Scalable, Accurate and Secure Privacy-Preserving Record Linkage

Von der Fakultät für Mathematik und Informatik
der Universität Leipzig
angenommene

Dissertation

zur Erlangung des akademischen Grades

Doctor Rerum Naturalium
(Dr. rer. nat.)

im Fachgebiet
Informatik

Vorgelegt

von M. Sc. Informatik Martin Franke
geboren am 18. März 1992 in Sömmerda

Die Annahme der Dissertation wurde empfohlen von:

1. Prof. Dr. Erhard Rahm, Universität Leipzig
2. Prof. Dr. Rainer Schnell, Universität Duisburg-Essen

Die Verleihung des akademischen Grades erfolgt mit Bestehen der Verteidigung am 16.04.2024 mit dem Gesamtprädikat *summa cum laude.*

Abstract

The digital revolution has led to rapid growth in the amount of data that is collected by organizations. Analyzing large data collections through the use of data mining and predictive analytics allows organizations to make data-driven decisions to improve efficiency, effectiveness, or profitability.

Much of the data collected by organizations, such as government agencies, healthcare providers, or insurance companies, is related to individuals. However, each organization typically maintains an independent database containing specific data for a concrete purpose. Therefore, different databases typically store different data (variables) about a certain group of individuals. Complex research questions often cannot be answered solely based on one database because either certain data (co-variables) are missing, or the sample size is too small. Comprehensive data analysis therefore often requires the integration (combination) of data from multiple autonomous and thus heterogeneous databases. For holistic medical research, for instance, medical care data could be combined with clinical trial data, data from disease registries, or social data to uncover hidden correlations and improve prevention, therapy, and care.

A crucial part of data integration is to identify records from different databases that refer to the same real-world entity, such as a patient. This task is known as record linkage and relies on comparing available quasi-identifiers, such as names, addresses, and dates of birth of patients. However, exchanging such personal data between different organizations conflicts with privacy and is often not permitted, since personal data is protected by strict legal regulations. Privacy-preserving record linkage (PPRL) addresses this problem by providing techniques for linking records while preserving the privacy of represented entities (individuals). PPRL approaches have to ensure that no private or confidential information is revealed during the linkage that would allow to (re-)identify an individual. Therefore, quasi-identifying attribute values are encoded (masked) and the linkage is conducted on encoded records.

PPRL techniques pose three key challenges that need to be addressed, namely (1) reaching high efficiency with scalability to large and potentially many databases; (2) achieving high linkage quality by avoiding false and missing matches; and (3) ensuring a high degree of privacy by providing secure encodings and linkage protocols. In this thesis, we present extensive research that addresses all three key challenges of PPRL by providing solutions to several problems and shortcomings identified in existing PPRL approaches.

First, we focus on improving the scalability and overall performance of PPRL by investigating locality-sensitive hashing (LSH) as a private blocking method to reduce the number of record pair comparisons. Furthermore, we develop parallel PPRL approaches that build on the modern distributed processing framework Apache Flink. We show that our approaches achieve high efficiency and effectiveness, scaling up to linking tens of millions of records.

Second, we address the challenge of achieving high linkage quality in privacy-constrained linkage scenarios. Most existing PPRL approaches rely on a simple threshold-based classification, and thus likely fail to achieve accurate linkage results when dealing with dirty data containing errors and inconsistencies. Therefore, we examine post-processing methods for removing match candidates that are unlikely to match. We show that post-processing raises the overall linkage quality by limiting the number of false matches.

Another problem is that measures to evaluate the quality of record linkage approaches require ground truth data that specifies known matches and non-matches. In many record linkage applications, however, ground truth data is not available. Additionally, in privacy-preserving linkage scenarios, a manual classification (clerical review) is generally not possible since inspecting actual attribute values of classified record pairs can reveal the identity of an individual. Therefore, we propose unsupervised approaches for estimating the quality of linkage results. The estimates can be used in practice, in particular, to optimize linkage configurations, such as the classification threshold.

Third, we focus on the privacy aspect and review hardening techniques that aim to improve the privacy (security) of encoding schemes based on Bloom filters. Bloom filter encodings are frequently used in both research and practical applications. However, several attacks have been proposed showing that Bloom filter encodings are susceptible to cryptanalysis. We comprehensively evaluate the proposed hardening techniques in terms of privacy and linkage quality to assess their practicability and their effectiveness in counteracting attacks.

Finally, we present Primat, an open-source toolbox for the definition and execution of tailored PPRL workflows. Primat offers several components for the different linkage participants that provide state-of-the-art PPRL methods, including various encoding and hardening techniques, LSH-based blocking, and post-processing methods.

Acknowledgments

The completion of this work would not have been viable without the support of numerous people. First, I would like to express my special thanks to my supervisor Prof. Dr. Erhard Rahm, who gave me the opportunity to do my doctorate. In numerous meetings, he supported me in developing my ideas and bringing them to paper. His suggestions always helped me to improve and finally publish my drafts. Thank you for your support, feedback, and guidance during my doctoral time.

I would also like to thank all my colleagues in the database group for their support and the very family-like working atmosphere. First and foremost, I would like to give special thanks to Dr. Victor Christen, with whom I was fortunate to work throughout my doctoral years. Victor's door was always open for an exchange of ideas, some encouragement, or a short off-topic conversation. I thank Victor very much for all his good advice and our excellent collaboration in both research and teaching. I would also like to thank Ziad Sehili and Florens Rohde for the many fruitful discussions about PPRL and for their mutual support. I thank Prof. Dr. Peter Christen from the Australian National University for our successful joint research and the great time we had in Australia. Furthermore, I would like to thank Andrea Hesse for her kind support in numerous organizational and administrative tasks and for reminding us that there are other topics besides computer science.

I owe special thanks to my parents Silke and Peter, my sister Lina, and my aunts who always lovingly supported me throughout my whole life. Last but not least, I would like to thank my dearest wife Sina for her love, patience, and efforts to support and encourage me. Thank you for being my best friend and soulmate in life.

Leipzig, 25.10.2023 Martin Franke

Dissertation-related Publications

- **Martin Franke**, Ziad Sehili, and Erhard Rahm. "Parallel Privacy-preserving Record Linkage using LSH-based Blocking." In: *Proceedings of the 3rd International Conference on Internet of Things, Big Data and Security (IoTBDS)*. SCITEPRESS - Science and Technology Publications, 2018, pp. 195–203. DOI: 10.5220/0006682701950203
- **Martin Franke**, Ziad Sehili, Marcel Gladbach, and Erhard Rahm. "Post-Processing Methods for High Quality Privacy-Preserving Record Linkage." In: *Data Privacy Management, Cryptocurrencies and Blockchain Technology (DPM, CBT)*. Springer, 2018, pp. 263–278. DOI: 10.1007/978-3-030-00305-0_19
- **Martin Franke**, Marcel Gladbach, Ziad Sehili, Florens Rohde, and Erhard Rahm. "ScaDS Research on Scalable Privacy-preserving Record Linkage." In: *Datenbank-Spektrum* 19.1 (2019), pp. 31–40. DOI: 10.1007/s13222-019-00305-y
- **Martin Franke**, Ziad Sehili, and Erhard Rahm. "PRIMAT: A Toolbox for Fast Privacy-Preserving Matching." In: *Proceedings of the VLDB Endowment*. Vol. 12. 12. 2019, pp. 1826–1829. DOI: 10.14778/3352063.3352076
- Florens Rohde, **Martin Franke**, Ziad Sehili, Martin Lablans, and Erhard Rahm. "Optimization of the Mainzelliste software for fast privacy-preserving record linkage." In: *Journal of Translational Medicine* 19.33 (2021). DOI: 10.1186/s12967-020-02678-1
- **Martin Franke**, Ziad Sehili, Florens Rohde, and Erhard Rahm. "Evaluation of Hardening Techniques for Privacy-Preserving Record Linkage." In: *Proceedings of the 24th International Conference on Extending Database Technology (EDBT)*. OpenProceedings.org, 2021. DOI: 10.5441/002/EDBT.2021.26
- Ziad Sehili, Florens Rohde, **Martin Franke**, and Erhard Rahm. "Multi-Party Privacy Preserving Record Linkage in Dynamic Metric Space." In: *Proceedings Datenbanksysteme für Business, Technologie und Web (BTW)*. Vol. P-311. LNI. Gesellschaft für Informatik, 2021, pp. 257–278. DOI: 10.18420/btw2021-13
- Florens Rohde, **Martin Franke**, Victor Christen, and Erhard Rahm. "Value-specific Weighting for Record-level Encodings in Privacy-Preserving Record Link-

age." In: *Proceedings Datenbanksysteme für Business, Technologie und Web (BTW)*. Gesellschaft für Informatik, 2023. DOI: 10.18420/BTW2023-21

- **Martin Franke**, Victor Christen, Peter Christen, Florens Rohde, and Erhard Rahm. "(Privately) Estimating Linkage Quality for Record Linkage." In: *Proceedings of the 27th International Conference on Extending Database Technology (EDBT)*. OpenProceedings.org, 2024

Contents

1 Introduction

1.1 Motivation

Every person produces data. This data can be produced consciously and actively, for example, by writing e-mails, purchasing products in online shops, or using social networks and exchanging messages, images, or video clips. This data can, however, also be produced more passively, for example, through mere birth and thus being registered at the residents' registration office, but also due to hospitalization or by getting employed. All these activities and events produce data that is associated with the person who triggered them. Even if not all data might be of interest at first glance, a wide variety of insights can be gained from analyzing them. For example, important insights into a country's demographics or current epidemiological trends can be derived from the aforementioned examples. However, for a comprehensive analysis of such data, it is often necessary to consider and combine data from different organizations and sources to capture more information and improve data quality [Chr12b; EIV07].

Let us assume that a study on road safety is to be conducted with the research question '*Do people with a mobile phone flat rate cause more traffic accidents with personal injury?*'. Different organizations store specific data related to a traffic accident, including the police, fire department, emergency medical services, as well as insurance companies, such as the health and vehicle insurance of those involved in the accident. Data is also recorded by hospitals or physicians who monitor or provide treatment for injuries. On the other hand, details about the mobile phone contracts of the persons involved are stored by their respective mobile network operators.

Combining the data from different organizations is challenging, especially for person-related data. Records from different databases referring to the same person have to be identified. A record within a particular database can typically be uniquely identified by its primary key. However, the primary key is often not valid as a global identifier across different databases. Additionally, the data stored by the different organizations might be sensitive. Especially, person-related data is subject to strict data protection regulations and laws. In the European Union, privacy and data protection are considered fundamental rights, enshrined in the Charter of Fundamental Rights of the European Union (CFREU) [Eur12]. The right to data protection is implemented within the General Data Protection Regulation (GDPR) [Eur16b], which is one of the strictest data protection and security laws in the world [God17; Gre18]. In Germany, data protection is mainly regulated by the General Right of Personality (Allgemeines Persönlichkeitsrecht, APR) and the Federal Data Protection Act (Bundesdatenschutzgesetz, BDSG), but can also be found in other laws such as the Telecommunications Act (Telekommunikationsgesetz, TKG) or the Telemedia Act (Telemediengesetz, TMG) [Buc12]. Protecting personal data is required, as the disclosure of sensitive or personal data represents an invasion in the person's privacy, and revealed information can lead to disadvantages for the person. For example, a person may be socially or professionally excluded due to the disclosure of a certain illness, or a revealed (e-mail) address may result in the receipt of unsolicited advertising or spam messages.

Techniques that are known as privacy-preserving record linkage (PPRL) have been developed in the last two decades [VCV13; Vat+17; Gko+21] in order to address the challenges of combining (linking) data without revealing any sensitive information about the entities being linked. The general aim of PPRL is to identify records that correspond to the same real-world entity, such as a patient or customer. In addition, the linkage process must preserve the entities' privacy. Thus, the linkage process must not reveal any private or sensitive information or other information that could be used to identify an entity. The general approach of PPRL techniques is to encode or encrypt sensitive identifying information and conduct the linkage using these encoded or encrypted values. The organizations involved in the linkage process only learn which of their records are matches. However, no organization learns any sensitive information about records contained in the databases of other organizations.

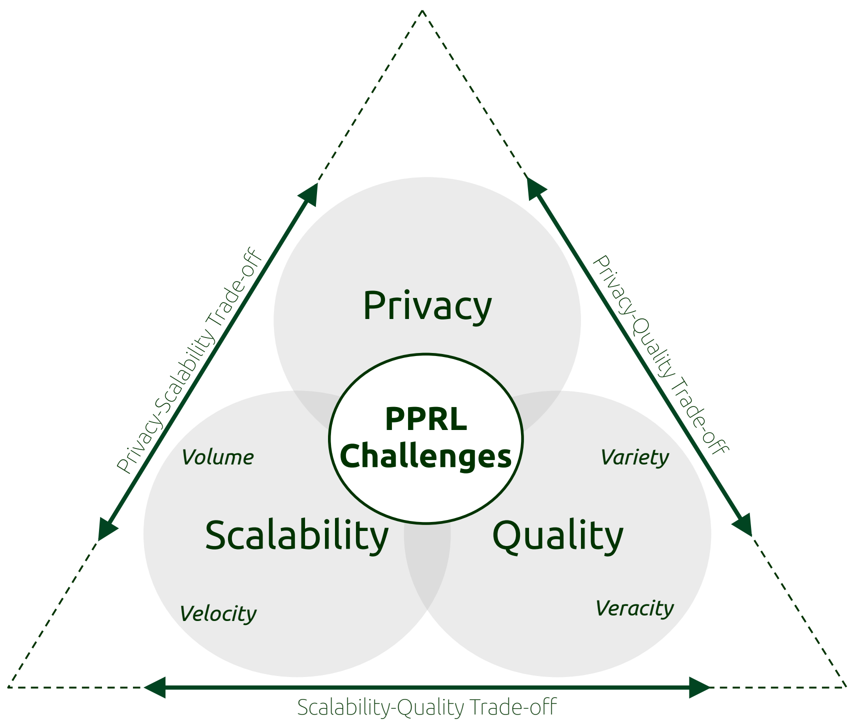
Linking sensitive data from different independent databases that are owned by different autonomous organizations (parties) is an essential task in research, administration, and business to improve data quality or facilitate advanced data analysis. An important application area is the medicine and healthcare domain, in which PPRL techniques are increasingly deployed [Kue+12; Kho+15; Gib+16; Luo+17; Lee+18; YW20; Xu+20]. Various medical care facilities, such as hospitals, medical offices, pharmacies, as well as health insurances hold patient data. Merging data from these different sources can improve the quality of patient care by making all health-related data available. Ideally, this can improve treatments or uncover previously unknown correlations, for example by revealing interactions between different drugs a patient is taking. In addition, costs can be saved because examinations do not have to be initiated again, for example, if medical evidence is missing. Often medical data is also combined with administrative databases or registers from municipal or federal authorities containing information about births, marriages, deaths, crimes, imprisonments, employments and so on [CRS20]. The combination of these data sources allows for a variety of social studies, for example, to analyze the consequences of childhood cancer on mortality or somatic, cognitive, psychological, and socio-economic outcomes [Kue+12].

PPRL techniques are also required by national security agencies and crime investigators to detect fraud or criminal suspects such as terrorists or black market traders [WCA04; JH06; Phu+12; HR15; Can+18; Arp+18; Kas+20]. In such scenarios, sensitive data from law enforcement agencies, financial institutions, internet service providers and businesses, such as airlines or railway companies, need to be combined and analyzed.

However, the task of linking sensitive data is challenging. Basically, the key challenges concern privacy (security), scalability, and quality as illustrated in Figure 1.1. In addition, PPRL is confronted with the characteristics of *Big Data* that are typically described by the five V's, namely ***v**olume*, ***v**elocity*, ***v**ariety*, ***v**eracity*, and ***v**alue*. Together, these properties represent, on the one hand, enormous amounts of data (volume) that need to be processed and analyzed in (near) real-time, often in the form of dynamic data streams (velocity). On the other hand, the data often originates from various databases and shows heterogeneous formats, structures, and data types (variety). In addition, the data is of varying quality and possibly contains errors, inconsistencies, bias, and noise (veracity). Finally, the analysis of the data should lead to new insights that enable predictions or process optimizations (value).

The three key challenges of PPRL are characterized as follows:

Privacy: Sensitive data must be protected against unauthorized access to avoid misuse and negative consequences for the individual. Sharing or exchanging sensitive (personal) data between different organizations is subject to privacy concerns and corresponding laws, regulations, and policies. Therefore, databases containing sensitive data need to be linked in such ways that no sensitive or confidential data is revealed during the linkage process. The protection of privacy is crucial for the entire linkage process, making the linkage task even more challenging. To fulfill this requirement, it is necessary to encode (mask) the records and to conduct the linkage only on the encoded data. However, the encoding has to preserve specific properties and relationships of the records that are needed for accurate linkage. Typically, this requires encoding techniques that preserve the similarities between records. At the same time, the encoding should make it infeasible to re-identify sensitive values or individual from the encoded (masked) records. Furthermore, the linkage must be performed according to specific protocols that



Privacy
PPRL Challenges
Scalability
Quality
Volume
Velocity
Variety
Veracity
Privacy-Scalability Trade-off
Privacy-Quality Trade-off
Scalability-Quality Trade-off

Figure 1.1: The three key challenges of PPRL.

regulate how parameters and data are exchanged between organizations participating in the linkage.

Quality: Identifying records referring to the same entity is complicated by heterogeneous, erroneous, outdated, and missing data [HS98; Chr12b; CRS20]. Thus, the exact matching of values will likely lead to low-quality linkage results, as already small differences between matching records, for instance, typos or different formats, would cause them to be missed. Therefore, PPRL techniques must be able to handle such real-world *dirty data* by supporting approximate matching techniques.

Scalability: As can be seen from the application scenarios described before, potentially many different organizations (data owners) are involved in a linkage process. In addition, each database can capture a large part of the population of a country or worldwide and thus contain tens of millions of records. For instance, the German telecommunications company *Deutsche Telekom AG* manages over 53 million mobile phone customers in Germany and over 248 million worldwide [Deu23]. Germany's largest public health insurance organization *AOK* (Allgemeine Ortskrankenkassen) also manages over 27 million customers, or roughly one-third of the German population [Bun23]. As a consequence, PPRL techniques should scale to millions of records from two or more data owners (organizations). The trivial approach to perform the linkage is to compare every possible pair of records from the input databases. Considering only two databases, this approach would result in a number of comparisons equal to the product of the sizes of the two databases. For example, if both databases contain 100 000 records, this would result in $100\,000 \cdot 100\,000 = 10^{10}$ comparisons. Even if 100 000 comparisons are performed in one second, it would take over one day ($27.78\,h$) to conduct the linkage. Consequently, the trivial comparison of all possible record pairs is not feasible in real-world applications. For this reason, the search space must be restricted by excluding dissimilar records from further comparisons at an early stage. To further improve scalability for large datasets, another option is to perform the linkage in parallel or distributed environments.

Given these key challenges, PPRL approaches should be capable of:

(1) preserving the privacy of individuals by protecting sensitive data from being revealed,

(2) correctly identify matching records to achieve high linkage quality, and

(3) efficiently processing large numbers of records from potentially many databases.

However, these three key challenges mutually influence each other, resulting in trade-offs that must be carefully balanced. To achieve a high linkage quality, encoding techniques need to enable approximate matching on encoded values in order to obtain approximate similarities between records, which is important to handle dirty data. However, the more properties of the original (plaintext) values are preserved by the encoding, the more vulnerable the encoding is to attacks as there are more properties to analyze and exploit. In contrast, encodings with high security (privacy) guarantees tend to incur high computational or communication costs, which limits their usability for linking large or many databases. Finally, if too many record pairs are excluded from a detailed comparison in order to achieve scalability, then this likely will reduce the linkage quality as matches with variations or inconsistencies might not be identified.

Current research does not thoroughly address the challenges identified. The scalability of existing PPRL approaches is still limited, even when using methods that efficiently reduce the number of record pair comparisons. In fact, large datasets containing millions of records will still lead to unacceptably long runtimes for real-world applications. Moreover, achieving high linkage quality is even more challenging on encoded data, in particular in the presence of errors or inconsistencies. At the same time, databases that cover a large part of a population will likely include structures such as families and households. It is also very likely that multiple individuals share common names, such as the surnames ‘Müller’ and ‘Meier’ in German-speaking countries, or ‘Smith’ and ‘Johnson’ in English-speaking countries. Both effects are hard to deal with in common PPRL approaches that rely on a simple threshold-based classification approach to decide whether two records are considered to match.

In terms of privacy, common PPRL techniques are known to disclose pieces of information that can be exploited through cryptanalysis or attack methods to gather sensitive information or even enable the re-identification of individuals. Finally, despite the large number of proposed PPRL methods, their practical use in real-world applications is limited due to the absence of convenient and powerful tools that allow easy implementation and configuration of appropriate PPRL approaches. This thesis addresses the identified problems and challenges through the contributions described in the following sections.

1.2 Scientific Contributions

In the following, we describe the scientific contributions made in this dissertation.

Speedup PPRL by Blocking and Distributed Processing

In order to improve the scalability of PPRL, we propose parallel PPRL (P3RL) approaches that are realized using a modern distributed processing framework to enable the utilization of large shared-nothing computer clusters. Ideally, this will speed up PPRL workflows in proportion to the number of CPUs in the cluster. In addition, we utilize blocking techniques that are able to efficiently reduce the number of record pair comparisons. In particular, we investigate blocking based on locality-sensitive hashing (LSH), a probabilistic blocking method that is applied solely on encoded data and thus reveals no additional information. Furthermore, we include optimizations for the LSH-based blocking, such as to avoid redundant record pair comparisons. We comprehensively evaluate the quality, efficiency, and scalability of our P3RL approaches for different parameter settings and large datasets with up to 16 million records in a cluster environment with up to 16 worker nodes. The P3RL approaches and the evaluation results were presented at the IoTBDS 2018 and published in the conference proceedings [FSR18].

LSH-based Blocking on Attribute-level Encodings

By default, LSH-based blocking requires that each record is transformed into a single encoding (record-level encoding). In contrast, real-world use cases may require generating multiple encodings per record - typically one encoding for each attribute of the record (attribute-level encoding). Such approaches tend to be more susceptible to cryptanalysis, but also often result in a higher linkage quality. In some use cases, this compromise must be made, especially in bio-medical application scenarios when achieving a very high linkage quality is of primary importance. Therefore, we investigate LSH-blocking schemes that can be applied on attribute-level encodings. We implement these approaches within the Mainzelliste, a web-based pseudonymization service that is used in various medical joint research projects. This work was published in 2021 in the Journal of Translational Medicine [Roh+21].

Post-processing Methods for Resolving Multiple Match Candidates

Obtaining a high linkage quality is one of the key challenges of PPRL. Ideally, a record linkage approach should find all matches (pairs of records referring to the same entity), despite possible data quality problems, like erroneous, outdated, or incomplete data, in the source databases. At the same time, false matches (two records referring to two different entities) should be avoided as much as possible, as otherwise conclusions based on incorrect assumptions may be drawn. Besides data quality issues, there are many factors that significantly influence the linkage quality, for instance, the encoding method or the blocking approach as well as their respective parametrization. Databases that contain many non-matching record pairs with a high similarity are particularly difficult to link. However, assuming that the source databases do not contain duplicates (there are no two records within a database that refer to the same real-world entity), then one record of a database can only match at maximum one record of another database. Therefore, we propose to add a post-processing step to the linkage process that aims to satisfy this property by selecting the record pairs that most likely match. We investigate different post-processing strategies and comparatively evaluate them on different datasets. The evaluation shows that applying post-processing can significantly increase the linkage quality. The approach was presented at the Workshop on Data Management and Privacy (DPM) held in conjunction with the ESORICS 2018 conference and published in the workshop proceedings [FSR18].

(Privately) Estimating Linkage Quality for Record Linkage

In practical linkage projects, assessing the quality of linkage results is a challenging endeavor. Commonly used performance measures, such as precision and recall, require a ground truth dataset containing true matching and true non-matching record pairs. Such ground truth data is often not available or incomplete and must be prepared manually, which is time- and resource-consuming. A manual inspection of (sampled) record pairs by domain exports during a clerical review process is also often not feasible, in particular when linking sensitive (personal) data or very large databases. In privacy-preserving linkage scenarios, a manual inspection of actual attribute values of matching or non-matching record pairs would violate the privacy of individuals since they can be identified in the databases to be linked. To overcome these problems, unsupervised techniques for assessing the linkage quality are needed that do not require ground truth data. Therefore, we review existing and propose improved unsupervised approaches for estimating the quality of linkage results. An evaluation using datasets from different domains shows that our novel approaches outperform existing methods and lead to accurate estimates. This work will be presented at the EDBT 2024 conference and published in the conference proceedings [Fra+24].

Analysis of the Privacy Properties of PPRL Encoding Techniques

Evaluating the privacy protection achieved by a PPRL method is of high importance. However, in contrast to linkage quality and scalability, privacy is a concept that is difficult to define, measure, and evaluate [WE18]. A widely used encoding technique for PPRL is based on Bloom filters [SBR09; SBR11]. Bloom filter encodings have been used in a variety of PPRL approaches in both research [VCV13; Gko+21] and real-world linkage applications [Ran+14; BRF15; Pow+17; Pit+18]. Several variants of such Bloom filter encodings have been proposed, in particular, to improve their resilience against cryptanalysis attacks [Sch15; Chr+18a]. We explore and categorize different Bloom filter encoding schemes and evaluate them in terms of privacy protection and linkage quality outcome. For this purpose, we propose measures that allow quantifying the privacy properties of different Bloom filter encoding variants. This work was presented at the EDBT 2021 and published in the conference proceedings [Fra+21].

A Toolbox for Fast Privacy-preserving Matching

There is a need for freely available software tools that support the implementation and configuration of PPRL workflows. For both, researchers and practitioners, such tools are important to better understand different linkage algorithms and to allow to experiment with different methods and their parameters. Ideally, this allows a comparative evaluation of different methods and thus the identification of suitable approaches for a specific application scenario. Therefore, we develop an open-source PPRL toolbox, the ***Pri**vate* ***Ma**tching* ***T**oolbox* (PRIMAT). It includes various state-of-the-art encoding and linking techniques covering the entire PPRL process, thus reducing the effort to deploy PPRL in academic or practical projects. PRIMAT also offers an evaluation framework to consistently compare PPRL methods in terms of linkage quality, privacy, and scalability. The toolbox was demonstrated at the VLDB 2019 and a description of its architecture and core features was published in the conference proceedings [FSR19].

1.3 Structure of Thesis

The remainder of this dissertation is structured as follows. We begin by discussing the preliminaries of privacy-preserving record linkage and relevant techniques in Chapter 2. In Chapter 3, we propose parallel PPRL approaches utilizing locality-sensitive hashing (LSH) for blocking and Apache Flink as a distributed execution engine to address the scalability challenge. In Chapter 4, we present approaches for LSH-based blocking on attribute-level encodings and their integration into a well-known pseudonymization software for bio-medical projects. Then, in Chapter 5, we investigate post-processing methods for resolving multiple match candidates to achieve high linkage quality in scenarios where the databases to be linked do not contain intra-source duplicates. Furthermore, we examine heuristics for estimating linkage quality in the absence of ground truth data in Chapter 6. Then, Chapter 7 deals with the assessment of the privacy properties of recently used encoding techniques for PPRL. In Chapter 8, we present our open-source toolbox PRIMAT, which contains various state-of-the-art encoding and linkage techniques to support the implementation of PPRL in academic and practical projects. Finally, we summarize our contributions and discuss future research directions in Chapter 9.

2 Background and Related Work

This chapter provides background knowledge that relates to our research problems and is useful for understanding the following chapters of this dissertation. We start with a short historical overview of record linkage in Section 2.1. In Section 2.2, we outline the legal background showing the need for privacy-preserving linkage techniques. Then, we provide a formal problem definition of privacy-preserving record linkage in Section 2.3, and an analysis of its complexity in Section 2.4. In Section 2.5, we discuss properties of attributes and recall the definitions for different types of identifiers. Afterward, we describe the general PPRL process in Section 2.6. In Section 2.7, we discuss protocols for conducting the linkage. Finally, in Section 2.8, we discuss encoding techniques based on Bloom filters which are currently most commonly used for PPRL.

2.1 Historic Overview of Record Linkage

The first scientific considerations of record linkage can be traced back to about the second half of the 1940s [Dun46]. At that time, record linkage was mainly used in the context of census data and demographics, for instance, for the registration of life events such as birth, marriage, and death [NK62; New67]. But also medical studies, such as the investigation of genetic defects or hereditary diseases, have already been considered as an area of application for record linkage [New+59].

The mathematical foundations of record linkage were given in the seminal work of Fellegi and Sunter when they introduced the probabilistic record linkage model [FS69]. Over time, numerous approaches for record linking have been developed, mainly focused on achieving high linkage quality and scalability to large datasets [Chr12b].

When linking personal data, data protection regulations and the privacy of individuals must be taken into account. Starting in the mid-1990s, this has led to the development of techniques, known as privacy-preserving record linkage (PPRL), which allow linkage without the need to exchange sensitive (plaintext) data between the linkage participants [DQB95; Qua+98].

2.2 Legal Background

Privacy is considered a universal human right, enshrined in Article 12 of the United Nations Universal Declaration of Human Rights [Uni48] that states:

> “ No one shall be subjected to arbitrary interference with his privacy, family, home or correspondence, nor to attacks upon his honour and reputation. Everyone has the right to the protection of the law against such interference or attacks. ”

Many national constitutions recognize the right to privacy that aims to restrict governmental and private actions that threaten the privacy of an individual. In the European Union, the right to privacy is enshrined in Article 8 of the European Convention of Human Rights (ECHR) [Cou50] and in Article 7 of the European Charter of Fundamental Rights (CFREU) [Eur12]. For instance, Article 8 of the ECHR states:

> “ 1. Everyone has the right to respect for his private and family life, his home and his correspondence.
>
> 2. There shall be no interference by a public authority with the exercise of this right except such as is in accordance with the law and is necessary in a democratic society in the interests of national security, public safety or the economic well-being of the country, for the prevention of disorder or crime, for the protection of health or morals, or for the protection of the rights and freedoms of others. ”

From the right to privacy arises the right to data protection that is enshrined in Article 8 of the CFREU:

> “ 1. Everyone has the right to the protection of personal data concerning him or her.
>
> 2. Such data must be processed fairly for specified purposes and on the basis of the consent of the person concerned or some other legitimate basis laid down by law. Everyone has the right of access to data which has been collected concerning him or her, and the right to have it rectified.
>
> 3. Compliance with these rules shall be subject to control by an independent authority. ”

The right to data protection (also known as data privacy) is implemented by the General Data Protection Regulation (GDPR) [Eur16b] of the European Union. Any organization that processes or stores the personal data of individuals must comply with the data protection and security requirements of the GDPR. According to Article 4 of the GDPR, personal data is “any information relating to an identified or identifiable natural person" [Eur16b]. An identifiable natural person is defined as "one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person" [Eur16b].

The rights to privacy and data protection need to be carefully balanced against other human rights or interests such as freedom of information, freedom of research (academic freedom), national security, and crime prevention [Cou50; Eur12]. For instance, airlines and travel agencies collect passenger name records (PNRs) that consist of the personal information of a passenger (e. g., full name, address, date of birth, and place of birth), the itinerary and other travel-related information (e. g., ticketing details, payment details, meal requirements). PNRs are routinely shared with government agencies, so-called Passenger Information Units, on the legal basis of directive 2016/681 of the EU parliament for the prevention, detection, investigation, and prosecution of terrorist offenses and serious crimes [Eur16a]. Basically, PNRs are matched to records from databases of wanted persons to identify persons that are linked to terrorist offenses or serious crimes. An example of such a database is the Schengen Information System (SIS) which stores data about persons who are legally not allowed to enter the Schengen area, are wanted for criminal activities, or are missing.

2.3 PPRL Problem Definition

Let $DO_1, \dots, DO_p$, with $p \geq 2$, be p database owners with their respective databases $D_1, \dots, D_p$. Let each database D_i be a single relation over sets $A^i_1, \dots, A^i_{\nu_i}$, that is $D_i \subseteq A^i_1 \times \dots \times A^i_{\nu_i}$, where $\times$ denotes the Cartesian product. Each set A^i_j represents an attribute and consists of all valid attribute values. We term the set of all attributes $\mathcal{A}_i = \{A^i_1, \dots, A^i_{\nu_i}\}$ the schema of database D_i. Each database D_i consists of records $r^i_1, \dots, r^i_{n_i}$, where $n_i = |D_i|$ denotes the number of records in database D_i (database size). Each record $r^i_j = (a^{i,j}_1, \dots, a^{i,j}_{\nu_i}) \in D_i$ is a ν_i-tuple consisting of attribute values $a^{i,j}_k \in A^i_k$ for $j \in \{1, \dots, n_i\}, k \in \{1, \dots, \nu_i\}$. To access the k-th attribute value of record r^i_j we use the notation $\pi_k(r^i_j)$ (projection on k-th component). Similarly, to restrict a record to the values of the attributes in $X \subset \mathcal{A}_i$ we write $\pi_X(r^i_j)$ or simply $r^i_j[X]$. This projection operation will discard (exclude) the attribute values that correspond to the other attributes not contained in X.

We assume that each record represents a real-world entity [Bey04; Che76] defined as a distinctly identifiable individual of the real world, such as a patient or a customer. As a consequence, the set of attributes $\mathcal{A}_i$ contains person-related attributes, such as first name, last name, date of birth, place of birth, gender, or address, as well as other context-dependent attributes containing content or payload data, for instance, medical data of patients (e. g., disease, medication, or body mass index).

Privacy-preserving record linkage takes as input the databases $D_1, \ldots, D_p$ and determines which of their records match. Therefore, a decision model (classifier) $\mathcal{DM}$ is used that assigns all possible candidate record pairs to the sets (classes) `M` of matches and `N` of non-matches. The set of all possible candidate record pairs `C`, the set of matches `M` and the set of non-matches `N` are defined as follows:

$$\mathtt{C} = \bigcup_{\substack{i,j \in \{1,\ldots,p\} \\ i \neq j}} D_i \times D_j = \bigcup_{\substack{i,j \in \{1,\ldots,p\} \\ i \neq j}} \{(a,b) \mid a \in D_i, b \in D_j\} \tag{2.1}$$

$$\mathtt{M} = \bigcup_{\substack{i,j \in \{1,\ldots,p\} \\ i \neq j}} \{(a,b) \mid a \overset{\mathcal{DM}}{\equiv} b, a \in D_i, b \in D_j\} \tag{2.2}$$

$$\mathtt{N} = \bigcup_{\substack{i,j \in \{1,\ldots,p\} \\ i \neq j}} \{(a,b) \mid a \overset{\mathcal{DM}}{\not\equiv} b, a \in D_i, b \in D_j\}. \tag{2.3}$$

With $\overset{\mathcal{DM}}{\equiv}$ we denote the equivalence relation under decision model $\mathcal{DM}$. Thus, $a \overset{\mathcal{DM}}{\equiv} b$ means that under decision model $\mathcal{DM}$ records a and b are considered to correspond to the same real-world entity. In contrast, two records $r_1 \in D_i$ and $r_2 \in D_j$ are considered equal ($r_1 = r_2$), if they have the same number of attributes and each attribute has the same value, i. e.,

$$r_1 = r_2 \iff \nu_i = \nu_j \;\wedge\; \forall k \in \{1,\ldots,\nu_i\} : [\pi_k(r_1) = \pi_k(r_2)] \tag{2.4}$$

In most scenarios, it is safe to assume that if two records are equal they also correspond to the same real-world entity, i. e.,

$$r_1 = r_2 \quad \Rightarrow \quad r_1 \equiv r_2 \tag{2.5}$$

Any decision model must be able to correctly classify record pairs of this trivial case (exact matches). However, since attribute values can be erroneous, missing, or out of date (dirty data), the opposite does not apply: if two records are not equal ($r_1 \neq r_2$), then this does not imply that the two records do not correspond to the same real-world entity. Accordingly, instead of conducting an equality check between attribute values, the decision model must conduct an approximate comparison that yields a similarity value (score). By using these similarity values, the decision model determines the probability that two records (approximately) match.

In contrast to traditional record linkage, the aim of PPRL is to ensure that the actual (plaintext) attribute values of the records contained in databases $D_1, \ldots, D_p$ are not disclosed during the linkage process. This means that no database owner DO_i nor any external party learns the actual attribute values of any record $r_k^j \in D_j$ with $i, j \in \{1, \ldots, p\}, i \neq j, k \in \{1, \ldots, n_j\}$.

In fact, no information should be disclosed that can be used to identify an individual (represented by a database record). At the end of the linkage process, the database owners only learn the set of matching record pairs in `M`. This typically means that only the record identifiers of the matched pairs in `M` and some selected attribute values of these records are shared between the database owners or with an external party, such as a research institute.

2.4 PPRL Computation Complexity

The complexity of PPRL depends on the number of record pairs to be compared. For simplicity, let each database D_i of database owner DO_i with $i \in \{1, \ldots, p\}$ contain $|D_i| = n$ records. The naive approach for conducting the linkage is to compare all possible pairs of records. Let $p = 2$, which corresponds to the linkage of the two databases D_1 and D_2. As can be seen from Equation 2.1, the number of all possible pairs of records is equal to the cardinality of the Cartesian product between databases D_1 and D_2. Consequently,

$$|D_1 \times D_2| = |D_1| \cdot |D_2| = n \cdot n = n^2 \tag{2.6}$$

record pairs must be compared with each other, which leads to a quadratic complexity of $\mathcal{O}(n^2)$. Similarly, for $p > 2$,

$$\sum_{\substack{i,j \in \{1,\ldots,p\} \\ i \neq j}} |D_i \times D_j| = \sum_{\substack{i,j \in \{1,\ldots,p\} \\ i \neq j}} |D_i| \cdot |D_j| = \sum_{\substack{i,j \in \{1,\ldots,p\} \\ i \neq j}} n^2 \tag{2.7}$$

record pairs must be compared leading to a complexity of

$$\mathcal{O}\left(\binom{p}{2} \cdot n^2\right) = \mathcal{O}\left(\frac{p \cdot (p-1)}{2} \cdot n^2\right) = \mathcal{O}\left(\left(\frac{p^2}{2} - \frac{p}{2}\right) \cdot n^2\right) = \mathcal{O}\left(p^2 \cdot n^2\right). \tag{2.8}$$

2.5 Keys and Identifiers

Keys and identifiers are used to uniquely identify a record (tuple) in a database (relation). In the following, different types of keys and identifiers are described, and their formal definitions are given. Before, we need to recall the definition of a functional dependency.

Functional dependency: Given a database D_i (single relation) with schema $\mathcal{A}_i$ and sets of attributes $X, Y \subseteq \mathcal{A}_i$, then X is said to functionally determine Y, denoted as $X \rightarrow Y$, if

$$\forall r_1, r_2 \in D_i : [r_1[X] = r_2[X] \;\; \Rightarrow \;\; r_1[Y] = r_2[Y]] \, . \tag{2.9}$$

That is, all records (tuples) $r_1, r_2 \in D_i$ with equal values for the X-attributes also have equal values for the Y-attributes. Thus, the values of the attributes from attribute set X uniquely determine the values of the attributes from attribute set Y.

Superkey: A set of attributes $K := \{A^i_1, \dots, A^i_k\} \subseteq \mathcal{A}_i$ is called superkey for schema $\mathcal{A}_i$ of database D_i if it uniquely identifies each record (tuple) contained in database D_i, that is,

$$\forall r_1, r_2 \in D_i : [r_1 \neq r_2 \;\; \Rightarrow \;\; \exists A' \in K : r_1[A'] \neq r_2[A']] \tag{2.10}$$

which is equivalent to $K \rightarrow \mathcal{A}$. According to the definition of a relation as a set of records (tuples), the set of all attributes is always a superkey. Otherwise, there could be two identical records in a relation, which would violate the definition of a set (collection of distinct values).

Candidate key: A set of attributes $K := \{A^i_1, \dots, A^i_k\} \subseteq \mathcal{A}_i$ is called candidate key for schema $\mathcal{A}_i$ of database D_i, if

$$K \rightarrow \mathcal{A} \quad \wedge \quad \forall A' \in K : [K - \{A'\} \not\rightarrow \mathcal{A}] \tag{2.11}$$

Thus, a candidate key is a minimal superkey. Therefore, if an attribute is removed from the candidate key, it is no longer a super key. On the other hand, if an attribute is added, then it is no longer minimal, but still a superkey.

Primary key: A primary key is a selected candidate key that is used as the main reference key. The decision is typically made by the designer of the database. Often, surrogate keys are used as primary keys in many real-world databases [Dat04]. Unlike a natural primary key, a surrogate key has no descriptive value and does not consist of real-world observations related to the real-world individual. Typically, surrogate keys are generated internally by the database system.

Prime attribute: An attribute that is present in any of the candidate keys is called a prime attribute.

Direct identifier: A prime attribute that is the only member of a candidate key is called a direct identifier.

Indirect identifier: A prime attribute that is not the only member of any candidate key is called an indirect identifier or quasi-identifier.

Example: Let $\mathcal{A} = \{\alpha, \beta, \gamma, \delta\}$ be the database schema with the functional dependencies $\{\alpha, \beta, \gamma\} \rightarrow \{\delta\}, \{\alpha, \beta\} \rightarrow \{\gamma, \delta\}$ and $\{\delta\} \rightarrow \{\alpha, \beta, \gamma\}$. Then, $\{\alpha, \beta, \gamma\}, \{\alpha, \beta\}$ and $\{\delta\}$

are superkeys but only $\{\alpha, \beta\}$ and $\{\delta\}$ are candidate keys. α, β and δ are prime attributes. δ is the only direct identifier while α, β and γ are indirect identifiers.

2.5.1 Lack of Global Identifiers

In many (research) projects it is desired to analyze data about individuals that is scattered across different independent databases. While each particular database (relation) typically has a primary key (and possibly other candidate keys) to uniquely identify records about individuals, these are generally only valid locally. This is due to the fact that the databases are maintained by different independent organizations that store different data depending on their operative business.

If a global identifier (also known as global candidate key or entity identifier), such as a unique personal identification number, is available in the databases to be matched, then the linkage becomes trivial and can be performed as a database join. However, not every country has issued a national identification number that could be used as a global identifier. A reason for this are privacy concerns regarding such national identification numbers, as they allow easy consolidation of data from different areas of life, facilitating the construction of person profiles. Such national identification numbers can also be exploited and misused by criminals, for instance, to steal the identity of a person. Besides, even national identification numbers can contain errors or might not be consistent over time. There also might be persons without or with several national identification numbers.

In Germany, for instance, there is no general-purpose national identification number. While there are sector-specific identifiers, their usage is legally and practically restricted to certain areas and purposes. Examples include the taxpayer identification number (where persons that are both employees and self-employed at the same time can have two identifiers), the social security number (issuance upon commencement of first employment), retirement insurance number, health insurance number, or the personal identification number for members of the German Armed Forces. In addition, organizations and authorities (government agencies) often operate on a decentralized basis. For example, due to the federal structure in Germany, administrative databases are usually not maintained on a nationwide basis, but at the level of a specific administrative unit (state, district, municipality). Also, the data of the health care system is scattered over around 1900 hospitals [Sta23b] and 170 health care insurance companies [Bun23].

Since global identifiers are often missing, the linkage needs to be conducted on indirect identifiers (quasi-identifying attributes) which are common in the databases to be linked. Even if global identifiers are present, a linkage is still required for individuals with inaccurate or missing global identifiers.

2.5.2 Indirect Identifiers and Data Quality Problems

The task of linkage relies on comparing common indirect identifiers (quasi-identifying attributes) in order to classify record pairs as matches and non-matches. In general, these are person-related attributes, such as name, date of birth, place of birth, gender, or address. Since indirect identifiers can reveal the identity of a person, these identifiers have to be protected against unauthorized access and should not be disclosed during the linkage process. Indirect identifiers are considered as 'dirty' containing inconsistencies and errors [HS98; Chr12b; CRS20] and thus hampering the linkage process.

The reasons for inconsistencies and errors in indirect identifiers are manifold [RD00; Chr12b]. Data that is collected by different organizations will likely be inconsistent as the data is recorded and processed at different points in time by different systems using different input modes and data representations. Moreover, each organization processes different data depending on its area of business. However, what data is relevant and useful for an organization can also change over time, e. g., to comply with new regulations or due to business reorganizations.

As a consequence, the database schemes $\mathcal{A}_1, \ldots, \mathcal{A}_p$ of the databases $D_1, \ldots, D_p$ to be linked will likely be different. In this respect, a distinction is typically made between naming and structural conflicts [RD00]. Naming conflicts include cases where the same name is used for different attributes (homonyms), as well as cases where different names are used for the same attribute (synonyms). Structural conflicts, in contrast, refer to a different representation of the same attribute(s), e. g., different formats, data types, or component structures. However, even attributes with the same name and type can use different representations for a value. For instance, the gender of a person could be represented with the string values 'F', 'M', 'O' or 'FEMALE', 'MALE', 'OTHER'. Therefore, the database owners likely need to conduct a (privacy-preserving) schema matching before linkage [Sca+07]. Schema matching aims at identifying semantic correspondences between the schema elements of different databases [BBR11]. Typically, each database schema $\mathcal{A}_i$ is mapped into a global database scheme $\mathcal{A}_G$ which is then used for the linkage. Often, the schema matching results in a limited number of indirect identifiers that are available among all databases and thus usable for linkage.

The causes of data quality problems are also very diverse and include, for example, bad database design (e. g., missing integrity constraints), poor recording processes (e. g., phone calls, handwritten forms, scanned documents), as well as changing data needs (e. g., new formats, additional attributes, removed attributes) [Chr12b]. Some typical data quality problems are described in the following [RD00; LR96; Sna07].

Missing values: Attribute values that are missing are typically represented by a null value or with an empty string. There are different reasons for missing attribute values [CRS20]. In some cases, the value is missing because it simply does not exist for each entity, e. g., not every person has a middle name. In other cases, the value is missing because it is not (exactly) known. A common practice is to insert default or dummy values if an attribute value is missing. For instance, if the day and month of birth are not exactly known for a person, then it might be set to the 1st of January.

Spelling variations: In general, spelling variations result from the fact that in many languages there is no one-to-one correspondence (bijection) between phonemes (units of sound) and graphemes (written symbols or letters). In English, for instance, the phoneme /f/ can correspond to the graphemes 'f' (as in fast), 'ff' (as in off), 'ph' (as in graph), and 'gh' (as in rough). In German, the phoneme /k/ can correspond to the graphemes 'c', 'k', 'q(u)'. A typical source of error is the dictation of information on the telephone, where, for example, certain spellings are assumed and no spelling clarification is requested.

Phonetic errors: In contrast to spelling variations, phonetic errors change the phonetic structure of a word. These errors are often caused by typos or by mishearing, e. g., 'Merlin' instead of 'Martin', or by mixing up similar sounding phonemes, such as /t/ and /d/, or /p/ and /b/.

