 2013.
122. C. O'Keefe, M. Yung, L. Gu, and R. Baxter. Privacy-preserving data linkage protocols. In *ACM WPES*, pages 94–102, 2004.
123. B. On, N. Koudas, D. Lee, and D. Srivastava. Group linkage. In *IEEE ICDE*, pages 496–505, 2007.
124. C. Pang, L. Gu, D. Hansen, and A. Maeder. Privacy-preserving fuzzy matching using a public reference table. In *Intelligent Patient Management*, volume 189 of *Studies in Computational Intelligence*, pages 71–89. Springer, 2009.
125. C. Phua, K. Smith-Miles, V. Lee, and R. Gayler. Resilient identity crime detection. *IEEE TKDE*, 24(3), 2012.
126. C. Quantin, H. Bouzelat, F. Allaert, A. Benhamiche, J. Faivre, and L. Dusserre. How to ensure data security of an epidemiological follow-up: quality assessment of an anonymous record linkage procedure. *IJMI*, 49(1):117–122, 1998.
127. C. Quantin, H. Bouzelat, and L. Dusserre. Irreversible encryption method by generation of polynomials. *Medical Informatics and the Internet in Medicine*, 21(2):113–121, 1996.
128. E. Rahm and H. H. Do. Data cleaning: Problems and current approaches. *IEEE Data Engineering Bulletin*, 23(4):3–13, 2000.

129. B. Ramadan, P. Christen, H. Liang, and R. W. Gayler. Dynamic sorted neighborhood indexing for real-time entity resolution. *ACM JDIQ*, 6(4):15, 2015.
130. T. Ranbaduge, P. Christen, and D. Vatsalan. Tree based scalable indexing for multi-party privacy-preserving record linkage. In *AusDM*, 2014.
131. T. Ranbaduge, D. Vatsalan, and P. Christen. Clustering-based scalable indexing for multi-party privacy-preserving record linkage. In *Springer PAKDD*, 2015.
132. T. Ranbaduge, D. Vatsalan, and P. Christen. Merlin–a tool for multi-party privacy-preserving record linkage. In *IEEE ICDMW*, pages 1640–1643, 2015.
133. T. Ranbaduge, D. Vatsalan, and P. Christen. Hashing-based distributed multiparty blocking for privacy-preserving record linkage. In *Springer PAKDD*, 2016.
134. T. Ranbaduge, D. Vatsalan, S. Randall, and P. Christen. Evaluation of advanced techniques for multi-party privacy-preserving record linkage on real-world health databases. In *IPDLN*, 2016.
135. S. M. Randall, A. M. Ferrante, J. H. Boyd, A. P. Brown, and J. B. Semmens. Limited privacy protection and poor sensitivity is it time to move on from the statistical linkage key-581? *Health Information Management Journal*, 2016.
136. S. M. Randall, A. M. Ferrante, J. H. Boyd, and J. B. Semmens. Privacy-preserving record linkage on large real world datasets. *Elsevier JBI*, 2014.
137. V. Rastogi, N. Dalvi, and M. Garofalakis. Large-scale collective entity matching. *Proceedings of VLDB*, 4:208–218, 2011.
138. C. Rong, W. Lu, X. Wang, X. Du, Y. Chen, and A. K. H. Tung. Efficient and Scalable Processing of String Similarity Join. *IEEE TKDE*, 25(10), 2013.
139. M. Roughan and Y. Zhang. Secure distributed data-mining and its application to large-scale network measurements. *ACM SIGCOMM Computer Communication Review*, 36(1):7–14, 2006.
140. T. Ryan, D. Gibson, and B. Holmes. *A national minimum data set for home and community care*. Australian Institute of Health and Welfare, 1999.
141. M. Scannapieco, I. Figotin, E. Bertino, and A. Elmagarmid. Privacy preserving schema and data matching. In *ACM SIGMOD*, pages 653–664, 2007.
142. D. A. Schneider and D. J. DeWitt. Tradeoffs in processing complex join queries via hashing in multiprocessor database machines. In *VLDB*, pages 469–480, 1990.
143. B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source code in C*. John Wiley & Sons, Inc., 2 edition, 1996.
144. R. Schnell. Privacy-preserving record linkage and privacy-preserving blocking for large files with cryptographic keys using multibit trees. *JSM*, 2013.
145. R. Schnell. An efficient privacy-preserving record linkage technique for administrative data and censuses. *Statistical Journal of the IAOS*, 30(3):263–270, 2014.
146. R. Schnell, T. Bachteler, and S. Bender. A toolbox for record linkage. *Austrian Journal of Statistics*, 33(1-2):125–133, 2004.
147. R. Schnell, T. Bachteler, and J. Reiher. Privacy-preserving record linkage using Bloom filters. *BMC Medical Informatics and Decision Making*, 9(1), 2009.
148. R. Schnell, T. Bachteler, and J. Reiher. A novel error-tolerant anonymous linking code. *German Record Linkage Center, WP-GRLC-2011-02*, 2011.