Name variations: While most normal words have only one correct spelling, names often have several valid variations. For example, the frequent German last names 'Müller' and 'Schmidt' have several variants, such as 'Mueller', 'Mühler', 'Möller', 'Miller', or 'Schmitt', 'Schmied', 'Schmitz', 'Schmiedel', 'Schmidtke', respectively.

Embedded values: One or multiple attribute values are embedded in one attribute, for example, both the first name and the middle name are entered in one field. This problem typically occurs due to free-form fields.

Attribute transposition: An attribute value is associated with the wrong attribute, or the values of two attributes are interchanged. This problem often occurs due to unintuitive, ambiguous, or unclear forms, for example, the values for first name and last name are interchanged.

Outdated values: Information can change over time and is likely to be recorded (or updated) in different databases at different points in time. Especially person-related attributes can change over time as persons can change their name, for instance, due to marriage or divorce. Also, address attributes, such as city, zip code, or street, are not stable over time and therefore often not up-to-date in each database.

OCR errors: Optical character recognition (OCR) errors occur when documents or forms are scanned and similar-looking characters are wrongly recognized, e. g., 1/i, 2/Z, 3/B, 4/A, 5/S, 6/G, l/I, or q/g.

Abbreviations: Abbreviations are often used out of convenience or due to a limitation in the maximum number of characters allowed in an input field. Again, names are particularly affected as only initials can be used instead of (given) names, or components of compound names, such as 'Hans-Peter' or 'Müller-Wohlfahrt', can be omitted. Moreover, nicknames, such as 'Vic' for 'Victor' or 'Chris' for 'Christopher', as well as alias (pen, stages) names could be used.

2.6 PPRL Process

The general linkage process consists of multiple steps as shown in Figure 2.1. Without loss of generality, we describe the linkage process for two database owners (parties) A and B with their corresponding databases D_A and D_B. However, the general process can easily be extended to the linkage of multiple (more than two) databases. The aim of the PPRL process is to identify which records in the database D_A of party A refer to the same real-world entity as records in the database D_B of party B. The individual steps of the linkage process are considered more closely in the following subsections.

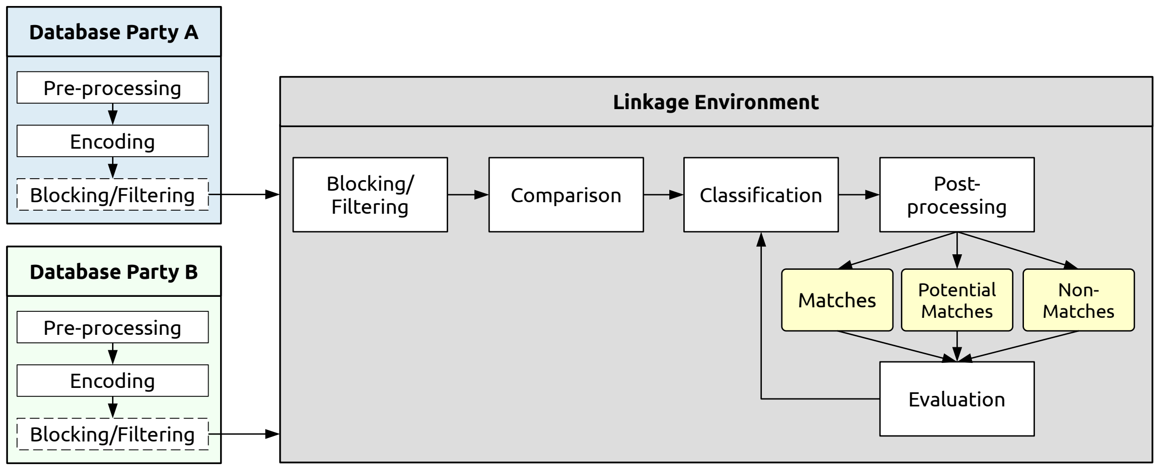


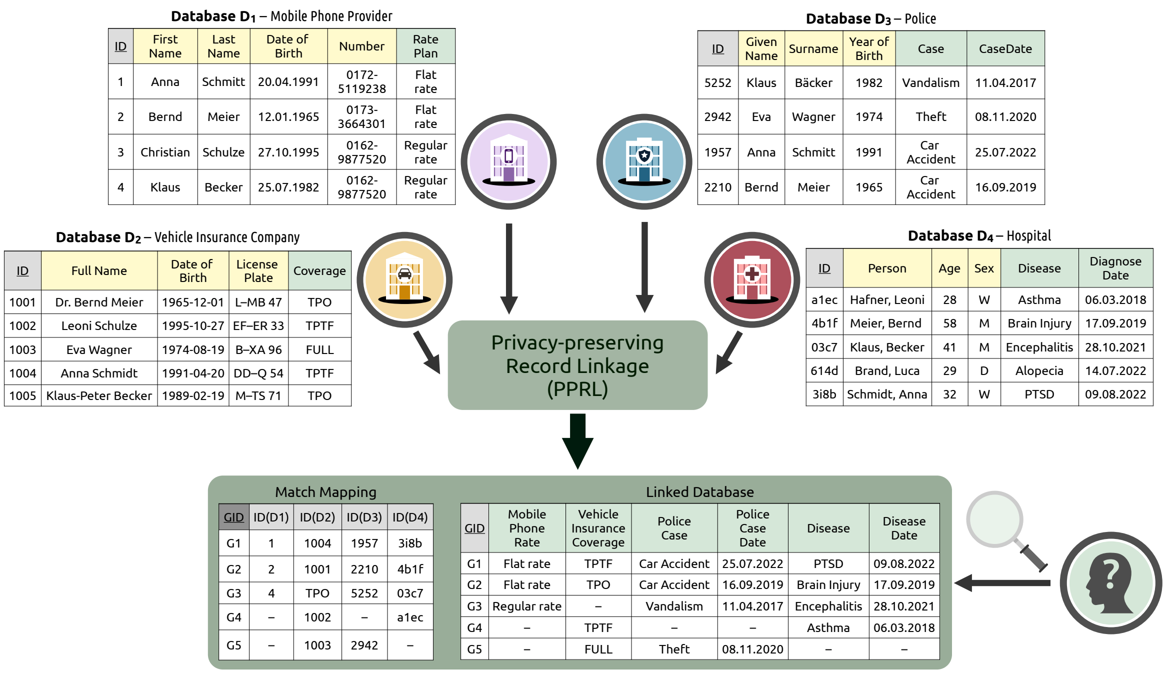
Figure 2.1: PPRL process for two database owners (parties). Dotted line boxes indicate optional steps.

The linkage process can be conducted under different protocols depending on the specific use case and its privacy requirements [CRS20]. The different linkage protocols are described in Section 2.7.

Before performing the linkage, the database owners must agree on the methods to be used, as well as on various parameters. This includes, in particular, the attributes to be used for linkage, i. e., for deciding whether two records refer to the same real-world entity or not. For this purpose, the database owners have to identify attributes within their database schemes that contain the same type of information (semantic correspondences).

As described in Section 2.5.2, this process is known as schema matching [Sca+07; BBR11] and is complicated by naming conflicts, such as synonyms (e. g., surname vs. last name), and structural conflicts, for instance, different representations for names or addresses (e. g., individual attributes vs. compound attribute).

The difference between the PPRL process in contrast to the traditional record linkage process is that privacy protection must be taken into account in each step. Therefore, the database owners encode (encrypt) their records locally and only exchange encoded records. Thus, the actual linkage is performed on encoded records only.



Database D₁ – Mobile Phone Provider

ID	First Name	Last Name	Date of Birth	Number	Rate Plan
1	Anna	Schmitt	20.04.1991	0172-5119238	Flat rate
2	Bernd	Meier	12.01.1965	0173-3664301	Flat rate
3	Christian	Schulze	27.10.1995	0162-9877520	Regular rate
4	Klaus	Becker	25.07.1982	0162-9877520	Regular rate

Database D₂ – Vehicle Insurance Company

ID	Full Name	Date of Birth	License Plate	Coverage
1001	Dr. Bernd Meier	1965-12-01	L–MB 47	TPO
1002	Leoni Schulze	1995-10-27	EF–ER 33	TPTF
1003	Eva Wagner	1974-08-19	B–XA 96	FULL
1004	Anna Schmidt	1991-04-20	DD–Q 54	TPTF
1005	Klaus-Peter Becker	1989-02-19	M–TS 71	TPO

Database D₃ – Police

ID	Given Name	Surname	Year of Birth	Case	CaseDate
5252	Klaus	Bäcker	1982	Vandalism	11.04.2017
2942	Eva	Wagner	1974	Theft	08.11.2020
1957	Anna	Schmitt	1991	Car Accident	25.07.2022
2210	Bernd	Meier	1965	Car Accident	16.09.2019

Database D₄ – Hospital

ID	Person	Age	Sex	Disease	Diagnose Date
a1ec	Hafner, Leoni	28	W	Asthma	06.03.2018
4b1f	Meier, Bernd	58	M	Brain Injury	17.09.2019
03c7	Klaus, Becker	41	M	Encephalitis	28.10.2021
614d	Brand, Luca	29	D	Alopecia	14.07.2022
3i8b	Schmidt, Anna	32	W	PTSD	09.08.2022

Match Mapping

GID	ID(D1)	ID(D2)	ID(D3)	ID(D4)
G1	1	1004	1957	3i8b
G2	2	1001	2210	4b1f
G3	4	TPO	5252	03c7
G4	–	1002	–	a1ec
G5	–	1003	2942	–

Linked Database

GID	Mobile Phone Rate	Vehicle Insurance Coverage	Police Case	Police Case Date	Disease	Disease Date
G1	Flat rate	TPTF	Car Accident	25.07.2022	PTSD	09.08.2022
G2	Flat rate	TPO	Car Accident	16.09.2019	Brain Injury	17.09.2019
G3	Regular rate	–	Vandalism	11.04.2017	Encephalitis	28.10.2021
G4	–	TPTF	–	–	Asthma	06.03.2018
G5	–	FULL	Theft	08.11.2020	–	–

Figure 2.2: An example PPRL application scenario aiming at investigating correlations between mobile phone usage and traffic accidents.

Throughout the following subsection, we will illustrate the steps of the PPRL process based on the example linkage scenario shown in Figure 2.2. Again, we assume that a study on road safety is to be conducted with the research question '*Do people with a mobile phone flat rate cause more traffic accidents with personal injury?*' (see Section 1.1). In this example, data from mobile phone providers (database D_1), vehicle insurance companies (database D_2), the police (database D_3), and hospitals (database D_4) is required. The linkage is conducted on personal quasi-identifying attributes (colored yellow) only. The result of the linkage is a match mapping that indicates which of the databases' records refer to the same real-world entity.

Based on the match mapping a linked database can be created for researchers or data analysts. The linked database generally contains only payload or aggregated data of matching records (green-colored attributes) and some form of identifiers, such as project-specific IDs. The linked database does not contain any personal quasi-identifying attributes that can be used to identify an individual. However, if there is a unique combination of attribute values of the payload data, re-identification of individuals may still be possible. In such a case, the payload data must be anonymized, e.g. using techniques such as k-anonymity or differential privacy [Dwo06].

2.6.1 Pre-processing

At first, the databases to be linked need to be pre-processed by the database owners. Pre-processing includes deduplication, data cleaning, and standardization. In each individual database, duplicate records may occur due to inconsistent or repetitive recording processes. Therefore, the database owners may internally link and deduplicate their databases to ensure that a record from one data source can only be linked to at maximum one record from another data source. Data cleaning and standardization are required since real-world data often contain erroneous, missing, incomplete, inconsistent, or outdated data (see Section 2.5.2). Data cleaning techniques aim at eliminating or weakening such errors. Typical data cleaning tasks are [Chr12b]:

- Filling in missing data, in particular, by utilizing functional dependencies between sets of attributes (see Section 2.5). For example, the name of a suburb or city can be inferred from a postcode or street address, or the gender of a person may be inferred from the first name attribute.
- Removing or transforming unwanted characters or words, e. g., umlauts, punctuation, special characters, whitespaces, variations of missing values (e. g., null, unknown, n/a).
- Smoothing of noisy data, such as outliers in numerical or date attributes (e. g., age, date of birth).
- Correcting inconsistencies, such as contradicting attribute values, e. g., zip code and city or first name and gender contradict each other.
- Extension or unification of abbreviations.

Moreover, different databases often use different formats and structures to represent data. Hence, standardization techniques are used to overcome heterogeneity by transforming data into well-defined and consistent forms [RD00]. This typically includes the segmentation of compound attributes, unification of different formats and attribute representations, and converting all letters into either upper or lower case.

Data cleaning and standardization techniques typically require look-up tables (e. g., name and address databases) and manually defined rule sets. All database owners should conduct the same pre-processing steps to reduce heterogeneity and facilitate high-quality linkage results. However, even extensive pre-processing may not resolve all data quality issues. In particular, inconsistencies, like contradicting or outdated values, are hard to detect in each individual database.

Example: In the running example from Figure 2.2, the databases to be linked show several differences in the data representation. Databases D_2 and D_4 use a single name attribute ('Full Name', 'Person'), while databases D_1 and D_2 use two separate name attributes ('First Name'/'Given Name' and 'Last Name'/'Surname'). Databases D_1 and D_2 also store the date of birth, while databases D_3 and D_4 contain only a year of birth and an age attribute, respectively. However, databases D_1 and D_2 use different date formats that need to be unified. Table 2.1 - Table 2.4 show the databases to be linked after resolving these issues by applying pre-processing. Only the attributes that are present in all databases are shown.

ID	First Name	Last Name	DoB	MoB	YoB
1	Anna	Schmitt	20	04	1991
2	Bernd	Meier	12	01	1965
3	Christian	Schulze	27	10	1995
4	Klaus	Becker	25	07	1982

Table 2.1: Database D_1 after pre-processing.

ID	First Name	Last Name	DoB	MoB	YoB
1001	Bernd	Meier	01	12	1965
1002	Leoni	Schulze	27	10	1995
1003	Eva	Wagner	19	08	1974
1004	Anna	Schmidt	20	04	1991
1005	Klaus-Peter	Becker	19	02	1989

Table 2.2: Database D_2 after pre-processing.

ID	First Name	Last Name	DoB	MoB	YoB
5252	Klaus	Bäcker	–	–	1982
2942	Eva	Wagner	–	–	1974
1957	Anna	Schmitt	–	–	1991
2210	Bernd	Meier	–	–	1965

Table 2.3: Database D_3 after pre-processing.

ID	First Name	Last Name	DoB	MoB	YoB
a1ec	Leoni	Hafner	–	–	1995
4b1f	Bernd	Meier	–	–	1965
03c7	Becker	Klaus	–	–	1982
614d	Luca	Brand	–	–	1994
3i8b	Anna	Schmidt	–	–	1991

Table 2.4: Database D_4 after pre-processing.

2.6.2 Encoding

Linking sensitive databases requires that no private or confidential information is revealed during the linkage [CRS20]. Therefore, the database owners have to perform an encoding step, where records are encoded or encrypted in such a way that sensitive data is secured from re-identification. Ideally, each plaintext record (i. e., its set of attribute values that are relevant for linkage) is transformed into an encoded representation that cannot (computationally infeasible) be reverted to its original form. This includes that the encoding should not disclose any encoded features, such as (parts) of the records' attribute values. Moreover, the number and the frequencies of encoded features should be obfuscated. At the same time, an ideal encoding function for PPRL needs to preserve the similarity of the plaintext attribute values to allow approximate linkage in the presence of errors and inconsistencies.

Encoding techniques for PPRL are typically divided into two main categories: (1) secure multi-party computation (SMC), and (2) perturbation-based approaches [Vid+23]. Secure multi-party computation approaches aim to enable multiple parties to jointly compute a (public) function over their (private) sensitive input data [Yao82; GMW87; LP09]. In the end, all parties learn the result (output) of the function, but nothing about the input values of the other parties. While encoding techniques based on SMC are provably secure, they generally are expensive in terms of communication and computation costs making them often not scalable and thus impractical for many real-world PPRL scenarios [Vat+14; CRS20]. Perturbation-based approaches, in contrast, transform (modify, generalize, or encode) the input data to prevent disclosure of the actual value [VCV13]. In general, perturbation-based approaches have a trade-off between privacy, scalability, and linkage quality they can provide. In this thesis, we will focus on perturbation-based approaches.

A straightforward approach that *appears* as a solution is the use of (keyed) cryptographic hash functions [Sta11; CZ18]. Hashing is routinely used to validate data integrity and to identify known content [Rou10]. A hash function takes as input an arbitrary string of binary data and produces a fixed-sized output, called *digest* or *hash value*, in a predefined range [Rou10]. Hash functions need to be deterministic, i. e., the same input value must always result in the same hash value. Moreover, the hash values should be uniformly distributed over the predefined output range, i. e., each hash value should be generated with (nearly) the same probability. If two different input values are mapped to the same hash value, this is called a *(hash) collision* [CZ18].

Cryptographic hash functions, such as `MD5` or `SHA-1`, are designed as collision resistant one-way functions [Rou09; Rou10]. On the one hand, finding two different input values x, y with the same hash value, i. e., $hash(x) = hash(y)$ should be computationally infeasible (collision resistance). On the other hand, given a hash value z, finding any input value x with the same hash value, i. e., $hash(x) = z$ (one-way property), should be computationally infeasible (or at least very expensive) too. Another basic property of cryptographic hash functions is the *avalanche effect* that causes significantly different hash values if the input value is only changed slightly [CZ18].

The avalanche effect is required for security, but it is obstructive for cryptographic hash functions to be directly utilized for the linkage of sensitive data. Even small differences in the input will result in distinct hash values, and therefore such hash functions are only suitable for exact matching of values. That is, only a binary decision is possible, whether two values are equal or not. However, as discussed in Section 2.5.2, real-world database records contain errors and inconsistencies, and therefore a binary match decision would generally lead to many false negative results. By using encrypted cryptographic hash functions, such as keyed-hash message authentication codes (HMACs), dictionary attacks (see Section 2.6.7.3) are infeasible for an external adversary. However, frequency attacks are still possible because the frequency distribution of the hash values corresponds to the frequencies of the plaintext values.

Nevertheless, hash functions are used as an essential building block in most similarity-preserving encoding techniques for PPRL that enable approximate matching of records [CRS20]. The first PPRL approach that used hashing was proposed by Quantin et al. [Qua+98]. Hashing is also used in several variants of *anonymous linkage codes* (ALCs) [Ran+19]. The basis for many such codes was the so-called *statistical linkage key* (SLK) that was proposed and used by the Australian Institute of Health and Welfare [Kar05; Cou+21]. The main idea of this technique is to encode a record by concatenating specific characters drawn from the values of the first name, last name, date of birth, and gender attributes. Several anonymous variants have been proposed that finally hash the resulting code to improve privacy protection [BAQ01; Web+12].

The weakness of using only a single anonymous linkage code lies in the error tolerance regarding the attributes selected for constructing the code [BG02; Ran+16]. In particular, a single ALC cannot adequately handle missing attribute values. As a consequence, only a low linkage quality is achievable in the presence of real-world data errors [Ran+16; SRB17]. In [Ran+19], the authors therefore proposed an approach for creating multiple ALCs by using different combinations of attributes. However, Vidanage et al. [Vid+20b] showed that this approach is vulnerable to frequency attacks and therefore presented several modifications to prevent such attacks.

A widely used approach that has become the quasi-standard in recent research and practical applications of PPRL are encoding techniques based on Bloom filters [VCV13; Vat+17; CRS20; Gko+21]. We will describe these types of encodings in Section 2.8. Over the years, several studies have shown that Bloom filters can be susceptible to frequency-based cryptanalysis [Kuz+11; Nie+14; KS14; Mit+16; Chr+18a; Chr+18b; Vid+20a; Vid+22; Vid+23]. To avoid privacy attacks, several alternative encoding schemes have been proposed in the literature, but these have not yet been used in real-world applications. An encoding method based on tabulation min-hash (TMH) has been proposed by Smith [Smi17]. Ranbaduge et al. [RCS20] proposed a two-step hash (2SH) encoding that strives to outperform Bloom filter encodings in terms of security/privacy, and TMH encodings in terms of computational cost. Recently, Christen et al. [Chr+22] proposed a novel encoding technique based on autoencoders that transform Bloom filters into vectors of real numbers. An autoencoder [Kra91; Kra92] is a type of artificial neural network that is typically used to learn an efficient representation (encoding) of a set of data and thus to extract substantial features.

The data owners that want to conduct the linkage need to know the encoding method and the parameters used. To prevent data owners from re-identifying each other's records, for example, by mounting a dictionary attack, the linkage process must follow specific protocols that define how data is exchanged between database owners and other linkage participants. The different types of linkage protocols are described in Section 2.7.

2.6.3 Blocking/Filtering

As discussed in Section 2.4, the trivial approach to link two databases is to compare all possible pairs of records, which leads to a quadratic complexity. As a consequence, blocking and filtering techniques are used to reduce the number of record comparisons [Chr12b]. This is achieved by pruning record pairs not fulfilling defined blocking or filter criteria and are hence unlikely considered as matches.

The output of this step is a set of candidate record pairs that need to be further compared. The set of candidate record pairs after blocking or filtering is denoted as C_B where $C_B \subsetneq C$ and $|C_B| \ll |C|$. The effort required for blocking must be significantly smaller than the effort that would be required to compare all possible record pairs C.

Both, blocking and filtering can be executed on encoded or plaintext data. Most privacy-preserving approaches perform blocking or filtering on encoded data [Vat+17]. State-of-the-art blocking techniques can significantly reduce the search space by applying blocking based on locality-sensitive hashing (LSH) [Dur12; KV13; KV16] or performing filtering based on multibit-trees [Bro+17; Sch15] or pivot-based filtering for metric distance functions [SR16]. Although these two methods have the same objective, they operate differently and could even be used in combination.

2.6.3.1 Blocking

The basic idea of blocking is to partition (group) records into blocks, and then compare only records within the same block with each other. Therefore, one or multiple blocking key values (BKV) are generated for each record. Records with the same blocking key value are assigned to the same block. Each corresponding blocking key represents a specific (potentially complex) criterion that records must satisfy in order to be considered as match candidates. A blocking key is defined by a function that takes as input one or more record attributes and outputs the blocking key value. The record attributes used for blocking may be different from the attributes used in the comparison step. If only one blocking key is used, the resulting blocks are disjoint. However, a single blocking criterion might be too restrictive and can consequently lead to many false negatives. As a consequence, multiple blocking keys are often used to increase the probability for records to share at least one blocking key. The drawback of using multiple blocking keys is that it leads to overlapping (non-disjoint) blocks and consequently to duplicate candidate record pairs.

Typically, blocking shows a trade-off between linkage quality and performance (scalability) depending on the type and number of blocking keys used. In general, using restrictive blocking criteria will lead to many small blocks and thus fewer candidate record pairs. At the same time, however, the chance of missing true matches increases. On the other hand, if less restrictive blocking criteria are selected, the number of blocks decreases while the block sizes and thus the number of candidates increases. As a consequence, more candidates need to be compared, leading to fewer false negatives but also less efficiency. Generally, the higher the number of blocking keys, the higher the number of resulting blocks and consequently the number of (duplicate) candidate record pairs.

Example: Considering database D_2 and D_4 from the running example, without blocking $|D_2| \cdot |D_4| = 5 \cdot 5 = 25$ record pairs need to be compared. Let $bkf(\cdot)$ be a blocking key function that outputs the first character of the last name attribute. Applied on the records in both databases, this will generate the blocking key values shown in Table 2.5 and Table 2.6. This will lead to the partitions (blocks) shown in Table 2.7. For each partition, each record from database D_2 needs to be compared with every record of database D_4. The block corresponding to blocking key value ‘B’ will lead to the candidate records pairs (1005, 4b1f), (1005, 03c7), and (1005, 614d). The block corresponding to blocking key value ‘S’ leads to the candidate record pairs (1002, 3i9b) and (1004, 3i9b). The block corresponding to the blocking key values ‘H’, ‘M’ and ‘W’ will lead to no candidate record pairs. Thus, only 5 record pairs need to be compared using blocking, in contrast to 25 pairs for the trivial approach. However, the pair (1002, a1ec) (being a match) is not compared, since the last name attribute differs, e. g., due to marriage.

2.6.3.2 Filtering

Filtering techniques use a distance (or similarity) metric and a corresponding distance (or similarity) threshold t (see Section 2.6.4) to efficiently remove dissimilar record pairs that cannot reach this threshold [SR16]. In contrast to blocking, filtering techniques generally do not involve any false negatives [Jia+14].

ID	BKV
1001	M
1002	S
1003	W
1004	S
1005	B

Table 2.5: Blocking key values (BKV) for database D_2.

ID	BKV
a1ec	H
4b1f	B
03c7	B
614d	B
3i9b	S

Table 2.6: Blocking key values (BKV) for database D_4.

BKV	Records	
	DB	ID
B	D_2	1005
	D_4	4b1f
		03c7
		614d
H	D_4	a1ec
M	D_2	1001
S	D_2	1002
		1004
	D_4	3i9b
W	D_2	1003

Table 2.7: Partitions resulting from blocking.

Filtering techniques employ one or more filters to substantially reduce the search space. Each filter evaluates a certain condition that must be fulfilled by a pair of records in order to reach the similarity threshold. Typical approaches are length, prefix, suffix, and position filters that utilize characteristics of the used similarity function [Chr12b; Vat+17]. Considering length filtering, for example, the basic idea is that if two objects (e. g., strings) are similar, their length can only differ by a certain factor (i. e., their length difference cannot be large). In this context, the meaning of length is ambiguous and varies depending on the objects of interest. For example, the number of elements (cardinality) is considered as the 'length' of a set. Properties of metric spaces are also utilized as filtering techniques to reduce the search space [CRS20; SR16; Seh+21].

Definition 2.6.3.1: Metric Space, (Distance) Metric

A metric space $\mathcal{M}(U, d)$ consists of a set of data objects U and a distance metric $d : U \times U \to \mathbb{R}$ to compute the distance between the objects. For any objects $x, y, z \in U$ it satisfies the following axioms:

$$d(x, x) = 0 \tag{2.12}$$

$$x \neq y \Rightarrow d(x, y) > 0 \quad \text{(positivity)} \tag{2.13}$$

$$d(x, y) = d(y, x) \quad \text{(symmetry)} \tag{2.14}$$

$$d(x, y) \leq d(x, z) + d(z, y) \quad \text{(triangle inequality).} \tag{2.15}$$

From this, the reverse triangle inequality can be derived as:

$$|d(x, z) - d(z, y)| \leq d(x, y) \tag{2.16}$$

The triangle inequality property is used to avoid computing the distance between every pair of objects. Instead, reference points (pivot elements) are used. When knowing the distance of a point x to a reference point z as well as the distance from a point y to z, then the distance from x to y can be approximated. The lower bound of the distance from x to y can be approximated by calculating the difference $d(x,z) - d(z,y)$. If this difference is larger than the threshold t, also $d(x,y)$ must be larger than t and the pair (x,y) can be pruned before calculating the actual similarity. More formally:

$$d(x,z) - d(z,y) > t \Rightarrow d(x,y) > t. \tag{2.17}$$

Example: Let $d(x,z) = 0.2$ and $d(z,y) = 0.3$. Then, $d(x,y) \leq d(x,z) + d(z,y) = 0.2 + 0.3 = 0.5$. Moreover, $d(x,y) \geq |d(x,z) - d(z,y)| = |0.2 - 0.3| = 0.1$ and thus $0.1 \leq d(x,y) \leq 0.5$.

2.6.4 Comparison

Each candidate record pair is compared using similarity functions that are applied on the records' attributes. A similarity function is defined as follows.

Definition 2.6.4.1: Similarity Function

For a set of data objects U, a similarity function $sim : U \times U \rightarrow [0,1]$ calculates a value (called similarity) that quantifies how similar two objects are. For objects $x, y \in U$, it satisfies the following axioms:

$$sim(x,y) = sim(y,x) \tag{2.18}$$

$$sim(x,y) = 1 \quad \Rightarrow \quad x = y. \tag{2.19}$$

The concept of a similarity function is strongly related to a distance metric. While a metric defines the distance between two objects, a similarity function measures the closeness. The larger the value $sim(x,y)$ is, the closer (the more similar) the two objects are. A similarity of 1 means that the objects are equal. A value of 0, on the other hand, means that the objects are completely different.

In contrast to a distance metric, a similarity function is not required to satisfy the triangle inequality. However, a semi-metric d [MC16] can be constructed from the similarity function:

$$d(x,y) = 1 - sim(x,y). \tag{2.20}$$

If the triangle inequality holds for d then it is in fact a metric. Similarly, a distance metric can be converted into a similarity function by using either

$$sim(x, y) = 1/d \quad \text{if } 0 < d \tag{2.21}$$

or

$$sim(x, y) = 1 - d \quad \text{if } 0 \leq d \leq 1. \tag{2.22}$$

A variety of similarity functions for different data types, such as strings, numbers, or dates, have been proposed, for instance, the Jaro-Winkler similarity or the Cosine similarity [Chr12b]. Set-based similarity measures are most commonly used because they can be applied to various domains. The two most important measures are the Jaccard [Jac12] and the Dice [Dic45] similarity, which are defined as follows.

Definition 2.6.4.2: Jaccard Similarity (Sets)

Let X and Y be two arbitrary sets. Then, their Jaccard similarity is defined by

$$sim_{Jaccard}(X, Y) = \frac{|X \cap Y|}{|X \cup Y|} = \frac{|X \cap Y|}{|X| + |Y| - |X \cap Y|} \tag{2.23}$$

Definition 2.6.4.3: Dice Similarity (Sets)

Let X and Y be two arbitrary sets. Then, their Dice similarity is defined by

$$sim_{Dice}(X, Y) = \frac{2 \cdot |X \cap Y|}{|X| + |Y|} \tag{2.24}$$

The Dice similarity can also be calculated based on the Jaccard similarity as

$$sim_{Dice}(X, Y) = \frac{2 \cdot sim_{Jaccard}(X, Y)}{1 + sim_{Jaccard}(X, Y)} \tag{2.25}$$

In general, each record comparison results in a similarity vector (or more generally a similarity matrix), where each vector entry represents the result of a specific similarity function evaluated on a specific pair of attributes.

The output of the comparison step is a set of candidate record pairs together with their similarity vector. This result can be considered as a similarity graph.

Definition 2.6.4.4: Graph, Similarity Graph

A graph is a pair $G = (V, E)$ of sets V (vertices) and E (edges) such that $E \subseteq [V]^2$, i. e., elements of E are 2-element subsets of V.

A similarity graph $SG = (V, E)$ is a graph in which vertices of V represent records and edges of E are the calculated similarities between records that have been compared. Thus, each edge connects two compared records and holds the resulting similarity vector.

2.6.5 Classification

In this step a decision model is used to assign each candidate record pair (based on its similarity vector) to one of the classes: matches (`M`), non-matches (`N`), and optionally potential matches (`PM`) [Chr12b]. The class of potential matches contains those candidate record pairs where the model was not able to make a clear decision.

2.6.5.1 Threshold-based Classification

While different classification techniques have been developed [Chr12b; BS22; Pap+21], many PPRL approaches rely on a simple threshold-based classification. Therefore, the similarity vector for a pair of records is aggregated into a single similarity or confidence score $sim_{\Delta}(\cdot, \cdot)$, mostly by calculating a weighted sum over the vector entries [Chr12b; CRS20].

The idea of assigning weights to attributes is part of the probabilistic record linkage model proposed by Fellegi and Sunter in [FS69]. This model is the basis for many record linkage approaches and is still frequently used and adapted [HSW07; Chr12b; CRS20; Roh+23].

The weighting of attributes addresses the fact that attributes have different importance and discriminatory power to distinguish matches and non-matches. On the one hand, each attribute has a different number of (possible) values, and these values follow a certain distribution. For example, an equal date of birth is a stronger indicator for a match than an equal gender as there are far more values (and thus each value occurs less often) for date of birth than for gender. On the other hand, attributes can also be erroneous or inconsistent (see Section 2.5.2), with some attributes being affected more often than others.

Consequently, for each attribute i two probabilities, namely the match probability (μ) and non-match probability (u), are defined as:

$$\mu_i = \mathbb{P}\left[\pi_i(a) = \pi_i(b), a \in D_A, b \in D_B \mid a \equiv b\right] \tag{2.26}$$

$$u_i = \mathbb{P}\left[\pi_i(a) = \pi_i(b), a \in D_A, b \in D_B \mid a \not\equiv b\right] \tag{2.27}$$

The μ-probability specifies the probability that the two records a and b have the same value for attribute i, given the records refer to the same entity. Ideally, $m_i = 1$, if all true matches agree on attribute i. However, this is only the case if attribute i does not contain any errors. If, for example, 10% of the duplicates have a non-equal value, e. g., due to typographical errors, then $m_i = 0.9$.

In contrast, the u-probability specifies the probability that two records have the same value for attribute i, given the records refer to different entities. The u-probability is low if the attribute has a high number of possible values. However, if, for example, an attribute i has only two possible and equally likely values, then $u_i = 0.5$ as the chance that two records agree only by chance on this attribute is 50%.

Using the μ- and u-probabilities, the weight w_i for attribute i is calculated as

$$w_i = \begin{cases} w_\mu = \log_2\left(\frac{\mu_i}{u_i}\right) & \text{if } \pi_i(a) = \pi_i(b) \\ w_u = \log_2\left(\frac{1-\mu_i}{1-u_i}\right) & \text{if } \pi_i(a) \neq \pi_i(b) \end{cases} \tag{2.28}$$

The match and non-match probabilities are typically frequency-dependent as a random agreement is more likely for frequent than for rare attribute values. Therefore, a value-specific weight can be incorporated to adjust the match and non-match weights based on the frequency of individual attribute values [HSW07; Roh+23].

Finally, using two threshold values $t_\uparrow$ and $t_\downarrow$ each record pair $(a, b) \in D_A \times D_B$ is classified as follows:

$$sim_\Delta(a,b) \geq t_\uparrow \quad \Rightarrow \quad (a,b) \in \mathtt{M} \tag{2.29}$$

$$t_\uparrow > sim_\Delta(a,b) \geq t_\downarrow \quad \Rightarrow \quad (a,b) \in \mathtt{PM} \tag{2.30}$$

$$sim_\Delta(a,b) < t_\downarrow \quad \Rightarrow \quad (a,b) \in \mathtt{N} \tag{2.31}$$

If $t_\uparrow = t_\downarrow$, then records are classified into two classes only (matches and non-matches). For the remainder of this thesis, we consider that both thresholds are the same ($t_\uparrow = t_\downarrow$).

Considering the linkage result as a similarity graph, each edge in the graph is weighted using sim_Δ and labeled by the class the connected record pair belongs to (match, non-match, or potential match). Vertices without edges are implicitly considered non-matches.

In record linkage applications, the sizes of the match and non-match classes are generally very imbalanced ($\mathtt{M} \ll \mathtt{N}$). This is known as the class imbalance problem [Chr12b] because, from all possible pairs of records, only a small fraction refers to the same entity, while the vast majority of record pairs are non-matches.

In Figure 2.3 a typical distribution of the similarities of matching and non-matching record pairs is shown (note the logarithmic scale on the y-axis). As can be seen, there are two peaks: (1) a large peak on the left side which is created by non-matching records and their similarity distribution; (2) a small peak on the right side which consists of matching records and their similarity distribution. Often, these distributions are overlapping and the aim of the linkage algorithm is to separate them as much as possible.

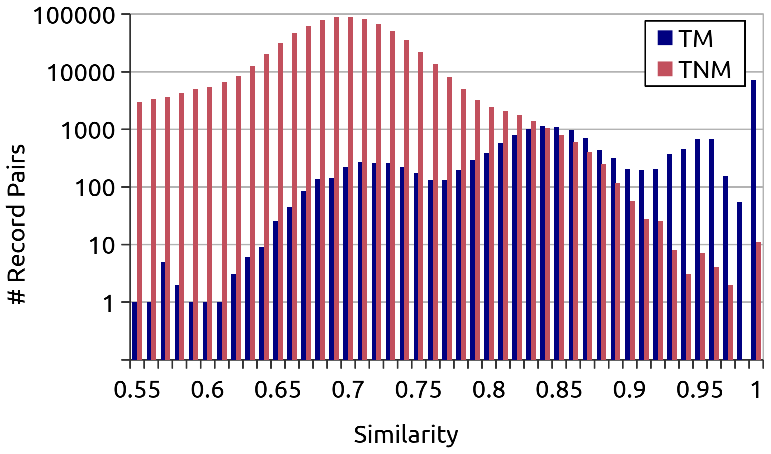


Figure 2.3: Example similarity distribution of true matching (TM) and true non-matching records (TNM) on an encoded dataset drawn from the North Carolina Voter Registration (NCVR) database [Nor23] using blocking.

2.6.5.2 Clustering

Another approach for deciding which candidate record pairs correspond to matches and which to non-matches is clustering. Clustering is the process of separating (partitioning, grouping) a set of elements (objects, entities) into a set of clusters (groups, subsets, categories) [XW05]. Given a measure of similarity, the objective of most clustering approaches is to generate clusters in such a way that elements within the same cluster are similar to each other (high intra-cluster similarity), while elements in different clusters are dissimilar to each other (low inter-cluster similarity) [XW05; HKP12]. Thus, elements within the same cluster should be more similar to each other than to elements of other clusters.

In the context of record linkage, ideally, each entity is represented by exactly one cluster. Thus, clustering will typically generate a large number of clusters where each cluster is very small and consists of only a few records. Clustering can be used for classification by classifying all elements within the same cluster as matches and elements of different clusters as non-matches. However, clustering can also be applied as a post-processing step after a pair-wise classification of record pairs has been conducted [Chr12b]. In that case, clustering techniques are used to refine and improve the classification results.

2.6.6 Post-processing

The output of the classification step is generally not the final outcome of a record linkage process. After the classification step, the similarity graph contains edges (links) of different types based on their vertex degree.

Definition 2.6.6.1: Vertex Degree, Edge Degree

The degree $\deg(a)$ of a graph vertex a is defined as the number of edges that are incident to that vertex [Die17]. If each vertex of the graph has the same degree δ, the graph is called a δ-regular graph [Die17]. Similarly, the degree $\deg(e)$ of a graph edge $e = (a, b)$ is given as the maximum degree of its endpoints (vertices) a and b, i. e., $\deg(e) = \max(\deg(a), \deg(b))$, where $(a, b) \in E$.

Edges (links) in the similarity graph can be categorized based on the degree of the vertices they are connecting:

- **One-to-one link:** A one-to-one link is an edge $e = (a, b)$ between two vertices $a, b \in V$ with $\deg(e) = 1$.
- **Multi-link:** A multi link is an edge $e = (a, b)$ between two vertices $a, b \in V$ with $\deg(e) > 1$.
- **One-to-many link:** A one-to-many link is an edge $e = (a, b)$ between two vertices $a, b \in V$ where $\deg(a) = 1$ and $\deg(b) > 1$.
- **Many-to-one link:** A many-to-one link is an edge $e = (a, b)$ between two vertices $a, b \in V$ where $\deg(a) > 1$ and $\deg(b) = 1$.
- **Many-to-many link:** A many-to-many link is an edge $e = (a, b)$ between two vertices $a, b \in V$ where $\deg(a) > 1$ and $\deg(b) > 1$.

Depending on the characteristics of the databases and the application that uses the linked database, certain link constraints must be satisfied [CRS20]. Therefore, each database is considered as clean or dirty depending on the absence/presence of duplicates.

Definition 2.6.6.2: Clean Database, Dirty Database

A database D is called **clean**, if

$$\nexists a, b \in D : a \equiv b \land a \neq b. \tag{2.32}$$

In contrast, a database D is called **dirty**, if

$$\exists a, b \in D : a \equiv b \land a \neq b. \tag{2.33}$$

In clean databases, there are thus no two records that refer to the same entity. A dirty database, however, contains at least two records that refer to the same entity. For instance, let $x = [\text{Vi\underline{c}tor}, \text{Christen}, 1988, \text{Leipzig}]$ and $y = [\text{Vi\underline{k}tor}, \text{Christen}, 1988, \text{Leipzig}]$, then $a \neq b$, while we can assume $a \equiv b$.

For simplicity, in the following, we assume without loss of generality the linkage of two databases only. Depending on whether the individual databases contain duplicates, four cases can be distinguished: clean-clean, clean-dirty, dirty-clean, and dirty-dirty.

If both databases are duplicate-free, then records from the same database are usually not compared. As a consequence, the similarity graph resulting from the linkage forms a bipartite graph [Die17]. Thus, V allows a division into two partitions, namely V_A and V_B where $V = V_A \cup V_B$, such that every edge has its ends in different partitions, i. e., vertices in the same partition are not adjacent. The partition V_A only consists of records from database D_A and partition V_B only of records from database D_B, respectively. Since we assume duplicate-free databases, any record of D_A can match to at maximum one record of D_B and vice versa. Thus, the similarity graph needs to be 1-regular and consequently must only contain edges with a degree of 1 (one-to-one links).

In this case, post-processing applies a one-to-one link restriction to the match result by resolving all multi-links (see Chapter 5). This is equivalent to finding a matching over the similarity graph [Die17]. Here, the term matching here refers to the graph-theoretic terminology.

Definition 2.6.6.3: (Graph) Matching

Given a graph $G = (V, E)$, a matching $M \subseteq E$ is a set of edges without common vertices, i. e., all edges are pairwise non-adjacent.

If one database is clean and the other is dirty (clean-dirty and dirty-clean cases), then one-to-one links and one-to-many links are permitted. If both databases are dirty (dirty-dirty case), then all types of links can potentially occur. In such a situation, each database owner could first individually run an intra-source deduplication process before performing the actual holistic (inter-source) linkage. While each database owner can optimize the deduplication configuration locally and potentially perform a manual assessment of the linking result, this approach has some drawbacks. At first, intra-source duplicates may be fused into a single record (cluster representative), e. g., by selecting attribute values that are more likely to be complete, accurate, and up-to-date. Using this approach, the amount of available information is reduced, which potentially leads to more false negatives. Secondly, errors in the deduplication process of a source are possible, where two records are considered as match while they actually refer to different entities (intra-source false positive). As a consequence, entities are wrongly fused and this error is propagated through the whole process, which in turn can lead to inter-source false positives [OSR19]. Therefore, it can be beneficial to retain intra-source duplicates. This approach requires comparing records from the same database, resulting in a similarity graph with intra-source links. By definition, this will make the similarity graph no longer bipartite. However, by removing all intra-source links, a bipartite subgraph can be obtained.

Besides enforcing link cardinality constraints as discussed before, post-processing can also ensure that the linkage result fulfills the transitive closure [Chr12b]. For records $a, b, c \in V$, this property guarantees that if both the pair (a, b) and (a, c) are classified as a match, then the pair (b, c) must also be a match. The transitive closure may be violated due to missed true matches, for example, during blocking, or because candidate pairs are classified independently of other candidate pairs [Chr12b].

2.6.7 Evaluation

The linkage process and the linkage result are evaluated under three main aspects [CRS20]: (1) linkage quality; (2) linkage complexity and scalability; and (3) privacy protection and security against attacks. In the following, we discuss how these aspects can be assessed.

2.6.7.1 Quality

Given a ground-truth (gold standard) dataset containing the true match status of a set of record pairs, four classification outcomes are possible for each pair of records:

- **True positive:** A true positive is a record pair that has been classified as a match and the pair is a true match. The two records refer to the same entity.
- **False positive:** A false positive is a record pair that has been classified as a match, but it is not a true match. The two records refer to different entities.

- **True negative:** A true negative is a record pair that has been classified as a non-match and it is a true non-match. The two records refer to different entities.
- **False negative:** A false negative is a record pair that has been classified as a non-match, but it is a true match. The two records refer to the same entity.

For a specific classification configuration, e. g., certain classification threshold t, this results in a confusion matrix [HKP12] reporting the total number of true positives (tp), true negatives (tn), false positives (fp) and false negatives (fn). Based on the confusion matrix, different quality measures can be calculated [HC18]. The most common measures are recall (R) and precision (P), which are defined as

$$R = \frac{tp}{tp + fn} \tag{2.34}$$

$$P = \frac{tp}{tp + fp} \tag{2.35}$$

Consequently, recall measures the proportion of true matches that have been correctly classified as matches after the linkage process. Precision is defined as the fraction of classified matches that are true matches. It is often desirable to combine recall and precision into a single number. For this purpose, frequently the F-measure (F) is calculated that is equal to the harmonic mean between both measures:

$$F = 2 \cdot \frac{P \cdot R}{P + R} \tag{2.36}$$

However, this way of aggregating recall and precision has several problematic aspects as discussed in [HC18; CHK23]. In [HCK21], therefore, a transformed version, called F-star (F^*), has been proposed:

$$F^* = \frac{F}{2 - F} = \frac{P \cdot R}{P + R - P \cdot R} \tag{2.37}$$

F^* is a monotonic transformation of the F-measure and therefore all results on the F-measure also hold for $F*$.

2.6.7.2 Scalability

The number of record pair comparisons mainly determines the complexity of the linkage. As described in Section 2.6.3, blocking or filtering techniques are used to reduce the number of record pair comparisons and thus have a great influence on the performance and scalability of PPRL approaches. The efficiency of blocking/filtering can be measured with the reduction ratio (RR) [EVE02; Chr12a] that is defined as:

$$RR = 1 - \frac{|C_B|}{C} \tag{2.38}$$

Consequently, the reduction ratio measures the ratio between the number of candidate record pairs after blocking/filtering and the number of all possible candidate pairs. A high reduction ratio means that the blocking/filtering technique was able to efficiently reduce the number of record pair comparisons. The reduction ratio for the blocking example in Section 2.6.3.1, where $|C| = 25$ and $|C_B| = 5$, is calculated as $RR = 1 - \frac{5}{25} = 0.8$.

2.6.7.3 Privacy

The privacy (security) properties of PPRL approaches are typically evaluated under specific attack models [VCV13; Vat+17]. For measuring privacy, there currently is no single accepted mathematical framework [CRS20]. However, privacy can be measured by the amount of information an adversary can gain under the attack model. Besides, privacy can be measured based on the risk of disclosure [Vat+14]. In general, the disclosure risk is assessed based on the likelihood that an individual can be associated with a record or an attribute (in the released/disclosed dataset) containing sensitive or confidential information [Lam93].

An attack model that is often assumed is the honest-but-curious model [HF10; Vat+14]. In this model, the parties follow the protocol and compute valid results (honest), but try to obtain as much information about the original data (plaintext records of other database owners) as possible. This includes that two or more database owners could *collude* to obtain sensitive information of the other database owners. Such a collaboration between database owners or between a database owner and a third party (see Section 2.7) is called a collusion.

Another attack model assumes that the parties participating in the linkage behave maliciously [Vat+14; Vat+17]. In this model, the parties can behave arbitrarily, i. e., the parties might not follow the protocol, abort the protocol at an arbitrary point, or send arbitrary input data. This model has, however, rarely been considered because the unpredictable behavior of the parties makes analysis difficult [Vat+14; Vat+17].

Different types of attacks have been considered in the PPRL context [VCV13; Vat+17]. The two most common types of attacks are described in the following [Vat+17].

Dictionary attacks: It is assumed that an attacker knows the used encoding (encryption/masking) method as well as the corresponding parameters so that the attacker can encode a publicly available dataset accordingly. Then, the attacker tries to find correspondences between plaintext and encoded values. To prevent such attacks, the available information must be separated between the parties (see Section 2.7), and keyed hash message authentication codes (HMACs) should be used in the encoding step.

Frequency attacks: In such attacks, the distribution of encoded values is analyzed. Using the distribution of known plaintext values, such as name or address frequencies, an attacker tries to align plaintext to encoded values. If, for instance, 'Müller' is the most frequent last name, then the most frequent encoded value will likely correspond to that last name. If an attacker can identify one or more encoded values, a known-plaintext attack can be mounted in order to identify the encoding method and its parameters. The use of HMACs does not change the frequency distribution of encoded values and thus does not provide any protection. To complicate frequency attacks, the frequency distribution needs to be artificially altered. One approach is to introduce dummy (fake) attribute values or records that contain values with lower frequencies. The drawback of this approach is that the fake values/records can have a negative impact on both, the linkage quality and scalability. On the one hand, the probability of wrongly classified records increases. On the other hand, more records might need to be matched and potentially sorted out after the linkage.

2.7 Linkage Protocols

Database owners need to know the encoding technique and the related parameters to encode their records in order to protect sensitive information. Therefore, the database owners cannot simply exchange all encoded records and individually perform the linkage because otherwise each database owner could mount a dictionary attack and try to re-identify the encoded records. As a consequence, the linkage must be performed under a specific protocol that regulates the exchange of information between the linkage participants. The most important participants (parties) of a linkage can be categorized as follows [CRS20]:

- **Database owners:** The database owners are the providers of the databases to be linked. Depending on the linkage protocol, the database owners may be involved in all linkage steps, or alternatively only pre-process, encode, and optionally block/filter their databases.
- **Linkage unit:** The linkage unit is a trusted third party, that typically provides no data to be linked.

- **Data consumer:** Data consumers process or analyze the linkage result, i. e., the linked database containing values from selected attributes. Database owners can be data consumers themselves, but also external researchers or data analysts.

In a linkage, each participant should only have access to the data it requires to perform its role in the specific protocol [CRS20]. This mechanism is known as separation principle. It includes that the participants involved in the actual linkage have access only to the quasi-identifying attributes that are necessary to perform the linkage. Participants that are involved in an analysis of the linked database (such as researchers or data analysts) have, in contrast, only access to payload or aggregated data of the matching records [CRS20]. The separation principle is also shown in the running example of Figure 2.2 where the personal quasi-identifying attributes are colored yellow and the payload data is colored green.

Basically, there are two types of linkage protocols, those with and those without using a linkage unit. In protocols without a linkage unit, the database owners communicate directly with each other to perform the linkage [Vat+17]. These protocols are, in general, more complex and expensive in terms of computation and communication because the database owners know the details of the used encoding technique, and thus it is harder to ensure that the database owners do not learn any sensitive information other than the set of matching record pairs. Typically, secure (multi-party) computation approaches [Yao82; LT05] are used as the basis for such protocols [Vat+17]. Protocols that use a linkage unit, in contrast, are generally more efficient as the actual linkage is performed by the linkage unit. After the database owners have pre-processed, encoded, and optionally blocked/filtered their databases, they send their encoded records to the linkage unit [Vat+17]. The linkage unit performs the actual linkage and sends back the match mapping (identifiers of matching record pairs) to the database owners. The database owners can then exchange selected attribute values (payload data) of the matched record pairs with each other, or with external data consumers.

2.8 Bloom Filter Encodings

Privacy-preserving record linkage requires that no private or confidential information is revealed during the linkage. Consequently, each record needs to be encoded (encrypted) to protect sensitive data. While different encoding techniques have been proposed in the literature, approaches utilizing Bloom filters as encoding technique have become the quasi-standard for recent PPRL approaches in both research and practical applications [VCV13; Vat+17; Gko+21]. In this thesis, we therefore also focus on encodings based on Bloom filters.

Bloom filters were originally proposed in 1970 by Burton H. Bloom as a space-efficient data structure for checking set membership [Blo70]. Bloom filters are frequently used in different domains, such as database applications or network protocols [BM04]. The use of Bloom filters for PPRL has been first proposed by Schnell and colleagues in [SBR09]. Since then, several encoding methods based on Bloom filters have been proposed for PPRL to improve the linkage quality [KGV18; VC14; VC16] or to reduce the re-identification risk [Dur12; Nie+14; Sch15; SBR11; SB16a]. In the following, we first describe the basics of Bloom filters and then explain how they can be used for PPRL to encode records containing sensitive data.

2.8.1 Bloom Filter Basics

A Bloom filter (BF) is a space-efficient probabilistic data structure for representing a set $\mathsf{E} = \{\mathsf{e}_1, \ldots, \mathsf{e}_n\}$ of n elements or features and testing set membership. Therefore, a bit vector v of fixed size m is allocated, and initially, all bits are set to zero.

Definition 2.8.1.1: Bit Vector

Let $\mathbb{B} = \{0, 1\}$ be the boolean domain. A bit vector (also known as bit array) $v \in \mathbb{B}^m = [b_0, ..., b_{m-1}](\forall i, 0 \leq i < m : b_i \in \mathbb{B})$ is a m-dimensional vector (or array) of bits, where $m = ||v||$ is called the size or length of the bit vector. An overview of the most important operations on bit vectors is given in Table 2.8.

A bit vector can be represented as a set containing the indices of all 1-bits, i. e., $v = \{i \mid 0 \leq i < m, b_i = 1\}$. This representation is called a bit set.

A set H of k independent (cryptographic) hash functions is selected, where each function $h_1, \ldots, h_k$ outputs a value in $[0, m-1]$, i. e., $h_i \in H : \mathsf{E} \to \{0, \ldots, m-1\}$. To represent the set E in the Bloom filter, each element is (hash) mapped to the bit vector v by using each of the k hash functions and setting the bits at the resulting positions to one, i. e., $\forall \mathsf{e} \in \mathsf{E}, \forall h \in H : v[h(\mathsf{e})] = 1$.

To check the membership of an element, the k hash functions are calculated and the bits at the resulting positions are checked. If all bits are set to one, the element *probably* is in the set. On the other hand, if at least one bit is zero, the element is definitively not in the set. Due to collisions, i. e., two or more elements may set the same bit position for the same or different hash functions, Bloom filters have a false positive probability (*fpp*) for an element to be not in the set (not represented by the Bloom filter). The false positive probability is defined as [BM04; MU17]:

$$fpp = \left(1 - \left(1 - \frac{1}{m}\right)^{k \cdot n}\right)^k \approx (1 - \mathrm{e}^{-\frac{k \cdot n}{m}})^k \tag{2.39}$$

Operation	Symbol	Description
#0-bits	$\|\| \cdot \|\|_0$	Number of bits equal to 0.
#1-bits	$\|\| \cdot \|\|_1$	Number of bits equal to 1 (cardinality/Hamming weight).
NOT	$\neg$	Flips all bits of the bit vector (complement).
AND	$\wedge$	Bit-wise logical AND operation between two bit vectors.
OR	$\vee$	Bit-wise logical OR operation between two bit vectors.
XOR	$\oplus$	Bit-wise logical exclusive OR operation between two bit vectors.
Concatenation	$\odot$	Appends one bit vector to the other.
Permutation	Π	Rearranges bits in the bit vector.
Projection/Slicing	π_i^j	Extracts a sub bit vector composed of the bits from position i (inclusive) to j (exclusive).
Bit Shift	$\ll, \gg$	Shifts bits to left or right, with or without discarding or adding bits.
Circular Shift	$\circlearrowleft, \circlearrowright$	Special type of permutation by moving the last (first) bit to the first (last) position, while shifting all other entries to the next (previous) position.

Table 2.8: Operations on bit vectors.

Figure 2.4 provides an example of a Bloom filter. The elements e_1 and e_2 are hashed $k = 3$ times. Each hash outputs a bit position and these bits are set to 1. For elements e_3 and e_4 it is checked if they are included in the Bloom filter. Therefore, the elements are also hashed and the corresponding bits are checked. The element e_3 is not in the set since at least one of the bits is 0. Element e_4 is considered to be in the set, although it was not added to the Bloom filter.

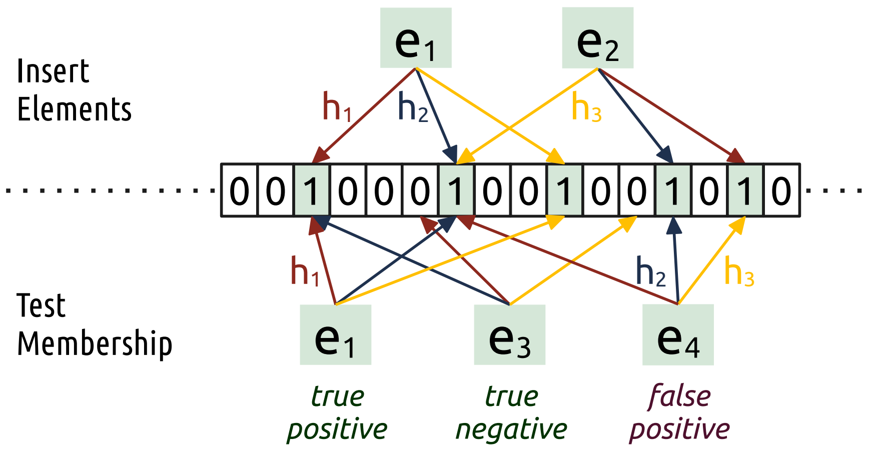


Figure 2.4: An example of a Bloom filter yielding a false positive.

As a consequence, an appropriate choice of the Bloom filter length m and the number of hash functions k is essential. Trivially, it must hold that $m > k \cdot n$. By using a fixed Bloom filter length, the optimal number of hash functions can be calculated as:

$$k_{opt} = \left\lceil \ln(2) \cdot \frac{m}{n} \right\rceil \tag{2.40}$$

The optimal Bloom filter length m_{opt} depends on k and the number of elements n to be mapped into the Bloom filter:

$$m_{opt} = \left\lceil \frac{k \cdot n}{\ln(2)} \right\rceil \tag{2.41}$$

The false positive probability can also be bounded to a certain value ϵ assuming that the optimal value for k is used [BM04]:

$$fpp \leq \epsilon \quad \Leftrightarrow \quad m \geq n \cdot \frac{\log_2(\frac{1}{\epsilon})}{\ln 2} \tag{2.42}$$

Assuming Bloom filters with the same size and set of hash functions, their union and intersection can be implemented with the bit-wise OR and AND operations, respectively.

While the union operation is lossless, i. e., the resulting Bloom filter will be equal to a Bloom filter that was built using the union of the two sets, the intersection operation produces a Bloom filter that may have a larger false positive rate [BM04]. Consider two Bloom filters representing sets E_1 and E_2 using the same bit vector length m and the same hash functions. A certain bit will be set in both Bloom filters, if the bit is either set by some element in $\mathsf{E}_1 \cap \mathsf{E}_2$, or simultaneously by an element in $\mathsf{E}_1 - (\mathsf{E}_1 \cap \mathsf{E}_2)$ and by another element in $\mathsf{E}_2 - (\mathsf{E}_1 \cap \mathsf{E}_2)$.

By using union and intersection on Bloom filters, set-based similarity measures can be used to calculate the similarity of two Bloom filters. The Bloom filter similarity is then an approximation of the similarity of the underlying (represented) sets. The most important similarity measures for Bloom filters are defined in Definition 2.8.1.2.

Example

Let $R = \{$‘a’, ‘b’, ‘c’, ‘d’$\}$ and $S = \{$‘a’, ‘b’, ‘c’, ‘e’$\}$ be two sets of elements. Let $h_1(x) := \texttt{ASCII}(x) \cdot 3 \bmod 16$ and $h_2(x) := \texttt{ASCII}(x) - 3 \bmod 16$. To represent the sets R and S in Bloom filters r and s of size $m = 16$, each element is hash mapped in the bit vector by using hash functions h_1 and h_2.

The hash values are $h_1(\text{‘a’}) = 97 \cdot 3 \bmod 16 = 3, h_1(\text{‘b’}) = 6, h_1(\text{‘c’}) = 9, h_1(\text{‘d’}) = 12, h_1(\text{‘e’}) = 15$ and $h_2(\text{‘a’}) = 14, h_2(\text{‘b’}) = 15, h_2(\text{‘c’}) = 0, h_2(\text{‘d’}) = 1, h_2(\text{‘e’}) = 2$. This will lead to the Bloom filters $r = [1101\,0010\,0100\,1011]$ and $s = [1011\,0010\,0100\,0011]$ with $||r||_1 = 8$, $||s||_1 = 7$, $||x \wedge y||_1 = ||[1001\,0010\,0100\,0011]||_1 = 6$, and $||x \vee y||_1 = ||[1111\,0010\,0100\,1011]||_1 = 9$.

As a consequence, $sim_{Jaccard}(r, s) = 2/3$, $sim_{Dice}(r, s) = 4/5$, and $sim_{Simpson}(r, s) = 6/7$, while $sim_{Jaccard}(R, S) = 3/5$, $sim_{Dice}(R, S) = 3/4$, and $sim_{Simpson}(R, S) = 3/4$.

Definition 2.8.1.2: Similarity Measures for Bloom Filters / Bit Vectors

Let $x, y \in \mathbb{B}^m$ be two bit vectors. Then, their similarity can be computed in different ways [CCT10]:

- Jaccard similarity:

$$sim_{Jaccard}(x,y) = \frac{||x \wedge y||_1}{||x \vee y||_1} = \frac{||x \wedge y||_1}{||x||_1 + ||y||_1 - ||x \wedge y||_1} \tag{2.43}$$

- Dice similarity:

$$sim_{Dice}(x,y) = \frac{2 \cdot ||x \wedge y||_1}{||x||_1 + ||y||_1} \tag{2.44}$$

- Overlap (Simpson) similarity:

$$sim_{Simpson}(x,y) = \frac{||x \wedge y||_1}{\min(||x||_1, ||y||_1)} \tag{2.45}$$

- Braun-Blanquet similarity:

$$sim_{BraunBlanquet}(x,y) = \frac{||x \wedge y||_1}{\max(||x||_1, ||y||_1)} \tag{2.46}$$

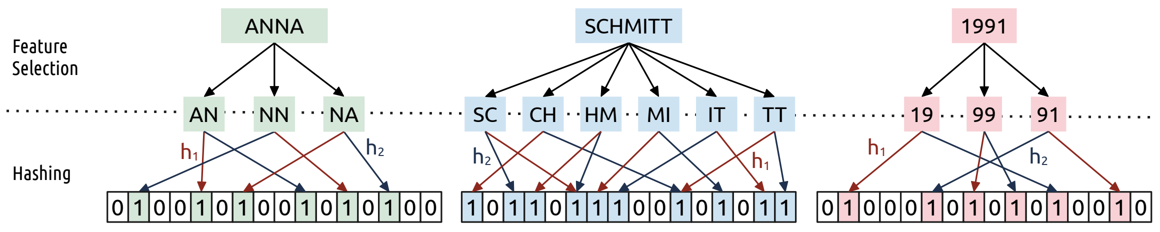
2.8.2 Utilization in PPRL

The main idea for utilizing Bloom filters in PPRL scenarios is to use a Bloom filter to represent the records attribute values, i. e., all quasi-identifying attributes of a person that are relevant for linkage, e. g., first name, last name, date of birth, and place of birth. The Bloom filters hash functions need to be cryptographic (one-way) hash functions that are keyed (seeded) with a secret key $\mathcal{S}$, i. e., keyed-hash message authentication codes (HMACs) like `MD5` or `SHA-1` [Nat08]. For approximate matching, the granularity of the record attributes is increased by segmentation into features. A widely used approach is to split the attribute values into small substrings of length q, called q-grams, typically setting $1 \leq q \leq 4$. Consequently, in PPRL, a Bloom filter represents a set of attribute value segments, that we term record (attribute) features. Thus, the number of common 1-bits of two Bloom filters approximates the number of common (overlapping) features between two records. In the following, we will describe the different types of Bloom filters and their privacy properties.

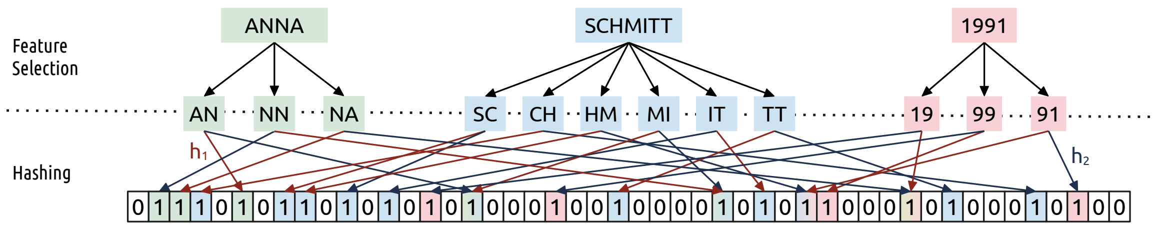
2.8.2.1 Types

There are two ways of encoding records into Bloom filters: either one Bloom filter is built for each record attribute, which is known as field- or attribute-level Bloom filter, or a single Bloom filter is built for all relevant attributes, which is known as record-level Bloom filter. For constructing record-level Bloom filters, there are two approaches: The first approach is called *Cryptographic Longterm Key* (CLK) [SBR11] and builds a single Bloom filter in which all record attributes are hashed. The second approach [Dur+14] first constructs attribute-level Bloom filters and then selects bits from these individual Bloom filters according to the weight of the respective attribute. In this work, we will focus on the first approach since it is heavily used in both research and practice [CRS20]. To support attribute weighting in the CLK approach as well, in [Vat+17] the authors propose to select different numbers of hash functions k for different attributes depending on their weight. Attributes with higher discriminatory power or lower error rates are therefore assigned more hash functions and thus set more bits in the Bloom filter.

The basic Bloom filter building process for attribute-level and record-level (CLK) Bloom filter is shown in Figure 2.5. In this example, the first name, last name, and year of birth attributes are segmented into q-grams (with $q = 2$), which are then mapped to the bit vector(s) using $k = 2$ hash functions. The hash functions for different attributes do not necessarily have to be the same. Either different hash functions or different secret keys can be used for each record attribute. Also, a different segmentation strategy can be used for different attributes, e. g., a different value for q.



(a) Attribute-level Bloom Filter



(b) Record-level Bloom Filter

Figure 2.5: Types of Bloom Filters.

The advantage of using attribute-level Bloom filters is that individual Bloom filters are produced allowing the use of sophisticated matching techniques known from traditional record linkage, for instance, classification based on attribute weights and attribute error rates (Fellegi-Sunter model), as well as approaches for handling null-valued attributes or composite fields, for instance, name attributes with compounds (multiple given names). However, as we discuss in the next section, attribute-level Bloom filters fulfill much weaker privacy properties compared to record-level Bloom filters.

2.8.2.2 Privacy Properties

The privacy-preserving properties of Bloom filters rely on the following aspects:

- An adversary has no information on how the record features are obtained, for instance, the selected attributes or length of substrings (q-grams).
- The selected hash functions, the secret key $\mathcal{S}$, and thus the hash mapping of record features to bit positions is unknown to an adversary. In particular, the use of keyed hash functions is essential to prevent dictionary attacks.
- Due to collisions, multiple record features will map to a single bit position in general. Keeping the Bloom filter size m fixed, the more hash functions are used, and the more features are mapped to the Bloom filter, the higher the number of collisions and thus the confusion.
- There is no coherence or positional information. Since a Bloom filter encodes a *set* of record features, it is not obvious from where features were obtained, i. e., within an attribute (which position) and for record-level Bloom filter even from which attribute.

However, Bloom filters are susceptible to frequency attacks as the frequencies of set bit positions correspond to the frequencies of record features [Vid+22; Vid+23]. Thus, frequently (co-)occurring record features will lead to frequently set bit positions or even to frequent Bloom filters in the case of attribute-level Bloom filters. By using publicly available datasets containing person-related data, e. g., telephone books, voter registration databases, social media profiles, or databases about persons of interest like authors, actors, or politicians, an adversary can estimate the frequencies of record features and then try to align those frequencies to the Bloom filters bit frequencies.

A successful re-identification of attribute values encoded in Bloom filters is a real threat as shown by several attacks proposed in the literature. Earlier attacks, namely [Kuz+11; KS14; Nie+14; Mit+16], often exploit the hashing method used in [SBR11], the double-hashing scheme, that combines two hash functions to implement the k Bloom filter hash functions. This hashing method can easily be replaced by using independent hash functions or other techniques as discussed in Section 7.2.2.1. Furthermore, these attacks rely on many unrealistic assumptions, for instance, that the encoded records are a random sample of a resource known to the adversary [Kuz+11; Nie+14] or that all parameters of the Bloom filter process, including used secret keys for the hash functions, are known to the adversary [Mit+16].

However, recent frequency-based cryptanalysis attacks, namely [Chr+18a] and in particular [Chr+18b], are able to correctly re-identify attribute values without relying on such assumptions. These attacks are more successful, the fewer attributes are encoded in a Bloom filter, and the larger the number of encoded records. Furthermore, Vidanage et al. [Vid+20a] proposed an attack on similarity graphs. Their attack is not limited to any specific PPRL method and aims to determine a mapping between the encoded records and plaintext records from a public database by using different graph features, for instance, weighted node degree and centrality measures.

Overall, the attacks show the risk of re-identification when using Bloom filters, especially attribute-level Bloom filters. Therefore, several techniques have been investigated to reduce the weaknesses of Bloom filters. We will describe and evaluate these techniques in Chapter 7.

3

Parallel Privacy-Preserving Record Linkage using LSH-based Blocking

This chapter is based on [FSR18]. To achieve high scalability of PPRL to large datasets with millions of records, we propose parallel PPRL (P3RL) approaches that build on modern distributed dataflow frameworks. The proposed P3RL approaches also include blocking for further performance improvements, in particular the use of LSH (locality-sensitive hashing) that supports a flexible configuration and can be applied on encoded records. We extensively evaluate the proposed LSH-based P3RL approaches on different datasets and cluster sizes.

3.1 Motivation

Nowadays, large amounts of person-related data are stored and processed, e. g., about patients or customers. For a comprehensive analysis of such data, it is often necessary to link and combine data from different data sources, e. g., for data integration in health care or business applications [Chr12b]. PPRL approaches aim at identifying records from different data sources referring to the same person without revealing personal identifiers or other sensitive information. PPRL is confronted with Big Data challenges, particularly high data *volumes* and different data representations and qualities (*variety*, *veracity*).

The aim of the work in this chapter is to improve the scalability and overall performance of PPRL by supporting both parallel PPRL (P3RL) and blocking. Our P3RL approaches enable the utilization of large shared nothing clusters running state-of-the-art distributed processing frameworks, such as Apache Flink [The23a] or Apache Spark [The23b], to reduce the execution time proportional to the number of processors in the cluster. Our P3RL approaches utilize blocking to partition the database records so that only records within the same block need to be compared. These comparisons are performed in parallel by distributing the blocks among all worker nodes within the computer cluster.

In this work, we focus on blocking based on locality-sensitive hashing (LSH) [IM98; Dur12] which can be applied on encoded data and is not domain-specific. To comparatively evaluate the efficiency of LSH in a parallel setting, we further developed a modified phonetic blocking approach based on Soundex [OR18]. We parallelize both approaches and compare them in terms of quality, efficiency, and scalability.

Following previous work, we realize our P3RL approach as a three-party protocol using a trusted third party to conduct the linkage, the linkage unit [VCV13]. The use of such a linkage unit is well-suited for P3RL because the linkage unit can maintain a high-performance computer cluster. As a privacy technique, we use the widely-used method by Schnell and colleagues [SBR11] to encode record attribute values in Bloom filters (see Section 2.8).

Specifically, we make the following contributions:

- We develop parallel PPRL (P3RL) approaches with LSH-based and phonetic blocking using a state-of-the-art distributed processing framework to efficiently execute PPRL on large-scale clusters. For LSH blocking, we include optimizations such as to avoid redundant match comparisons.
- We comprehensively evaluate the quality, efficiency, scalability, and speedup of our P3RL approaches for different parameter settings and large datasets with up to 16 million records in a cluster environment with up to 16 worker nodes.

After a discussion of related work in the next section, we describe the fundamentals of LSH-based blocking in Section 3.3. Then, in Section 3.4, we present our P3RL approaches using Apache Flink for both LSH and phonetic blocking. In Section 3.5, we evaluate our approaches for different datasets and cluster sizes. Finally, we conclude this chapter in Section 3.6.

3.2 Related Work

Record linkage approaches aim at achieving a high match quality and scalability to large datasets [KR10]. To reduce the number of record comparisons, blocking techniques are frequently used [Chr12a]. As discussed in Section 2.6.3.1, the standard blocking method defines one or multiple blocking keys to group records into blocks such that only records of the same block are compared. A blocking key is determined by applying a function on one or more selected record attributes [Fis+15]. For example, one could block persons based on the Soundex value of their last name (phonetic blocking) or on the concatenation of the initial two letters of their first name and the city of birth.

Analogous to traditional record linkage, blocking techniques are applied to make PPRL scalable to large datasets. A common approach is to use phonetic codes to enable blocking based on phonetic similarities of attribute values [KV09]. LSH-based blocking has been shown to achieve high match quality as well as scalability to large datasets [Dur12; KV15; KV16].

Parallel record linkage (PRL): For a further improvement of the scalability, several parallelization techniques have been considered for traditional record linkage, as surveyed in [CSS18]. On the one hand, graphics processing units (GPUs) have been used to speed up similarity computations of candidate record pairs [For+13; Ngo+13]. Another widely-used approach is to utilize parallel processing frameworks, such as Hadoop MapReduce, to conduct the linkage in a parallel and distributed fashion [San+07; Wan+10; BGH11; KTR12; Eft+17; Pap+17; GH21].

Parallel privacy-preserving record linkage (P3RL): We are aware of only a few studies on parallel approaches for PPRL. In [Seh+15], graphics processors are utilized for a parallel matching of Bloom filters (bit vectors). In [Gla+18], a distributed PPRL approach is proposed using a pivot-based filtering method.

Furthermore, the authors of [KV13] and [KV14] proposed the use of MapReduce to improve the scalability of PPRL based on Bloom filters using LSH-based blocking. Like in the MapReduce approaches for record linkage [KTR12], the map function is used to determine the blocking key values (LSH keys) in parallel. Then, the records are grouped based on their LSH keys and compared block-wise in the reduce step. To address the problem of duplicate candidate pairs in different blocks, two MapReduce jobs are chained. While the first job only emits the IDs of candidate record pairs, the second job groups equal candidate pairs in the reduce step to calculate the similarity of each pair only once. Here the usage of MapReduce shows some limitations: In contrast to modern distributed processing frameworks like Apache Flink, MapReduce does not support complex user-defined functions and requires expensive job chaining and other workarounds instead. Moreover, the evaluation is limited to two and four nodes and small datasets of only about 300 000 records. As a result, the scalability of the approach to larger datasets with millions of records and larger clusters remains open.

More recently, in [KK23], the authors investigated further parallel PPRL approaches utilizing Apache Spark and blocking based on phonetic codes.

3.3 Locality-sensitive Hashing

Locality-sensitive hashing (LSH) was proposed to solve the nearest neighbor problem in high-dimensional data spaces [IM98]. For LSH, a family of hash functions that is sensitive to a distance measure d is used (see Definition 2.6.3.1). Such a family is defined as follows:

Definition 3.3.0.1: LSH Family

Let d_1, d_2 with $d_1 < d_2$ be two distances according to a distance measure d over a metric space $\mathcal{M}(U, d)$. Moreover, let pr_1 and pr_2 with $pr_1 > pr_2$ be two probabilities. A family of hash functions $\mathcal{F}$ is called (d_1, d_2, pr_1, pr_2)-sensitive if for all $f \in \mathcal{F}$ and for all elements $x, y \in U$ the following conditions are met:

- $d(x, y) \leq d_1 \quad \Rightarrow \quad \mathbb{P}\left[f(x) = f(y)\right] \geq pr_1$
- $d(x, y) \geq d_2 \quad \Rightarrow \quad \mathbb{P}\left[f(x) = f(y)\right] \leq pr_2$

With that, the probability that a function $f \in \mathcal{F}$ returns the same output for two elements with a distance smaller or equal to d_1 is at least pr_1. Otherwise, if the distance is greater or equal to d_2, then the probability that f returns the same output is at most pr_2.

For applying LSH as a blocking method for PPRL, the two hash families approximating the Jaccard and the Hamming distance are most relevant [Dur12]. We focus on the hash family $\boldsymbol{\mathcal{F}_{\mathcal{H}}}$ that is sensitive to the Hamming distance (HLSH). Each function $f_i \in \mathcal{F}_{\mathcal{H}}$ with $0 \leq i < m$ maps a Bloom filter Bf_j representing a record r_j to the bit value on position i of Bf_j. For LSH-based blocking, a set $\Theta = \{f_{\lambda_1}, \ldots, f_{\lambda_\Psi} \mid f_{\lambda_\iota} \in \mathcal{F}\}$ of $\boldsymbol{\Psi}$ hash functions is used. To group similar records, a blocking key $BK_\Theta(\text{Bf}_j)$ is generated by concatenating the output values of the hash functions $f_\lambda \in \Theta$, such that $BK_\Theta(\text{Bf}_j) = f_{\lambda_1}(\text{Bf}_j) \odot \ldots \odot f_{\lambda_\Psi}(\text{Bf}_j)$, where $\odot$ denotes the concatenation of hash values. As a blocking key consists of $\Psi < m$ function values, Ψ defines the length of the blocking (LSH) key. Based on the probabilistic assumption of LSH, the blocking keys of two similar Bloom filters with a distance smaller or equal to d_1 may be different. For this reason, $\boldsymbol{\Lambda}$ blocking keys $BK_{\Theta_1}, \ldots, BK_{\Theta_\Lambda}$ are used to increase the probability that two similar Bloom filters have at least one common blocking key.

The parameters Ψ and Λ influence the efficiency and effectiveness of an LSH-based blocking approach. The higher Ψ (LSH key length) the higher is the probability that only records with a high similarity are assigned to the same block. Thus, the number of records per block will be smaller. However, a higher Ψ also raises the probability that matching records are missed due to erroneous data. Λ determines the number of blocking keys. Hence, a higher value of Λ increases the probability that two similar Bloom filters have at least one common blocking key. However, an increasing Λ leads to more computations and can deteriorate scalability. Basically, Λ should be as low as possible while Ψ is high enough to build as many blocks so that the search space is greatly reduced. An optimal value for Λ can analytically be determined dependent on

Ψ and on the similarity of true matching Bloom filter pairs [KV14]. This is difficult to utilize in practice since the similarity of true matches is generally unknown. Using HLSH, Ψ should be sufficiently large because each function $f \in \mathcal{F}_{\mathcal{H}}$ can only return 0 or 1 as output. Thus, at most 2^{Ψ} blocking key values (blocks) exist for one HLSH key.

Example: Let $\Theta_1 = \{f_7, f_1\}$, $\Theta_2 = \{f_0, f_5\}$ and $\mathrm{Bf}_1 = \mathbf{11}011\mathbf{0}\mathbf{11}$, $\mathrm{Bf}_2 = \mathbf{10}011\mathbf{0}\mathbf{11}$ two Bloom filters. We get $BK_{\Theta_1}(\mathrm{Bf}_\mathbf{1}) = f_7(\mathrm{Bf}_1) \odot f_1(\mathrm{Bf}_1) = 11$, $BK_{\Theta_2}(\mathrm{Bf}_\mathbf{1}) = f_0(\mathrm{Bf}_1) \odot f_{15}(\mathrm{Bf}_1) = 10$ and $BK_{\Theta_1}(\mathrm{Bf}_\mathbf{2}) = 10$, $BK_{\Theta_2}(\mathrm{Bf}_\mathbf{2}) = 10$. Hence, Bf_1 and Bf_2 agree on BK_{Θ_2} and will be put into the same block for that LSH key, and finally be compared in detail.

3.4 Parallel PPRL (P3RL)

We now explain our P3RL framework based on Apache Flink. At first, we give a brief introduction to Flink and outline the basic concepts of our framework. We then describe the implementation of the PPRL process using HLSH and phonetic blocking. For our HLSH-based blocking approach, we propose two optimizations that aim at avoiding duplicate match comparisons and restrict the choice of HLSH keys by avoiding the most frequent 0/1-bit positions.

3.4.1 Apache Flink

We based our implementation on Apache Flink [Car+15] which is an open-source framework for the in-memory processing of distributed dataflows. The use of distributed dataflow systems, such as Apache Flink, simplifies the development of distributed programs because these systems provide common data processing operations and handle all the technical aspects of parallelization. Therefore, Flink programs can be automatically executed in parallel on large-scale computer clusters.

A Flink program is defined through *streams* and *transformations*. Streams are collections of arbitrary data objects. Since we have a fixed set of input records, we use bounded streams of data, called *DataSets*, and the associated *DataSet API*. Transformations produce new streams by modifying existing ones. Multiple transformations can be combined to perform complex user-defined functions. Flink offers a wide range of transformations, some of which are adopted from the MapReduce paradigm [DG08]. Basic transformations of the DataSet API include Map, FlatMap, Reduce, GroupReduce, CoGroup, and Join.

At runtime, Apache Flink deploys two types of processes, called JobManager and TaskManager [Car+15]. A JobManager coordinates the distributed execution of a Flink program by scheduling tasks and managing resources. A TaskManager, in contrast, corresponds to a worker node and executes parts of the parallel program, the tasks of a dataflow. To control how many tasks a TaskManager accepts, the number of its task slots is defined. Each task slot represents a fixed subset of resources (e. g., managed memory) of the TaskManager. One or more subtasks of the Flink program are assigned to the task slots and executed there in separate threads. Each TaskManager is a Java virtual machine (JVM) that processes these threads.

3.4.2 General Approach

The general approach of our distributed framework using Flink is illustrated in Figure 3.1. At first, each party individually performs a pre-processing step where records are encoded, and static blocking keys can be defined [VCV13]. Then, the parties send their encoded records as Bloom filters to the linkage unit. The linkage unit utilizes an HDFS cluster and stores the encoded records distributed and replicated among the cluster nodes. To conduct the linkage, the data is read in parallel by the nodes. If the encoded records do not contain blocking keys, the linkage unit generates them. Afterward, the blocking step is conducted to group together similar records for search space reduction. Hence, the records are distributed and redirected among the cluster nodes based on their blocking keys. Thereby, all records with the same blocking key value are sent to the same worker node. Finally, the workers build candidate pairs, optionally remove duplicates and perform the similarity calculations in parallel. The IDs of the matching pairs are then sent back to the data owners.

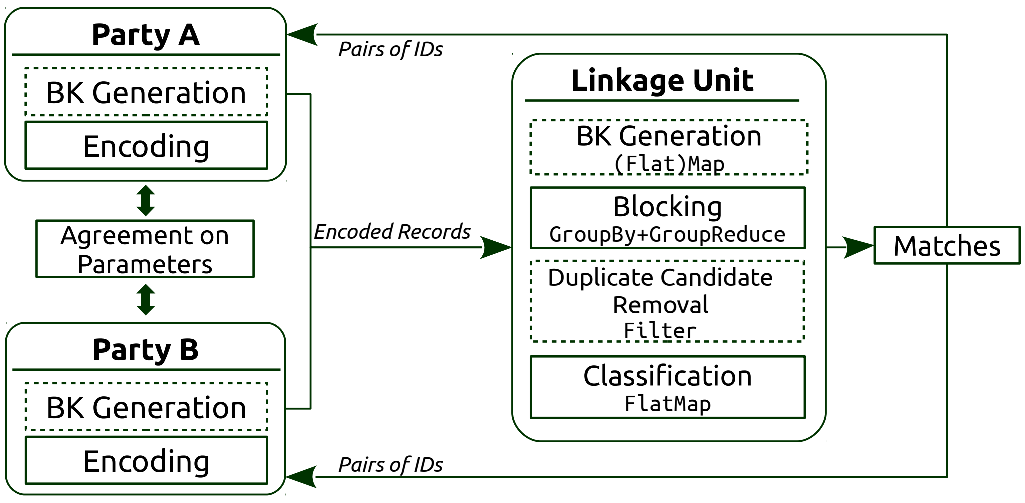


Figure 3.1: General approach of the P3RL using Apache Flink. A dotted box indicates an optional step.

3.4.3 Hamming LSH

The first step of our HLSH blocking approach is to calculate the blocking keys BK_{Θ_1}, $BK_{\Theta_2}, \ldots, BK_{\Theta_\Lambda}$ for every input Bloom filter Bf_i within a `FlatMap` function. Such a function applies a user-defined `Map` function to each record and returns an arbitrary number of result elements.

We choose $f_{\lambda_1}, \ldots, f_{\lambda_\Psi}$ randomly from $\mathcal{F}_\mathcal{H}$ for each blocking key, but using each function $f_{\lambda_\iota} \in \mathcal{F}_\mathcal{H}$ only once so that each HLSH key uses different bit positions. More formally, we set $\Theta_x \cap \Theta_y = \emptyset,\ \forall x, y \in \{1, \ldots, \Lambda\}$. The output of the `FlatMap` function are tuples of the form `(keyID, keyValue, record)`. Each Bf_i is replicated Λ times since it produces a tuple $T^i_j = \left(j,\ BK_{\Theta_j}(\mathrm{Bf}_i),\ \mathrm{Bf}_i\right)$ for every $j \in \{1, \ldots, \Lambda\}$. The first two fields of T correspond to the HLSH key with its ID and the third field consists of the input Bloom filter.

Then, on the first two fields of each tuple T^i_j a `GroupBy` function is applied. By that, the tuples are redistributed so that tuples with the same HLSH key value are assigned to the same block and worker node. By using a `GroupReduce` function, every pair of tuples within a block builds a candidate pair $C^{i',i''}_j = (\mathrm{Bf}_{i'}, \mathrm{Bf}_{i''})$ if the Bloom filters originate from different parties. A `GroupReduce` function is similar to a `Reduce` function, but it gets the whole group (block) at once and returns an arbitrary number of result elements.

Finally, the similarity of all candidate pairs is computed within a `FlatMap` function to output only candidates with a sufficient similarity value. Afterward, the matches are written into the Hadoop distributed file system (HDFS).

3.4.3.1 Removal of Duplicate Candidate Pairs

By using multiple blocking keys, LSH generates overlapping blocks. Consequently, pairs of encoded records may occur in multiple blocks so that they are compared several times. To avoid these redundant similarity calculations, we adapted the approach from [KTR13], so that for every tuple T^i_j additionally a list of all HLSH keys until the $(j-1)$-th key is emitted. We get tuples $\hat{T}^i_j = (j,\ BK_{\Theta_j}(\mathrm{Bf}_i),\ \mathrm{Bf}_i,\ \mathit{keys}_j(\mathrm{Bf}_i))$ with $\mathit{keys}_j(\mathrm{Bf}_i) = \{BK_{\Theta_1}(\mathrm{Bf}_i), \ldots, BK_{\Theta_{j-1}}(\mathrm{Bf}_i)\}$.

For each candidate pair $\hat{C}^{i',i''}_j = \left(\left(\mathrm{Bf}_{i'},\ \mathit{keys}_j(\mathrm{Bf}_{i'})\right), \left(\mathrm{Bf}_{i''},\ \mathit{keys}_j(\mathrm{Bf}_{i''})\right)\right)$ it is checked, if the HLSH key lists are disjoint, i. e., if $\mathit{keys}_j(\mathrm{Bf}_{i'}) \cap \mathit{keys}_j(\mathrm{Bf}_{i''}) = \emptyset$. If they are disjoint, then BK_{Θ_j} is the least common HLSH key and the candidate pair is compared. Otherwise, a $BK_{\Theta_{j'}}$ with $j' < j$ exists so that the candidate pair is already considered in another block and can be pruned for BK_{Θ_j}. We realized this overlap check with a `Filter` function which is applied to each candidate pair. If the function evaluates to true, i. e., the key lists do not overlap, the similarity of the candidate pair will be calculated. Otherwise, the filter function evaluates to false and the candidate pair is pruned.

The avoidance of redundant match comparisons leads to additional computational effort. At maximum $\mathcal{O}(\Lambda - 1)$ HLSH keys need to be compared for each candidate pair. Moreover, the tuple objects are larger leading to higher network traffic. As mentioned in [KTR13] the removal of duplicate candidate pairs can also lead to load balancing issues. By focusing on the least common HLSH keys, a Bloom filter pair will be only processed for the HLSH key with the smallest index. Consequently, a pair with a higher index (with respect to the HLSH key) is more likely to be considered a duplicate, potentially introducing skew effects.

3.4.3.2 HLSH Key Restriction (HLSH-KR)

HLSH uses randomly selected bits from bit vectors (Bloom filters) to construct the blocking key values. By that, HLSH applies probabilistic blocking on the presence or absence of certain q-grams in the attribute values of a record.

With growing data volume some q-grams can occur in many records, because of limited real-world designations and namespaces that can be built by linguistic units (e. g., morphemes, phonemes) of natural languages. For example, most residential addresses end with suffixes like ‘street’ or ‘road’. Even if such suffixes are abbreviated, many records will produce q-grams like ‘st’ or ‘rd’ resulting in the same 1-bits in the Bloom filters. Another example is the attribute gender, assuming only two possible values (‘female’, ‘male’). If mapped into Bloom filters, every Bloom filter will contain the same 1-bits corresponding to q-grams resulting from the substring ‘male’. If such frequently occurring 1-bits are used to construct an HLSH key, many records will share the same HLSH key and are assigned to the same block. This will lead to large blocks with records only agreeing on q-grams having a low discriminatory power.

On the other hand, bit positions where the majority of bits are 0-bits can exist. For example, some q-grams are very rare, because they are not common or are only existing due to erroneous data (typographical errors). Again, using these bit positions for constructing HLSH keys can lead to large blocks, because many records share the property that they do not contain this information.

To overcome these issues, very frequent or infrequent q-grams could be treated as stop words to avoid them being encoded into the Bloom filters. However, the identification of such q-grams depends on the domain and on the used record attributes. Removing q-grams also influences the linkage quality by changing the similarity values of Bloom filter pairs.

Based on these observations, we propose to consider only those bit positions for the HLSH keys that are not frequently set to 0 or 1, respectively. For this purpose, we count the number of 1-bits for each position and for all input Bloom filters. The resulting list of bit positions is sorted with respect to the number of 1-bits at the corresponding position in ascending order. Then, we remove $\frac{1}{v}$ bit positions at the beginning (frequent 0-bits) and at the end of the list (frequent 1-bits) resulting in a bit position list P. Here v denotes the pruning proportion for frequent bits. For the HLSH key generation, we choose the hash functions randomly from $\tilde{\mathcal{F}_{\mathcal{H}}} = \{f_\iota \in \mathcal{F}_{\mathcal{H}} \mid \iota \in P\}$.

3.4.4 Phonetic Blocking

To comparatively evaluate our HLSH approach, we consider phonetic blocking (PB) as a baseline for comparison. The idea of phonetic blocking is to use a phonetic encoding function that produces the same output for input values with a similar pronunciation. Usually, attributes like surname or given name are used to group persons with a similar name while ignoring typographical variations. For phonetic blocking, the blocking key is constructed during the pre-processing step. Each party individually builds for every record a phonetic code for a selected attribute.

Phonetic codes inherently provide some degree of privacy by producing the same encoding for multiple similar-sounding values (one-to-many mapping). However, providing phonetic codes as plaintext reveals some information about the encoded records. For example, Soundex reveals the first letter of an attribute value, thereby providing an entrance point for cryptanalysis. Therefore, we encode the phonetic code for a record r_i into a separate Bloom filter that is used as a regular blocking key. Consequently, two Bloom filters are sent to the linking unit for each record. As with the HLSH approach, records are first grouped into blocks using a `GroupBy` function, and then all candidate pairs for each block are generated by a `GroupReduce` function.

Since frequent/rare phonetic codes are potentially identifiable by analyzing the frequency distribution of the blocking Bloom filter values, we consider a second variant denoted as salted phonetic blocking (SPB). Similar to [Sch15], we select a record-specific key used as salt for the Bloom filter hash functions to affect the hash values and corresponding Bloom filter bit positions. If the salting keys of two records differ, it is very unlikely that the same phonetic code will result in the same Bloom filter. Consequently, the attribute(s) used as the salting key should be free of errors because otherwise, many false negatives will occur.

Following this idea, for salted phonetic blocking, we use a second phonetic code as salt such that each hash function gets as input both phonetic codes concatenated. Since only records agreeing in both phonetic codes are assigned to the same block, the number of blocks increases and the block sizes decrease. However, if only one attribute used for blocking contains an error that cannot be compensated by the phonetic encoding function, then even two similar records are assigned to different blocks, and thus excluded from a detailed comparison.

3.5 Evaluation

In this section, we evaluate our P3RL approaches in terms of quality, scalability, and speedup. Before presenting the evaluation results, we describe our experimental setup and the datasets and metrics we used.

3.5.1 Experimental Setup

We conducted our experiments using a cluster with 16 worker nodes. Each worker is equipped with an Intel Xeon E5-2430 CPU with 6×2.5 GHz, 48 GB RAM, and two 4 TB SATA disks running openSUSE 13.2. The nodes are connected via 1 Gbit Ethernet. We use Hadoop 2.6.0 and Flink 1.3.1. Flink is executed standalone with 1 JobManager and 16 TaskManagers, each with 6 TaskSlots and 40 GB JVM heap size.

3.5.2 Datasets

We generated synthetic datasets using the data generator and corruption tool GeCo published in [CV13]. We replaced the lookup files for attribute values by German names and address lists because the lookup files shipped with GeCo are very small, which makes them not suitable for generating large datasets. We also added realistic frequency values for these names and addresses, which are drawn from German census data [Sta23a].

To consider different PPRL scenarios, we generated datasets D_S and D_R. With $\mathbf{D_S}$ a *statewide* linkage is simulated by using the complete look-up files. In contrast, we restrict the addresses for records from $\mathbf{D_R}$ to a certain region by only considering cities whose zip code starts with '04'. With that, D_R simulates a *regional* linkage scenario which imitates a linkage of patient records from local health care providers.

For both datasets, we build subsets D_{S_n} and D_{R_n} of size $n \cdot 10^6$ for all $n \in \{2^b \mid b \in \mathbb{N} : b \leq 4\}$. Each obtained dataset consists of $(4n/5) \cdot 10^6$ original and $(n/5) \cdot 10^6$ randomly selected duplicate records. To simulate dirty data, each duplicate is corrupted by choosing α attributes that are modified. A maximum of β modifications are then introduced into each of these attributes. We get datasets $D_{S_n}^{(\alpha,\beta)}$ and $D_{R_n}^{(\alpha,\beta)}$, respectively.

We consider two levels of corruptions (*moderate* and *high*) by generating $D^M_{S_n}$, $D^M_{R_n}$ with $M = (2,1)$ and $D^H_{S_n}$ with $H = (3,2)$.