149. Z. Sehili, L. Kolb, C. Borgs, R. Schnell, and E. Rahm. Privacy preserving record linkage with PPJoin. In *BTW Conference*, 2015.
150. Z. Sehili and E. Rahm. Speeding up privacy preserving record linkage for metric space similarity measures. *Datenbank-Spektrum*, pages 1–10, 2016.
151. D. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. *IEEE Symposium on Security and Privacy*, pages 44–55, 2000.
152. L. Sweeney. K-anonymity: A model for protecting privacy. *International Journal of Uncertainty Fuzziness and Knowledge Based Systems*, 10(5):557–570, 2002.

153. K.-N. Tran, D. Vatsalan, and P. Christen. GeCo: an online personal data generator and corruptor. In *ACM CIKM*, pages 2473–2476, 2013.
154. S. Trepetin. Privacy-preserving string comparisons in record linkage systems: a review. *Information Security Journal: A Global Perspective*, 17(5):253–266, 2008.
155. E. Turgay, T. Pedersen, Y. Saygın, E. Savaş, and A. Levi. Disclosure risks of distance preserving data transformations. In *Springer SSDBM*, pages 79–94, 2008.
156. J. Vaidya, Y. Zhu, and C. W. Clifton. *Privacy Preserving Data Mining*, volume 19 of *Advances in Information Security*. Springer, 2006.
157. E. Van Eycken, K. Haustermans, F. Buntinx, et al. Evaluation of the encryption procedure and record linkage in the Belgian National Cancer Registry. *Archives of public health*, 58(6):281–294, 2000.
158. D. Vatsalan and P. Christen. An iterative two-party protocol for scalable privacy-preserving record linkage. In *AusDM, CRPIT*, volume 134, 2012.
159. D. Vatsalan and P. Christen. Sorted nearest neighborhood clustering for efficient private blocking. In *Springer PAKDD*, volume 7819, 2013.
160. D. Vatsalan and P. Christen. Scalable privacy-preserving record linkage for multiple databases. In *ACM CIKM*, 2014.
161. D. Vatsalan and P. Christen. Privacy-preserving matching of similar patients. *Elsevier JBI*, 59:285–298, 2016.
162. D. Vatsalan, P. Christen, C. M. O'Keefe, and V. S. Verykios. An evaluation framework for privacy-preserving record linkage. *JPC*, 2014.
163. D. Vatsalan, P. Christen, and V. S. Verykios. An efficient two-party protocol for approximate matching in private record linkage. In *AusDM*, 2011.
164. D. Vatsalan, P. Christen, and V. S. Verykios. Efficient two-party private blocking based on sorted nearest neighborhood clustering. In *ACM CIKM*, 2013.
165. D. Vatsalan, P. Christen, and V. S. Verykios. A taxonomy of privacy-preserving record linkage techniques. *Elsevier JIS*, 38(6):946–969, 2013.
166. R. Vernica, M. J. Carey, and C. Li. Efficient parallel set-similarity joins using MapReduce. In *Proceedings of ACM SIGMOD*, pages 495–506, 2010.
167. V. S. Verykios, A. Karakasidis, and V. Mitrogiannis. Privacy preserving record linkage approaches. *IJDMMM*, 1(2):206–221, 2009.
168. G. Wang, H. Chen, and H. Atabakhsh. Automatically detecting deceptive criminal identities. *Communications of the ACM*, 47(3):70–76, 2004.
169. Q. Wang, D. Vatsalan, and P. Christen. Efficient interactive training selection for large-scale entity resolution. In *PAKDD*, 2015.
170. Z. Wen and C. Dong. Efficient protocols for private record linkage. In *Proceedings ACM Symposium on Applied Computing*, pages 1688–1694, 2014.
171. W. E. Winkler. Methods for evaluating and creating data quality. *Elsevier JIS*, 29(7):531–550, 2004.
172. C. Xiao, W. Wang, X. Lin, and J. X. Yu. Efficient Similarity Joins for Near Duplicate Detection. In *WWW*, pages 131–140, 2008.
173. M. Yakout, M. Atallah, and A. Elmagarmid. Efficient private record linkage. In *IEEE ICDE*, pages 1283–1286, 2009.
174. P. Zezula, G. Amato, V. Dohnal, and M. Batko. *Similarity search: the metric space approach*, volume 32. Springer, 2006.
175. X. Zhang, C. Liu, S. Nepal, and J. Chen. An efficient quasi-identifier index based approach for privacy preservation over incremental data sets on cloud. *Journal of Computer and System Sciences*, 79(5):542–555, 2013.
176. X. Zhang, C. Liu, S. Nepal, S. Pandey, and J. Chen. A privacy leakage upper bound constraint-based approach for cost-effective privacy preserving of intermediate data sets in cloud. *IEEE TPDS*, 24(6):1192–1202, 2013.