Moreover, the records are encoded into CLK Bloom filters by tokenizing the values of the attributes $\mathcal{A} = \{$surname, first name, date of birth (dob), city, zip code$\}$ into a set of trigrams. We set $k = 20$.

To determine m_{opt}, we estimate the average number of trigrams $\omega(A)$ each attribute $A \in \mathcal{A}$ produces. We obtain $\omega(A) = (\Gamma_A - q) + 1$, where Γ_A is the average length of values from attribute A. Our estimation using character padding [Sch15] to build trigrams is shown in Table 3.1. To approximate the average total number of trigrams, we calculate $\sum_{A \in \mathcal{A}} \omega(A) = 42$ leading to $m_{opt} = 1212$ (see Section 2.8). For the phonetic blocking approaches PB and SPB, we choose the attribute surname and use the first name attribute as salt. This leads to an optimal blocking Bloom filter length of 29 using 20 hash functions.

	Name	**Surname**	**Dob**	**Zip + City**
Avg. attribute length	6	7	8	5 + 10
q-grams (q = 3)	8	9	10	15

Table 3.1: Estimated average attribute length and the resulting number of q-grams.

3.5.3 Evaluation Metrics

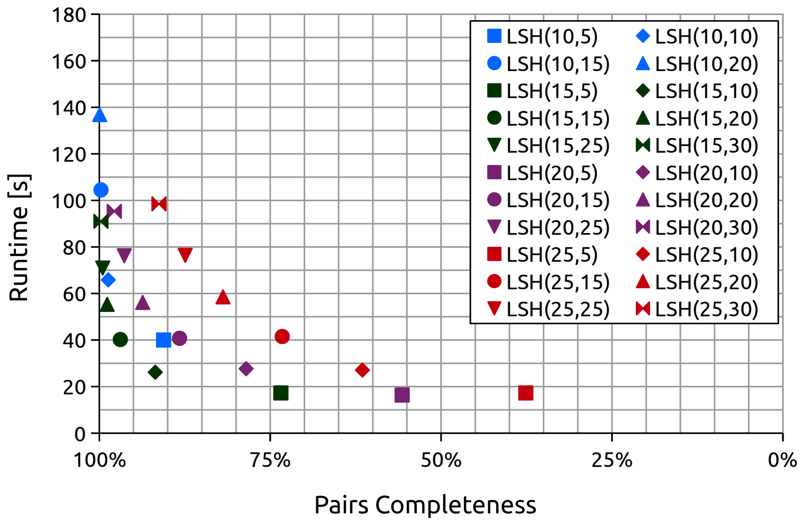
To assess the match quality of our blocking schemes, we measure the pairs completeness (PC) that is the ratio of true matches ($PC \in [0,1]$) that can be identified within the determined blocks [Chr12b]. To evaluate scalability, we measure the execution times for several datasets of different sizes. We also calculate the reduction ratio (RR) which is defined as the fraction of record pairs that are removed by a blocking method compared to the evaluation of the Cartesian Product (see Section 2.6.7.2). The achievable speedup is analyzed by utilizing different cluster sizes (Υ).

3.5.4 Experimental Results

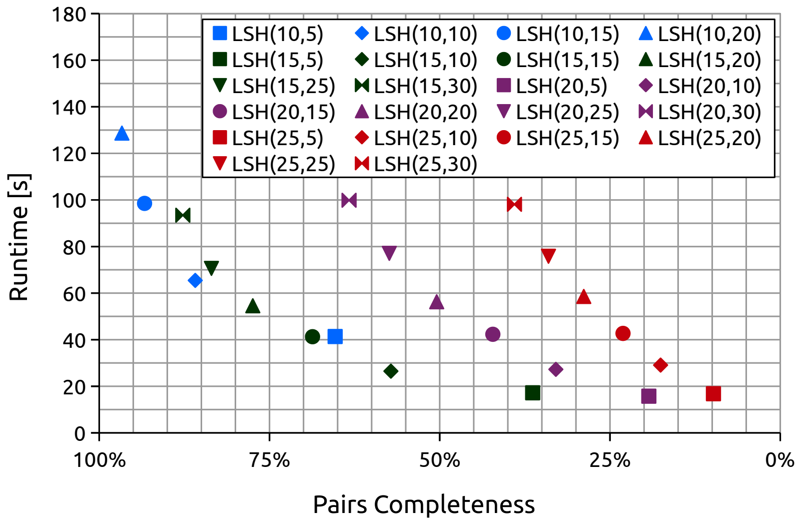
In the following, we present the results of our experimental evaluation.

3.5.4.1 HLSH Parameter Evaluation

To analyze effectiveness and efficiency, we evaluate different HLSH parameter settings by considering several HLSH key lengths with $\Psi \in \{10, 15, 20, 25\}$ and varying the



(a) $D^M_{S_1}$



(b) $D^H_{S_1}$

Figure 3.2: Pairs completeness against runtime of different HLSH parameter settings for $D^M_{S_1}$ and $D^H_{S_1}$ and $\Upsilon = 4$.

number of HLSH keys $\Lambda \in \{5, 10, 15, \ldots, 30\}$. For this experiment we use $D_{S_1}^M$ and $D_{S_1}^H$ (1 million records) setting $\Upsilon = 4$.

Figure 3.2 shows the PC values compared to the runtimes for the different HLSH settings, denoted as LSH(Ψ, Λ), for both moderate and high degrees of corruption. For a moderate degree of corruption (Figure 3.2a), several configurations achieve a good PC of over 95%. A high PC result is favored by low key lengths Λ (e. g., 10 or 15) and thus relatively large blocks. The best results for $\Psi = 10$ requires relatively high runtimes due to the larger block sizes compared to configurations with larger key lengths (since per key at most 2^{10} blocks are generated for $\Psi = 10$). A near-optimal PC with better runtimes is achieved with $\Psi = 15$ and $\Lambda = 20$. A large key length such as $\Psi = 25$ misses many matches and achieves only relatively low PC values even with many blocking keys, e. g., $\Lambda = 30$. This is even more apparent for the results with a high degree of corruption (Figure 3.2b). Here, choosing $\Psi \geq 20$ achieves unacceptably low PC values of less than 65%. The best results are achieved for $\Psi = 10$ and $\Psi = 15$ where the latter setting needs already a high number of blocking keys of at least 30. The best trade-off between PC and runtime can be achieved by using $\Psi = 15$. However, facing very dirty data as simulated with $D_{S_1}^H$, Λ should be sufficiently large with $\Lambda \geq 30$.

Dataset	Method	PC [%]	RR [%]	# Blocks	# Candidates
$D_{S_1}^M$	LSH(10,10)	98.68	98.47	1024	2446.25
	LSH(10,15)	**99.76**	97.69	1024	3702.66
	LSH(15,15)	96.92	99.90	32 425	161.56
	LSH(15,20)	98.85	99.87	32 448	213.11
	PB	86.70	99.62	2347	610.77
	SPB	73.11	**99.99**	**79 864**	**19.19**
$D_{S_1}^H$	LSH(10,10)	85.90	98.52	1,024	2375.22
	LSH(10,15)	**93.34**	97.76	1024	3580.99
	LSH(15,15)	68.68	99.91	32 489	151.01
	LSH(15,20)	77.45	99.88	32 504	199.74
	PB	75.50	99.66	2737	551.34
	SPB	54.90	**99.99**	**84 071**	**18.36**

Table 3.2: Comparison of the different blocking schemes considering PC, RR, the number of blocks (per HLSH key) and the number of candidates (in millions) for $D_{S_1}^M$ and $D_{S_1}^H$.

3.5.4.2 Comparison of HLSH and Phonetic Blocking

In the following, we compare the best-performing HLSH settings to the phonetic blocking approaches PB and SPB. The results using the same datasets as before are shown in Table 3.2.

We observe that PB and SPB achieve quite low pair completeness even for $D^M_{S_1}$. The reason is that phonetic blocking is more sensitive with respect to data errors compared to HLSH blocking. All approaches except those with $\Psi = 10$ achieve a high RR of over 99% and are thus able to greatly reduce the search space. However, the number of candidates varies significantly between the methods, even if the RR values are very close. For instance, the number of candidates for PB is more than a factor of 3.5 higher compared to LSH(15,15). SPB instead generates 30 times fewer candidate pairs compared to PB. This is due to the low number of blocks that are generated by PB. While Soundex theoretically can produce $26 \cdot 7^3$ blocking key values (blocks), less than $8 \cdot 7^3$ blocks are actually built, making PB not feasible for large datasets.

3.5.4.3 HLSH Optimizations

We evaluate the HLSH optimizations described in Sections 3.4.3.1 and 3.4.3.2 for the datasets D^M_S and D^M_R setting $v = \frac{1}{8}$ and $\Upsilon = 4$. Figure 3.3 shows the runtimes and the number of candidates (logarithmic scale on the right-side y-axis) achieved by LSH(15,20) while enabling our optimizations.

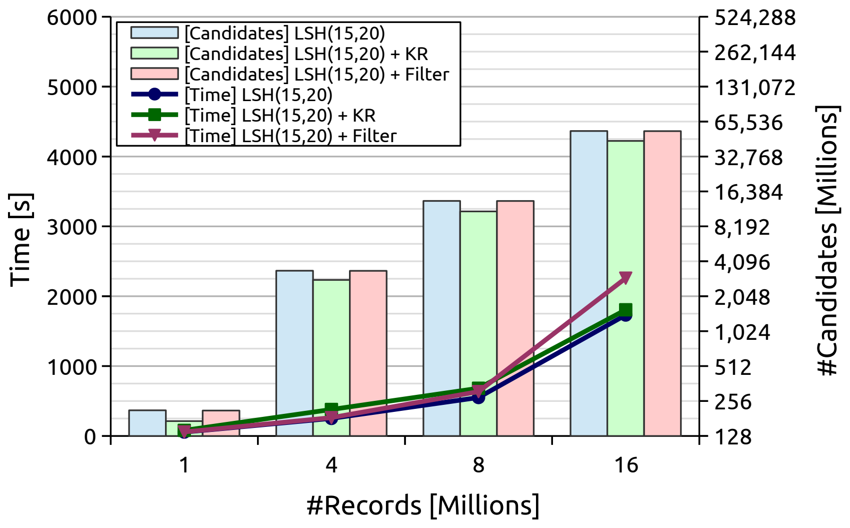
In general, the number of candidates for D^M_R is about 2.4 times as high as for D^M_S due to a higher degree of similarity between the records and thus larger blocks. For both datasets, the removal of duplicate candidates could reduce the number of candidates very little by at most 1%. Hence, the overhead checking the lists of HLSH keys was more significant, significantly increasing the overall run time.

By contrast, the key restriction approach HLSH-KR reduces the number of candidates substantially by up to 20% for D^M_S and even 40% for D^M_R. While for D^M_S the overhead for calculating the bit frequencies neutralizes the savings, for $D^M_{R_8}$ and $D^M_{R_{16}}$ HLSH-KR leads to significant runtime savings. HLSH-KR generally improves with growing data volumes since the data space becomes denser such that more frequent 0/1-bits occur, and thus avoiding overly large blocks becomes more beneficial. The overhead to determine frequent 0/1-bits is linear and becomes negligible for large datasets because the complexity of PPRL remains quadratic.

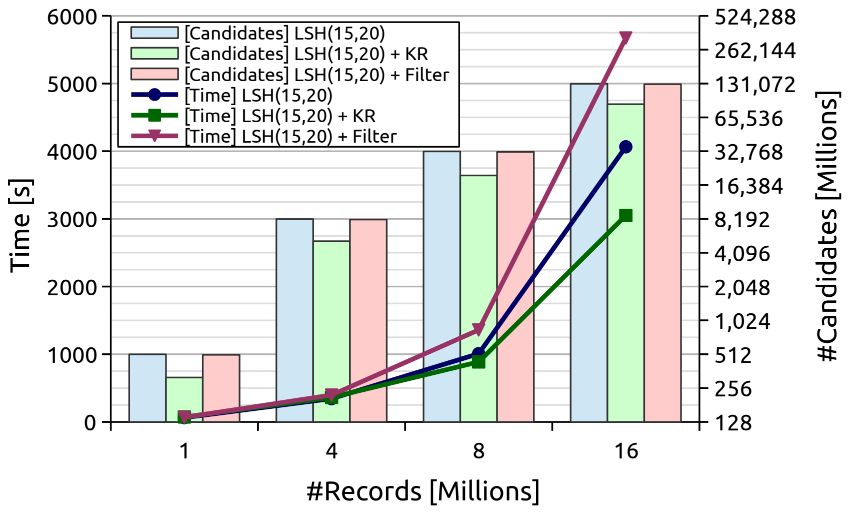
Using HLSH-KR also influences the linkage quality, since fewer bit positions are considered. For both datasets, we measured a loss of PC of around 1% while enabling HLSH-KR.

3.5.4.4 Scalability and Speedup

To evaluate the scalability, we execute our blocking approaches on the datasets $D^M_{S_1}$, $D^M_{S_2}, \ldots, D^M_{S_{16}}$ setting $\Upsilon = 16$. The runtime results depicted in Figure 3.4 show that



(a) D_S^M



(b) D_R^M

Figure 3.3: Evaluation of HLSH duplicate candidate filter and HLSH key restriction (denoted as KR) for D_S^M and D_R^M setting $\upsilon = \frac{1}{8}$ and $\Upsilon = 4$.

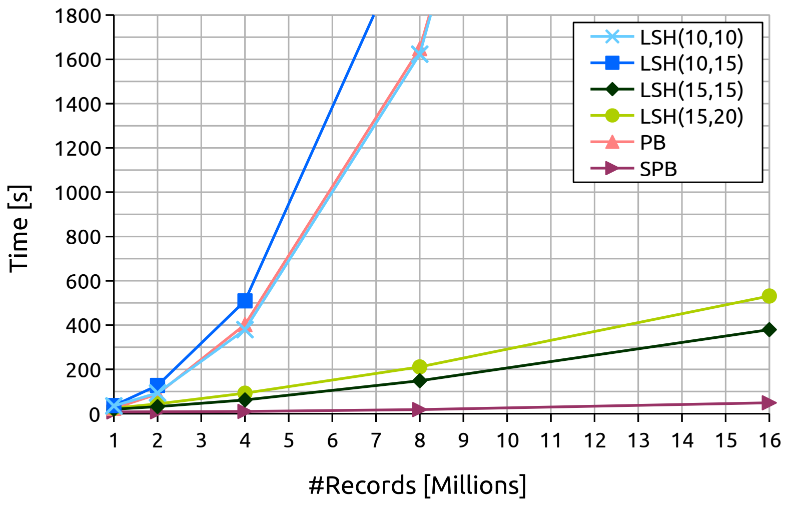


Figure 3.4: Execution times for different blocking methods for D_S^M with $\Upsilon = 16$.

LSH(10,10), LSH(10,15), and PB do not scale with respect to the number of records, which is due to the low number of blocks. Already for the dataset $D_{S_4}^M$, these approaches run more than 8 times slower than the other. For $D_{S_8}^M$ and $D_{S_{16}}^M$ the runtime increases drastically, so that PB, for example, requires around $1650\,s$ and $7120\,s$, respectively. By contrast, SPB can achieve the lowest execution times, albeit for an unacceptably low PC (Table 3.2). In general, even a slightly higher reduction ratio can lead to a huge performance improvement. The LSH blocking approaches with $\Psi = 15$ are able to efficiently link large datasets with a high match quality. For example, by using LSH(15,20) the linkage of dataset $D_{S_{16}}^M$ is performed in around nine minutes. It is also interesting that the runtime of LSH(15,20) is higher by a factor of 1.4 compared to LSH(15,15). Hence, the increase of the runtime corresponds to the higher value for Λ $(\frac{4}{3})$.

Finally, we evaluate the speedup by utilizing up to 16 worker nodes using $D_{S_1}^M$ and $D_{S_{16}}^M$. The results in Figure 3.5 show that for $D_{S_{16}}^M$ and up to eight worker nodes, the speedup is nearly linear for both LSH methods, while degrades after this.

In contrast, PB achieves a much lower speedup that degrades already for more than four workers. This is because of the low number of blocks (Table 3.2) and because of data skew effects introduced by frequent names, leading to large blocks that need to be processed on a single worker. This results in an uneven workload, causing some workers to determine the overall runtime. Even if parallelism is increased by adding more workers, each block is assigned to only one worker, resulting in the same workload for that node.

The speedup results for $D_{S_1}^M$ are lower compared to $D_{S_{16}}^M$ indicating that the lower data volume can be handled already by fewer workers. This is confirmed by the speedup of the SPB approach, which already degrades for more than two worker nodes. PB achieves a slightly higher speedup for $D^M_{S_1}$ compared to the $D^M_{S_{16}}$ dataset, which indicates that the performance problems introduced by data skew effects become more severe with increasing data volume.

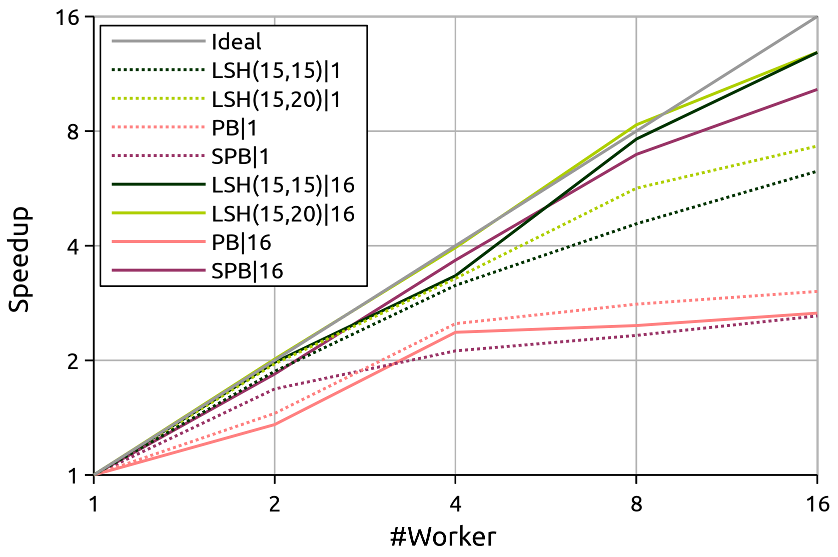


Figure 3.5: Speedup evaluation for the blocking methods.

3.6 Conclusion

We proposed and evaluated parallel PPRL approaches using LSH-based blocking executed within Apache Flink as a state-of-the-art distributed processing framework. Our evaluation using large datasets and different degrees of data quality showed the high efficiency and effectiveness of our LSH-based P3RL approaches, which clearly outperform approaches based on phonetic blocking. We also found that the overhead for finding duplicate candidates cannot be outweighed by the achievable savings. By contrast, avoiding the most frequent 0/1-bits for LSH blocking proved to be beneficial for very large datasets. For future work, we want to explore how optimal LSH parameters can be determined automatically. Moreover, we plan to investigate further P3RL approaches utilizing blocking and filtering to reduce the number of candidate record pairs.

4

LSH-based Blocking on Attribute-level Bloom Filters

This chapter is based on [Roh+21]. As shown in Chapter 3, utilizing locality-sensitive hashing (LSH) as a blocking method can significantly improve the scalability of PPRL by considerably reducing the number of candidate record pairs. At the same time, LSH-based blocking can correctly identify the vast majority of true matches, so that only a few matches are missed. Moreover, LSH-based blocking does not require plaintext data but can be applied directly on encoded records (bit vectors). Therefore, LSH-based blocking does not reveal (additional) information about the plaintext attribute values. However, LSH-based blocking has so far only been applied to record-level Bloom filter encodings where a single bit vector represents all identifying data of a person. We therefore investigate approaches for LSH-based blocking using attribute-level encodings. We implement these approaches in an identify management software, called Mainzelliste [LBÜ15], which is widely used for (privacy-preserving) record linkage applications in the medical domain in Germany.

4.1 Motivation

The advantage of attribute-level encodings is that they allow more sophisticated matching techniques compared to record-level encodings (see Section 2.8.2.1). Especially in the medical domain, Bloom filter encodings on attribute level are therefore preferred, as they promise a higher linkage quality and thus more reliable linkage results [Ran+19]. In particular, the Mainzelliste [TMF21], a web-based open-source software for identity management, focuses on attribute-level Bloom filters [LBÜ15] for privacy-preserving record linkage by default. However, the Mainzelliste shows poor runtime and scalability since it misses support for blocking. As a consequence, every new patient record has to be compared with every already known (previously added) record. To improve runtimes, we extend the Mainzelliste to support blocking based on locality-sensitive hashing (LSH) that shows high efficiency and effectiveness (see Chapter 3). However, LSH-based blocking has so far only been applied to record-level Bloom filter approaches, where all attributes are mapped into a single Bloom filter (bit vector).

In this work, we therefore propose methods for LSH-based blocking on attribute-level encodings that result in multiple bit vectors. Although we implement and evaluate these methods within the Mainzelliste, their application is not limited to this particular software, and they can be included in other PPRL tools as well.

Specifically, we make the following contributions:

- We propose two methods for LSH-based blocking on attribute-level Bloom filter encodings.
- We implement our LSH-based methods within the Mainzelliste, an identity management software that is used in several medical research projects.
- We evaluate our blocking extensions to assess the runtime improvements and to identify suitable default parameter settings to achieve high linkage quality.

The rest of this chapter is structured as follows. In Section 4.2, we give a brief description of the Mainzelliste software and its use in medical research projects. We then present our methods for LSH-based blocking on attribute-level Bloom filter encodings in Section 4.3. In Section 4.4, we evaluate our approaches in terms of linkage quality and scalability. Finally, in Section 4.5, we conclude.

4.2 Background

The core functionalities of the Mainzelliste are the pseudonymization and de-pseudonymization of patients, accessible via a RESTful interface that enables self-explanatory usage through widely used web technologies. The pseudonymization process includes a configurable record linkage process, which by default uses an error-tolerant matching algorithm [Con+05] to compute the similarity between pairs of records and find duplicates even in the presence of typos, interchanged attributes, or missing values.

Since its first release in 2013, the Mainzelliste has been used by a constantly growing number of national medical research networks [LSÜ18; Hav+14], centralized biobanks [Ber+16], research platforms [Skr+16], commercial data capture and analysis suites [Cau15], registry software solutions [Mus+14; Sto+17], and disease registries [BW15; ChI15]. The software is under continuous development, incorporating community contributions from various research institutions [LBÜ23].

The Mainzelliste can be used for conventional record linkage on original (plaintext) as well as for PPRL on encoded attribute values. A variety of other open-source record linkage tools exists [Chr12b], but most of them are limited to one-time batch matching. A comparison of the Mainzelliste with other tools for incremental matching on plaintext data was carried out in [LBÜ15]. While PPRL has already been applied for several medical use cases in different organizations [Kue+12; Gib+16; Luo+17], to the best of our knowledge the Mainzelliste is the only publicly available PPRL tool with a RESTful web interface that has been used in numerous real applications. In contrast to many other PPRL tools, it is ready-to-use and easily deployable in medical applications, rather than being a prototype or library adding functionality to other programs. A detailed comparison of available PPRL tools is given in Section 8.3.

4.3 Approaches for LSH Blocking on Attribute-level Bloom Filters

LSH has been used as a blocking method for PPRL in several approaches (see Section 3.2). However, LSH-based blocking has so far only been applied to record-level Bloom filter encodings where a single bit vector represents all attributes of a record. Therefore, in the following, we propose two methods to apply LSH on a set of attribute-level Bloom filters $\{bv_1, \ldots, bv_p\}$ where p denotes the number of Bloom filters (corresponding to attributes/fields) used for blocking. Both methods are illustrated in Figure 4.1.

4.3.1 Attribute-level LSH

As a first approach, we consider an attribute-dependent selection strategy, where a certain number Λ_i, $i \in \{1, \ldots, p\}$, of LSH blocking keys is constructed for each attribute separately. All bits of a single LSH key are drawn from the same attribute-level Bloom filter, and hence each key is affected by exactly one attribute. For the example in Figure 4.1, we have chosen a single key of length four for each of the three considered attributes. The two sample records a and b have the same blocking key for two of the three keys (BK_2 and BK_3).

The main benefit of this approach is that it is error-tolerant even if several attribute values are different or missing. At least one matching attribute is sufficient to assign two records to the same block. On the other hand, as each blocking key solely depends on a single attribute-level Bloom filter, the resulting blocks can become large when there are only a few different attribute values or frequent attribute values, e. g., popular last or first names.

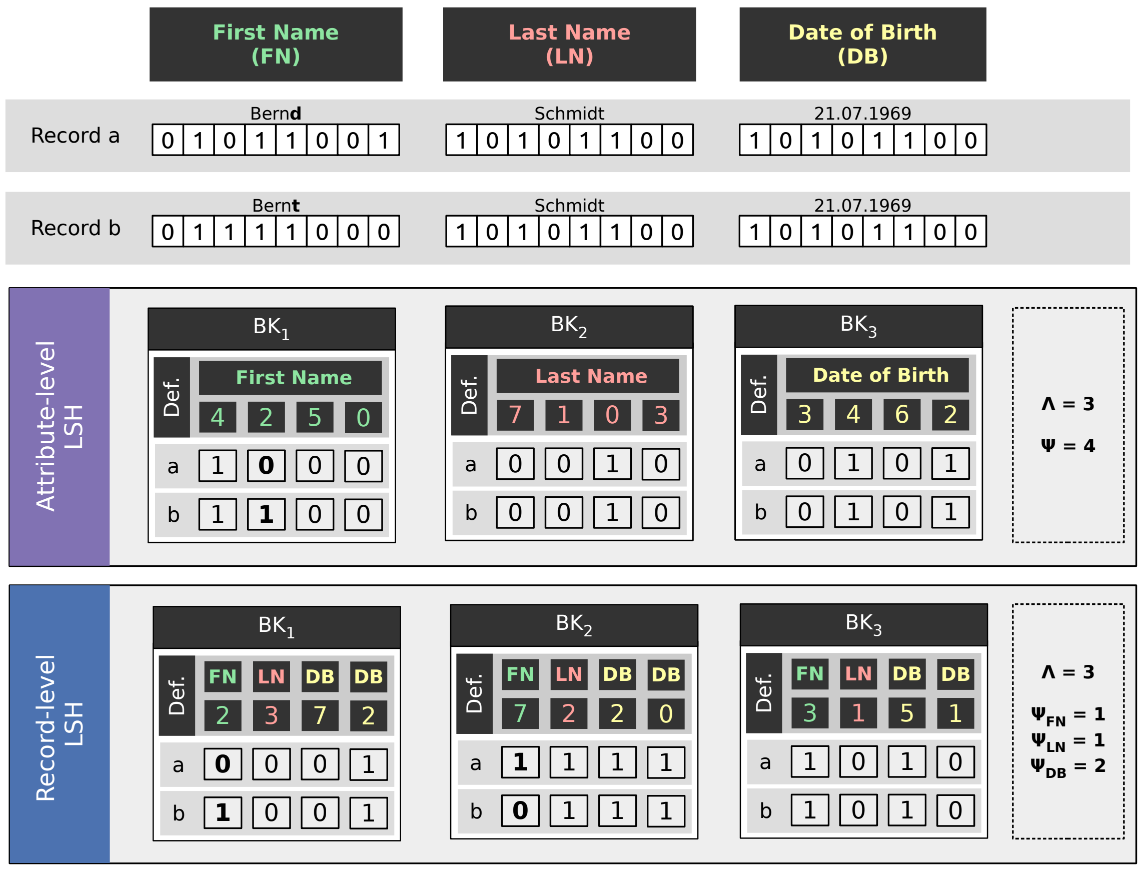


Figure 4.1: LSH-blocking variants on attribute-level Bloom filters.

4.3.2 Record-level LSH

We also consider a multi-attribute selection strategy. For each LSH blocking key bk_i with $i \in \{1, \ldots, \Lambda\}$ we select a certain number $\{\Psi_1, \ldots, \Psi_p\}$ of positions from each attribute-level Bloom filter. As a consequence, the $\Psi = \sum_{i=1}^{p} \Psi_i$ bits of each LSH key will be drawn from the different attribute-level Bloom filters.

For the example in Figure 4.1, we have again $\Lambda = 3$ blocking keys of length $\Psi = 4$, but the bits are selected from all three attributes. Here, 1 bit is selected from the first name attribute ($\Psi_{FN} = 1$), 1 bit from the last name attribute ($\Psi_{LN} = 1$), and 2 bits from the date of birth attribute ($\Psi_{DB} = 1$). As a result, only the key BK_3 has the same value for the two considered records.

In contrast to the attribute-level LSH approach, the record-level strategy can lead to smaller blocks as each LSH blocking key depends on several Bloom filters and thus attributes. Therefore, the record-level LSH strategy is assumed to produce fewer candidates and consequently fewer record pair comparisons. However, the record-level strategy may also be less error-tolerant compared to the attribute-level strategy. In particular, if attributes are erroneous or contain missing values, then the probability that these attributes will affect several or even all LSH blocking keys increases. As a consequence, such cases can lead to missing matches (false negatives). Thus, more LSH keys may be needed to avoid or limit this problem.

4.4 Evaluation

In this section, we evaluate the approaches for LSH-based blocking on attribute-level Bloom filters in terms of their efficiency and effectiveness. Before presenting the evaluation results, we describe our experimental setup and the datasets and metrics we use.

4.4.1 Datasets

For the evaluation, we use one real-world and four synthetically generated, near-real datasets each with the attributes first name, last name, and date of birth. Table 4.1 shows the main features of the five datasets, in particular their sizes and error rates.

Dataset R is based on a real-world dataset with approximately 50 000 person records that were drawn from the civil register of a German city. This dataset is of high quality and contains only 565 duplicate records. An analysis of the duplicates shows that approximately 80% are equal in all of their attributes, but the remaining duplicates contain missing values, diacritics, and multiple names in first and last name attributes. All records of dataset R are sequentially inserted so that each additional record is matched against the records already stored in the Mainzelliste database.

To systematically evaluate the impact of the dataset size and data quality, we synthetically generated four additional datasets with near-real person names derived from look-up files and frequency distributions from German census data. For this purpose, we employ a customized version of the GeCo data generation and corruption tool used in previous research on record linkage [TVC13]. The G datasets are generated in three sizes to evaluate the scalability of the linkage: small, medium, and large with 10 000, 100 000, and 1 000 000 records in total. For these datasets, we assume that a subset A of 70% of the records is already inserted in the Mainzelliste database and that the records of the remaining subset B are added (matched and inserted) one by one. For the large dataset $\mathrm{G_L}$ the runtimes without blocking are already too high, so we only evaluate it for a randomly selected subset of B encompassing 10% of its records. The quality regarding the G datasets is lower than for the real dataset R since we assume a relatively high share of duplicate records (50% of the records in subset B). Furthermore, 30% of the duplicates are assumed to contain one or two erroneous attribute values, as indicated in the last column of Table 4.1.

For quality evaluation, we additionally consider the ‘dirty’ dataset $\mathrm{D_M}$. Dataset $\mathrm{D_M}$ has the same size as $\mathrm{G_M}$ but more errors, e. g., phonetic variation, OCR errors, and typos, that are introduced by GeCo’s corruption component. In $\mathrm{D_M}$ 40% of the duplicate records are erroneous, including 5% with errors in all three attributes, to provide a pessimistic scenario for achieving high match quality.

DS	$\|A\|$	$\|B\|$	$\|A \cap B\|$	% records : errors
R	0	51 380	565	81% : 0
				18% : 1
				01% : 2
$\mathrm{G_S}$	7000	3000	1500	70 % : 0
				27% : 1
				03% : 2
$\mathrm{G_M}$	70 000	30 000	15 000	70% : 0
				27% : 1
				03% : 2
$\mathrm{G_L}$	700 000	300 000	150 000	70% : 0
				27% : 1
				03% : 2
$\mathrm{D_M}$	70 000	30 000	15 000	60% : 0
				25% : 1
				10% : 2
				05% : 3

Table 4.1: Description of the datasets, each with the size of the initial patient list $|A|$, the number of inserted patients $|B|$, the number of duplicate records $|A \cap B|$ and the proportion of records with a certain amount of erroneous fields.

4.4.2 Bloom Filter Encoding

Record linkage on encodings based on Bloom filters requires the pre-processing steps to be done before the actual encoding and therefore by the data owners. Table 4.2 shows the data cleaning methods used for each attribute. For dataset R an additional step was performed to split compound attributes as described above. After pre-processing, all attributes are split into bigrams (q-grams of length 2) that are mapped into the Bloom filters. The three components of the birthday have been encoded in a joint Bloom filter. An essential parameter for encoding is the ratio of the number of hash functions to the length of the Bloom filter. The larger the ratio, the more bits are set on average in the bit vector. The applied encoding parameters (shown in Table 4.2) result in an average share of approximately 25% 1-bits.

Attribute	Preprocessing	Padding	k	m
First name	Trim whitespace Lower case conversion	yes	20	384
Last name	Umlaut substitution Diacritics removal	yes	20	512
Day of birth	–	yes	20	512
Month of birth	–	no	40	
Year of birth	Only last two digits	yes	20	

Table 4.2: Bloom filter encoding used for the evaluation with k as the number of hash functions and m as the length of the Bloom filter.

4.4.3 Evaluation Metrics

Runtime for inserting patients is measured within the Mainzelliste. Therefore, it does not include the network latency (delay) of the HTTP requests. Please note that the time for inserting a patient includes the retrieval of records from the database, the actual matching, as well as the time needed for persistence. Furthermore, we determine the average number of candidates for each record and calculate the reduction ratio (see Section 2.6.7.2).

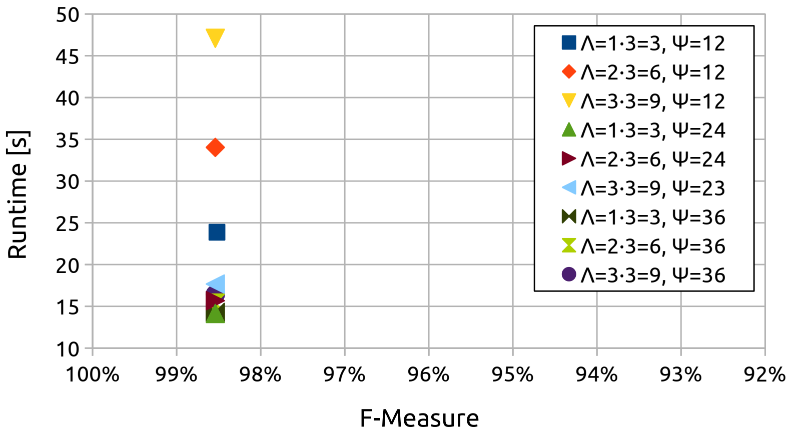
4.4.4 Benchmark Setup

All experiments are conducted on a desktop computer equipped with an Intel i7-6700, 32 GB main memory, and an SSD running Ubuntu 18.04, MySQL 5.7, and Tomcat 8.5. We compare the LSH-based blocking to traditional blocking on plaintext attributes. Therefore, we use the Soundex encoding function [OR18] on the first and last name attributes. As a result, two records are compared if they share the same Soundex value for either the first or the last name.

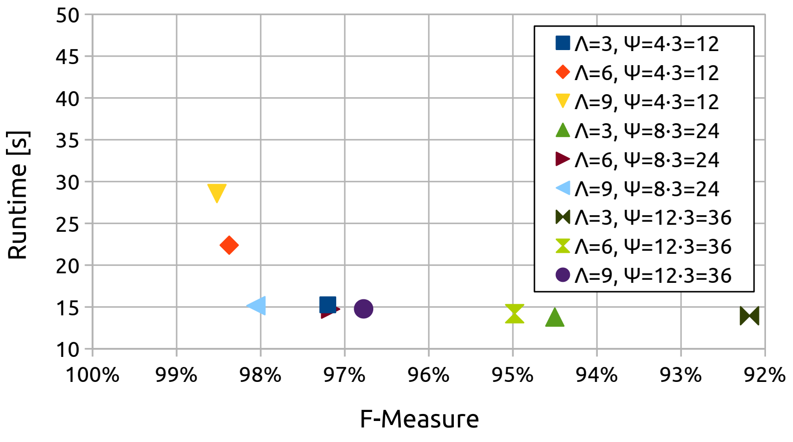
4.4.5 Results and Discussion

LSH blocking requires the configuration of the two parameters Λ and Ψ (number and length of blocking keys). We therefore evaluated different settings on dataset G_M to determine suitable default parameters for each LSH method. Figure 4.2 shows the obtained F1-score and runtime results for different values for Λ and Ψ.

For attribute-level LSH (Figure 4.2a) the F1-scores are very stable as at least one of the three attributes per record is error-free for G_M. We therefore choose $\Lambda = 3$, corresponding to one key per attribute, and $\Psi = 36$ as it results in shorter runtimes. However, for record-level LSH (Figure 4.2b) a higher number of blocking keys $\Lambda = 9$



(a) Attribute-level LSH



(b) Record-level LSH

Figure 4.2: F-Measure against runtime for different numbers of LSH keys (Λ) and LSH key lengths (Ψ) determined for dataset G_M.

and shorter keys with $\Psi = 24$, i. e., 8 hashes for each attribute ($8 \cdot 3 = 24$), yield a good compromise between linkage quality and runtime.

Additionally, we apply the key restriction approach proposed in Section 3.4.3.2 to exclude bit positions that are frequently set to 0 or 1 as they can cause larger block sizes. The bit frequencies are determined at runtime based on the first 1000 inserted records, applying a prune ratio of 0.5.

Table 4.3 shows the results of all evaluations for the five datasets with and without Bloom filters, as well as with and without (Soundex or LSH) blocking. Rows without blocking correspond to the original implementation of the Mainzelliste whereas rows with blocking represent the respective results with our improvements. For each of the five configurations per dataset, the table shows the linkage quality results as well as

DS	BF	Blocking	t	Rec. [%]	Prec. [%]	F1 [%]	Insert [ms]	# Blocks	# Cand.	RR [%]
R	✗	No	0.90	98.05	99.64	98.84	394	0	25 283	0.00
		Soundex	0.90	98.05	99.64	98.84	13	5753	185	99.27
	✓	No	0.95	98.05	99.64	98.84	352	0	25 283	0.00
		F-LSH	0.95	98.05	99.64	98.84	14	45 436	69	99.73
		R-LSH	0.95	98.05	99.64	98.84	15	**409 876**	**5**	**99.98**
$\mathrm{G_S}$	✗	No	0.80	98.00	100.00	98.99	108	0	7400	0.00
		Soundex	0.80	98.00	100.00	98.99	12	1438	91	98.77
	✓	No	0.85	99.00	100.00	99.50	81	0	7387	0.00
		F-LSH	0.85	**99.00**	100.00	**99.50**	10	9810	42	99.43
		R-LSH	0.85	97.73	100.00	98.85	11	**72 011**	**2**	**99.98**
$\mathrm{G_M}$	✗	No	0.80	98.12	99.53	98.82	1018	0	73 966	0.00
		Soundex	0.80	98.09	99.53	98.80	22	1950	862	98.83
	✓	No	0.85	98.55	98.53	98.54	724	0	73 879	0.00
		F-LSH	0.85	**98.55**	98.53	**98.54**	14	27427	406	99.45
		R-LSH	0.85	97.20	**98.93**	98.06	15	**554 785**	**14**	**99.98**
$\mathrm{G_L}$	✗	No	0.80	*98.18*	*96.77*	*97.47*	*9525*	0	*701 971*	0.00
		Soundex	0.80	98.18	96.28	97.22	147	2260	8549	98.78
	✓	No	0.85	*98.63*	*90.92*	*94.62*	*7151*	0	*701 767*	0.00
		F-LSH	0.85	**98.62**	90.47	94.37	55	36 100	3996	99.43
		R-LSH	0.85	96.93	**92.42**	**94.62**	**15**	**2 970 247**	**145**	**99.98**
$\mathrm{D_M}$	✗	No	0.80	90.09	99.51	94.57	1059	0	74 970	0.00
		Soundex	0.80	89.45	99.51	94.21	22	2454	842	98.88
	✓	No	0.85	91.84	98.31	94.96	731	0	74 717	0.00
		F-LSH	0.85	**91.82**	98.31	**94.95**	14	29 993	388	99.48
		R-LSH	0.85	89.41	**98.73**	93.84	15	**565 698**	**14**	**99.98**

Table 4.3: Mainzelliste evaluation results using different blocking approaches.

the average (insert) runtime per record, the number of blocks, the number of match candidates, and the achieved reduction ratios.

A detailed comparison between matching on plaintext and encoded data without blocking can be found in [Roh+21].

The newly introduced blocking methods led to dramatic improvements in the runtime of the Mainzelliste software by several orders of magnitude. Figure 4.3 illustrates the average insert time per record vs. the dataset size. In the original implementation without blocking (left part of Figure 4.3) these execution times rise linearly with the number of records. This leads to an unacceptably long runtime per record for dataset $\mathrm{G_L}$ of up to 9.5 (7) seconds for plaintext (Bloom filter) matching and thus to execution times of more than one month for 300 000 records. Applying blocking (right part of Figure 4.3 with different scaling of the y-axis) leads to drastically improved execution times, e. g., by a factor of almost 500 using record-level LSH on dataset $\mathrm{G_L}$.

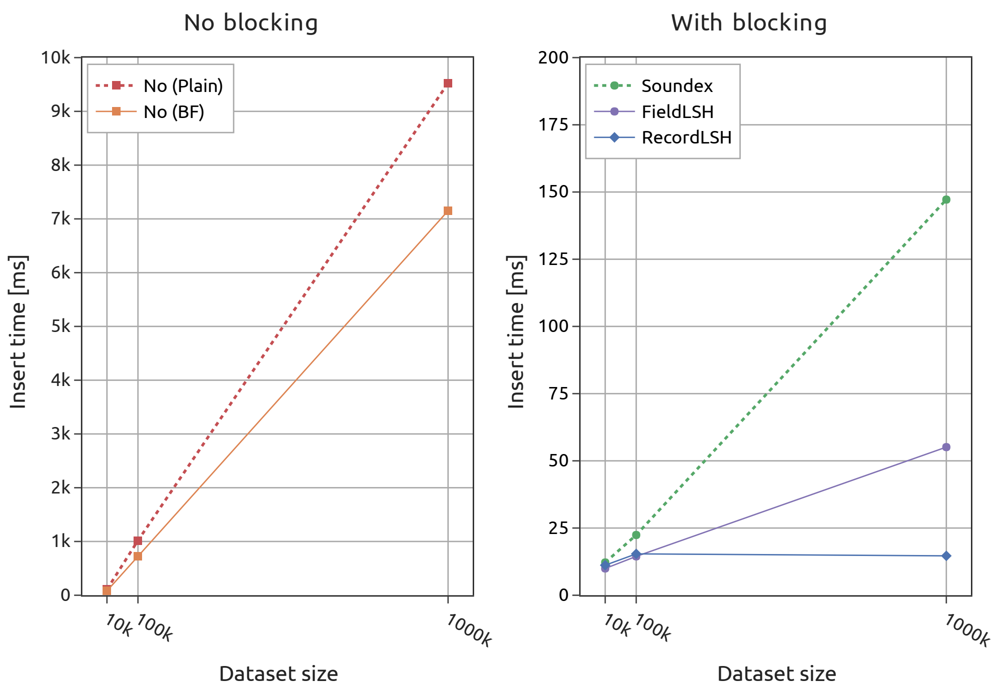


Figure 4.3: Comparison of average insertion times per patient record on datasets G_S, G_M and G_L without (left) and with (right) blocking.

Moreover, runtimes are stable for record-level LSH on datasets of different sizes. Attribute-level LSH and especially Soundex are more dependent on the data volume and experience an increase in runtimes with more records. This is because their number of blocks increases only modestly with more data so that the average size of blocks and thus the number of comparisons per record increases with larger data volumes. Still, for dataset G_L, the execution time for blocking with attribute-level LSH (Soundex) is a factor of 130 (65) faster than without blocking. The reduction ratios achieve even better values of up to 99.98%, i. e., a factor 5000 in the number of comparisons.

These high runtime improvements are achieved without reduction in linkage quality, as can be seen from the F1-score values in Table 4.3. There are some relatively small differences between the two LSH variants. Attribute-level LSH leads to larger blocks than record-level LSH, thereby enabling a slightly better recall. On the other hand, the smaller blocks of record-level LSH favor better precision, especially for the large dataset G_L. Record-level LSH is much faster than attribute-level LSH for the large dataset G_L, but the runtimes are only slightly worse for the smaller datasets. This is because the reported insert times are only partially determined by the match time but also include the time to store new records and their blocking keys in the database. The latter persistence step needs slightly more time for record-level LSH than for attribute-level LSH because of the higher number of LSH blocking keys (9 vs. 3).

Given the comparable linkage quality and runtimes for both attribute-level LSH and record-level LSH in most cases, we recommend attribute-level LSH as the default blocking strategy except for very large datasets. This is because it is much easier to configure than record-level LSH, and a simple approach with a single blocking key per attribute proved to perform very well.

4.5 Conclusion

We proposed two methods for LSH-based blocking on attribute-level Bloom filters. We implemented both methods within the Mainzelliste software and evaluated their performance in terms of linkage quality and scalability. Our results using real-world and near-real datasets showed drastically improved runtimes without compromising the quality of the linkage.

5

Post-processing Methods for High Quality PPRL

This chapter is based on [Fra+18]. The use of encoded data makes it challenging to achieve high linkage quality in particular for dirty data containing errors or inconsistencies. Moreover, person-related data is often dense, e. g., due to frequent names or addresses, leading to high similarities for non-matches. Both effects are hard to deal with in common PPRL approaches that rely on a simple threshold-based classification to decide whether a record pair is considered to match. In particular, dirty or dense data likely leads to many multi-links where persons are wrongly linked to more than one other person. Therefore, we propose the use of post-processing methods for resolving multi-links and outline three possible approaches. In our evaluation using large synthetic and real datasets, we compare these approaches with each other and show that applying post-processing is highly beneficial and can significantly increase linkage quality in terms of both precision and F-measure.

5.1 Motivation

A high linkage quality is essential for the practical applicability of PPRL, especially in the medical domain. Ideally, a PPRL approach should find all matches, despite possible data quality problems, such as erroneous or inconsistent data, in the source databases. On the other hand, false matches should be strictly avoided, as otherwise (medical) conclusions based on incorrect assumptions could be made.

To decide whether a pair of records represents a match or a non-match, classification models are used. For traditional record linkage, sophisticated classification models based on supervised machine learning approaches, e. g., support vector machines or decision trees, can be used to achieve highly accurate linkage results [Chr12b]. Moreover, linkage results can be manually reviewed to increase final quality or to adjust parameter configurations.

In contrast, currently most PPRL approaches only apply threshold-based classification based on a single threshold, as supervised machine learning approaches require training data which is usually not available in a privacy-preserving context [VCV13]. In general, it is also not feasible to manually inspect actual quasi-identifier values of records because this would be a violation of privacy. Moreover, recent encoding techniques often aggregate all attribute values into a single binary encoding, making it hard to deploy attribute-wise or rule-based classification [VCV13; Vat+17]. All these effects likely reduce the linkage quality of PPRL, indicating the demand for refined classification techniques [Dur12; Ran+19].

In the following, we study post-processing methods for improving the linkage quality of PPRL in terms of precision. By using simple threshold-based classification approaches, only low linkage accuracy is likely achieved in PPRL scenarios dealing with dirty or dense data. Dirty data such as missing or erroneous attribute values can lead to a low similarity between matching records that are thus easily missed with a higher similarity threshold. Another problem case is dense data, where many non-matching records can have a high similarity. For example, members of a family often share the same last name and address, leading to a high similarity for different persons. Datasets focusing on a specific city or region also tend to have many persons with similar addresses. For such dense data, there can be many non-matching record pairs with a similarity above a fixed threshold.

A key drawback of classification approaches based on a single threshold is that they often produce multi-links, i. e., one record is linked (matched) to many records of another source, and moreover, each record pair exceeds the similarity threshold. However, assuming deduplicated source databases, each record can at most match one record of another source. Hence, the linkage result should exclusively contain one-to-one links as otherwise precision deteriorates. Therefore, we analyze methods that can be executed after any (threshold-based) classification to clean multi-links, i. e., to transform the linkage result such that only one-to-one links occur in the final result. In particular, we make the following contributions:

- We study three post-processing strategies for the cleaning of multi-links, or selection of match candidates respectively, to increase the overall linkage quality of PPRL, especially when dealing with dense or dirty data.
- We evaluate the different post-processing approaches using large synthetic and real datasets showing different data characteristics and difficulty levels.

- In our evaluation, we consider both linkage quality in terms of recall, precision, and F-measure, as well as efficiency in terms of runtime.

In the following, we outline the problems of simple threshold-based classification techniques (Section 5.2) and discuss related work (Section 5.3). Then, we formalize the problem of cleaning multi-links that we want to address with post-processing (Section 5.4) and describe approaches for solving it (Section 5.5). In Section 5.6, we evaluate selected approaches in terms of quality and performance. Finally, we conclude our work in Section 5.7.

5.2 Background

As discussed in Section 2.6, a PPRL pipeline contains multiple steps. Following previous work, we assume a three-party protocol, where a (trusted) third party, called linkage unit, is required [VCV13]. The linkage unit conducts the actual linkage of encoded records from two or more database owners. While we focus on only two database owners (parties) A and B with their respective databases D_A and D_B, the PPRL process, and the post-processing strategies can be extended to multiple database owners.

In this work, we focus on record linkage with the additional challenge to preserve the privacy of referenced individuals. Consequently, each record needs to be encoded to protect sensitive data. Most recent PPRL approaches use record-level encodings because they generally provide better privacy protection than field-level encodings. The drawback of such record-level encodings, however, is that their comparison typically yields only a single similarity score (see Section 2.6.4). As a consequence, most PPRL approaches use a single similarity threshold to classify candidate record pairs into matches and non-matches.

However, by using simple threshold-based classification approaches, multi-links occur in the final match result. Since commonly deduplicated databases are assumed, the desired outcome should be a linkage result consisting of only one-to-one links between records. We address this problem by introducing a post-processing step after classification to clean multi-links in the linkage result. The main problem in the post-processing step is to decide which candidates should be selected, leading to only one-to-one links and high linkage quality.

5.3 Related Work

In order to achieve a high linkage quality, previous work mostly focuses on developing or optimizing encoding techniques to support approximate matching, attribute weighting, or different data types [KGV17; VCV13; VC16; Vat+14]. Besides, efficient blocking and filtering techniques have been proposed that do not compromise the linkage quality outcome [Vat+17].

The problem of post-processing corresponds to weighted bipartite graph matching problems [Wes01]. In fact, applying a one-to-one matching restriction, i. e., to clean multi-links, is highly related to problems in graph theory like the assignment problem (AP) or the stable marriage problem (SMP). Various algorithms have been developed to solve such kinds of problems [IM08; Wes01]. The most prominent approaches are variants of the Hungarian algorithm (Kuhn-Munkres algorithm) [Mun57] for solving assignment problems as well as variants of the Gale-Shapley algorithm [GS62; GI89; Irv94; MW70] for solving the stable marriage problem.

For PPRL, post-processing methods have only been studied to a limited extent so far: Though several approaches were considered for traditional record linkage, they were not comparatively evaluated in a PPRL context [Böh+12; Chr12b; Dur12; EIV07; Len06]. As a consequence, it is unknown to which degree post-processing is useful and which method is suited best for PPRL. In general, the matching on encoded data allows only simple approaches for match classification so that the need for post-processing is increased for PPRL.

Recently, Papadakis et al. [Pap+22] conducted an extensive empirical evaluation of eight different bipartite graph matching algorithms for non-private clean-clean entity resolution.

A similar post-processing problem, namely selecting the most probable correspondences from a mapping, has been studied in the field of schema matching [DR02; MG07; MGR02] and ontology matching [MS07]. In [MGR02] and [DR02] best match selection strategies, called MaxN or Perfectionist Egalitarian Polygamy, are used to enforce a one-to-one cardinality constraint by selecting only candidates offering the best similarity scores. Additionally, algorithms for solving the maximum weighted bipartite graph matching problem and the stable marriage problem have been considered as selection strategies [MG07; MS07].

5.4 Problem Definition

After the classification step (see Section 2.6.5) all candidate record pairs resulting from blocking/filtering C_B are classified into matches $\mathtt{M}$ and non-matches $\mathtt{N}$.

We assume, that a simple threshold-based approach is used for classification. Thus, the class of matches $\mathtt{M}$ contain all candidate record pairs with a similarity score $sim_\Delta(\cdot,\cdot)$ above a single predefined similarity threshold $\mathtt{t}$, i. e., $\mathtt{M} = \{(a,b) \mid a \in D_A, b \in$

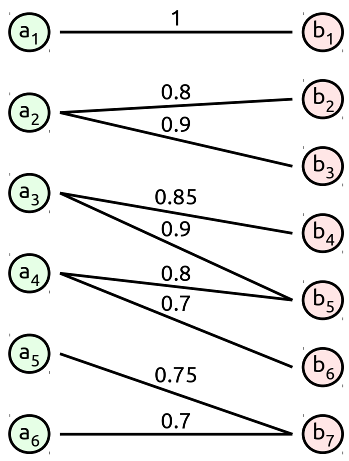


Figure 5.1: Example linkage graph containing several multi-links.

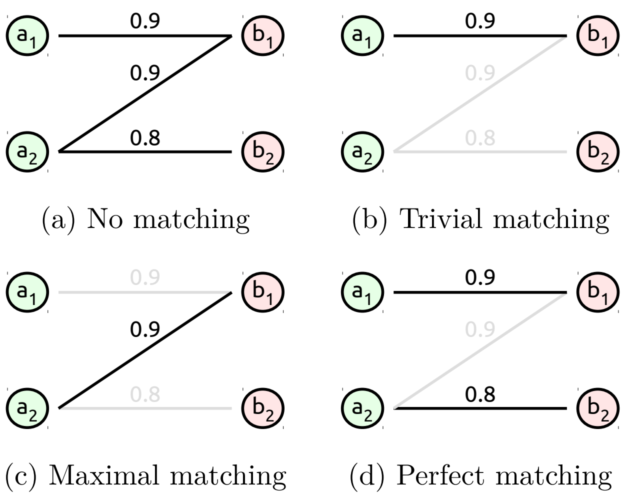


Figure 5.2: Different types of matchings.

$D_B, sim_{\Delta}(a,b) \geq t\}$. We also assume that the databases to be linked are deduplicated before linkage.

The set of matches M constitutes a weighted bipartite similarity graph $\mathbf{SG} = (V_A \cup V_B, E)$ (see Section 2.6.6). Let $\mathbf{V_A}$ and $\mathbf{V_B}$ be two partitions consisting of vertices representing records (entities) from database D_A or database D_B respectively, which occur in the linkage result, i. e., are part of a record pair classified as a match. Thus, $V_A = \{a \in D_A \mid \exists b \in D_B : (a,b) \in \mathtt{M}\}$ and $V_B = \{b \in D_B \mid \exists a \in D_A : (a,b) \in \mathtt{M}\}$. **E** denotes the set of edges representing links between records classified as matches. Each edge (link) has a property for the similarity score of the record pair. Between records of the same database, no direct link exists. An example linkage graph is depicted in Figure 5.1.

After classification, it is still possible that the linkage graph contains multi-links, i. e., one-to-many, many-to-one, or many-to-many links. Since deduplicated databases are assumed, only one-to-one links should be present in the final linkage result. Hence, the aim of post-processing is to find a *matching* over SG (see Section 2.6.6). A matching $\mathbf{M} \subseteq E$ is a subset of links such that each record in $V = V_A \cup V_B$ appears in at most one link, i. e., contributes to at most one matching record pair. As a consequence, post-processing applies a one-to-one link (cardinality) restriction on the set of classified matches M.

In general, several matchings over SG can be found. Thus, the challenge of post-processing is to select the matching yielding the best linkage quality in terms of either recall, precision, or F-measure. Ideally, no true match should be pruned (no loss of recall) while resolving all multi-links to improve precision. Links providing high similarity scores should be favored over those with low similarity, e. g., near t, as very high similarities typically indicate definite matches. Also, other link features, like link degree, can be used for link prioritization [SPR18].

A matching can be selected in such a way that it fulfills certain properties. Basic types of matchings are trivial, maximal, maximum and perfect matchings [Wes01]. A matching M is called maximal, if any link not in M is added to M, then M would be no longer a matching. If a matching is not maximal, then it is a trivial matching. Furthermore, if a matching contains the largest possible number of edges (links), then it is a maximum matching. Each maximum matching is also maximal, but not vice versa. Finally, a perfect matching is defined as a matching where every vertex of the graph is incident to exactly one edge of the matching. Every perfect matching is maximum and hence maximal. However, not for every linkage graph a perfect matching can be obtained. The different types of matchings are illustrated in Figure 5.2.

Since PPRL is confronted with potentially large datasets containing millions of records [Vat+17], post-processing approaches need to be scalable and efficient.

5.5 Post-processing Strategies for PPRL

We now present post-processing strategies for PPRL to enable a one-to-one link restriction on the linkage result. We chose three frequently used approaches known from schema matching for obtaining matchings in bipartite graphs. The approaches are described in detail below.

5.5.1 Symmetric Best Match

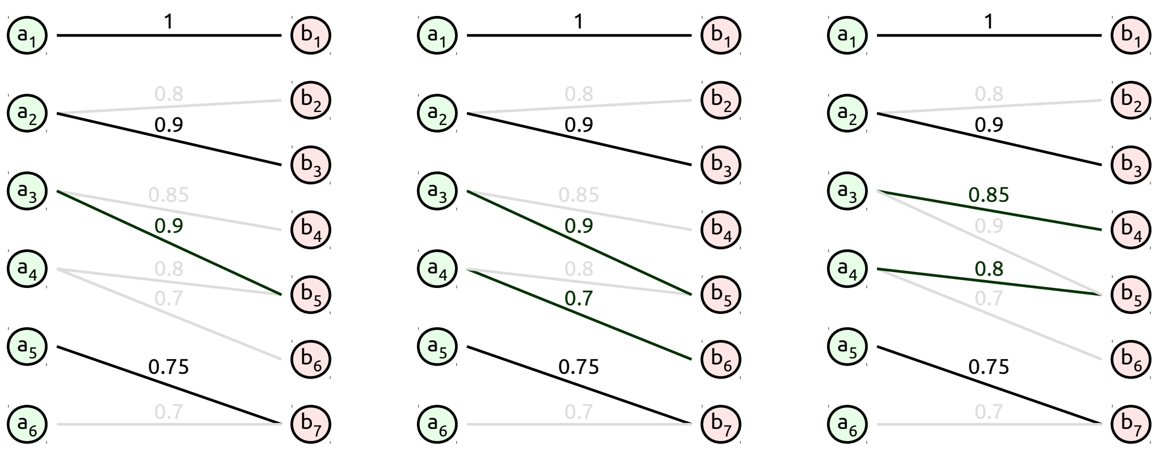
At first, we consider a symmetric best match strategy (SBM) as proposed in [MGR02] and [DR02]. The basic idea is that for every record only the best matching record of the other source is accepted. A record $a \in V_A$ may have links to several records $b \in V_B$. From these links, only the one with the highest similarity score called best link, is selected. This approach is equivalent to a MaxN strategy, which extracts the maximum N correspondences for each record setting $N = 1$ (Max1).

To obtain a matching M over a linkage graph SG for every record of both partitions V_A and V_B the best link is extracted. Thus, two sets E_A^{Max1} and E_B^{Max1} are built containing the best links for each record of the respective partition, e. g.,

$$E_A^{Max1} = \{(a, b) \in E \mid \forall b' \in V_B : (b \neq b' \wedge (a, b') \in E) \rightarrow (sim_\Delta(a, b') \leq sim_\Delta(a, b))\}.$$

Then, the final matching is obtained by building the intersection of these two sets, i. e., $M_{SMB} = M_{Max1\text{-}both} = E_A^{Max1} \cap E_B^{Max1}$. Since the best links from both partitions are considered, this strategy is also called Max1-both.

In Figure 5.3a Max1-both is applied on the linkage graph from Figure 5.1. It is important to note that the obtained matching is not maximal. Since only record pairs with a common best link are accepted, other record pairs are excluded from the matching even if they do not violate the one-to-one link restriction and have a relatively high similarity.



(a) Max1-both (SBM) (b) SM (c) MWM

Figure 5.3: Illustration of the resulting linkage graph from Figure 5.1 after applying different post-processing methods. For Max1-both (a) the link a_4-b_6 is removed since the best link for a_4 is to b_5. In contrast, in the stable matching (b) the link a_4-b_6 is included as it does not violate the one-to-one-link restriction nor the stable property. For the maximum weight matching (c), the links a_3-b_4 and a_4-b_5 are included in the matching as the sum of their similarities is higher than for a_3-b_5 and a_4-b_6. However, the maximum weight matching is not stable due to the links a_3-b_4 and a_4-b_5, as a_3 and b_5 prefer each other over their current matching records.

5.5.2 Stable Marriage and Stable Matchings

The stable marriage problem (SMP) [GS62] is the problem of finding a stable matching (SM) between two sets of elements given a (strictly) ordered preference list for each element. A matching is defined as stable if there are no two records of the different partitions that both have a higher similarity to each other than to their current matching record. Used as a post-processing method for PPRL, several extensions to the classic stable marriage problem need to be considered [IM08; MW70]:

Unequal Sets: Usually, an SMP instance consists of two sets of elements having the same cardinality. The partitions of the linkage graph are in general of different sizes, i. e., $|V_A| \neq |V_B|$, as not every record may have a duplicate in the other source.

Incomplete preference lists with ties: In the traditional stable marriage problem, each element has a preference list that *strictly* orders *all* members of the other set. Since blocking or filtering techniques are used for PPRL to reduce the number of record pair comparisons, not every record $a \in V_A$ has a link to a record $b \in V_B$ and vice versa. However, one simple approach is to add dummy links with a similarity score of zero or $-\infty$. Moreover, a record may have two links with the same similarity score to two different records of the other source, called tie or indifference [Irv94], e. g., $sim_\Delta(a, b_1) = 0.9$ and $sim_\Delta(a, b_2) = 0.9$ where $a \in V_A$ and $b_1, b_2 \in V_B$. The simplest way to handle indifference is to break ties arbitrarily [Irv94]. Also, secondary link features can be used for resolving ties [SPR18].

Symmetry: For the stable marriage problem, it is not required that two elements prefer each other the same (asymmetric preference). In our case, the stable marriage problem is symmetric since the similarity of a record pair is symmetric (see Definition 2.6.4.1).

To obtain a stable matching the Gale-Shapley algorithm [GS62] or one of its variants taking the described extensions into account [Irv94; IM08; MW70] can be used. A simple approach is to order all links (or candidate pairs) based on their similarity score and process them iteratively in descending order. The current link is added to the final matching if it does not violate the one-to-one link restriction. The algorithm stops if all links have been processed [MG07]. In Figure 5.3b a stable matching for the linkage graph from Figure 5.1 is depicted. In contrast to matchings obtained by the symmetric best match strategy, stable matchings are maximal. In general, multiple stable matchings may exist for a linkage graph.

5.5.3 Maximum Weight Matchings

As a third method, we consider finding a maximum weight matching (MWM). A maximum weight matching is a matching that has maximum weight, i. e., that maximizes the sum of the overall similarities between records in the final linkage result. This problem corresponds to the assignment problem which consists of finding a maximum weight matching in a weighted bipartite graph. To solve the assignment problem on bipartite graphs in polynomial time, the Hungarian algorithm (Kuhn-Munkres algorithm) can be used [Mun57]. The Hungarian algorithm has a complexity of $\mathcal{O}(n^4)$ but can be reduced to $\mathcal{O}(n^3)$ where n is the number of records. For the linkage graph from Figure 5.1 the corresponding maximum weight matching is depicted in Figure 5.3c. Each maximum weight matching is maximal but does not have to be stable.

5.6 Evaluation

In this section, we evaluate the introduced post-processing methods for the cleaning of multi-links in terms of linkage quality and efficiency. Before presenting the evaluation results, we describe our experimental setup as well as the datasets and metrics we use.

5.6.1 Experimental Setup

All experiments are conducted on a desktop machine equipped with an Intel Core i7-6700 CPU with $8 \times 3.40\,\text{GHz}$, 32 GB main memory and running Ubuntu 16.04.4. and Java 1.8.0_171.

5.6.2 PPRL Setup

Following previous work, we implement the PPRL process as a three-party protocol utilizing Bloom filter as a privacy technique as proposed by Schnell [SBR11]. To overcome the quadratic complexity, we utilize LSH-based blocking utilizing the family of hash functions which is sensitive to the Hamming distance (HLSH) [Dur12]. The respective hash functions are used to build overlapping blocks in which similar records are grouped. For HLSH-based blocking mainly the two parameters Ψ, determining the number of hash functions used for building a blocking key, and Λ, defining the number of blocking keys, are important for high efficiency and linkage quality outcome [FSR18]. Based on [FSR18] we empirically set Ψ and Λ individual for each dataset as outlined in Table 5.1 leading to high efficiency and effectiveness. Finally, we apply the Jaccard similarity to determine the similarity of candidate record pairs [Jac12].

5.6.3 Datasets

For evaluation, we use synthetic and real datasets containing one million records with person-related data. An overview of all relevant dataset characteristics and parameters is given in Table 5.1.

The synthetic datasets $\mathbf{G_1}$ and $\mathbf{G_2}$ are generated using the data generator and corruption tool GeCo [CV13]. We customize the tool by using lookup files containing German names and addresses with realistic frequency values drawn from German census data [Sta23a]. Moreover, we extend GeCo by a *family* and *move rate* used for G_2. The family rate determines how many records of a dataset belong to a family. All records of the same family agree on their last name and address attributes. The size of each family is chosen randomly between two and five. To simulate moves, we add a move rate that

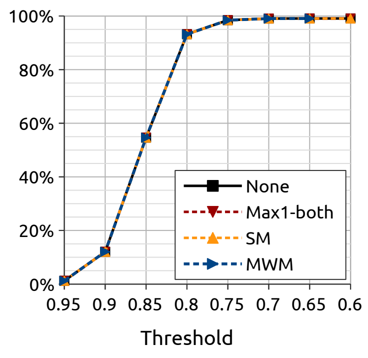
Characteristic	$\mathbf{G_1}$	$\mathbf{G_2}$	**N**
Type	Synthetic (GeCo)		Real (NCVR)
$\lvert D_A \rvert$	800 000	700 000	500 000
$\lvert D_B \rvert$	200 000	300 000	500 000
$\lvert D_A \rvert + \lvert D_B \rvert$	1 000 000		
$\lvert D_A \cap D_B \rvert$	200 000 (100%)	150 000 (50%)	250 000 (50%)
Attributes	First name, last name, city, zip, date of birth		First name, middle name, last name, city, year of birth
q-grams	$q = 2$ (bigrams), no character padding		
g	28		25
$\lvert Errors \rvert / record$	2	0 - 2 : 0 (40%) 1 (30%) 2 (10%)	
$\lvert Errors \rvert / attr$	0 - 1	0 - 2	
$Moves$		20%	
$Families$		25%	
m	1024		
k	26		29
BF type	CLK with random hashing [SBR11; SB16a]		
HLSH key length Ψ	16		
HLSH keys Λ	20		30

Table 5.1: Dataset characteristics and used parameters for the evaluation of different post-processing methods.

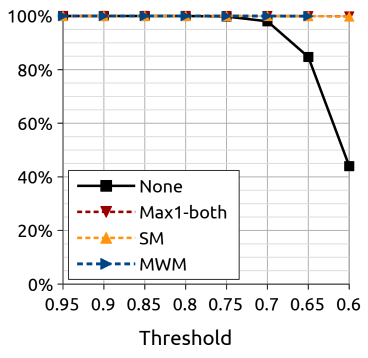
defines in how many records the address attributes will be altered. The move rate does not introduce data errors like typos, instead, it simulates inconsistencies between data sources.

A generated dataset D consists of two subsets D_A, D_B to be linked with each other. While the original tool requires all records of D_B to be duplicates of records of D_A, we modify the tool to support arbitrary degrees of overlaps between D_A and D_B. As a consequence, records in both D_A and D_B may have no duplicate record, which is more realistic.

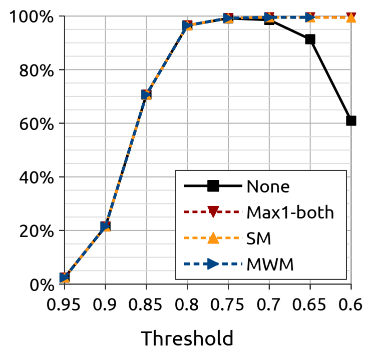
We also use a refined model to corrupt records by allowing a different number of errors per record instead of a fixed maximum number of errors for all records. We may thus generate duplicates such that 50% of the duplicates contain no error, 20% one error, and 10% two errors while the remaining 20% have an address change (move rate). For the real dataset **N**, we use subsets of two snapshots of the North Carolina voter registration database (NCVR) [Nor23] at different points in time.



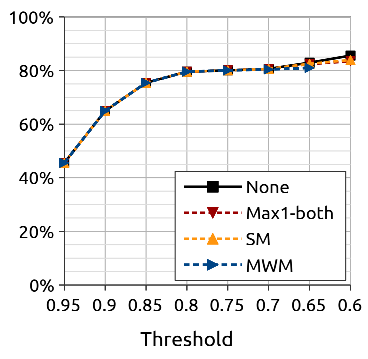
(a) Recall G_1



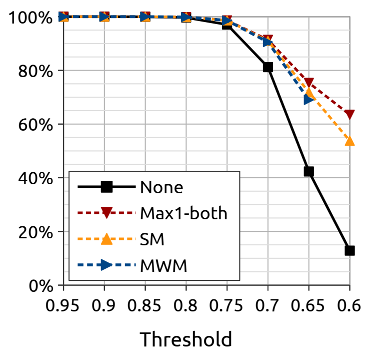
(b) Precision G_1



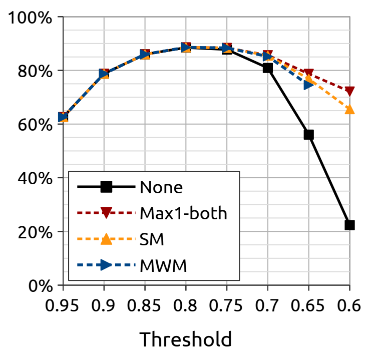
(c) F-measure G_1



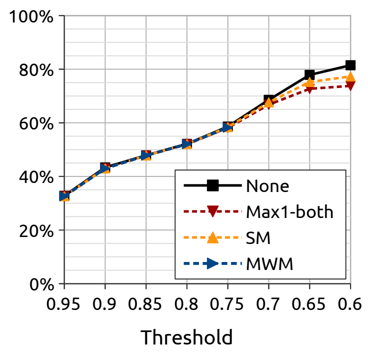
(d) Recall G_2



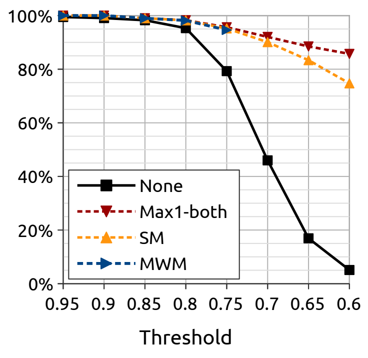
(e) Precision G_2



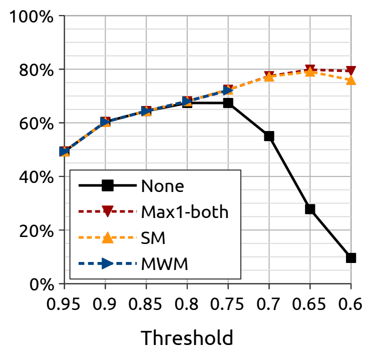
(f) F-measure G_2



(g) Recall N



(h) Precision N



(i) F-measure N

Figure 5.4: Quality results for the datasets G_1, G_2 and N using different post-processing methods.

5.6.4 Evaluation Metrics

To assess the linkage quality, we measure recall, precision, and F-measure (Section 2.6.7.1). To evaluate efficiency, we measure the *execution times* of the post-processing methods in seconds.

5.6.5 Evaluation Results

In order to analyze the impact of post-processing on the linkage quality, we compare the three strategies described in Section 5.5 to the standard PPRL without post-processing. The aim of post-processing is to optimize precision while recall is ideally preserved. The results in Figure 5.4 show the obtained linkage quality for datasets G_1, G_2, and N.

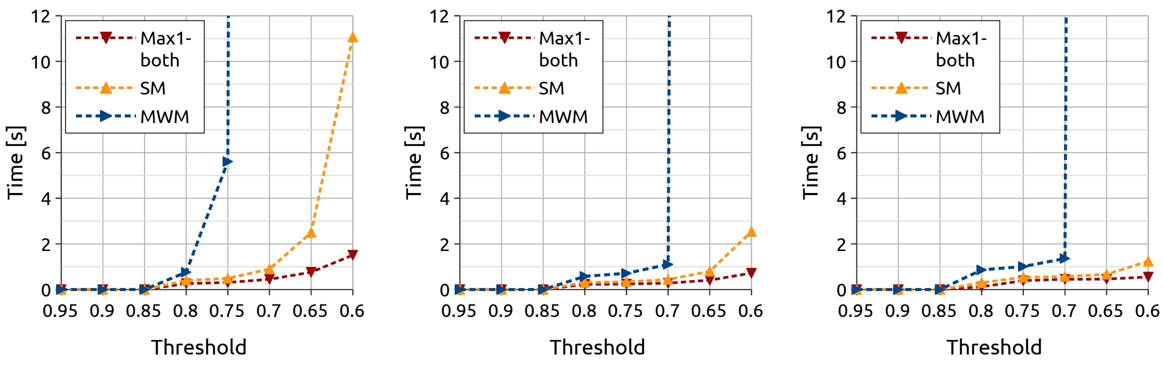
Dataset G_1 is based on settings of the original GeCo tool with 100% overlap and a fixed error rate. We observe that a high linkage quality is achieved even if post-processing is disabled, with near-perfect recall for $t \leq 0.8$ and near-perfect precision for $t \geq 0.7$. The high degree of precision is made possible by the assumption of 100% overlap between the data sources, which minimizes the likelihood of wrongly matching a record.

The three post-processing methods achieve very similar results for G_1. While recall remains stable, precision and consequently F-measure can be significantly improved to almost 100% even for low thresholds $t \leq 0.7$. This is due to the high overlap of the two subsets, making false matches after post-processing only possible if a record has a higher similarity to a record having no duplicate than to its actual true match. Despite this best-case situation simulated with G_1, only low precision is achieved for low threshold values without post-processing.

For datasets G_2 and N overall a lower linkage quality is obtained since the data is more dense making it harder to separate matches and non-matches. Similar to G_1, precision significantly decreases for G_2 using lower threshold values. All post-processing strategies can again improve precision for lower threshold values. The best results are achieved for the symmetric best match approach (Max1-both) outperforming obtaining a stable matching (SM) or a maximum weight matching (MWM). The stable matching yields slightly better results than the maximum weight matching. For the synthetic datasets G_1 and G_2, post-processing does not increase the top F-measure score, but the best linkage quality is reached with a wider range of threshold settings, thereby simplifying the choice of a suitable threshold.

The post-processing methods are most effective for the real dataset N. Here a higher recall can only be achieved for lower threshold values $t \leq 0.7$ but precision drops dramatically in this range without post-processing due to a high number of multi-links. As a result, the best possible F-measure is limited to only 67%. In contrast, the use of post-processing can maintain a high precision even for lower thresholds at only a small decrease in recall compared to disabled post-processing. As a result, the top F-measure is substantially increased to around 80% underlining the high effectiveness and significance of the proposed post-processing. Again, the use of Max1-both is most effective, followed by the stable matching approach.

Additionally, we comparatively evaluate the post-processing strategies in terms of runtime. The results depicted in Figure 5.5 show that Max1-both achieves the lowest execution times even for low thresholds. The extended Gale-Shapley algorithm that we use for calculating the stable matching shows a significant performance decrease for lower similarity thresholds, most notably for dataset N and $t \leq 0.7$. For higher thresholds $t > 0.7$, the runtimes are very similar to those of Max1-both. The computation of the maximum weight matching by using the Hungarian algorithm incurs a high computational complexity and massive memory consumption. As a consequence, we are not able to obtain a maximum weight matching for low threshold values (compare Figure 5.4). Hence, we consider the MWM approach as not scalable enough for large datasets with millions of records.



(a) G_1 (b) G_2 (c) N

Figure 5.5: Runtime results for the datasets G_1, G_2 and N using different post-processing methods.

In conclusion, both the symmetric best match (Max1-both) approach and the stable matching approach are able to significantly improve the linkage quality of PPRL, especially at low thresholds, while showing good performance. In our setup, the execution of the entire PPRL process takes only a few minutes. Therefore, introducing post-processing that takes only a few seconds for execution does not affect the overall performance. In general, Max1-both can achieve the best linkage quality in terms of precision and F-measure. For applications favoring recall over precision, the stable matching approach should be applied.

5.7 Conclusion

We evaluated different post-processing methods for PPRL to restrict the linkage result to only one-to-one links. Our evaluation for large synthetic and real datasets containing one million records showed that without post-processing only low linkage quality is achieved, especially when dealing with dense or dirty data. In contrast, using a symmetric best match strategy for post-processing is a lightweight approach to improve the overall linkage quality. As a side effect, by using post-processing, the similarity threshold used for classification can be selected lower without compromising linkage quality. Since in practical applications, an appropriate threshold is hard to define, this fact becomes highly beneficial. In the future, we plan to investigate further post-processing strategies using further link features and other heuristics. We also plan to analyze post-processing methods for multi-party PPRL where more than two databases need to be linked.

6 (Privately) Estimating Linkage Quality for Record Linkage

This chapter is based on [Fra+24]. To evaluate the quality of (privacy-preserving) record linkage approaches, the performance measures of precision, recall, and F-measure are commonly used. These measures require ground truth data that specifies known matches and non-matches. However, in practical linkage applications, there typically is no such ground truth data available. Although linkage quality can be assessed manually by domain experts, such a clerical review process is time- and resource-consuming and generally not feasible when linking databases that are very large or that contain sensitive (personal) data. We review existing and propose improved unsupervised approaches for estimating the quality of linkage results. We evaluate our approaches on multiple datasets from three different domains. This evaluation shows that our approaches outperform existing methods and lead to estimates that are close to the actual linkage quality.

6.1 Motivation

Record linkage is a challenging task due to data quality, scalability, as well as privacy and confidentiality issues [CRS20]. Most importantly, record linkage algorithms must achieve high linkage quality, as this is essential for their practical applicability and utility. In many record linkage applications, however, there is no ground truth (gold standard) data available that specifies if two records refer to the same entity or not (true match status) [Chr12b]. One possibility to acquire ground truth data is to manually generate such data by (smartly) sampling record pairs and manually classifying them as a match or a non-match [Chr12b]. Similarly, domain experts can manually assess linkage results by (visually) inspecting classified record pairs in order to confirm or reject match decisions [Kum+14]. However, such a manual classification (also known as clerical review) is time- and resource-consuming as well as error-prone, especially for datasets that are large and/or difficult to classify. It can therefore lead to many potential matches, i. e., candidates for which it is unclear if they refer to the same entity or not.

Evaluating linkage quality becomes even more challenging when personal or sensitive data needs to be linked [CRS20]. This problem is addressed by privacy-preserving record linkage (PPRL) techniques [Gko+21], where linkage is conducted on encoded data using secure protocols such that no sensitive information is revealed during the linkage process to protect the privacy of individuals [Vid+23].

In privacy-constrained scenarios, it is generally not possible to inspect actual (quasi-identifying) attribute values of classified record pairs because these can be sensitive and reveal the identity of an individual. Furthermore, the organizations conducting the linkage are generally not allowed or willing to share ground truth or training data. There is limited work [Cha+21; Kum+14] that investigates approaches for manual clerical reviews working on partially (visually) masked quasi-identifying attribute values. Such approaches, however, will again be time- and resource-consuming while making the clerical review process likely to be less accurate than if complete attribute values were available for manual assessment.

Unsupervised approaches for estimating linkage quality are therefore required to overcome this lack of ground truth data. So far, however, only a few such approaches have been proposed [HKN14; NL13]. As we show in our work, in many scenarios the estimated measures do not correlate well with the actual linkage quality. We therefore propose several extensions to existing approaches, as well as novel heuristics, to improve estimates for linkage quality. In particular, we make the following contributions:

- We adapt existing and propose novel unsupervised methods for estimating linkage quality based on a given similarity graph. Our methods address various data quality issues such as heterogeneity of records and duplicates in the same database. Our methods can be used in practice for both traditional and privacy-preserving record linkage applications, in particular, to optimize linkage configurations, such as the classification threshold, which is often a challenging task.
- To estimate the overlap between datasets, our methods require a set of attributes where the values for true matching records are mostly the same, while the values for non-matches mostly differ. To achieve this aim, we develop an apriori-like strategy to automatically determine suitable attribute combinations, in particular for heterogeneous datasets where a manual selection of attributes is hard.

- We comprehensively evaluate our methods for estimating linkage quality against two baseline methods proposed in the literature [NdM12; NL13] using real-world datasets from three different domains (persons, music, and cameras).

The remainder of this chapter is structured as follows. We define the problem of estimating linkage quality in Section 6.2 and discuss related work in Section 6.3. In Section 6.4, we present our novel approaches for estimating the linkage quality for both clean and dirty databases. In Section 6.5, we discuss the privacy aspects of using similarity graphs and cryptosets. In Section 6.6, we evaluate our approaches on different datasets to validate their practical applicability. Finally, we conclude the chapter in Section 6.7.

6.2 Problem Definition

Let D_A and D_B be two databases from database owners A and B, respectively. Let $SG = (V_A \cup V_B, E)$ be the similarity graph resulting from a record linkage process using a certain linkage configuration. Based on an analysis of the two databases and the given similarity graph SG, we aim to estimate the linkage quality in terms of the total number of true positives (tp), false positives (fp) and false negatives (fn). From these estimates precision and recall, as well as aggregated measures such as the F-measure, can be calculated. We assume that no ground truth data is available that can be used, for example, due to privacy or data protection concerns.

6.3 Related Work

Existing methods to estimate linkage quality in the context of record linkage can be roughly divided into three categories, which we describe in the following.

6.3.1 Manual Assessment

The result of a linkage is manually inspected by domain experts in order to assess the linkage quality outcome [Chr12b]. The disadvantage of such approaches is that they can be very time- and resource-consuming. To limit this effort, often only a small sample of record pairs is revised, in particular edge cases. These are pairs that are hard to classify and thus have high uncertainty [Chr12b].

A simple sampling method is proposed by Boyd et al. [Boy+16] where record pairs at different threshold values are sampled and clerically reviewed. The obtained results are then applied to the entire dataset and provide estimates for the number of false positives and false negatives.

Marchant and Rubinstein proposed OASIS [MR17], a tool that takes an unlabeled dataset as input and intelligently selects items to be (manually) labeled to provide an estimate of the linkage quality. To minimize the amount of labeling required, OASIS uses an adaptive importance sampling method.

In privacy-preserving settings, however, such manual inspection is even harder to employ. Initial work [Kum+14] addresses this problem by visual masking and partly hiding actual attribute values, in order to allow manual link decisions without compromising the privacy of individuals. However, manual decisions based on masked attribute values might also be less accurate compared to reviews based on fully visible attribute values.

6.3.2 Supervised Approaches

Linkage quality can be estimated based on ground truth (training) data, where the match status of a set of record pairs is known. Such training data need to be of high quality and contain a large diversity of example pairs, especially those that are difficult to classify. Heise et al. [HKN14] proposed a sampling-based approach for duplicity assessment, which estimates the number and sizes of duplicate record clusters in a dataset. The main benefit of their approach is that it can efficiently approximate the number of duplicates while only performing a fraction of the candidate comparisons compared to what an actual record linkage process would take.

Binette et al. [Bin+22] estimate linkage quality from samples by using (partial) ground truth data. Similarly, in [DH19; HDG20] partial ground truth data is submitted to the linkage process in the form of positive/negative controls. Positive controls are records that are known to be a match. In contrast, negative controls are records that should definitely not match any other record. In [Moo+14], such controls are used for the linkage of prisoner records and a register of deaths. In that specific scenario, for a subset of prisoners, it is known that they died in prison (positive controls) while for another subset of prisoners, it is known that they were alive at the time of the linkage (negative controls). By counting the number of correctly classified control records, the linkage quality can be calculated. In general, the control records can also be artificially created jointly by the database owners and then employed in the linkage process.

Again, in privacy-preserving record linkage scenarios, database owners might not be able or willing to prepare and exchange training data due to privacy constraints [CRS20].

6.3.3 Unsupervised Approaches

These approaches do not have access to the characteristics of true matching and non-matching record pairs. Lamiroy and Sun [LS11] propose an approach to measure recall and precision in the absence of ground truth data. Their method requires access to different competing approaches, such as different classifiers, in order to establish a ranking and find an overall consensus between these approaches. The drawback of this approach is that it is sensitive to collective bias, namely if the competing approaches are consistent in their errors. Similarly, Platanios et al. [PBM14] propose methods for estimating the accuracy of different competing classifiers based on their agreement rates over unlabeled data. The authors show that their approach is able to estimate accuracy if the competing classifiers do not make independent errors. Other unsupervised approaches rely on dataset characteristics and the similarities between pairs or groups of records. Such approaches are closely related to clustering approaches that can be used for classification, as well as for post-processing [Chr12b]. Clustering is the process of partitioning data objects into subsets (called clusters), such that intra-cluster similarity is maximized while inter-cluster similarity is minimized. This means that objects in the same cluster have a high similarity, while objects in different clusters have a low similarity to each other [HKP12]. Clustering techniques utilize different heuristics but are generally executed in an unsupervised fashion. In [NdM12], Nikolov et al. propose an unsupervised approach that aims to estimate linkage quality in the absence of labeled data. Therefore, pseudo-precision (PP) and pseudo-recall (PR) measures are used which are defined as follows:

$$PP = \frac{|\{a \in V_A | \exists b \in V_B : (a,b) \in E\}|}{\sum_{a \in V_A} |\{b \in V_B | (a,b) \in E\}|} \tag{6.1}$$

$$PR = \frac{|E|}{\min(|V_A|, |V_B|)} \tag{6.2}$$

Assuming the databases to be linked are clean, the pseudo-precision measure is based on the following fact: If there are multiple links originating from the same record, at most one can be correct. The other links are necessarily errors. The pseudo-recall measure considers the number of records in the smaller partition (database) as the maximum number of possible matches. However, this is only the case if one database is a subset of the other database. This, in turn, will be rarely the case in most record linkage scenarios [Chr12b]. As a consequence, pseudo-recall tends to (strongly) underestimate the actual recall, if the overlap between the two databases is low. Besides, pseudo-recall can result in values greater than 1, namely if $|E| > \min(|V_A|, |V_B|)$. To overcome this issue, Ngomo and Lyko [NL13] refined the approach by specifying an alternative pseudo-recall variant, which is defined as:

$$PR_{Alt} = \frac{|\{a \in V_A | \exists b \in V_B : (a,b) \in E\}| + |\{b \in V_B | \exists a \in V_A : (a,b) \in E\}|}{|V_A| + |V_B|} \tag{6.3}$$

This pseudo-recall measure indicates how well the records in both databases are covered by the linkage result. A pseudo-recall value of 1 means that every record of database D_A is linked to at least one record of database D_B and vice versa. While the results in [NdM12] are promising, the authors in [NL13] achieved varying results, with both positive and negative correlations between the estimated and the actual linkage quality. However, the results are hard to interpret as only F-measure values were reported and compared (a weakness of the F-measure reported by others [CHK23]). It is thus difficult to assess if the ambiguous correlations reported are due to the recall or precision estimates.

6.4 Estimating Linkage Quality using Similarity Graphs

The key idea of our methods for estimating linkage quality in the absence of ground truth data is to analyze both the input data and the similarity graph generated by a record linkage algorithm. For assessing the quality in terms of recall and precision, the number of true positives (*tp*), false positives (*fp*), and false negatives (*fn*) need to be approximately determined. In the following, we discuss different strategies considering the degree of vertices, similarity of edges, and cryptosets, to determine the relevant counts required for calculating precision and recall.

Because of possibly different data quality levels regarding duplicates in a database, we distinguish our methods as being suitable for deduplicated databases (clean) and databases containing (intra-source) duplicates (dirty). Assuming that the databases to be linked are duplicate-free, the number of possible matches for each record is limited to one (see Section 2.6.6), and therefore our heuristics need to be more strict. In general, we assume that the similarity graph was generated without applying a one-to-one cardinality restriction as part of a post-processing step. By applying a one-to-one cardinality restriction, the most likely matching record out of a set of candidates would be selected (see Chapter 5). This would lead to a loss of information about the ambiguity of possible match candidates.

6.4.1 Deduplicated Databases

The main difference between clean and dirty databases is that with the former a record $a \in D_A$ can correspond to at maximum one record $b \in D_B$. Otherwise, the database D_B is not duplicate-free if a record $b' \in D_B$ exists where $b' = a$. Due to the transitive closure of a regarding equality [Chr12b], record b would be equal to b', which contradicts the assumption of duplicate-free databases.

We utilize this constraint and the degree of nodes in the similarity graph as indicators for true positives. A one-to-one link implies that there is exactly one match candidate for a record. In contrast, a multi-link implies that there are several match candidates for a record, leading to uncertainty regarding the decision of which records to match. While for clean databases each additional match candidate will be a false positive (without a post-processing step), for dirty databases multiple match candidates may form an intra-source duplicate (as we will discuss in Section 6.4.2). In addition to the edge degree, the edge weight (the aggregated similarity sim_Δ or a confidence value) is also an important criterion. The higher the edge weight and the greater the difference to the similarity threshold value is, the more certain a match decision will be.

In the following, we describe the different strategies using the vertex degree and the edge similarity to estimate the number of true positives and false positives, as well as cryptosets to determine the number of false negatives.

6.4.1.1 Vertex Degree

Due to the constraint for duplicate-free databases, we can approximate the set of true positives by the records of a database D_A that have been linked to at most one record from the other database D_B. We can formalize the set of estimated true positives TP_A regarding database A as follows:

$$TP_A = \{a \in V_A \mid \exists b \in V_B : (a, b) \in E\} \tag{6.4}$$

Using the estimation of the set of true positives, we can approximately determine precision with Equation 6.5 where $|E|$ represents the number of edges. In this approximation, the number of true positives is limited by the minimum number of expected true positives regarding the set of records from TP_A and TP_B being linked. To relax the assumption of one-to-one links, Equations 6.6 and 6.7 considers the average of the number of records from TP_A and TP_B. Equation 6.7 limits the number of links by the minimum of $|V_A|$ and $|V_B|$ motivated by the duplicate-free assumption.

$$PP_{1:1} = \frac{\min(|TP_A|,\ |TP_B|)}{|E|} \tag{6.5}$$

$$PP_{1:n} = \frac{|TP_A| + |TP_B|}{2 \cdot |E|} \tag{6.6}$$

$$PR_{AltMin} = \frac{|TP_A| + |TP_B|}{2 \cdot \min(|V_A|, |V_B|)} \tag{6.7}$$

6.4.1.2 Similarity Scores

In addition to the graph structure, similarity graphs provide information for each edge representing how likely a match between the linked records is. Therefore, we utilize the similarities to calculate for each edge a probability to be a true positive depending on its adjacent edges. As we discuss below, the intuition is that we can select for each record only one edge, and therefore we utilize the similarities of adjacent edges as a probability to select one edge per record. The calculated probability for each edge and the restriction of edges based on the duplicate-free assumption can then be used to calculate the expectation of the number of true positives for the given similarity graph.

For calculating the probability of a true positive given an edge $e = (a, b) \in E$, we determine two probabilities, $p_{tp}^{A}(e)$ and $p_{tp}^{B}(e)$ representing how likely e is a true positive considering records $a \in V_A$ and $b \in V_B$. The probability $p_{tp}^{A}(e)$ defined in Equation 6.8 (with $p_{tp}^{B}(e)$ calculated in a similar way) is based on the similarity of edge e and normalized by the sum of the similarities of its adjacent edges associated with a record a. Here, $N(v)$ denotes the neighborhood of a vertex v, which is the set of vertices adjacent to v [Die17].

$$p_{tp}^{A}(e) = \mathbb{P}[e = (a, b) \in TP \mid a \in V_A] = \frac{sim_\Delta(e)}{\sum_{b' \in N(a)} sim_\Delta(a, b')} \tag{6.8}$$

The probabilities $p_{tp}^{A}(e)$ and $p_{tp}^{B}(e)$ are used to determine a joint probability indicating how likely it is that e is a true positive. To estimate the number of true positives, we then calculate the expected value of true positives based on the joint probability of e where $e \in E_{Sel}$. The set E_{Sel} defined in Equation 6.9 consists of edges maximizing the similarity for at least one incident vertex a or b regarding the edges being adjacent with the vertices of the neighborhood of a respectively of b. Due to our assumption that the databases are deduplicated, we assume that for each record the edge with the largest similarity is most likely a true link.

$$E_{sel} = \Big\{(a, b) \in E \;\Big|\; \max_{b' \in N(A)} (sim_\Delta(a, b')) = sim_\Delta(a, b) \vee \max_{a' \in N(B)} (sim_\Delta(a', b)) = sim_\Delta(a, b)\Big\} \tag{6.9}$$

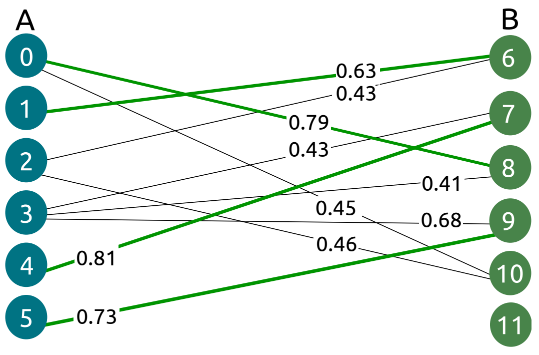


Figure 6.1: Example similarity graph of records from databases A and B. True positive links are shown with thick green lines.

To calculate the expected value of true positives considering the edges of E_{sel}, we determine the sum of the joint probability over all edges $e \in E_{sel}$ that is formally defined as follows:

$$\mathbb{E}(TP) = \sum_{e=(a,b)\in E_{sel}} p^A_{tp}(e) \cdot p^B_{tp}(e) \tag{6.10}$$

We can then define our new precision estimate PP_{prob} as:

$$PP_{prob} = \frac{\mathbb{E}(TP)}{|E|} \tag{6.11}$$

Example: Using the various methods, we can estimate the number of true positives for our example shown in Figure 6.1. In this example, $|E| = 10, |V_A| = |V_B| = 6$ as well as $|TP_A| = 6$ and $|TP_B| = 5$. The determined sets are used to calculate $PP_{1:1} = {}^{\min(5,6)}/_{10} = 0.5$, $PP_{1:n} = {}^{(5+6)}/_{2\cdot 10} = 0.55$ and $PP = {}^{6}/_{10} = 0.6$. For calculating PP_{prob}, we need to calculate the probabilities $p^A_{tp}(e)$ and $p^B_{tp}(e)$ for each edge $e \in E_{sel} := \{(0,8),(1,6),(2,10),(3,9),(4,7),(5,9)\}$ being aggregated by $\mathbb{E}(TP)$. For instance, the probabilities $p^A_{tp}((3,9))$ and $p^B_{tp}((3,9))$ for the edge $(3,9)$ are $p^A_{tp}((3,9)) = {}^{0.68}/_{(0.68+0.41+0.43)} \approx 0.45$ and $p^B_{tp}((3,9)) = {}^{0.68}/_{(0.68+0.73)} \approx 0.48$, respectively. Overall, the expected number of true positives is $\mathbb{E}(TP) = 2.662$ resulting in $PP_{prob} = 0.266$.

6.4.1.3 Cryptosets

To approximate recall, we need to determine the number of overlapping records. In order to guarantee the privacy of sensitive personal information, we utilize cryptosets [SMR15]. The main idea of using cryptosets is the analysis of histograms consisting of record-depending information from both databases. An example of how cryptosets are generated is illustrated in Figure 6.2.

For each record, a private identifier is constructed by applying specific functions on a set of attributes. These private identifiers do not need to be unique, but the number of

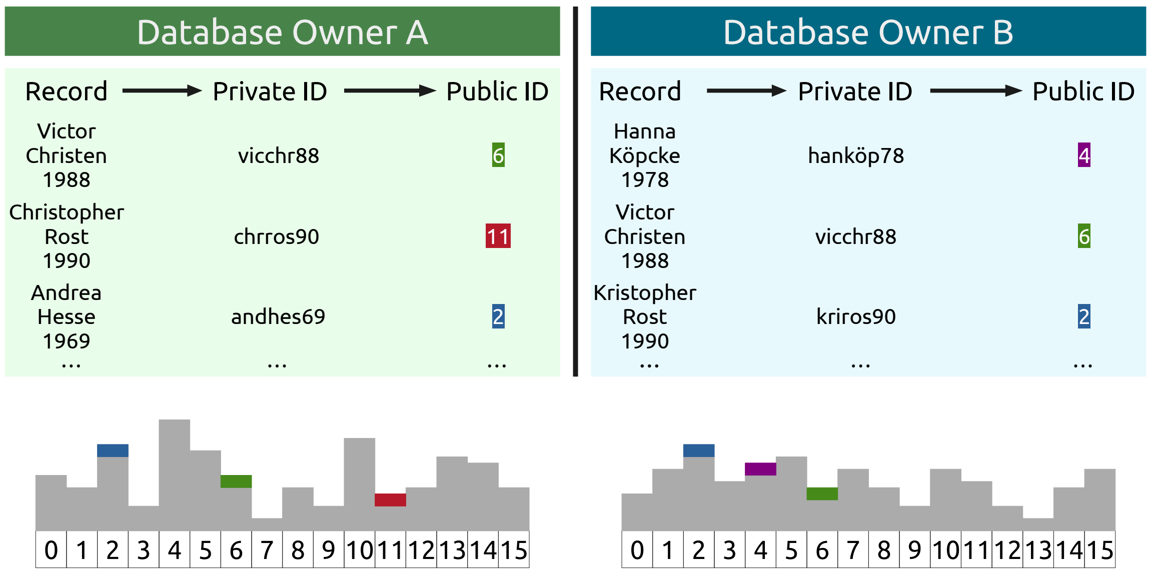


Figure 6.2: Illustration of the cryptoset approach to estimate the overlap of two (private) datasets (adapted from [SMR15]).

records with the same private identifier should be kept small. On the one hand, the more unique the private identifiers are, the more accurate the cryptoset estimate of the overlap between the two databases will be. On the other hand, the construction of the private identifiers should be error-tolerant. Records that refer to the same entity but contain errors or inconsistencies, such as typos or missing values, should ideally produce the same private identifier otherwise the overlap will be underestimated. In our example shown in Figure 6.2, the private identifiers are generated by concatenating the first three characters of the first name and last name and the last two digits of the year of birth.

The resulting private identifier id_{priv} is transformed to a public identifier id_{pub} in the range $[0, L-1]$ using a one-way cryptographic hash function h, i. e., $id_{pub} = h(id_{priv})$ [CRS20]. Then, each database owner initializes a histogram of length L and increments for each record the count at the position $id_{pub} \bmod L$ corresponding to the public ID of the record (bottom of Figure 6.2).

Cryptosets have a trade-off between estimation error and security risk [SMR15]. This trade-off is controlled by the cryptoset length L. Longer cryptosets result in fewer collisions because fewer public identifiers (records) are mapped to the same position. While this makes the estimates more accurate, the cryptosets become less secure.

Overlap Estimation

Assuming two (sensitive) databases D_A and D_B for which cryptosets C_A and C_B have been constructed using the same protocol, then the overlap of records $CE(C_A, C_B)$ (**c**rypotset **e**stimation) in these two databases, $|D_A \cap D_B|$, can be estimated as follows:

$$CE(C_A, C_B) = pc(C_A, C_B) \cdot \sqrt{\frac{\max(|D_A|, |D_B|)}{\min(|D_A|, |D_B|)}} \tag{6.12}$$

where $pc(\cdot,\cdot)$ is the Pearson correlation coefficient. Note that $|D_A| = \sum_{i=0}^{L-1} C_i^A$ and $|D_B| = \sum_{i=0}^{L-1} C_i^B$. The Pearson correlation coefficient is defined in Equation 6.13 based on the covariance between the cryptosets of C_A and C_B normalized by the product of the standard deviations of C_A and C_B, where $\overline{C_A}$ and $\overline{C_B}$ are the means of frequencies of the id_{pub} distribution of C_A and C_B, respectively.

$$pc(C_A, C_B) = \frac{\sum_{i=0}^{L-1}(C_A[i] - \overline{C_A})(C_B[i] - \overline{C_B})}{\sqrt{\sum_{i=0}^{L-1}(C_A[i] - \overline{C_A})^2} \cdot \sqrt{\sum_{i=0}^{L-1}(C_B[i] - \overline{C_B})^2}} \tag{6.13}$$

We can now determine recall by utilizing the cryptoset-based approximation of the overlap from Equation 6.12 and the approximation of the number of true positives based on $|TP_A|$ and $|TP_B|$, or $\mathbb{E}(TP)$ as calculated in Equation 6.4 and Equation 6.10 for the databases D_A and D_B.

$$PR_{CE1:1} = \frac{\min(|TP_A|, |TP_B|)}{CE} \tag{6.14}$$

$$PR_{CE1:n} = \frac{|TP_A| + |TP_B|}{2 \cdot CE} \tag{6.15}$$

$$PR_{CEprob} = \frac{\mathbb{E}(TP)}{CE} \tag{6.16}$$

Generation of Private Identifiers

Due to the importance of private identifiers for estimating the overlap, automatic approaches are required if the databases consist of heterogeneous or sensitive data, which makes manual selection infeasible. We propose a method, as outlined in Algorithm 1, that automatically selects a subset of attributes to generate the identifiers based on the attribute characteristics such as uniqueness as well as value distribution. The selected attributes representing the private identifier influence the estimated overlap. A high number of records with the same identifier results in an overestimated overlap, whereas a small number leads probably to an underestimated overlap since the private identifiers are too unique and thus the intersection of the resulting histograms is small.

ALGORITHM 1: Apriori-like approach to determine attributes for generating meaningful private identifiers.

```
Input:  $\mathbf{D}$: dataset from a certain party, $\mathbf{A}$: set of attributes
        $mr$: threshold for the ratio of missing attribute values
        $t_{info}$: threshold to filter uninformative attribute combinations
Output: $\mathbf{AC}$ set of attribute combinations to generate private IDs
1  $\mathbf{AC} \leftarrow \emptyset$
2  $\mathbf{A}_{valid} \leftarrow filterAttributes(D, A, mr)$
3  $tempAttCombs \leftarrow apriori(A_{valid})$
4  do
5      $filteredCombs \leftarrow \emptyset$
6      for $ac \in tempAttCombs$ do
7          $u \leftarrow computeUniqueness(\mathbf{D}, ac)$
8          $s \leftarrow computeWeightedUniformitySim(\mathbf{D}, ac, u)$
9          $info \leftarrow 2 \cdot {}^{(s \cdot u)}/_{(s+u)}$
10         if $info \geq t_{info}$ then
11             $filteredCombs \leftarrow filteredCombs \cup \{ac\}$
12             $\mathbf{AC} \leftarrow \mathbf{AC} \cup \{ac\}$
13     $tempAttCombs \leftarrow apriori(filteredCombs)$
14 while $tempAttCombs \neq \emptyset$
15 return AC
```

Therefore, we propose an automatic approach for selecting a subset of attributes satisfying different criteria so that the resulting identifiers enable an effective estimation [RCS21]. The approach follows an apriori-like strategy [AS94], where we start with attribute sets of size one and combine them. An attribute combination is added to the final result set **AC** if the harmonic mean based on the uniqueness (u) and the weighted similarity (s) regarding a uniform distribution is above a threshold t_{info} (Algorithm 1 line 5-13). The attribute combination is also added to the candidate set *filteredCombs* to generate larger attribute combinations *tempAttCombs* being validated in the next iteration. The generation process stops if we cannot derive larger attribute sets satisfying the defined criteria in terms of uniqueness and similarity to a uniform distribution.

We define uniqueness (u) as the ratio of distinct values regarding an attribute combination and the number of records. Moreover, the similarity s is determined by computing the histogram intersection between the value distribution regarding a certain attribute combination and a uniform distribution. To avoid a high impact of combinations leading to a high uniqueness, we weigh the similarity by the uniqueness of an attribute combination with ${}^{u \cdot (1-u)}/_{0.25}$ mitigating the impact of combinations with a high ($u \approx 1$) or small uniqueness ($u \approx 0$). To reduce the number of attribute combinations, we filter the possible attributes based on the number of existing values at first (line 2). Our assumption here is that attributes or combinations with a high number of missing values result in ineffective identifiers for representing the underlying records.

6.4.2 Dirty Databases

The proposed methods in the previous section assume one-to-one links between the two databases. Consequently, if we use these methods for databases with duplicates, we would underestimate the number of true positives since multi-links are possible.

For estimating the number of true positives, we rely on the assumption that records being the same entity are similar to each other, which is also reflected in the similarity graph. As a result of the linkage process, records representing the same entity are elements of one connected component. A connected component CC is a maximal-connected subgraph, i. e., CC is not part of any larger connected subgraph [Die17]. The records of a connected component should be similar to each other which is explicitly represented by the computed similarities. Nevertheless, the similarities can be different due to quality issues or edges missing due to the specified threshold. In this case, we cannot utilize the similarities directly to quantify the number of true positives.

We reformulate the assumption that each record is similar to the other records using the personalized PageRank [Pag+99]. In the context of the personalized PageRank considering a certain record, each record of a connected component should be reachable with roughly the same probability. Otherwise, a record is more (dis)similar to a subset of records, indicating that not all records refer to the same entity.

To quantify the number of true positives, we introduce a true positive score $tp_{score}(a, b)$ for each edge $e = (a, b)$ based on the personalized PageRank $pp(a, b)$ of the adjacent records a and b as well as the similarity sim_{Δ}. Ideally, the probability of reaching a record b starting from a is equal to the probability by randomly selecting a record b' from the connected component CC of a since each record should be similar to the other records. The probability of randomly selecting a record of a connected component is $p_{uni} = \frac{1}{|CC|}$. Using the probabilities, we define the true positive score of an edge as:

$$tp_{score}\,(e = (a, b)) = (1 - |pp(a, b) - p_{uni}|) \cdot (1 - |pp(b, a) - p_{uni}|) \cdot sim_{norm}(a, b) \quad (6.17)$$

The first factor and the second factor represent the probability difference reaching node b starting from a and reaching node a starting from b, respectively. The third factor weighs the two differences using the min-max normalized similarity $sim_{norm}(a, b)$ between a and b. The smaller the differences and higher the similarity, the higher the tp_{score} for the edge $e = (a, b)$. The total number of true positives TP_{score} is estimated by the sum of $tp_{score}(e)$ overall identified matches $e \in E$. The resulting estimation is used to compute the precision PP_{dup} as follows:

$$PP_{dup} = \frac{TP_{score}}{|E|} \quad (6.18)$$

Dataset	Attributes	#Records	#Matches	Blocking Key	Similarity Function
Music Brainz	Artist, title, album, year, length, language, number	20 000	16 250	preLen1(album)	Trigram(title)
Dexter	Heterog. key-value pairs	21 023	185 839	mfr. name, model number	Trigram(model names, product code, sensor type), Euclid(opt./digital zoom, camera dim., price, weight, resolution)

Table 6.1: Characteristics and linking configuration of MusicBrainz and Dexter datasets.

To measure recall, we use the estimated number of true positives TP_{score} compared to the estimated overlap CE by using cryptosets.

6.5 Discussion of Privacy Aspects

Our methods for estimating linkage quality rely on analyzing similarity graphs as well as cryptosets of the databases to be linked. In the context of PPRL, there are only a few works that propose attacks on similarity graphs [Vid+20a]. Such attacks aim to determine a mapping between the encoded data and publicly available plaintext data by using graph features (such as weighted node degrees). Our estimation methods, however, utilize existing similarity graphs which are typically generated by the linkage unit in PPRL scenarios. Therefore, our approaches do not add any privacy flaws but rather rely on the security of the method that was used to generate the similarity graph.

Cryptosets can be seen as a summary of the databases' contents that can be shared in public, untrustworthy environments to measure the overlap between private databases. In the literature, cryptosets are considered as information-theoretic secure [SMR15] as it is not possible to determine which records are in a private database based on its cryptoset. For the overlap estimation, the cryptosets of the databases to be linked need to be shared with the linkage unit or between the database owners. Each cryptoset is a vector of length L containing the counts of public identifiers. Those public identifiers are determined by mapping the private identifiers of records representing (parts of) attribute values to an integer value in the range $[0, L-1]$ using a cryptographic one-way hash function. Setting $L \ll \min(|D_A|, |D_B|)$ results in a many-to-one relationship between private and public identifiers where the number of records being mapped to the same public identifier is typically large and thus impeding the alignment of specific records [SMR15]. Consequently, even if an adversary knows the encoding function, dictionary-based attacks are not feasible.

In addition to this theoretical argument, information gain [CRS20] can be used as a measure to quantify how much information is exposed by a cryptoset C_A compared to a theoretically optimal cryptoset C_U consisting of all possible values in the domain of private identifiers. Due to the large number of possible values in a domain, the public identifiers in C_U are approximately uniformly distributed so that each position in C_U is set with a probability of $^1/_L$. We can calculate the information gain $I\left(C_A \,\middle\|\, C_U\right)$ using both entropies of C_A and C_U as shown in Equation 6.19, where smaller information gain values represent higher privacy. If information gain is high, the frequency distribution of a cryptoset can potentially be used in a cryptanalysis attack to align it to a public value distribution, such as telephone books or census data for names. However, no such attack has so far been developed.

$$I\left(C_A \,\middle\|\, C_U\right) = -\sum_{i=0}^{L-1} \frac{1}{L} \cdot log_2 \frac{1}{L} - \left(-\sum_{i=0}^{L-1} \frac{C_A[i]}{|D_A|} \cdot log_2 \frac{C_A[i]}{|D_A|}\right) \tag{6.19}$$

Securely computing the intersection of private databases is an intensively studied problem with various approaches showing different security and complexity properties [Kum+21]. In general, two parties want to compute the intersection of their private sets without revealing anything to the other party other than the (number of) elements in the intersection. For a detailed discussion of different private set intersection protocols, we refer to [Pin+19]. Many approaches focus only on two parties, compute only the exact overlap (not considering errors or inconsistencies between matching records), or do not account for duplicate elements (multiset intersection) [AES03; FNP04; Ege+15]. Therefore, we employ cryptosets as a specific solution to the private set intersection cardinality problem that meets our requirements. However, our approaches for estimating linkage quality are not strictly limited to cryptosets.

6.6 Experimental Evaluation

In this section, we evaluate the proposed approaches for linkage quality assessment using datasets from three distinct domains with different characteristics. In the following, we describe the datasets we used as well as the methods and parameter settings for generating the similarity graphs.

6.6.1 Datasets

We use datasets from three different domains: voter records (personal information), records about music albums, and records about consumer products (cameras). In contrast to the voter datasets, the music and camera datasets are more heterogeneous (have different attribute structures) and show diverse types of errors. The voter and music datasets are clean (duplicate-free), while the product dataset is dirty and contains intra-source duplicates. We use the voter dataset to estimate the linkage quality in a PPRL scenario. The music and product datasets, in contrast, are used for estimating the quality in a non-privacy-oriented linkage context.

6.6.1.1 NCVR

We first consider a dataset provided by Panse et al. [Pan+21] that is based on the North Carolina Voter Registration (NCVR) database (`https://www.ncsbe.gov/`). This dataset contains over 120 million historic voter records with person-related attributes such as first name (FN), middle name (MN), last name (LN), year of birth (YOB), place of birth (POB), city, ZIP code, and sex. Compared to the other two datasets, MusicBrainz and Dexter as described next, it represents a homogeneous dataset in terms of the number of attributes and the characteristics of attribute values such as length distribution and amount of missing values. From this dataset, we extracted subsets A and B with $|A| = |B| = 200\,000$ and varying degrees of overlap (number of matches):

- $NCVR_H$ (high overlap) where $|A \cap B| = 160\,000$.
- $NCVR_{MH}$ (medium-high overlap) where $|A \cap B| = 120\,000$.
- $NCVR_M$ (medium overlap) where $|A \cap B| = 100\,000$.
- $NCVR_{LM}$ (low-medium overlap) where $|A \cap B| = 80\,000$.
- $NCVR_L$ (low overlap) where $|A \cap B| = 40\,000$.

Each singleton record is drawn from the NCVR snapshot of '2021-01-01'. Each duplicate pair (a, b) consists of records $a \in A$ and $b \in B$ where record a is drawn from a snapshot between '2008-01-01' (inclusive) and '2021-01-01' (exclusive), while record b is from snapshot '2021-01-01'. Moreover, there is a difference or error in at least one attribute that is not the year of birth: $\forall a, b : (YOB(a) = YOB(b)) \wedge \exists attr \in \{FN, MN, LN, POB, SEX\} : attr(a) \neq attr(b)$.

As we use this dataset to estimate the linkage quality in a PPRL scenario, we utilize Bloom filters as proposed by Schnell et al. [SBR11] as an encoding technique. Bloom-filter-based encodings have become the quasi-standard for recent PPRL approaches in both research and real applications [CRS20; Vat+17]. We use record-level Bloom filters with a length of $m = 1024$, trigrams, and attribute weighting. To overcome the quadratic complexity of linkage, we use LSH-based blocking based on the Hamming distance like in previous work (see Chapter 3). To determine the similarity of candidate record pairs, we use the Jaccard coefficient [CRS20].

6.6.1.2 MusicBrainz

The MusicBrainz dataset is a synthetically generated dataset from the MusicBrainz (`https://musicbrainz.org/`) database. The dataset is corrupted by [Hil+20] consisting of five sources with duplicates for 50% of the original records. While each database is duplicate-free, the records are heterogeneous regarding the characteristics of attribute values such as the number of missing values, length of values, and ratio of errors. The similarity graphs we used in our evaluation have been utilized in several previous studies [SPR18; SPR20; LSR21; SDR21]. The linkage configuration is shown in Table 6.1.

6.6.1.3 Dexter

This dataset is derived from the camera dataset of the ACM SIGMOD 2020 Programming Contest (`http://www.inf.uniroma3.it/db/sigmod2020contest/index.html`). The dataset consists of 23 sources with approximately 21 000 records and intra-source duplicates. Each database consists of source-specific attributes. We used the same linkage configuration as in previous studies [SDR21; SPR20] (see Table 6.1).

6.6.2 Results

In the following, we evaluate our proposed methods for estimating the linkage quality on both clean (deduplicated) and heterogeneous/dirty databases. For each dataset, we analyze the recall, precision, and the resulting F-measure estimates and compare them with the actual results as calculated from ground truth data.

6.6.2.1 Clean Databases

An essential part of estimating the linkage quality is the estimation of the overlap of two databases utilizing the cryptoset method we described in Section 6.4.1.3. Due to independence regarding various similarity graphs, we evaluate the approach considering different manual-defined private identifiers on the NCVR datasets representing a homogeneous dataset. We consider three different private identifier configurations: `[2FN,2LN,YOB]`, `[3FN,3LN,YOB]`, and `[SD_FN,SD_LN,YOB]`, where $2A/3A$ extract the first 2/3 letters from the value of attribute A. Similarly, `SD_`A computes the Soundex [HM02] from the value of attribute A. We empirically set the cryptoset length to $L = 8192$ as this results in more accurate estimates with a lower standard deviation.

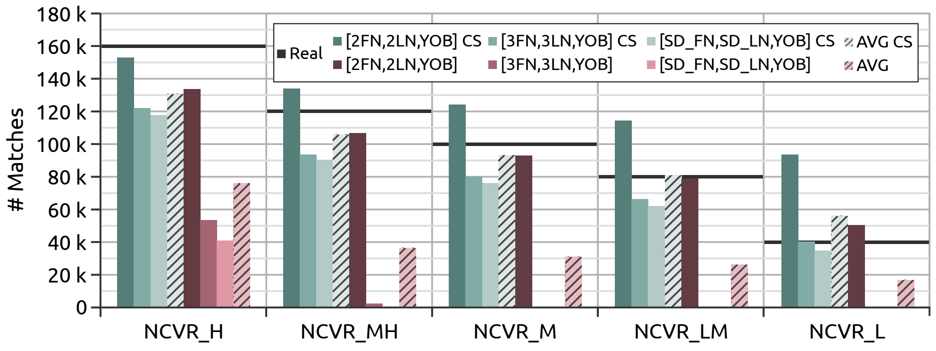


Figure 6.3: Evaluation of cryptoset approach to measure recall. Results based on ground truth are shown as horizontal lines.

The results, as depicted in Figure 6.3 (green bars), show that the cryptoset approach is able to estimate the overlap for different private identifier generation configurations. The estimates for NCVR$_H$ (with 160 000 matches) range from roughly 117 000 (73.1%) to 153 000 (95.6%), for NCVR$_M$ (100 000 matches) from 76 000 to 124 000 and for NCVR$_L$ (40 000 matches) from 35 000 (87.5%) to 94 000 (235%). The configuration `[3FN,3LN,YOB]` provides the best estimate with an average absolute difference of around 19 660 matches between the estimate and the actual number of matches over all datasets. To reduce the impact of different configurations to construct the private identifiers, we calculate the average estimated overlap over a set of configurations. Using this average leads to the best estimate with an average difference of only around 13,400 matches to the actual number of matches. We therefore use the average over the estimated overlaps as default in the following experiments.

We also compared the cryptoset approach against estimating the overlap using the private identifiers directly. The results (red bars) show that only the configuration `[2FN,2LN,YOB]` leads to similar results compared to the cryptoset estimate. Therefore, the non-private overlap estimation is more sensitive regarding the used configuration. The overhead of the encryption is negligibly small, with an average runtime of 1.9s compared to 0.9s using the non-private estimation considering all configurations and datasets.

The results of the different quality estimation approaches for the NCVR dataset are shown in Figure 6.4. The precision estimates for $NCVR_H$ (high overlap) are very close to each other and also to the real precision. With decreasing overlap, the estimates PP, $PP_{1:1}$, and $PP_{1:n}$ are increasingly overestimating the actual precision. In such cases,

PP_{prob} is providing better estimates. In terms of recall, PR, PR_{Alt} and PR_{AltMin} are underestimating the actual recall, in particular for the datasets with medium and low overlap. Due to the same size of both databases PR_{Alt} and PR_{AltMin} are equal.

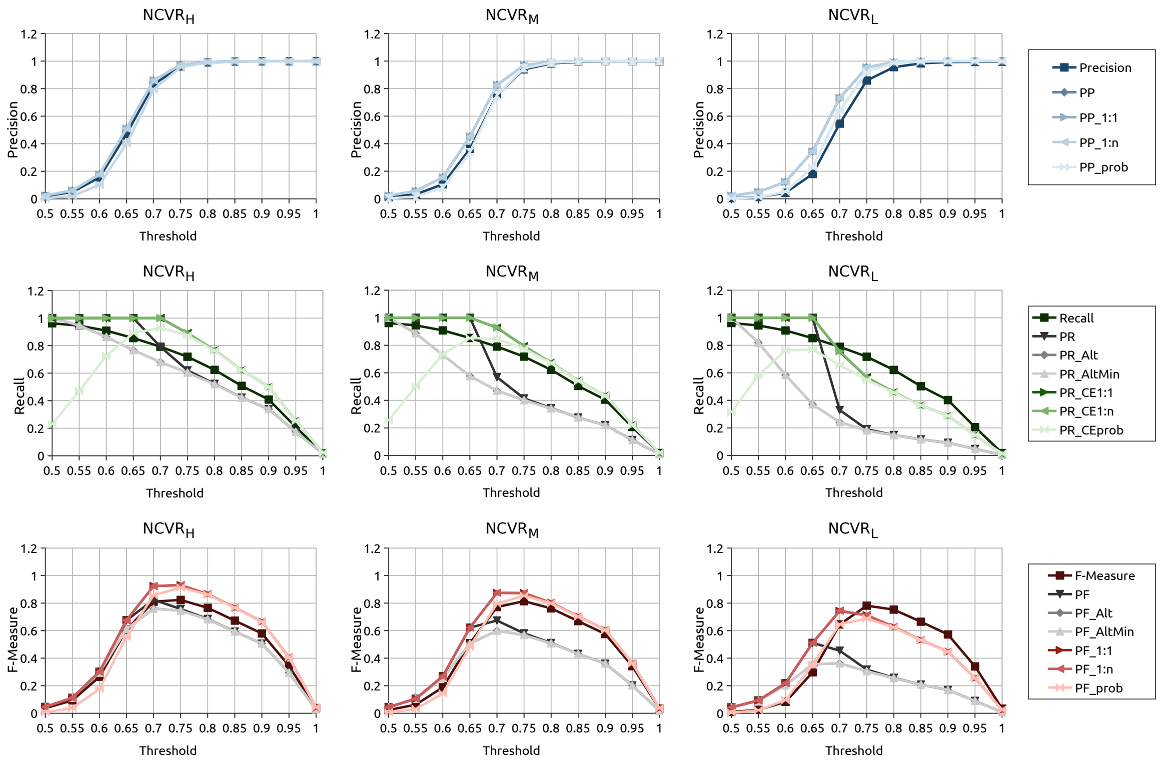


Figure 6.4: Results on NCVR datasets with different overlaps considering different thresholds to generate the similarity graph.

The cryptoset approach, in contrast, provides estimates that are much closer to the actual recall, especially for NCVR$_M$ and NCVR$_L$. The recall estimates for PR_{CEprob} are dropping below a certain threshold which is caused by the small number of expected true positives for small thresholds using the probability-based estimation method. Using low thresholds results in graphs with low similarities and a high number of edges for each record. Thus, the number of true positives is underestimated if the threshold is too low and the difference to the optimal threshold is too high because of the high ambiguity in terms of the similarities of correct and incorrect matches. To avoid the effect of dropping recall estimates considering thresholds $t_1 < t_2$, the estimated recall for threshold t_1 can be bounded to the recall value obtained by threshold t_2 assuming that lower thresholds will not result in fewer true positives.

For dataset $NCVR_H$, all approaches provide estimates that are relatively close to the actual F-measure. Here, the approaches PF, PF_{Alt} and PF_{AltMin} (slightly) underestimate the actual F-measure, while the other approaches (slightly to moderately) overestimate the actual F-measure. The highest F-measure of 0.823 is reached for $t = 0.75$, followed closely by an F-measure of 0.811 for $t = 0.7$ and 0.766 for $t = 0.8$. Using the estimations for the threshold selection, the estimation methods $PF_{1:1}$, $PF_{1:n}$, and PF_{prob} lead to the optimal threshold configuration. In contrast, the approaches PF, PF_{Alt} and PF_{AltMin} reach their maximum estimated F-measure at $t = 0.7$.

For $NCVR_M$, the estimates of PF, PF_{Alt} and PF_{AltMin} begin to diverge more from the actual F-measure. The F-measure is heavily underestimated for thresholds $t > 0.65$, with a maximum at $t = 0.7$, while the optimal threshold is at $t = 0.75$. This trend continues for dataset $NCVR_L$ where the estimates of PF, PF_{Alt} and PF_{AltMin} are even worse. For $NCVR_M$, PF_{prob} achieves the best estimates where the predicted F-measure slightly differs from the actual F-measure by at most 0.06. Considering the threshold selection, the estimation results in selecting the optimal threshold of $t = 0.75$.

In addition to the quality estimation, we analyzed the privacy of the cryptosets for the person datasets $NCVR_L$, $NCVR_M$ and $NCVR_H$. Each entry of a cryptoset is set by on average 24 elements (with a standard deviation ranging from 4.98 to 6.55). We also calculated information gain as described in Section 6.5 using the proposed identifier configurations and $L = 8192$. The information gain values are small, similar to the original work [SMR15], and range from around 0.03 to 0.052 as shown in Table 6.2. Moreover, the more specific the private identifier is, the more evenly the public identifiers are distributed in the cryptoset resulting in a smaller information gain. We also observe that the information gain increases with a higher overlap, which is because the private identifiers are generated from a finite set of values (such as first/last names). Therefore, the number of identifiers mapped to one public identifier increases with the number of records in the overlap, while the number of identifiers of non-overlapping records remains constant.

Configuration private ID	**$NCVR_L$**	**$NCVR_M$**	**$NCVR_H$**
`[2FN, 2LN, YOB]`	0.0497	0.0506	0.0522
`[3FN, 3LN, YOB]`	0.0317	0.0320	0.0316
`[SD_FN, SD_LN, YOB]`	0.0302	0.0314	0.0317

Table 6.2: Averaged information gain $I(C_A||C_U)$ and $I(C_B||C_U)$.

6.6.2.2 Heterogeneous and Dirty Databases

In contrast to the voter datasets, MusicBrainz and Dexter contain heterogeneous records regarding the characteristics of values. Therefore, we utilize our proposed automatic selection method to determine the private identifiers being utilized to estimate the overlap. To generate the private identifiers, we extract the first three characters from the value of an attribute. For the MusicBrainz dataset, we use all available attributes as candidates. For the Dexter dataset, we utilize a subset of the available attributes, such

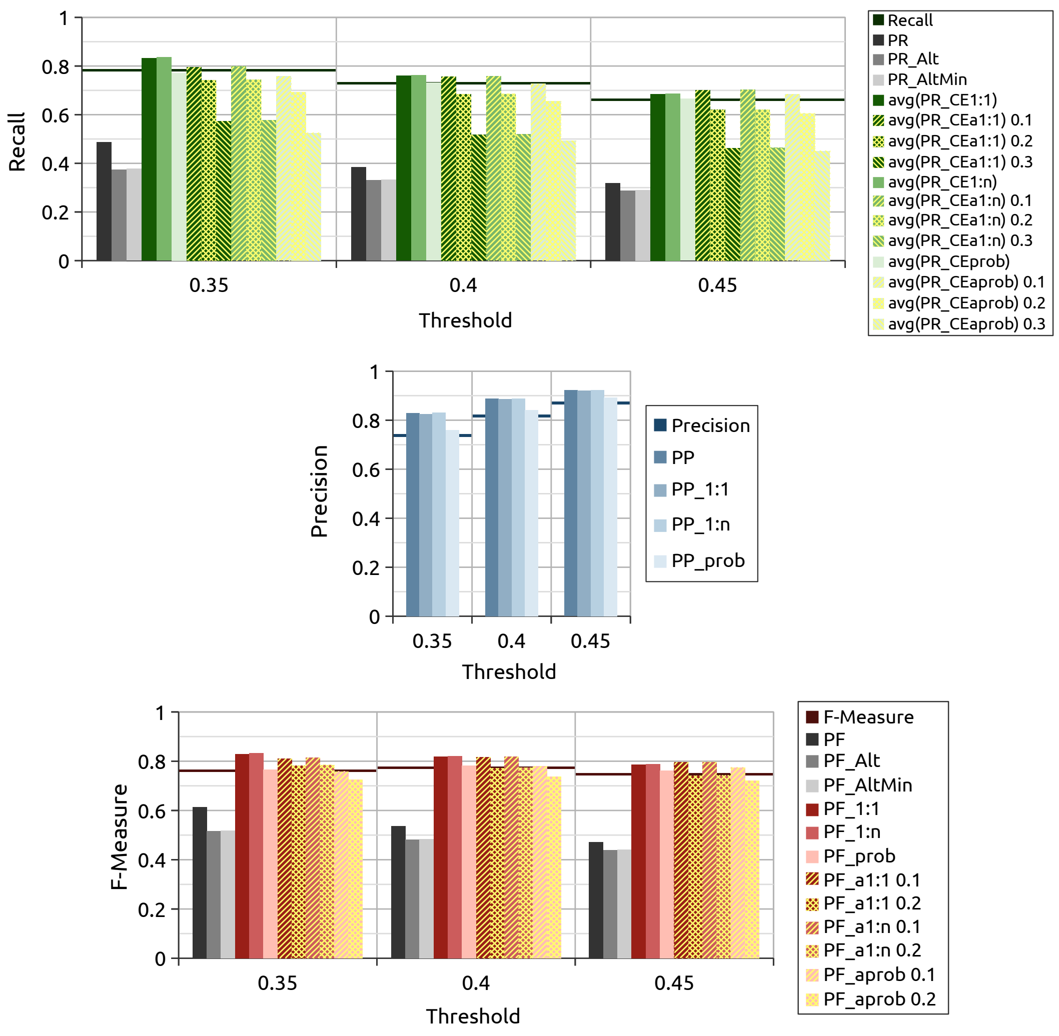


Figure 6.5: Results on MusicBrainz dataset. Results based on ground truth are shown as horizontal lines.

as product name, brand, and model. Both datasets consist of more than two databases, we therefore calculate the macro precision and recall (the average of pairwise precision and recall values) [SCF10].

The results for the MusicBrainz dataset are shown in Figure 6.5. The recall estimates based on cryptosets in combination with the automatic generation of private identifiers are abbreviated with '*..a..*'. To determine the estimated overlap, we follow the same aggregation strategy as for the NCVR dataset, where we averaged the estimations regarding the generated private identifiers. Moreover, we compare the automatic selection method of the private identifiers with manually selected identifiers.

The cryptoset method in combination with the private identifier generation approach results in recall estimates where the average difference between the true and estimated recall values over all thresholds is below 0.03 for $t_{info} = 0.1$. In contrast to the cryptoset estimation, the baseline estimations PR, PR_{Alt} and PR_{AltMin} lead to an average difference

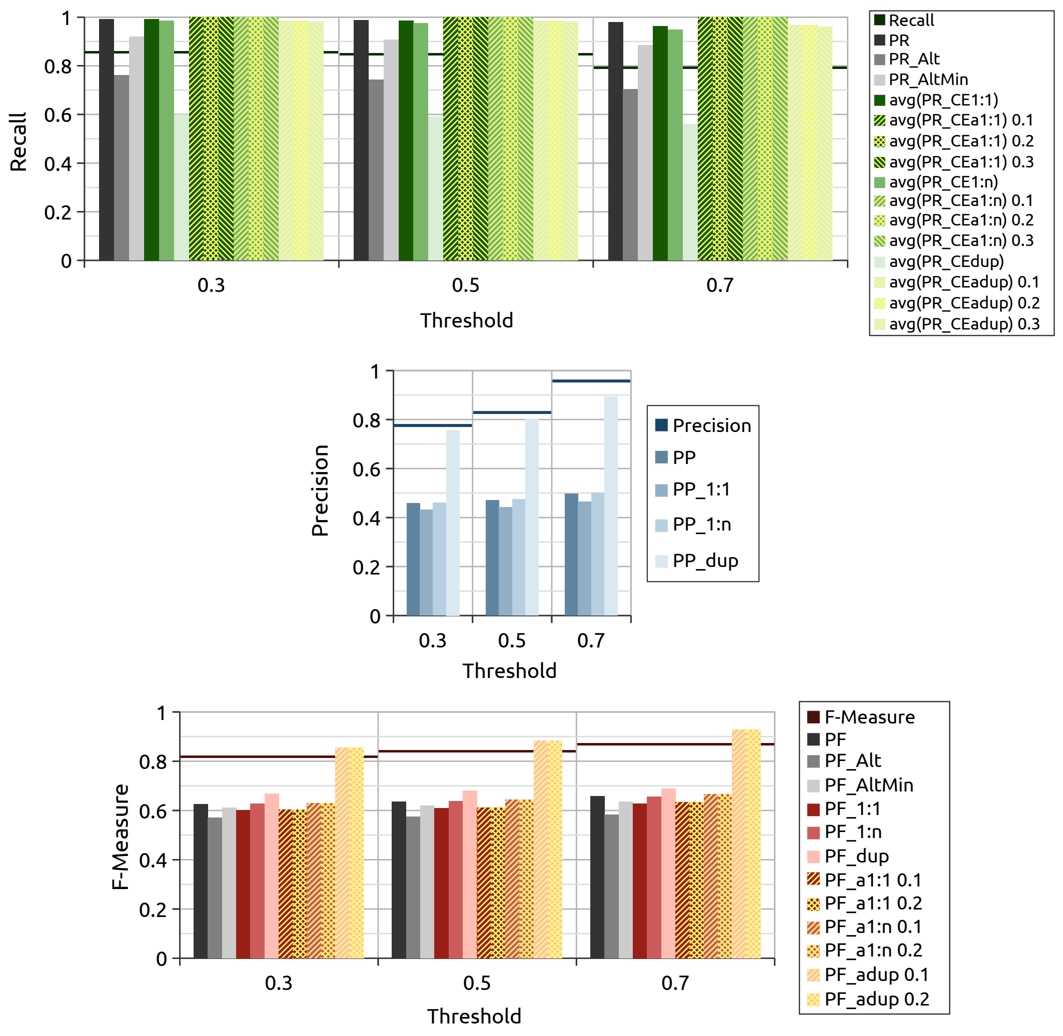


Figure 6.6: Results on Dexter dataset. Results based on ground truth are shown as horizontal lines.

ranging from around 0.33 (PR) up to around 0.4 (PR_{Alt}, PR_{AltMin}). t_{info} highly influences the quality of the cryptoset estimation, resulting in different overlaps so that the recall differs up to 0.2 using $t_{info} = 0.3$ compared to $t_{info} = 0.1$. Comparing the automatic approach with the manually defined private identifiers, the method achieves comparable results using $t_{info} = 0.1$. However, due to the manual effort of selecting appropriate attribute combinations, we suggest using an automatic method.

Considering the estimates of precision, the probabilistic method PP_{prob} achieves the best results for the applied thresholds with differences below 0.018. In contrast to the probabilistic method, the baseline approach PP as well as the adapted $PP_{1:1}$ and $PP_{1:n}$ lead to similar estimates being far away from the real precision, with a difference ranging from around 0.04 to around 0.09.

The combination of PP_{prob} and $PR_{CEaprob}$ to estimate precision and recall leads to the best F-measure approximation PF_{aprob} with an average difference of around 0.01 to the actual value. However, due to the harmonic mean, $PP_{1:1}$ and $PR_{CEa1:1}$ with $t_{info} = 0.2$ achieve similar results regarding the F-measure with a difference of 0.011 because of the neutralization effect of an overestimated precision and an underestimated recall.

The Dexter dataset consists of heterogeneous databases containing intra-source duplicates. Consequently, the previous methods for estimating the true positives will lead to inaccurate estimates since the number of true positives for each record is limited to one. Therefore, we apply our method based on the personalized PageRank (described in Section 6.4.2) to determine the precision estimate (PP_{dup}) that incorporates intra-source similarities as well. Due to the heterogeneity regarding various attributes and different types of quality issues, the manual selection of attributes for computing the private identifiers is a challenging task. Nevertheless, we use the brand and name attribute to determine the private identifiers, resulting in reasonable results regarding a small manual effort. The results for the Dexter dataset are shown in Figure 6.6.

Considering the estimations of recall using the cryptoset method in combination with the automatic private identifier generation, recall values are highly overestimated due to the underestimation of the overlap between the databases. In contrast, the manually defined cryptoset methods highly underestimate recall values due to an overestimated overlap. The overestimation indicates that the private identifiers are not specific enough to represent records in this dataset.

As expected, the baseline and the modified precision estimations utilizing the one-to-one assumption result in poor estimates being almost half of the actual precision. The precision estimate PP_{dup} achieves values that differ only slightly by at most 0.03 compared to the real precision for thresholds from 0.3 and 0.5. The accuracy of PP_{dup} is also reflected by the achieved F-measure estimates differing by at most 0.06.

Overall, our results show that the probabilistic and the personalized PageRank-based methods accurately estimate the number of true positives for clean and dirty databases. Moreover, the cryptoset-based approach improves the recall estimations significantly compared to the baseline approaches.

The automatic generation of private identifiers for the cryptoset-based estimation leads to comparable results as the application of manually defined rules. However, for very heterogeneous datasets such as the Dexter dataset, our method does not always lead to accurate results, showing the need for further work.

6.7 Conclusion

Typically, quality measures for record linkage results, such as precision and recall, are calculated based on ground truth data. However, in most real-world linkage scenarios, such ground truth data is not available. A manual inspection of linkage results is also often not feasible, in particular, due to privacy constraints when linking sensitive data.

In this chapter, we presented different approaches for estimating the quality of a linkage result given in the form of a similarity graph. We showed that our methods outperform existing approaches and lead to accurate estimates on different datasets. These estimates can be used in practical applications to identify suitable linkage methods and to optimize their parameters, such as the classification threshold. In future work, we plan to investigate clustering-based approaches for estimating the linkage quality in deduplication scenarios.

7

Evaluation of Hardening Techniques for PPRL

This chapter is based on [Fra+21]. Privacy-preserving record linkage aims at integrating person-related data from different sources while protecting the privacy of individuals by securely encoding and matching quasi-identifying attributes. For this purpose, Bloom-filter-based encodings have been frequently used in both research and practical applications. Simultaneously, however, weaknesses and attack scenarios were identified, emphasizing that Bloom filters are in principle susceptible to cryptanalysis.

To counteract such attacks, various encoding variants and tweaks, also known as hardening techniques, have been proposed. Usually, these techniques bear a trade-off between privacy (security) and the linkage quality outcome. Currently, a comprehensive evaluation of the suggested hardening methods is not available. In this chapter, we will therefore review and categorize available Bloom-filter-based encoding schemes and hardening techniques. We also comprehensively evaluate the approaches in terms of privacy (security) and linkage quality to assess their practicability and their effectiveness in counteracting attacks.

7.1 Motivation

Over the last years, numerous PPRL approaches have been published [Vat+17]. However, many approaches are not suited for real-world applications as they either are not able to sufficiently handle dirty data, i. e., erroneous, outdated, or missing values, or do not scale to larger datasets. More recent work mainly focuses on encoding techniques utilizing Bloom filters [Blo70] as an error-tolerant and privacy-preserving method to encode records containing sensitive information.

While Bloom-filter-based encodings have become the quasi-standard in PPRL approaches, several studies analyzed weaknesses and implemented successful attacks on Bloom filters [Kuz+11; Nie+14; KS14; Mit+16; Chr+18a; Chr+18b; Vid+20a; Vid+22; Vid+23]. In general, it was observed that Bloom filters carry a non-negligible re-identification risk because they are vulnerable to frequency-based cryptanalysis. In order to prevent such attacks, various Bloom filter hardening techniques were proposed [Sch15; Chr+18a]. Such techniques aim at reducing patterns and frequency information that can be obtained by analyzing the frequency of individual Bloom filters or (co-occurring) 1-bits.

Previous studies on Bloom filter hardening techniques only consider individual methods and do not analyze the effects of combining different approaches. Moreover, many of the proposed hardening techniques have received only limited evaluation on small synthetic datasets, making it hard to assess the possible effects on the linkage quality.

The aim of this work is to review hardening techniques proposed in the literature and to evaluate their effectiveness in terms of achieving high privacy (security) and linkage quality. In particular, we make the following contributions:

- We survey Bloom filter variants and hardening techniques that have been proposed for use in PPRL scenarios to allow secure encoding and matching of sensitive person-related data.
- We categorize existing hardening techniques to generalize the Bloom filter encoding process, and thus highlight the different possibilities for building tailored Bloom filter encodings that meet the privacy requirements of individual application scenarios.
- We explore additional variants of hardening techniques, in particular, salting utilizing blocking approaches and attribute-specific salting on groups of attributes.
- We propose and analyze measures that allow us to quantify the privacy properties of different Bloom filter variants.
- We comprehensively evaluate different Bloom filter variants and hardening techniques in terms of privacy (security) and linkage quality using two real-world datasets containing typical errors and inconsistencies.

The rest of this chapter is structured as follows. In Section 7.2, we describe different Bloom filter variants and hardening techniques. Then, we discuss and introduce new measures to assess the privacy/security properties of Bloom filter encodings in Section 7.3. In Section 7.4, we describe our experimental setup and then present a comprehensive evaluation of different hardening techniques in terms of linkage quality and privacy/security properties. Finally, we conclude in Section 7.6

7.2 Bloom Filter Variants and Hardening Methods

The use of Bloom filters [Blo70] for PPRL has been proposed by Schnell and colleagues [SBR09] and has become the quasi-standard for recent PPRL approaches in both research and real applications [Vat+17]. We discussed Bloom filters and their utilization in PPRL applications in Section 2.8.

In the following, we review different variations within the Bloom filter encoding process. In general, these variations will affect both the Bloom filter's privacy and similarity-preserving (matching) properties. Approaches that try to achieve a more uniform frequency distribution of individual Bloom filters or set bit positions are also known as hardening techniques as they are intended to make Bloom filter encodings more robust against cryptanalysis. An overview of these techniques is given in Table 7.1. We divide the approaches into three categories, namely into approaches that

(1) alter the way of selecting features from the records attributes values,

(2) modify the Bloom filter hashing process, and

(3) modify already existing Bloom filters by changing or aggregating bits.

In the following subsections, we will describe the approaches of each category.

7.2.1 Record Feature Selection

We will first focus on how features are selected from the record's attributes. In the encoding process, at first, all attribute values are pre-processed to bring them into the same format and to reduce data quality issues. After that, all linkage-relevant attributes, i. e., the quasi-identifiers of a person, are transformed into their respective feature sets. Such features are pieces of information that are usually obtained by segmenting the attribute values into chunks or tokens. This is necessary because instead of a binary decision for equality (true or false), approximate linkage is desired, resulting in similarity scores ranging from zero (completely different) to one (equal).

7.2.1.1 Standardization of Attribute Lengths

Quasi-identifiers, such as names and addresses, show high variation and skewness, leading to significant differences in the length of attribute values [FL83]. For instance, multiple given names, middle names, or compound surnames (e. g., 'Hans-Wilhelm Müller-Wohlfahrt') will lead to exceptionally long attribute values and consequently a comparatively large amount of 1-bits in the resulting Bloom filter. The same applies to very short names (e. g., 'Ed Lee') resulting in very few 1-bits in the Bloom filter.

Subject of modification	**Technique**	**Reference**	**Description**
Bloom filter input	Avoidance of padding	[Nie+14; Sch15]	No use of padded q-grams as BF input due to their higher frequency.
	Standardization of attribute lengths	[Nie+14; Sch15]	The length of attribute values is unified to avoid exceptionally short or long values.
	Feature Expansion	[SB18]	Expand actual q-gram set by randomly adding q-grams based on frequent co-occurrences.
Hashing mechanism	Increasing the number of hash functions (k)	[SBR09; SBR11]	Using more hash functions (k) while keeping the Bloom filter size (m) fixed will lead to more collisions and thus a higher number of features that are mapped to each position.
	Random hashing	[Nie+14]	Replacement for the double-hashing scheme [SBR09] which can be exploited in attacks [Nie+14].
	Attribute weighting	[Dur+14; Vat+14]	Record features are hashed with a different number of hash functions (k) depending on the weight of the attribute from which they were obtained.
	Salting	[SBR11; Nie+14]	Record features are hashed together with an additional attribute-specific and/or record-specific value.
Output Bloom filter	Balancing	[SB16a]	Each Bloom filter is concatenated with a negative copy of itself and then the underlying bits are permuted.
	XOR-folding	[SB16b]	Each Bloom filter is split into halves which are then combined using the bit-wise XOR-operation.
	Rule90	[SB18]	Each Bloom filter bit is replaced by the XOR-sum of its two neighboring bits.
	Windowing-based XORing	[RS20]	Sliding window approach where the Bloom filter bits in each window are XORed to obtain the hardened Bloom filter.
	Re-sampling	[RS20]	Two Bloom filter bits are selected randomly and XORed to set bits in the hardened Bloom filter.
	Linear Diffusion Layer	[AHS23]	Each bit in the hardened Bloom filter is a linear combination (XOR-sum) of secretly chosen bits from the original Bloom filter.
	Re-hashing	[Sch15]	Sliding window approach where the Bloom filter bits in each window are used to generate a new set of bits.
	Random noise	[SBR09; AGK12; Nie+14; Sch15; SB16a; VRC19]	Bloom filter bits are changed randomly.
	Autoencoders	[Chr+22]	Bloom filters are transformed into vectors of real numbers by using autoencoders.
	Fake injections	[KVC12]	Addition of artificial records and thus Bloom filters.

Table 7.1: Overview of Bloom filter hardening techniques.

By analyzing the number of 1-bits in a set of Bloom filters, an adversary can gain information on the length of encoded attribute values. To address this problem, the length of the quasi-identifiers should be standardized by sampling, deletion, or stretching of the attribute values [Sch15]. Stretching can be implemented by concatenating short attribute values with (rarely occurring) character sequences.

7.2.1.2 Segmentation Strategy

The standard segmentation strategy adopted from traditional record linkage is to transform all quasi-identifiers into their respective q-gram set. A q-gram set is a set of all consecutive character sequences of length q that can be built from the attribute's string value by using a sliding window approach. For instance, setting $q = 3$ the value 'Smith' will produce the q-gram set {'Smi', 'mit', 'ith'}. The idea behind building these q-gram sets is that they allow approximate string comparisons by calculating the number of q-grams two sets have in common. To directly obtain a similarity value, any set-based similarity measure, e. g., Jaccard coefficient, can be used.

The choice of q is important since it can affect the linkage quality. Usually, q is selected in the range $[1, 4]$ while most approaches set $q = 2$. In general, larger values for q are more sensitive to single character differences, e. g., the values 'Smith' and 'Smyth' will have two bigrams ($q = 2$), i. e., 'Sm' and 'th', but zero trigrams ($q = 3$) in common. However, choosing a larger q also increases the number of possible q-grams, e. g., for $q = 2$ at maximum $26^2 = 676$ while for $q = 3$ at maximum $26^3 = 17\,576$ are possible. Overall, larger values for q tend to be less error-tolerant and thus possibly lead to missing matches. On the other hand, larger q's are more distinctive and thus tend to reduce false positives.

As can be seen from the example above, for $q > 1$ each character will contribute to multiple q-grams except the first and last character. Thus, a common extension is to construct padded q-grams by surrounding each attribute value with $q - 1$ special characters at the beginning and the end. For our example, the padded q-gram set will be { '++S', '+Sm', 'Smi', 'mit', 'ith', 'th-', 'h--' }.

By using padded q-grams, strings with the same beginning and end but variations in the middle will reach larger similarity values, while strings with different beginning and end will produce lower similarity values compared to standard q-grams [Chr12b]. It is important to note, that padded q-grams are among the most frequent q-grams and thus can ease any frequency alignment attacks.

There are several other extensions for generating q-grams, two of which have been used in traditional record linkage, but so far not for PPRL: positional q-grams and skip-grams [Chr12b]. Positional q-grams add the information from which position the q-gram was obtained. For our running example, the positional q-gram set for $q = 3$ is { (‘Smi’,0), (‘mit’,1), (‘ith’,2) }. When determining the overlap between two positional q-gram sets, only the q-grams at the same position or within a specific range are considered. Positional q-grams will be more distinctive and thus tend to reduce false positives and even the frequency distribution. The idea of skip-grams is to not only consider consecutive characters but to skip one or multiple characters. Depending on the defined skip length, multiple skip-gram sets can be created and used in addition to the regular q-gram set.

So far, only a few alternatives to q-grams have been investigated. In [KGV18] and [VC16] the authors explore methods for handling numerical attribute values. Besides, arbitrary substrings of individual length or phonetic codes, such as Soundex [Chr12b], are possible approaches that can be used for feature extraction.

7.2.1.3 Feature Expansion

To obfuscate the actual encoded features, the generated feature (q-gram) set can be extended with other features (q-grams) according to a probabilistic language model. This approach was proposed by Schnell and Borgs in [SB18] where they use Markov chaining [Ros10] as a hardening technique for Bloom filters. The key idea is to encode each q-gram `q` with `c` additional q-grams that are randomly selected based on their probability to occur after `q` (co-occurrence). The parameter `c` is called the *chain length*, where larger values provide more privacy.

7.2.2 Modification of the Hashing Mechanism

After transforming all quasi-identifiers in their respective feature set, the features of each set are hashed into one record-level Bloom filter. As discussed in Section 2.8.2.2, we do not further consider field-level Bloom filters due to their vulnerabilities. For PPRL several modifications of the standard hashing process of Bloom filters have been proposed which we will discuss below.

7.2.2.1 Hash Functions

As described in Section 2.8.2.2, by default k independent (cryptographic) hash functions are used in conjunction with a private key $\mathcal{S}$ to prevent dictionary attacks. However, the authors of [SBR11] proposed the usage of the so-called double-hashing scheme. This scheme only uses two independent hash functions G_1, G_2 to implement the Bloom filters k hash functions. Each hash function is then defined as $H_i(x) = (G_1(x) + (i-1) \cdot G_2(x)) \bmod m, \forall i \in \{1, \ldots, k\}$.

The attacks described in [KS14; Nie+14] showed that this specific scheme can be successfully exploited. As a consequence, an alternative method, called random hashing, was proposed [Nie+14] that utilizes a pseudo-random number generator to calculate the hash values. Therefore, the random number generator is seeded with the private key $\mathcal{S}$ and the actual input of the hash function, i. e., a certain record feature. No attacks against this method are known at present.

7.2.2.2 Salting

Salting is a well-known technique in cryptography that is often used to safeguard passwords in databases [MT79]. The idea is to use an additional input, called salt, for the hash functions to flatten the frequency distribution.

Already in [SBR11] it is mentioned that a different cryptographic secret key $\mathcal{S}_a$ can be used for each record *attribute a*. We term such kind of key as attribute salt. By using this approach, the same feature will be mapped to different positions if it originates from different attributes. For instance, given the first name ‘thomas’ and the last name ‘smith’ the bigram ‘th’ will produce different positions.

This approach will smoothen the overall frequency distribution and also reduce false negatives since features from different attributes will not produce common 1-bits (except due to collision). However, the Bloom filter’s ability to match exchanged attributes, e. g., transposed first and middle name, is lost. If such errors occur repeatedly, this will lead to missing matches. As a compromise, we propose to define groups of attributes, where transpositions are expected. Then, the same key is used for each attribute from the same group. For instance, all name-related attributes (first name, middle name, last name) could form a group.

Another salting variant is proposed in [Nie+14], where *for each record* a specific salt is selected and then used as a key for the Bloom filters k hash functions. Therefore, we term such keys as record salt, since they depend on a specific record.

Records salts can also be combined with the aforementioned attribute salts. Only if the record salt is identical for two records, the same feature (q-gram) will set the same bit positions in the corresponding Bloom filters. However, if the record salts are different, then the probability that the same bit positions are set in the corresponding Bloom filters is very low. Thus, if the attributes (from which the record salts are extracted) contain errors, this will lead to many false negatives. For this reason, only commonly available, stable, and small segments of quasi-identifiers, such as year of birth, are suitable as salting keys. Consequently, this technique is only an option in PPRL scenarios where the attributes used for salting are guaranteed to be of very high quality, which might rarely be the case in practice.

To reduce the aforementioned problem of salting with record-specific keys, we propose to generate the salt by utilizing blocking approaches. Blocking [Chr12b] is an essential technique in (privacy-preserving) record linkage to overcome the quadratic complexity of the linkage process since in general each record must be compared to each record of another source.

The idea of blocking is to partition records into small blocks and then to compare only records within the same block to reduce the number of record pair comparisons (see Section 2.6.3). For this purpose, one or more blocking keys are defined, where each blocking key represents a specific, potentially complex criterion that records must meet to be considered as potential matches. For example, the combination of the first letter of the first and last name and the year of birth might be used as a blocking key.

If the attributes used for blocking contain errors, then the blocking key will also be affected, leading to many false negatives, in particular if the blocking key is very restrictive. Hence, often multiple blocking keys are used to increase the probability for records to share at least one blocking key. However, this will lead to duplicate candidates since very similar records will share most blocking keys.

The challenge of both, salting and blocking, is to select a key that is as specific as possible (to increase privacy, or to reduce the number of record pair comparisons) and at the same time not prone to errors. For record-dependent salting, only the use of attribute segments was suggested. In contrast, for blocking more sophisticated approaches have been considered, in particular using phonetic codes, e. g., Soundex, or locality-sensitive hashing schemes, e. g., MinHash [Bro97].

7.2.2.3 Dependency-based Hashing

In traditional record linkage, sophisticated classification models are used to decide whether a record pair represents a match or a non-match. Often, these models deploy an attribute-wise or rule-based classification, considering the discriminatory power and expected error rate of the attributes [Chr12b].

In contrast, PPRL approaches based on record-level Bloom filters only apply classification based on a single similarity threshold since all attribute values are aggregated (encoded) in a single Bloom filter. However, as discussed in Section 2.8.2.1, the record-level Bloom filter variant proposed in [Dur+14] also considers the weight of attributes by selecting more bits from the field-level Bloom filters of attributes with higher weights. In [Vat+14] the authors proposed an extension to the approach of [SBR11] to allow attribute weighting. While still only a single Bloom filter is constructed, a different number of hash functions is selected for different attributes according to their weights.

Consequently, the higher the weight of an attribute, the more hash functions will be used and thus the more bits the attribute will set in the Bloom filter.

The idea of varying the number of hash functions can be generalized to dependency-based hashing. For instance, not only the weights of attributes can be considered but also the frequency of input features or their position within the attribute value (positional q-grams).

7.2.3 Bloom Filter Modifications

While the methods described so far modify the way Bloom filters are created, the following approaches are applied directly on the obtained Bloom filters (bit vectors).

7.2.3.1 Balanced Bloom Filters

Balanced Bloom filters were proposed in [SB16a] for achieving a constant Hamming weight among all Bloom filters. A constant Hamming weight should make the elimination of infrequent patterns more difficult. Balanced Bloom filters are constructed by concatenating a Bloom filter with a negative copy of itself and then permuting the underlying bits. Since the size of the Bloom filters is doubled, balanced Bloom filters will increase computing time and required memory for Bloom filter comparisons.

Example: Using balancing on Bloom filter [10011001] will give [10011001]$\odot$[01100110] = [1001100101100110] as output before applying the permutation.

7.2.3.2 XOR-Folding

XOR-folding of bit vectors is a method originating from chemo-informatics to speed up database queries [Che+05]. In [SB16b] the authors adopted this idea for Bloom-filter-based PPRL for preventing bit pattern attacks. To apply the XOR-folding a Bloom filter is split into halves and then the two halves are combined by the XOR-operation. The folding process may be repeated several times. Since the size of the Bloom filters is halved, XOR-folding will decrease computing time and required memory for Bloom filter comparisons. The initial evaluation in [SB16b] using unrealistic datasets with full overlap and low error rates shows that one-time folding does not significantly affect linkage quality. However, folding multiple times drastically increases the number of false positives.

Example: Applying XOR-folding on Bloom filter [11000101] will give [1100] $\oplus$ [0101] = [1001] as output.

7.2.3.3 Rule90

In [SB18] the use of the so-called Rule90 was suggested to increase the resistance of Bloom filters against bit-pattern-based attacks. The Rule90 is also based on the XOR-operation which is applied on the two neighboring values of each Bloom filter bit. Consequently, there are 8 possible combinations (patterns), which are listed in Table 7.2. So each bit b_i $(0 \leq i \leq m-1)$ is replaced by the result of XOR-ing the two adjacent bits at positions $(i-1) \bmod m$ and $(i+1) \bmod m$. By using the modulo function, the first and the last bit are treated as if they were adjacent.

Pattern	111	110	101	100	011	010	001	000
New Bit Value	0	1	0	1	1	0	1	0

Table 7.2: Transformation rules for Rule90.

Example: Applying Rule90 to the Bloom filter [11000101] will lead to the following patterns 1**1**1, 1**1**0, 1**0**0, 0**0**0, 0**0**1, 0**1**0, 1**0**1, 0**1**1 where the middle bit corresponds to the bit at position $i \in \{0, m-1\}$ of the Bloom filter. After applying the transformation rules (Table 7.2) we obtain [01101001].

7.2.3.4 Windowing based XORing (WXOR)

This hardening technique was proposed in [RS20] and applies a sliding window approach to select bits from the original Bloom filter, which are then combined using the bit-wise XOR-operation to obtain the hardened Bloom filter. This approach uses two windows w_1 and w_2, both of size w, that are iteratively moved over the original Bloom filter Bf of length m. In each iteration, window w_1 is positioned at bit position i $(0 \leq i < m-w)$, and window w_2 is positioned at bit position $(i+1) \bmod m$. Then, the bits that are covered from each window are extracted, i. e., $w_1 = \pi_i^{i+w}(\text{Bf})$ and $w_2 = \pi_{(i+1) \bmod m}^{(i+1+w) \bmod m}(\text{Bf})$, and combined using the bit-wise XOR-operation, i. e., $w_1 \oplus w_2$. Finally, the resulting XORed bit pattern is used to update the Bloom filter bits according to window w_1. Then, both windows are moved forward one bit and these steps are repeated until the complete Bloom filter is processed. In [RS20], the authors argue that a larger window size w will make the hardening process more efficient, but will also yield less accurate similarity calculations.

Example: Applying the WXOR approach on Bloom filter [11000101] setting $w = 4$ will lead to windows $w_1 = [1100]$ and $w_2 = [1000]$ in the first iteration. Since $w_1 \oplus w_2 = [0100]$, the hardened Bloom filter after the first iteration is [**0100**0101]. In the second iteration we obtain windows $w_1 = [1000]$ and $w_2 = [0001]$ with $w_1 \oplus w_2 = [1001]$ leading to Bloom filter [0**1001**101]. After the third iteration we get Bloom filter [01**0101**01] and after the fourth iteration Bloom filter [010**1111**1]. Finally, we obtain Bloom filter [0101**0001**].

7.2.3.5 Re-Sampling based XORing

Another hardening technique that is based on the XOR-operation is the re-sampling approach proposed by Ranbaduge and Schnell in [RS20]. This approach randomly samples bits from the original Bloom filter Bf and applies the XOR-operation on these bits. To obtain a hardened Bloom filter Bf_H of length m, the approach applies m sampling steps. In each sampling step ι, with $0 \leq \iota < m$, two bits b_i and b_j are randomly selected (with replacement) from the original Bloom filter, with $0 \leq i, j < m$. Then, the two bits are XORed and the resulting bit value is used to set the bit at position ι in the hardened Bloom filter, i. e., $\mathrm{Bf}_H[\iota] = b_i \oplus b_j$.

7.2.3.6 Linear Diffusion Layer

Recently, Armknecht et al. proposed in [AHS23] to extend Bloom filters by a linear diffusion layer. The core idea is that each output bit of the encoding is a linear combination (XOR-sum) of secretly chosen Bloom filter bits. In a theoretical and experimental analysis, their approach showed improved security and similar linkage quality compared to standard Bloom filters. This approach is very similar to the re-sampling based XORing proposed in [RS20]. In the re-sampling approach, each bit in the hardened Bloom filter is built by combining (XORing) two randomly selected bits of the original Bloom filter. In this approach, in contrast, the number of bits to be combined can be chosen arbitrarily. Besides, the approach by Armknecht et al. uses a greedy approach to ensure that each bit of the original Bloom filter is used on average in the same number of linear combinations.

7.2.3.7 Re-Hashing

The idea of re-hashing [Sch15] is to use consecutive bits of a Bloom filter to generate a new bit vector. Therefore, a window of width w bits is moved over the Bloom filter, where in each step the window slides forward s positions (step size). At first, a new bit vector v of size m' is allocated. Then, the w bits, which are currently covered by the window, are represented as an integer value. The integer value is then used in combination with a secret key as input for a random number generator. With that, r new integer values are generated with replacement, each in the range $[0, m' - 1]$. Finally, the bits at these r positions are set to one in the bit vector v. The evaluation in [SB16a] uses unrealistic datasets (full overlap, no errors) and shows no clear trend. However, this technique is highly dependent on the choice of the parameters m', w, s and r as well as on the original Bloom filters, in particular the average fill factor (amount of 1-bits).

Example: Given the Bloom filter [11000101] and setting $w = 4, s = 2$ will lead to three windows, namely $w_1 = [1100], w_2 = [0001], w_3 = [0101]$. By transforming the bits in each window into an integer value, we obtain the seeds $12, 1$, and 5. Setting $r = 2$ the random number generator might generate the positions $(4, 2), (2, 5), (8, 6)$ for the respective seeds which finally results in the bit vector [001011101].

7.2.3.8 Random Noise

In order to make the frequency distribution of Bloom filters more uniform, random noise can be added to the Bloom filters [Sch15; SB16a]. Trivial options are to randomly set bits to one/zero or to flip bits (complement). Additionally, the amount of random noise can depend on the frequency of mapped record features. For instance, for Bloom filters containing frequent q-grams, more noise can be added. In [AGK12], an ϵ-differential private Bloom filter variant, called BLoom-and-flIP (BLIP), based on permanent randomized response is proposed. Each bit position $b_i, \forall i \in \{0, \ldots, m-1\}$ is assigned a new value b'_i based on a flip probability f. In the original approach the new value for each bit position b_i is defined as:

$$b'_i = \begin{cases} 1 & \text{if } b_i = 0 \text{ with probability } f \\ 0 & \text{if } b_i = 1 \text{ with probability } f \\ b_i & \text{with probability } 1 - f. \end{cases} \tag{7.1}$$

Schnell and Borgs [SB16a] first applied BLIP in the context of PPRL, but using a slightly different approach:

$$b'_i = \begin{cases} 1 & \text{with probability } \frac{1}{2}f \\ 0 & \text{with probability } \frac{1}{2}f \\ b_i & \text{with probability } 1 - f. \end{cases} \tag{7.2}$$

The difference between the two approaches is that in the second approach (Equation 7.2) the bits are flipped independently of their original state. Suppose, for example, for a Bloom filter length of $m = 1024$ the flip probability is set to $f = 0.125$. Using Equation 7.1, 128 randomly selected bits are flipped, while 896 bits remain unchanged. By using Equation 7.2 instead, 128 randomly selected bits are set to 0 or 1, both with equal probability. It is therefore very likely that less than 128 bits will be changed. As a consequence, if a Bloom filter has less than 50% 1-bits, then it will have more than 50% 1-bits after applying Equation 7.1. A higher number of 1-bits increases the similarities between hardened Bloom filters, and therefore can potentially lead to more false positive matches.

Using BLIP as a hardening technique is expected to reduce the linkage quality as Bloom filter bits are flipped randomly. To overcome this weakness, Vaiwsri et al. [VRC19] proposed a hardening technique based on reference values as an extension of the BLIP approach. Their approach uses reference values from a publicly available database to determine specific bit positions to be flipped. The idea is that similar plaintext values will likely have similar sets of reference values. Therefore, Bloom filters encoding similar plaintext values will be randomized similarly. Their evaluation shows that their approach yields improved linkage quality compared to the original BLIP approaches, but requires more computational effort.

7.2.3.9 Autoencoder

To overcome the vulnerability of Bloom filters to frequency attacks, Christen et al. [Chr+22] proposed to use a second layer of encryption that applies a continuous transformation φ from Bloom filters (bit vectors) into vectors of real numbers, i. e., $\varphi : \mathbb{B}^m \longrightarrow \mathbb{R}^{m'}$ where $m < m'$. The transformation is based on autoencoders [Kra91; Kra92] which are neural networks that can generate lower-dimensional representations with a small information loss for high-dimensional inputs. The reduction of dimensions results in information loss, which is intended to hide potentially vulnerable bit patterns in the Bloom filters. Moreover, existing attacks on frequent 1-bit patterns are not applicable on vectors of real numbers.

7.2.3.10 Fake Injections

Another option to modify the frequency distribution of Bloom filters is to add artificial records or attribute values [KVC12]. By inserting random strings containing rarely occurring q-grams, the overall frequency distribution will become more uniform, making any frequency alignment less accurate. The drawback of fake records is that they produce computational overhead in the matching process. Moreover, it is possible that a fake record will match with another record by chance. Thus, after the linkage, fake records need to be winnowed.

7.3 Bloom Filter Privacy Measures

Several attacks on Bloom filters have been described in the literature (see Section 2.8.2.2), which show that Bloom filters carry the risk of re-identification of attribute values and even complete records. Currently, the privacy of encoding schemes based on Bloom filters is mainly evaluated by simulating attacks and inspecting their results, i. e., the more attribute values and records can be correctly re-identified by an attack, the lower the assumed degree of privacy of the encoding scheme. However, this way of measuring privacy strongly depends on the used attacks, their assumptions, and the used reference dataset. Besides, only a few studies investigated evaluation measures for privacy [Vat+17]. These measures either calculate the probability of suspicion [Vat+14] or are based on entropy and information gain between masked and unmasked data [SB16a]. The disadvantage of these measures is that they strongly depend on the reference dataset used. In the following, we therefore propose privacy measures that solely depend on a Bloom filter dataset.

To evaluate the disclosure risk of Bloom filter encoding schemes, we propose to analyze the frequency distribution of the Bloom filter 1-bits. As described in Section 2.8.2.2, attacks on Bloom filters mostly try to align the frequencies of frequent (co-occurring) bit patterns to frequent (co-occurring) record features (q-grams). Thus, the more uniform the frequency distribution of 1-bits is, the less likely an attack will be successful.

To measure the uniformity of the bit frequency distribution of a Bloom filter dataset $\mathcal{B}$, we calculate for each Bloom filter bit position (column) $0 \leq i < m$ the number of 1-bits, given as $\mathbf{c}_i = \sum_{\mathrm{Bf}\in\mathcal{B}} \mathrm{Bf}[i]$, where $\mathrm{Bf}[i]$ returns the Bloom filter's bit value at position i. The total number of 1-bits is then $\mathbf{b} = \sum_{i=0}^{m-1} c_i = \sum_{\mathrm{Bf}\in\mathcal{B}} ||\mathrm{Bf}||_1$ where $||\mathrm{Bf}||_1$ denotes the cardinality of a Bloom filter (see Section 2.8). We can then calculate for each column its share of the total number of 1-bits, i. e., $\mathbf{p}_i = c_i/b$. Ideally, for a perfect uniform bit distribution, c_i will be close to b/m for all $i \in \{0, m-1\}$.

In mathematics and economics, there are several measures that allow assessment of the (non-) uniformity of a certain distribution. Consequently, we are adapting the most promising of these measures to our problem. At first, we consider the Shannon entropy since uniform probability will yield maximum entropy. The Shannon entropy is defined as

$$\mathcal{H}(\mathcal{B}) = -\sum_{i=0}^{m-1} p_i \cdot \log_2(p_i) \tag{7.3}$$

where the maximum entropy is given as

$$\mathcal{H}_{max}(\mathcal{B}) = \log_2(m) \tag{7.4}$$

We define the normalized Shannon entropy ranging from 0 (high entropy - close to uniform) to 1 (low entropy) as

$$\widetilde{\mathcal{H}}(\mathcal{B}) = 1 - \frac{\mathcal{H}(\mathcal{B})}{\mathcal{H}_{max}(\mathcal{B})} \tag{7.5}$$

Next, we consider the Gini coefficient [CV12; Gas72], which is well-known in economics as a measure of income inequality. The Gini coefficient can range from 0 (perfect equality

– all values are the same) to 1 (maximal inequality – one column has all 1-bits and all others have only 0-bits) and is defined as

$$G(\mathcal{B}) = \frac{\sum_{i=0}^{m-1} \sum_{j=0}^{m-1} |c_i - c_j|}{2m \cdot b} \tag{7.6}$$

Moreover, we calculate the Jensen-Shannon divergence (JSD) [FT04] which is a measure of similarity between two probability distributions. The Jensen-Shannon divergence is based on the Kullback-Leibler divergence (KLD) [KL51], but has better properties for our application: In contrast to the Kullback-Leibler divergence, the Jensen-Shannon divergence is a symmetric measure. In fact, the square root of the Jensen-Shannon divergence is a metric known as Jensen-Shannon distance (D_{JS}) [ES03]. For discrete probability distributions P and Q defined on the same probability space, the Jensen-Shannon divergence is defined as

$$\text{JSD}(P \,||\, Q) = \frac{1}{2}\,\text{KLD}(P \,||\, M) + \frac{1}{2}\,\text{KLD}(Q \,||\, M) \tag{7.7}$$

where

$$\text{KLD}(P \,||\, Q) = \sum_{s \in \mathcal{S}} P(s) \cdot \log_2 \left(\frac{P(s)}{Q(s)} \right) \tag{7.8}$$

and

$$M = \frac{1}{2}(P + Q). \tag{7.9}$$

The Jensen-Shannon divergence also provides scores between 0 (identical) to 1 (maximal distance). Since we want to measure the uniformity of the bit frequency distribution of a Bloom filter dataset $\mathcal{B}$, we calculate the Jensen-Shannon distance as

$$D_{\text{JS}}(\mathcal{B}) = \sqrt{\text{JSD}(\mathcal{B})} \tag{7.10}$$

where

$$\text{JSD}(\mathcal{B}) = \frac{1}{2} \left(\sum_{i=0}^{m-1} \frac{1}{m} \cdot \log_2 \left(\frac{\frac{1}{m}}{\frac{1}{2} \cdot (p_i + \frac{1}{m})} \right) \right) + \frac{1}{2} \left(\sum_{i=0}^{m-1} p_i \cdot \log_2 \left(\frac{p_i}{\frac{1}{2} \cdot (p_i + \frac{1}{m})} \right) \right)$$

Finally, we measure how many different record features (q-grams) are mapped to each bit position, which we denote as feature ratio (fr). The more features are mapped to each position, the harder becomes a one-to-one assignment between bit positions and record features, which will limit the accuracy of an attack.

7.4 Evaluation Setup

Before presenting the evaluation results, we describe our experimental setup as well as the datasets and metrics we use.

7.4.1 PPRL Setup

We implement the PPRL process as a three-party protocol that requires a trusted linkage unit [VC16]. Furthermore, we set the Bloom filter length $m = 1024$. To overcome the quadratic complexity of linkage, we use LSH-based blocking based on the Hamming distance [FSR18]. We empirically determined the necessary parameters leading to high efficiency and effectiveness. As a result, we set $\Psi = 16$ (LSH key length) and $\Lambda = 30$ (number of LSH keys) as default. Finally, we calculate the Jaccard coefficient to determine the similarity of candidate record pairs. We classify every record pair with a similarity equal to or greater than t as a match. Finally, we apply a one-to-one matching constraint, i. e., a record of one source can match to at maximum one record of another source, utilizing a symmetric best match approach (see Section 5.5.1).

7.4.2 Datasets

For evaluation, we use two real datasets that are obtained from the North Carolina voter registration database (NCVR) (`https://www.ncsbe.gov/`) and the Ohio voter files (OHVF) (`https://www.ohiosos.gov/`). For both datasets, we select subsets of two snapshots at different points in time. Due to the time difference, records contain errors and inconsistencies, e. g., due to marriages/divorces or moves.

Please note that we do not insert artificial errors or otherwise modify the records. We only determine how many attributes of a record have changed and use this information to construct subsets with a specific amount of records containing errors.

An overview of all relevant dataset characteristics is given in Table 7.3. Each dataset consists of two subsets, S_A and S_B, to be linked with each other. The two subsets are associated with two data owners (or sources) A and B, respectively.

7.4.3 Metrics

To assess the linkage quality we determine recall, precision, and F-measure (Section 2.6.7.1). To assess the privacy (security) of the different Bloom-filter-based encoding schemes, we analyze the frequency distribution of the Bloom filter's 1-bits in order

Characteristic	**Dataset**	
	N	**O**
Type	Real (NCVR)	Real (OHVF)
$\lvert S_A \rvert$	50 000	120 000
$\lvert S_B \rvert$	50 000	80 000
$\lvert S_A \cap S_B \rvert$	10 000	40 000
Attributes	First name Middle name Last name Year of birth (YOB) City	First name Middle name Last name Date of birth (BD) City
|Errors|/record	0 (40%) 1 (30%) 2 (20%) 3 (10%)	0 (37.5%) 1 (55%) 2 (6.875%) 3 (0.625%)

Table 7.3: Characteristics of datasets N and O used for the evaluation of hardening techniques.

to determine the normalized Shannon entropy, the Gini coefficient, and the Jensen-Shannon distance (see Section 7.3). Furthermore, we calculate the feature ratio (fr) that determines how many record features are mapped on average to each bit position.

7.4.4 Q-Gram Frequencies

Before we begin our evaluation on Bloom filters, we analyze the plaintext frequencies of our datasets N and O as well as the complete NCVR and OHVF datasets. At first, we measure the relative bigram frequencies as shown in Figure 7.1. What can be seen in this figure is the high dispersion of bigrams. For the complete NCVR and OHVF the non-uniformity is a bit higher than in our datasets, which is mainly due to the larger number of infrequent bigrams. Since our datasets are only subsets from the respective voter registrations (NCVR/OHVF), some of these rare bigrams do simply not occur in our dataset subsets.

In Figure 7.2, we plot the Lorenz curves [Gas72] for the plaintext datasets as well as Bloom filters (see Section 7.5). These diagrams again illustrate the high dispersion for the plaintext values. Comparing bigrams and trigrams, it can be seen that the non-uniformity for trigrams is even higher than for bigrams.

Our observations are confirmed by our uniformity (privacy) measures (see Section 7.3) which we calculate for the datasets as listed in Table 7.4. We use these values as a baseline for the Bloom filter privacy analysis. The closer the values for a set of Bloom

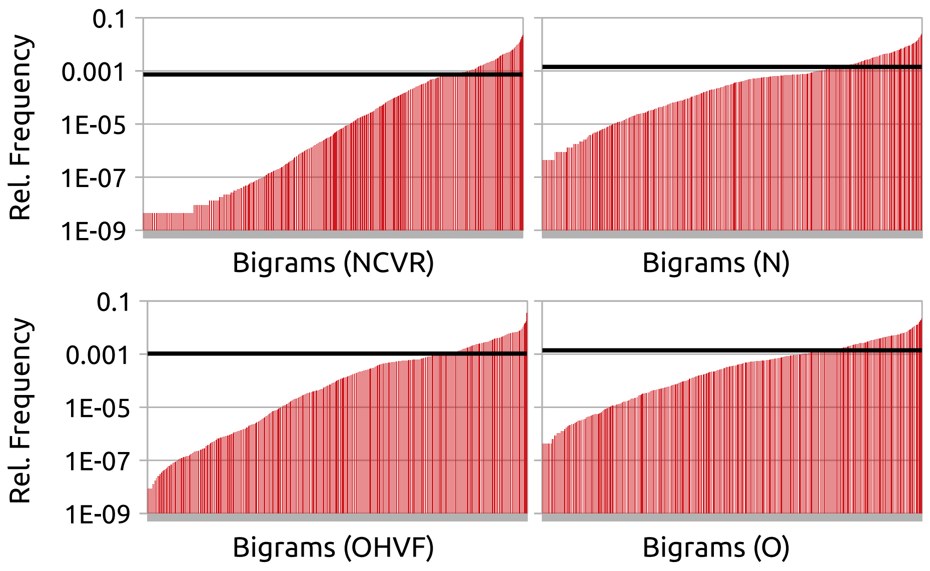


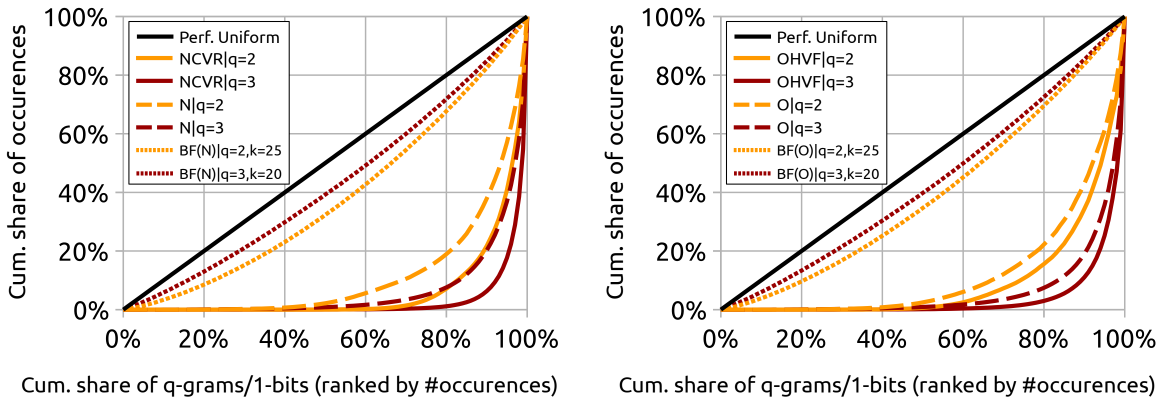
Figure 7.1: Relative bigram frequencies for datasets N and O.

Measure	Datasets							
	Bigrams				Trigrams			
	N	NCVR	O	OHVF	N	NCVR	O	OHVF
$\widetilde{\mathcal{H}}$	0.1848	0.2497	0.1670	0.2027	0.2151	0.2781	0.2142	0.2491
G	0.7709	0.8728	0.7466	0.8047	0.8705	0.9425	0.8729	0.9189
D_{JS}	0.6315	0.7516	0.6107	0.6724	0.7340	0.8392	0.7362	0.8007

Table 7.4: Analysis of q-gram frequency distribution.

filters are to these values, the more likely a frequency alignment will be successful. On the other hand, the larger the difference between the values for plaintext and Bloom filters, the better the Bloom filters can hide the plaintext frequencies and thus the less likely a successful frequency alignment becomes.

Comparing our three measures, it can be seen that the values for the normalized Shannon entropy ($\widetilde{\mathcal{H}}$) are much lower than the values for the Gini coefficient (G) and the Jensen-Shannon distance (D_{JS}). However, all measures clearly indicate the differences in the frequency distribution of bigrams and trigrams. Comparing both datasets, it can be seen that the non-uniformity of bi- and trigrams is slightly higher for the NCVR than for the OHVF dataset.



(a) NCVR (b) OHVF

Figure 7.2: Comparison of Lorenz curves for plaintext and encoded records (Bloom filters).

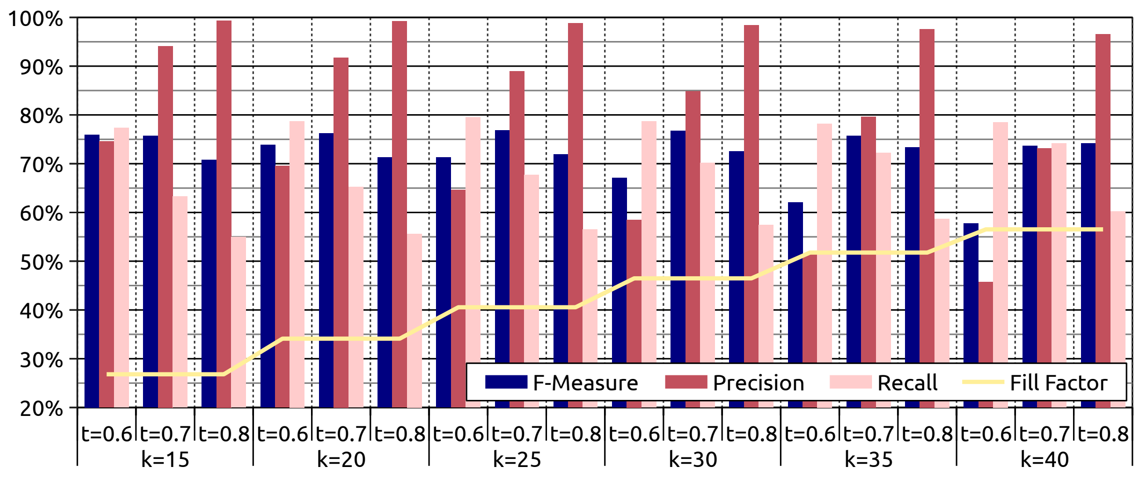
7.5 Results and Discussion

In this section, we evaluate various Bloom filter variants and hardening techniques in terms of linkage quality and privacy (security).

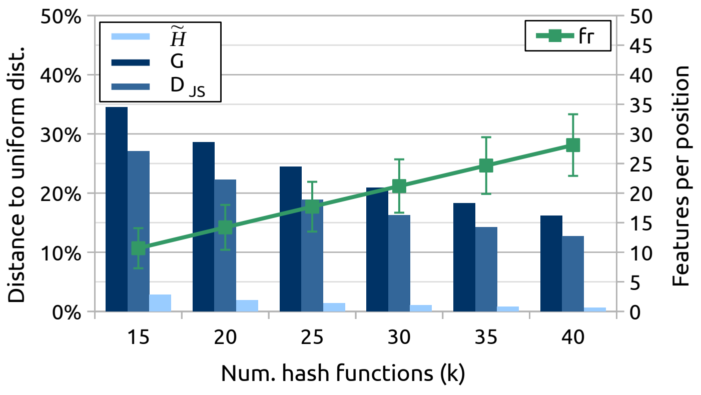
7.5.1 Hash Functions and Fill Factor

In the following, we evaluate the linkage quality outcome and the privacy properties of basic Bloom filters by inspecting the frequency distribution of the Bloom filter's 1-bits compared to the q-gram frequencies. At first, we vary the number of hash functions (k), selecting $k \in \{15, 20, \ldots, 40\}$ and bigrams ($q = 2$), to adjust the fill factor (amount of 1-bits) of the Bloom filters. The results for dataset N are depicted in Figure 7.3a.

The results show, that for the high similarity thresholds of $t = 0.8$, all configurations achieve high precision $\geq$96.58%, but low recall $\leq$60.19%, leading to a max. F-measure of 74.16%. For lower similarity thresholds ($t = \{0.7, 0.6\}$), precision is reduced drastically the more hash functions are used. For instance, setting $t = 0.6$ and $k = 15$, the highest precision of 75.93% is achieved, while for $k = 40$ the precision is only 45.74%. In contrast, the higher the number of hash functions, the higher the recall. For instance, setting $t = 0.6$ and $k = 15$, the recall is 73.28%, while for $k = 40$ it increases to 78.51%. However, the impact on precision is much higher (difference of around 34%) than on recall (difference of around 5%). Overall, the configuration with $t = 0.7$ and $k = 25$ achieves the best F-measure of 76.89%. However, the other configurations except those with a fill factor over 50% ($k \in \{35, 40\}$) achieve only slightly less F-measure. When averaging precision and recall for each k over all thresholds, the configurations with



(a) Quality



(b) Privacy

Figure 7.3: Evaluation of standard Bloom filters using bigrams without padding for varying number of hash functions (k) on dataset N.

$k \leq 25$ achieve a mean F-measure of over 75%, while for larger k it declines from around 74% for $k = 30$ to around 71% for $k = 40$.

Next, we analyze our privacy measures, which are depicted in Figure 7.3b. The figure shows that the more hash functions are used (and thus the higher the fill factor of the Bloom filters) the higher the average number of features that are mapped to each bit position. Even for the lowest number of hash functions, on average around 10 different bigrams are mapped to each individual bit position. Compared to the plaintext frequencies (see Table 7.4), we see that basic Bloom filters have a significantly more uniform frequency distribution than the original plaintext dataset. For instance, using $k = 25$ hash functions, we obtain a Gini coefficient of 0.2443 and a Jensen-Shannon distance of 0.1891 compared to 0.7709 and 0.6315 for the non-encoded dataset. Although for the Shannon entropy also a difference is visible, i. e., from 0.1848 for plaintext to 0.0137 for Bloom filters setting $k = 25$, the values are in general much closer to zero and thus less intuitive to compare. As a consequence, in the following, we will focus on the other two privacy measures. Finally, the privacy measures indicate, that the more hash functions are used, the closer the 1-bit distribution will get to uniform. However, the effect is not linear, such that the privacy gain is continuously getting lower, in particular for $k \geq 30$.

7.5.2 Choice of q and the Impact of Padding

In Table 7.5 we compare the linkage quality of Bloom filters using different configurations for $q \in \{2, 3\}$ and padding for dataset N.

Without the use of padding the best configuration for bigrams, i. e., $k = 25$ and $t = 0.7$, achieves a slightly less F-measure of 76.89% than the best configuration for trigrams, i. e., $k = 20$ and $t = 0.6$, of 77.45%. However, considering the mean over all configurations, using bigrams achieves a slightly higher F-measure of 75.07% compared to 74.69% for trigrams. Surprisingly, using trigrams results in an overall higher recall but lower precision if we average the results over all configurations.

Moreover, the use of padding leads to a higher linkage quality, i. e., the best configurations for bigrams achieve an F-measure of 82.75% while for trigrams even 85.10% is attained. Averaged over all configurations, by using padding, recall is increased about 5% for bigrams and around 2.8% for trigrams. Interestingly, also precision is increased by around 2.11% for bigrams and around 4.02% for trigrams. Thus, for both bigrams and trigrams, the mean F-measure can be increased by padding by more than 3%.

We repeat the experiments on dataset O and report the best configurations in Table 7.6. The results confirm our previous observation that trigrams with padding lead to the highest linkage quality. Here, the best configuration using trigrams and padding outperforms that with bigrams and padding even slightly more than for dataset N, i. e., F-measure increases 2.35% for N and 4.49% for O.

Figure 7.4 shows our privacy measures for the best configuration in each group. In general, the use of bigrams leads to a less uniform distribution of 1-bits and thus lower privacy. Also, the use of padding leads to a higher dispersion of the Bloom filters 1-bits. However, even the worst configuration, namely bigrams using padding, leads to a significantly lower Gini coefficient compared to the plaintext datasets. For N, for instance, the Gini coefficient is reduced from 0.7709 to 0.3801 (see Table 7.4 and Figure 7.2). Also, the Jensen-Shannon distance reduces drastically, e. g., for dataset N from 0.6315 for the plaintext dataset to 0.3045 for the Bloom filter dataset using bigrams with padding. In contrast, the use of trigrams leads to a more even distribution of 1-bits, so that despite using padding, a slightly more uniform frequency distribution is achieved than with bigrams and without using padding.

q	Pad.	k	t	Recall [%]	Precision [%]	F-Measure [%]	Mean Recall [%]	Mean Precision [%]	Mean F-Measure [%]
2	No	15	0.6	77.34	74.58	75.93	85.21	67.09	75.07
			0.7	63.27	94.12	75.67			
			0.8	54.96	**99.36**	70.77			
		20	0.6	78.71	69.54	73.84			
			0.7	65.21	91.75	76.24			
			0.8	55.64	99.19	71.29			
		25	0.6	**79.51**	64.58	71.27			
			0.7	67.71	88.95	**76.89**			
			0.8	56.50	98.81	71.89			
		30	0.6	78.68	58.51	67.11			
			0.7	70.14	84.80	76.78			
			0.8	57.43	98.37	72.52			
	Yes	10	0.6	81.48	84.06	**82.75**	90.21	**69.20**	78.32
			0.7	64.56	97.71	77.75			
			0.8	55.08	**99.67**	70.95			
		15	0.6	**83.77**	76.18	79.79			
			0.7	68.46	96.17	79.98			
			0.8	56.02	99.53	71.69			
		20	0.6	83.66	66.15	73.88			
			0.7	72.33	93.07	81.40			
			0.8	57.42	99.37	72.78			
3	No	15	0.6	69.83	86.85	77.42	91.12	63.28	74.69
			0.7	59.49	96.99	73.75			
			0.8	53.71	**99.57**	69.78			
		20	0.6	72.30	83.38	**77.45**			
			0.7	60.71	96.19	74.44			
			0.8	54.30	99.54	70.27			
		25	0.6	73.53	79.83	76.55			
			0.7	62.08	94.93	75.07			
			0.8	54.76	99.41	70.62			
		30	0.6	74.18	75.71	74.94			
			0.7	63.64	93.34	75.68			
			0.8	55.30	99.15	71.00			
		35	0.6	**74.25**	71.66	72.93			
			0.7	65.22	91.43	76.13			
			0.8	55.96	98.86	71.47			
	Yes	10	0.6	79.17	91.99	**85.10**	**93.93**	67.30	**78.42**
			0.7	61.77	98.91	76.05			
			0.8	53.87	**99.70**	69.95			
		15	0.6	**80.87**	86.61	83.64			
			0.7	66.74	98.00	79.40			
			0.8	54.82	99.63	70.72			
		20	0.6	80.31	75.42	77.79			
			0.7	71.96	95.60	82.11			
			0.8	56.20	99.48	71.82			

Table 7.5: Comparison of Bloom filter encodings using bi- and trigrams with and without padding for dataset N.

q	Padding	k	t	Recall [%]	Precision [%]	F-Measure [%]
2	No	25	0.7	68.17	88.32	76.95
	Yes	10	0.6	93.99	83.17	88.25
3	No	15	0.6	72.68	82.76	77.39
	Yes	10	0.6	95.49	90.14	92.74

Table 7.6: Comparison of Bloom filter encodings using bi- and trigrams with and without padding for dataset O.

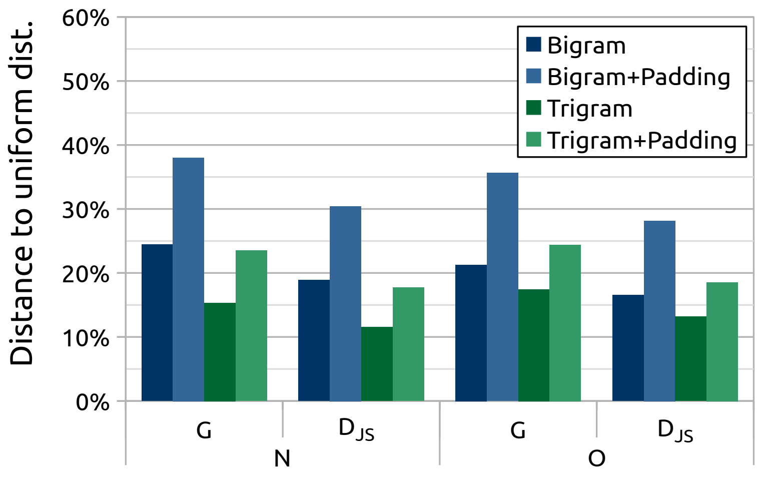


Figure 7.4: Comparison of Bloom filter privacy for bigrams and trigrams with and without using padding for datasets N and O.

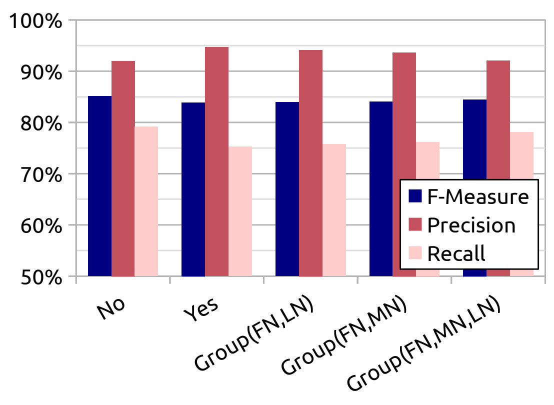
To summarize, the highest linkage quality is achieved by using padding which indeed leads to less uniform 1-bit distribution making frequency-based cryptanalysis more likely to be successful. However, this can be compensated by using trigrams leading even to a slightly better linkage quality than for bigrams. Consequently, for our following evaluation, we select the best configuration using trigrams and padding with $k = 10$ as a baseline for our experiments.

7.5.3 Salting and Weighting

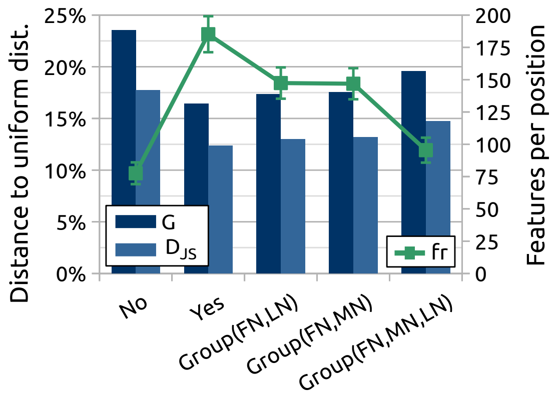
In this section, we evaluate the impact of methods that alter the Bloom filter's hashing process by varying the number of hash functions and using salting keys to modify the hash mapping.

7.5.3.1 Attribute Salts

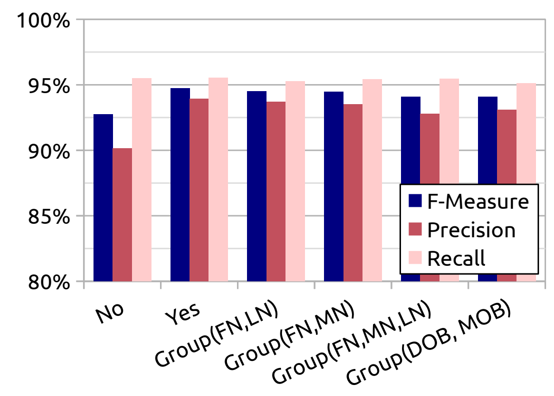
Figure 7.5 depicts the results for Bloom filters where the used hash functions are keyed (seeded) with a salt depending on the attribute a feature belongs to.



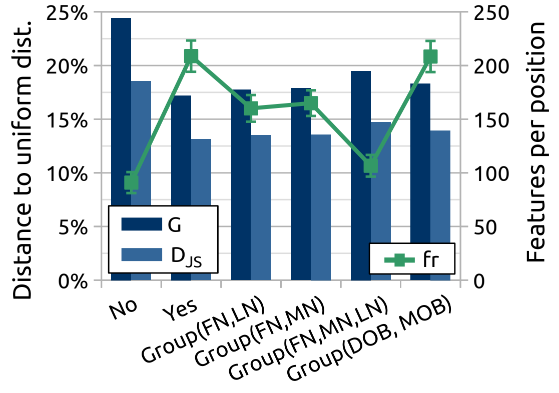
(a) Quality – N dataset



(b) Privacy – N dataset



(c) Quality – O dataset



(d) Privacy – O dataset

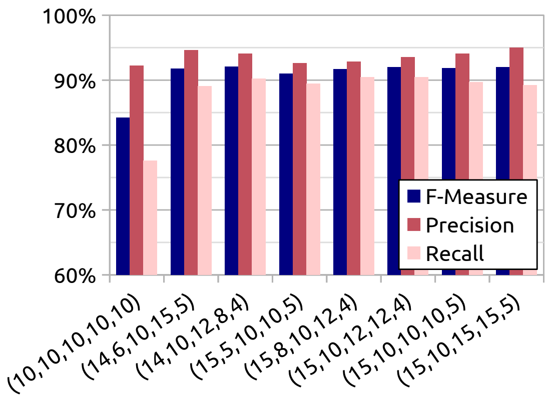
Figure 7.5: Impact of attribute salting.

For dataset N, we observe that using an individual salt for each attribute increases precision from 91.99% to 94.69% but also decreases recall from 79.17% to 75.26% leading to an F-measure loss of around 1.2%. Surprisingly, for dataset O, precision increases from 90.14% to 93.93% while recall remains stable. Simultaneously, the average number of features that are mapped to each bit position increases by more than a factor of two for both datasets (Figures 7.5b and 7.5d). Furthermore, also the Gini coefficient and the Jensen-Shannon distance are significantly decreased, thus indicating an additional smoothing of the 1-bit distribution.

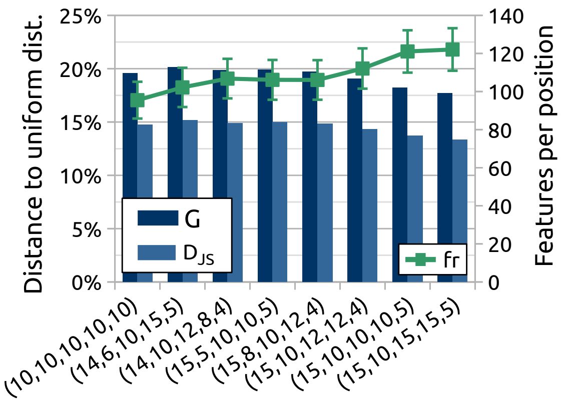
To be tolerant of swapped attributes, we build groups containing name-related attributes, i. e., one group for first name (FN) and last name (LN), one for first name and middle name (MN), and one for all three name components. Additionally, for dataset O, we build a group containing the day and month of birth (DOB, MOB). For all attributes within one group, the same attribute salt is used.

For dataset N, we observe that all groups can slightly increase F-measure, while the group `(FN,MN,LN)` performs best and can increase F-measure to 84.48%. Compared to the variant without using attribute salts, the F-measure is therefore only decreased by 0.6%. On dataset O, all groups achieve similar results, whereby precision and thus F-measure is always slightly lower than without using groups. Accordingly, swapped attributes seem to occur only rarely in dataset O. Using attribute salt groups also reduces the feature ratio and is less effective in flattening the 1-bit distribution.

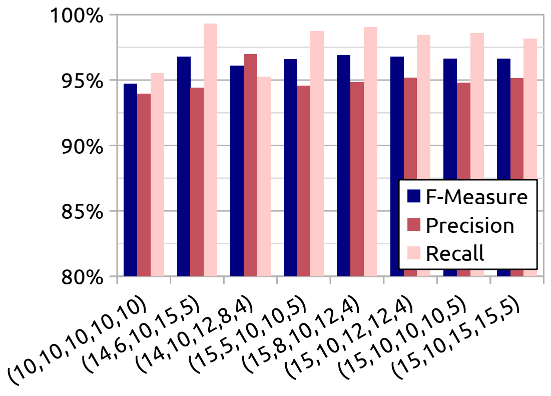
Overall, however, the use of attribute salts can significantly reduce the dispersion of 1-bits while maintaining a high linkage quality. Building attribute salt groups can be beneficial for linkage quality, namely for applications where attribute transpositions are likely to occur. In the following, we include attribute salting as a baseline for our experiments, where for dataset N the group `(FN,MN,LN)` is used.



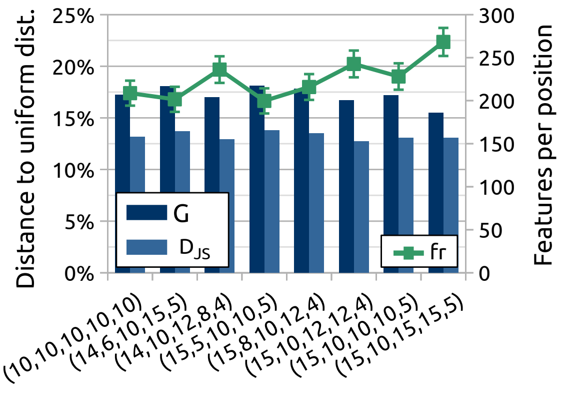
(a) Quality – N dataset



(b) Privacy – N dataset



(c) Quality – O dataset



(d) Privacy – O dataset

Figure 7.6: Evaluation of varying number of hash functions based on attribute weights.

7.5.3.2 Impact of Attribute Weighting

In the following, we evaluate the impact of attribute weighting. Therefore, the number of hash functions is varied for each attribute depending on attribute weight. We tested several configurations and report the results in Figure 7.6. The number of hash functions for each attribute is denoted in the order `(FN,MN,LN,YOB/BD,City)`. We observe that using attribute weighting strongly affects the linkage quality. All configurations that use a lower number of hash functions to map the attribute *city* can significantly increase both recall and precision. As a consequence, the F-measure is improved by more than 6% to over 91% for dataset N and by around 2% to over 96% for dataset O.

Analyzing the privacy results depicted in Figures 7.6b and 7.6d, we observe that most weighting configurations can slightly increase the feature ratio and also slightly decrease the non-uniformity of 1-bits. By comparatively analyzing linkage quality and privacy, we select the configuration `(15,10,15,15,5)` as a new baseline since it achieves the highest privacy while F-measure is only minimally less than for `(14,10,12,8,4)` (dataset N) and `(15,10,12,12,4)` (dataset O).

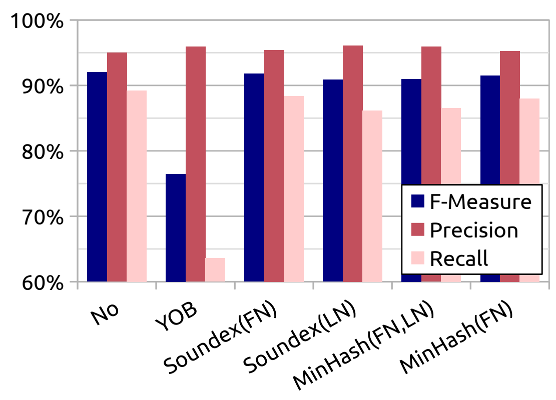
7.5.3.3 Record Salts

We now evaluate the approach of using a hash function salt that is individually selected for *each record*. The record salt is used in addition to the attribute salt we selected in the previous experiment. We tested several configurations using different attributes (year of birth, first name, last name).

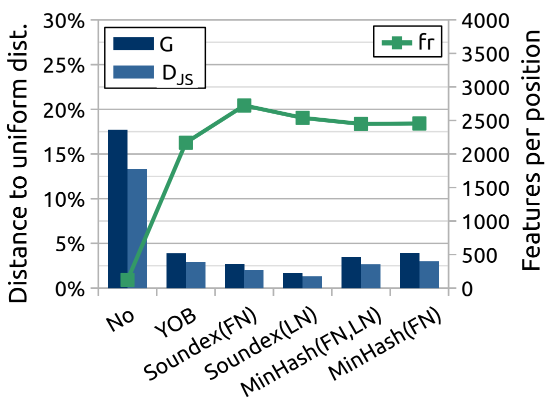
As Figures 7.7a and 7.7c illustrate, record salts highly affect the linkage quality outcome. If we use the person's year of birth (YOB) as a record-specific salt for the Bloom filter's hash functions, recall drops drastically from 89.20% (baseline) to only 63.58% for dataset N. Apparently, in this dataset, this attribute is often erroneous and thus not suitable as record salt. In contrast, applying this configuration on dataset O, recall is only slightly reduced while precision is slightly increased, resulting in nearly the same F-measure.

In order to compensate for erroneous attributes, we test two techniques that are often utilized as blocking approaches, namely Soundex and MinHashing that we apply on the first and/or last name attribute. All tested approaches can slightly increase precision as they make the hash-mapping of the record features more unique. However, the Soundex and MinHash-based approaches also decrease recall, depending on the attribute(s) used. For instance, using Soundex on last name leads to relatively low recall in both datasets indicating many errors, e. g., due to marriages or divorces. Nevertheless, with the approaches using the first name, a similar high F-measure (loss $\leq 1\%$) can be achieved as with the baseline.

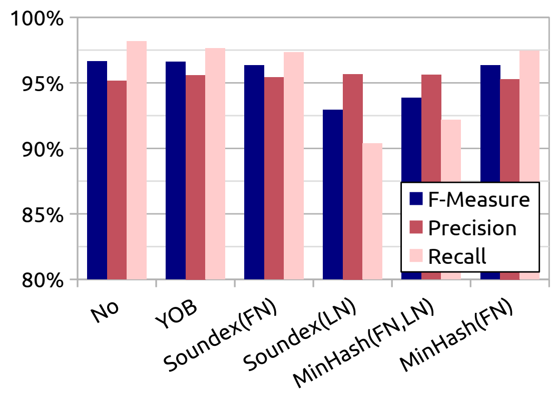
Inspecting the privacy results depicted in Figures 7.7b and 7.7d, we observe that the number of features that are mapped to each individual bit position is greatly increased by at least a factor of 10. At the same time, using record salts leads to a much more uniform 1-bit distribution. For instance, the Gini coefficient can be reduced from 0.1772 (baseline dataset N) and 0.1549 (baseline dataset O) to less than 0.04 for all tested



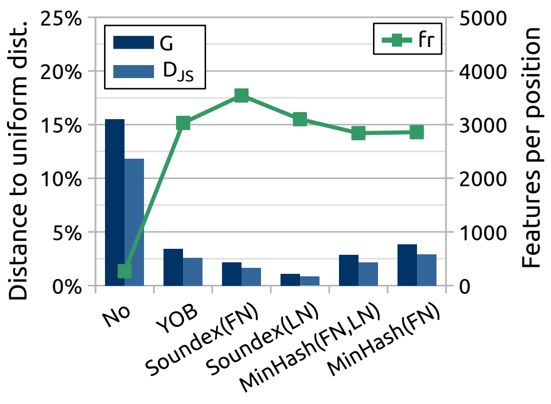
(a) Quality – N dataset



(b) Privacy – N dataset



(c) Quality – O dataset



(d) Privacy – O dataset

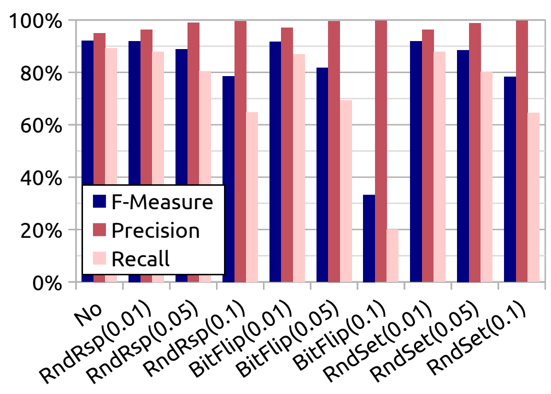
Figure 7.7: Impact of record salting.

approaches. The most uniform 1-bit distribution is achieved by using Soundex applied on the last name attribute, which leads to a Gini coefficient of less than 0.02. This implies that the 1-bit distribution is almost perfectly uniform, which will make any frequency-based attack very unlikely to be successful.

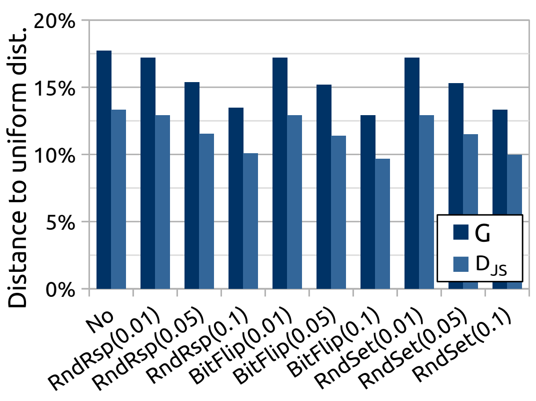
By analyzing privacy in relation to quality, we conclude that Soundex, applied to the first name, performs the best for both datasets and is able to achieve high linkage quality while effectively flattening the 1-bit distribution.

7.5.4 Modifications

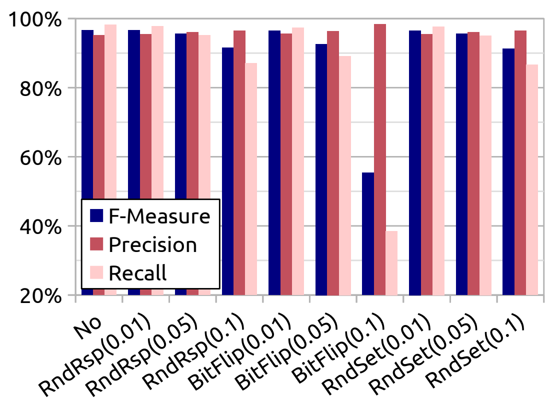
In the following, we evaluate hardening techniques that are applied directly on Bloom filters (bit vectors).



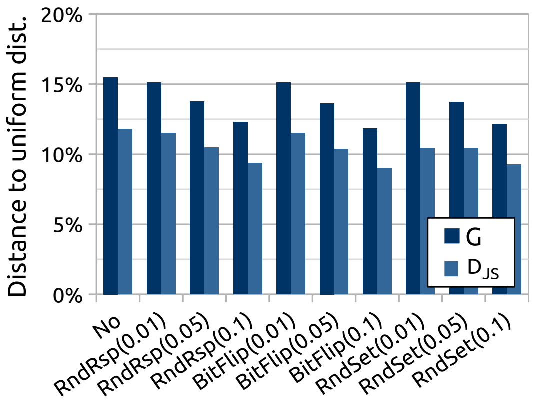
(a) Quality – N dataset



(b) Privacy – N dataset



(c) Quality – O dataset



(d) Privacy – O dataset

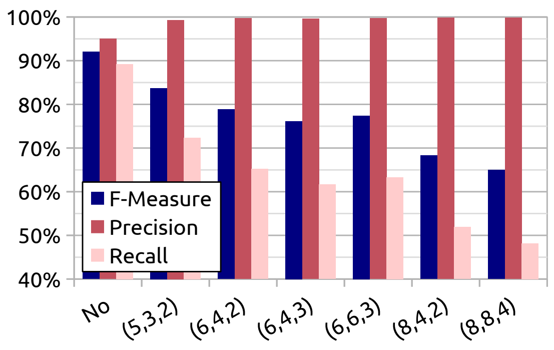
Figure 7.8: Evaluation of random noise approaches.

7.5.4.1 Adding Random Noise

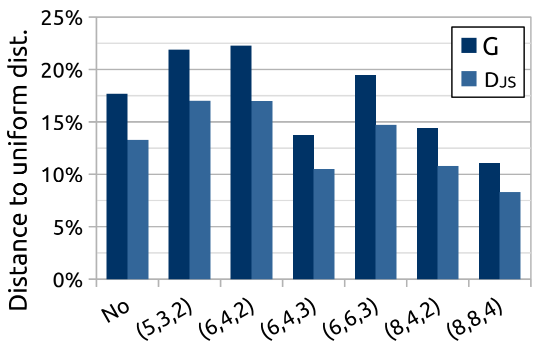
There are several ways of adding random noise to a Bloom filter (see Section 7.2.3.8). We compare the randomized response technique (RndRsp), random bit flipping (BitFlip) and randomly setting bits to one (RndSet) with each other. We vary the probability of changing an individual bit by setting $\rho = \{0.01, 0.05, 0.1\}$. The results are depicted in Figure 7.8.

As expected, recall and F-measure decrease with increasing ρ. While for $\rho = 0.01$ the loss is relatively small, it becomes significantly large for $\rho = 0.1$, in particular for the bit flip approach where recall drastically drops below 20% for N and below 40% for O. Interestingly, precision can be raised for all approaches and configurations up to 4.7% (for $\rho = 0.1$). Overall, the bit-flipping approach leads to the highest loss in linkage quality.

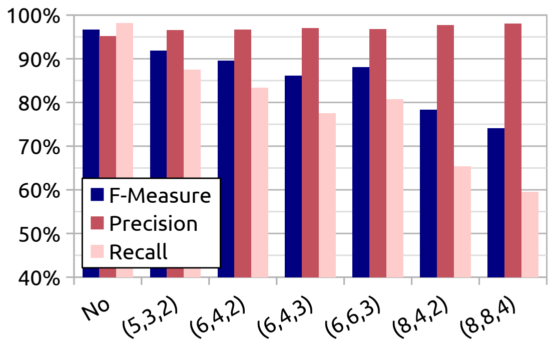
By analyzing the privacy results shown in Figures 7.8b and 7.8d, it is evident that all random noise approaches can flatten the frequency distribution only a little. Only



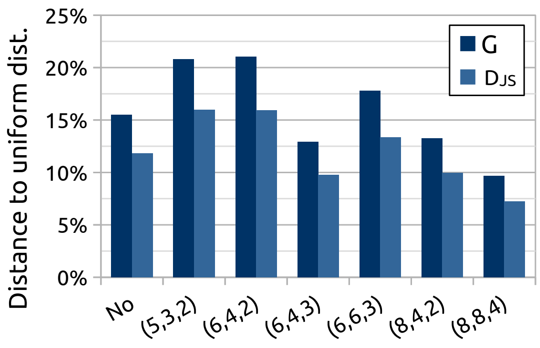
(a) Quality – N dataset



(b) Privacy – N dataset



(c) Quality – O dataset



(d) Privacy – O dataset

Figure 7.9: Evaluation of re-hashing.

at a high ρ-value of 0.1 the Gini coefficient can be reduced by up to 4.81% and the Jensen-Shannon distance up to 3.65%.

Analyzing the trade-off between linkage quality and privacy, we observe that a high value for ρ leads to an unacceptable loss in linkage quality. For lower ρ-values both techniques, randomized response and randomly setting bits to one, lead to relatively small losses in linkage quality. However, they are not able to significantly flatten the 1-bit distribution. Nevertheless, hardening techniques based on random noise may impede deterministic attacks by increasing the number of unique bit patterns.

7.5.4.2 Re-Hashing

To evaluate re-hashing, we test several configurations regarding window size w, step size s, and the number of re-hashed values r. In Figure 7.9 we report the best configurations which are denoted in the order `(w,s,r)` setting $m' = m$.

The results regarding linkage quality show that re-hashing increases precision but, in contrast, drastically decreases recall. In general, the larger the window size w, the lower the recall that can be achieved. This effect is due to the fact that with larger windows, there is a higher probability that a bit in the window is different for two similar Bloom filters. This will result in another integer value (seed) on which the re-hashed 1-bit positions are selected. Even for the configuration with the smallest window size $w = 5$, recall decreases by more than 16% for both datasets. We could not further decrease the window size, as with $w = 4$ only 16 different bit patterns are possible. Consequently, the re-hashed values will be often the same.

This observation is confirmed by inspecting the privacy measures illustrated in Figures 7.9b and 7.9d. Surprisingly, several configurations, namely `(5,3,2)`, `(6,4,2)` and `(6,6,3)`, will increase the non-uniformity of 1-bits. This is because the re-hashed values will be mapped only to a small range and thus increase the frequencies of these bits. In contrast, the configuration `(6,4,3)` and those with $w = 8$ can flatten the similarity distribution moderately. At the same time, however, these configurations will lead to an unacceptable low recall, e. g., for dataset N to only 61.66% for `(6,4,3)` or even less than 50% for `(8,8,4)`. As illustrated by the two configurations `(8,8,4)` and `(8,4,4)`, a reduction of the step size can increase recall since configurations with $s < w$ will lead to overlapping windows and thus a higher chance of finding overlapping bit patterns between two Bloom filters. However, this again increases the unequal distribution of 1-bits.

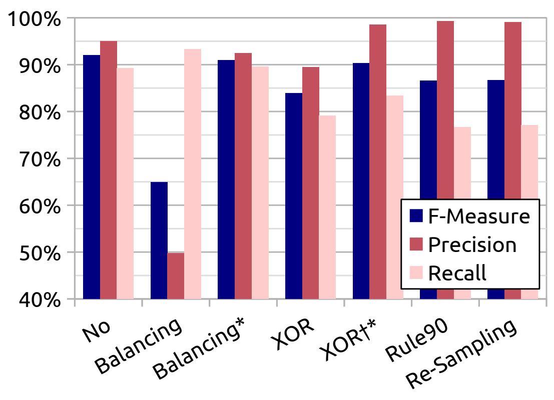
To summarize, we observe that re-hashing will decrease linkage quality while being not effective in increasing the uniformity of 1-bits. Therefore, we can not recommend this method for practical applications.

7.5.4.3 Balancing, xor-folding, Rule90, Re-Sampling

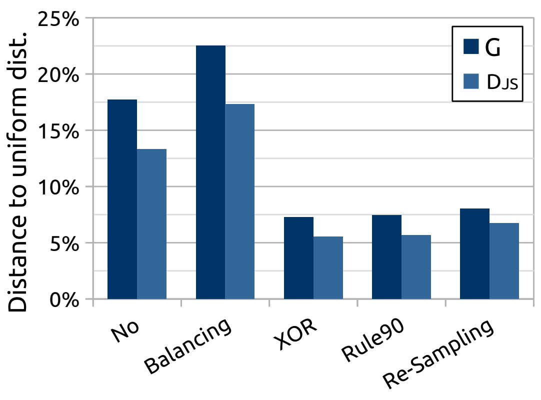
Finally, we evaluate balanced Bloom filters, XOR-folding and applying Rule90 in terms of linkage quality and privacy. The results are depicted in Figure 7.10.

The evaluation shows that balancing reduces precision. While for dataset O precision decreases moderately by 6.93%, for dataset N it drops drastically by 45.19%. In contrast, recall remains stable for dataset O whereas it is slightly increased for dataset N. We found that changing our basic similarity threshold from 0.6 to 0.7 can significantly improve linkage quality using balanced Bloom filters. This might be due to the fact that balancing doubles the size of the Bloom filters. Thus, we included the starred version of balancing, indicating that a different threshold was used. With this configuration, balancing reduces the F-measure only slightly for both datasets. For dataset N, this is due to a little less precision and a little higher recall than for the baseline. For dataset O, however, it is the other way around, i. e., less recall and higher precision.

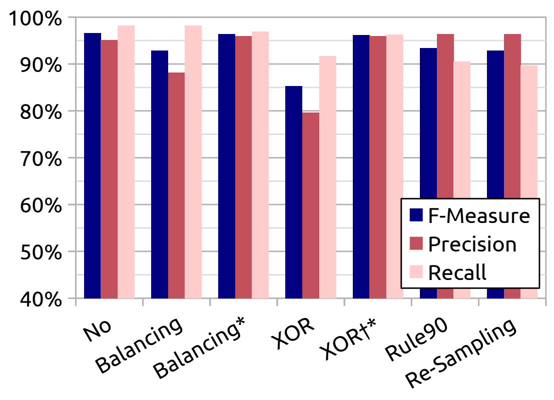
XOR-folding also causes a reduction in linkage quality for both datasets. Since XOR-folding halves the size of the Bloom filters, LSH-based blocking is affected in such a way that the amount of bits selected for the LSH keys is comparatively large. We therefore



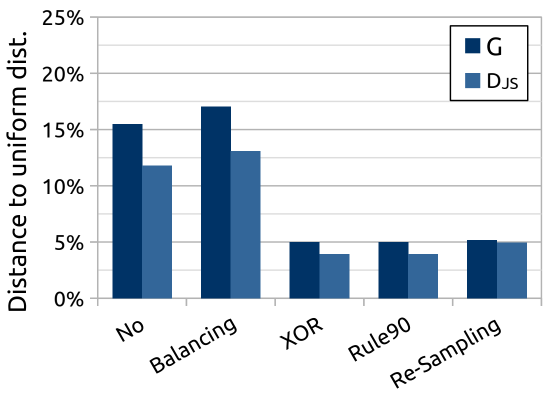
(a) Quality – N dataset



(b) Privacy – N dataset



(c) Quality – O dataset



(d) Privacy – O dataset

Figure 7.10: Evaluation of balanced Bloom filters, XOR-folding, Rule90 and re-sampling.

reduced the LSH key length from $\Psi = 16$ to $\Psi = 10$ and indicated this configuration with a dagger (†). By using this configuration and setting $t = 0.7$, for both datasets, XOR-folding results in a minor loss of F-measure of less than 1% compared to the baseline. Primarily accountable for the high F-measure is the high precision, which is slightly increased.

Furthermore, we observe that applying Rule90 also leads to a relatively high loss of recall of around 12.5% for dataset N and 7.6% for dataset O. Again, Rule90 increases precision slightly, thus leading to a moderate loss of F-measure of around 5.5% for dataset N and 3.2% for dataset O.

The re-sampling approach produces similar results as Rule90 for both datasets. That meets our expectations as both approaches rely on combining certain bits using the XOR-operation. While Rule90 considers the two neighboring bits, re-sampling randomly samples two bits from the original Bloom filter.

Examining Figures 7.10b and 7.10d, we can see that balancing interestingly increases the dispersion of 1-bits. For dataset N, for instance, the Gini coefficient is increased by around 4.8% and the Jensen-Shannon distance by around 4%. In contrast, XOR-folding, Rule90, and re-sampling lead to a more uniform distribution of 1-bits. These three approaches reduce the Gini coefficient by about 10% and the Jensen-Shannon distance by more than 7.5%.

Considering both, linkage quality and privacy, we conclude that XOR-folding performs the best by maintaining high linkage quality while effectively flattening the 1-bit distribution. From a practical point of view, however, XOR-folding requires adjusting the linkage configuration as it halves the length of the Bloom filters. Rule90 and re-sampling, in contrast, do not require any additional parameters. Consequently, they allow an easier configuration with similar results in terms of quality and privacy protection.

7.5.4.4 Window-based xor

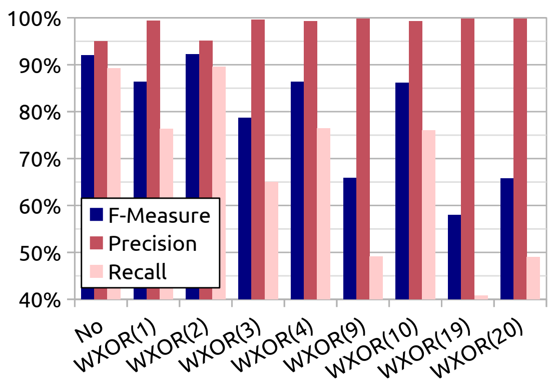
To evaluate the window-based XOR (WXOR) approach, we test different window sizes $1 \leq w \leq 20$. The results are shown in Figure 7.11.

As expected, the overall quality of the linkage in terms of the F-measure decreases with increasing window size, which is due to the substantially decreasing recall values. However, larger windows increase precision up to almost 100%. Interestingly, the quality results alternate according to the window size w, with odd window sizes leading to a greater decrease than even window sizes. Using a window size of $w = 2$ yields almost the same quality results as the baseline.

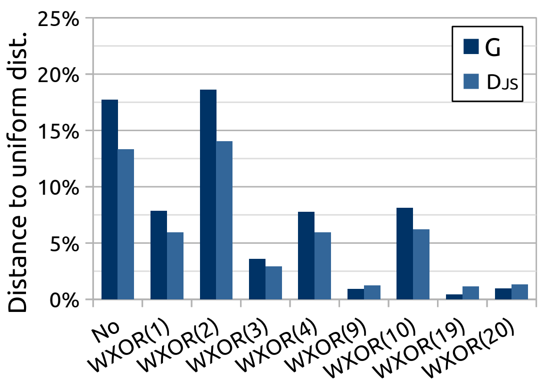
Also in terms of privacy, the WXOR approach shows alternating results depending on the window size as plotted in Figures 7.11b and 7.11d. In general, larger windows result in a more even distribution of 1-bits, with odd window sizes being more effective at smoothing the 1-bit distribution. Again, a window size of $w = 2$ produces only slightly different results than the baseline without hardening, keeping the 1-bit distribution nearly the same. Compared to the other XOR-based hardening techniques - XOR-folding, Rule90, and re-sampling - the WXOR approach actually achieves slightly better results and leads to an almost uniform 1-bit distribution.

Overall, only a window size of $w = 1$ leads to results that provide a good balance between linkage quality and privacy protection. Larger windows result in a more uniform 1-bit distribution but also reduce the linkage quality considerably. The high sensitivity of the w parameter complicates its configuration, so we cannot recommend the WXOR approach for practical applications.

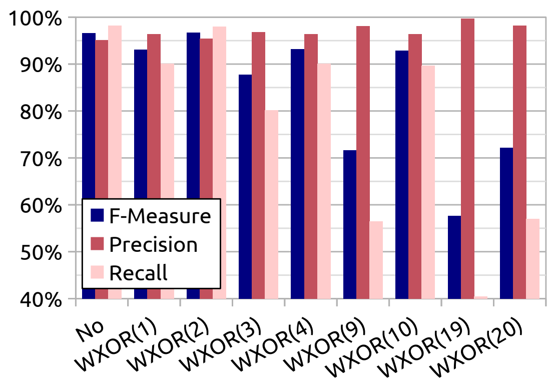
The alternating quality and privacy results are likely due to the fact that even window sizes lead to fewer bit changes. Since both windows are always moved forward by only one bit, successive XOR-operations may neutralize each other. This is because of the properties of the XOR-operation: (1) the XOR-operation is commutative and associative; (2) the XOR-operation is self-inverse, i. e., applying XOR on the same



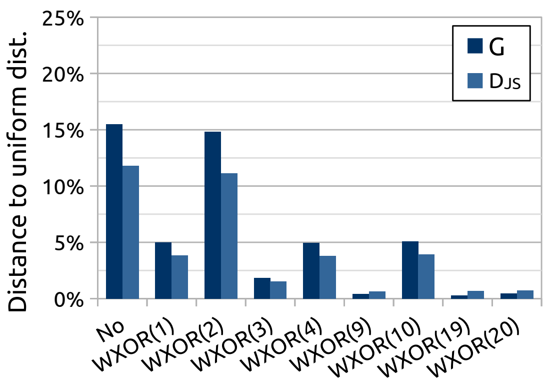
(a) Quality – N dataset



(b) Privacy – N dataset



(c) Quality – O dataset



(d) Privacy – O dataset

Figure 7.11: Evaluation of window-based XOR using different windows sizes.

arguments will result in 0; and (3) 0 is the identity element, i. e., if one of the two arguments is 0, then the remaining argument is left unchanged. Thus, in a sequence of XOR-operations, all pairs of duplicate values can be removed without affecting the result, i. e., $(b_i \oplus b_j) \oplus b_j = b_i \oplus (b_j \oplus b_j) = b_i \oplus 0 = b_i$ for bits b_i and b_j with $0 \leq i, j < m$.

7.6 Conclusion

Bloom filters are frequently used in both research and practice for PPRL applications. In this chapter, we reviewed and classified various Bloom filter variants and hardening techniques that aim at making Bloom filters more robust against cryptanalysis.

Currently, no privacy measure exists that allows comparison of different encoding schemes in terms of privacy (security) and is independent of any reference dataset. Therefore, we proposed three privacy measures that allow for assessing the privacy properties of Bloom filter encodings. These measures are based solely on a set of Bloom filters and do not need any reference dataset or other information.

Moreover, we comprehensively evaluated the Bloom filter variants and hardening techniques in terms of both linkage quality and privacy. The evaluation showed that multiple hardening techniques drastically reduce linkage quality and are thus not applicable in real-world scenarios. However, in particular two techniques, namely salting and XOR-folding, drastically reduce any frequency information while maintaining high linkage quality. Carefully selected Bloom filter parameters in combination with these techniques will make any frequency-based cryptanalysis very unlikely to be successful. For future work, we aim to evaluate these approaches against modern Bloom filter attacks described in the literature to further verify our findings.

8

PRIMAT: A Toolbox for Fast Privacy-preserving Matching

This chapter is based on [FSR19]. Privacy-preserving record linkage (PPRL) is increasingly demanded in real-world applications, e. g., in the healthcare domain, to combine person-related data for data analysis while preserving the privacy of individuals. However, the adoption of PPRL is hampered by the absence of easy-to-use and powerful PPRL tools covering the *entire* PPRL process. Therefore, we demonstrate Primat, a flexible and scalable tool that enables the definition and application of tailored PPRL workflows as well as the comparative evaluation of different PPRL methods. We introduce the main requirements for PPRL tools and discuss previous tool efforts that do not fully meet the requirements and have not been applied in practice. By contrast, Primat covers the whole PPRL life-cycle and improves applicability by providing various components for data owners and the central linkage to be executed by a trusted linkage unit.

8.1 Motivation

The integration of person-related data, e. g., for customers or patients, is needed in many applications for improved data analysis. However, stricter legal data protection requirements increasingly ask for privacy-preserving data integration that does not reveal the identity of persons for whom data is combined and analyzed. These requirements are met by privacy-preserving record linkage (PPRL) techniques that encode identifying attributes, e. g., name and birthdate, and often perform the linkage of encoded records in a separated, trusted environment. A large number of such PPRL methods have been proposed in the last years, as surveyed in [VCV13; Vat+17]. Some of these approaches have also been applied, primarily in medical research studies to combine patient-related data from different hospitals or medical offices, e. g., to analyze diseases or treatments [Gib+16; LBÜ15; Ran+14]. Despite the large number of proposed PPRL schemes, their practical use in real applications is still limited due to the absence of convenient tools and the high complexity of properly selecting and configuring a suitable PPRL approach. In fact, the relative strengths and weaknesses of different PPRL approaches and configurations regarding privacy, effectiveness, and efficiency are largely unknown and ask for more comparative evaluations. While there are some available PPRL implementations and prototypes (see Section 8.3) they focus on research aspects and do not provide enough functionality for use in practice. Previous PPRL applications in medicine are based on tailored solutions that are not usable in different applications. There is also proprietary PPRL software in use, e. g., in an Australian record linkage center supporting medical research projects [Gib+16; Luo+17].

There is therefore a strong need for easy-to-use, powerful, and open-source tools to facilitate the adoption of PPRL in real applications. We have thus started developing an open-source [Fra23] PPRL toolbox, named Primat (**Pri**vate **Ma**tching **T**oolbox). It includes our previously developed methods for fast and scalable PPRL based on the use of blocking and parallel matching of encoded records (see Chapter 3) as well as for post-processing to select the best matches (see Chapter 5). Primat focuses on practical usability and provides different linkage modes, protocols, and components that support creating individual linkage workflows. Moreover, Primat offers an evaluation framework to uniformly compare PPRL methods regarding their efficiency and effectiveness.

In the following, we first introduce the main requirements for PPRL tools such as Primat (Section 8.2) and discuss related implementations (Section 8.3). Then, we provide an overview of Primat and its components (Section 8.4) and finally conclude.

8.2 Requirements for PPRL Tools

To achieve broad acceptance and operability, the several requirements should be satisfied by a PPRL toolbox. These requirements are described in detail in the following.

8.2.1 Tackle PPRL Key Challenges

PPRL has three key challenges that need to be carefully addressed. In particular, a high degree of *privacy* has to be ensured by providing state-of-the-art encoding techniques that reduce the risk of data breaches. Moreover, a high *linkage quality* must be achieved, i. e., the number of false and missing matches should be minimized. Finally, PPRL needs to be scalable to large data volumes and possibly many data owners where up to millions of records need to be linked. Hence, blocking or filtering methods [Chr12b] as well as parallel and distributed processing are beneficial to reduce the complexity of linkage and to speed up similarity computations.

8.2.2 Support of Entire PPRL Process

PPRL demands a multi-step process involving data owners and trusted third parties, e. g., the linkage unit, to ensure high efficiency and effectiveness. Data owners need to prepare their data to ensure comparability and encode their records to support privacy. Data preparation includes unification to resolve schema differences as well as cleaning procedures, e. g., to resolve data entry errors. The actual linkage also requires multiple steps, in particular, blocking or filtering to avoid comparing every possible pair of records, similarity calculation, and a classification step to decide whether a record pair is considered a match. Consequently, supporting the entire linkage process is essential to bring PPRL into applications. In order to support various use cases, all components must be individually configurable to comply with the specific needs of each application scenario. Moreover, additional or future techniques should be easy to integrate.

8.2.3 Enabling of Batch and Incremental Linkage

Typically, PPRL is executed in batch mode, where *all* records of a fixed number of static databases are linked *at once*. This linkage mode is also known as offline matching [Chr12b; AKM13] and is illustrated in Figure 8.1a. All database owners provide their complete encoded databases (Figure 8.1a step 1). The offline batch processing results in a match mapping, e. g., in the form of a similarity graph, when the entire linkage process has been completed (Figure 8.1a step 2).

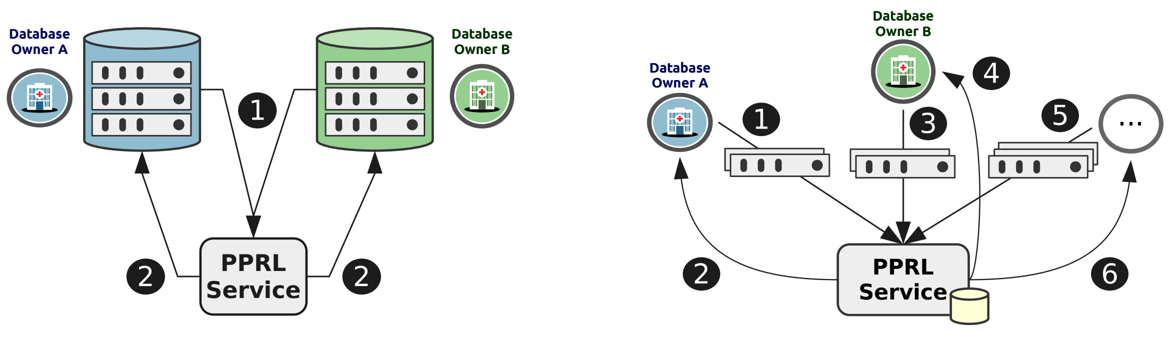
Neither the input records nor the match result are stored permanently. As a consequence, a complete re-computation is needed for new records or new databases to be linked. Due to the high complexity of the linkage (see Section 2.4) this leads to a high overhead even if performance optimizations, such as blocking or parallel/distributed processing, are used. As a consequence, the timeliness of results is likely limited, especially for a higher number of databases that are of large size. Batch linkage, however, allows a flexible adjustment of methods and parameters on re-execution and therefore enables dealing with changing requirements, for example regarding data quality or privacy.

In some practical applications, it is desired to deploy PPRL in an incremental fashion such that new records can be added continuously without having to repeat linkage for previously known records. This linkage mode is also known as online matching and is illustrated in Figure 8.1b. In this linkage mode, database owners can dynamically add new sets of records, including a single record or even an entire new database. The inserted records, calculated similarities between records, and match results are stored in a database for future inquiries. For each inserted/queried record, the party submitting the record receives the match status and additional information depending on the use case. In Figure 8.1b, at first, database owner A sends a set of records to the PPRL service (typically deployed at the linkage unit) and gets their match status as a response (steps 1-2). Then, database owner B does the same with a set of its records (steps 3-4). Further sets of records are then sent by an arbitrary party.

Adding individual records and querying their match status on-demand must be very fast [LBÜ15], for instance, to trigger certain actions in emergency situations. Domains that require such a real-time linkage include healthcare, law enforcement, and national security, as well as the finance sector [Chr12b; CRS20]. In these areas, the queries aim to check whether a person is present in a particular database, which corresponds to an identity or solvency check, for example [Phu+12]. In real-time linkage scenarios, a response typically must be provided within at most a few seconds. While incremental linkage can provide real-time match information, the drawback is that there is no global match decision possible. Each subset of records is matched based on the current linkage configuration as well as the records contained at that time. This can lead to match decisions in earlier stages which turn out as sub-optimal in later stages when more records have been inserted.

In incremental linkage, each party can dynamically add new records that need to be matched to previously added records. Therefore, the incremental linkage adds the challenge of match cluster management, also known as entity clustering [NR18]. A match cluster or entity cluster is a group (subset) of records that are assumed to represent the same real-world entity. Ideally, each real-world entity is represented by exactly one cluster (see Section 2.6.5.2). Each cluster is typically assigned a unique cluster ID, that can be considered as an entity identifier (see Section 2.5.1). The task of entity clustering is to determine and maintain a set of disjoint entity clusters such that all records within a cluster match with each other. These records are thus considered referring to the same entity, while records of different clusters refer to different entities.

For batch linkage, the input is a fixed and static set of records from different databases, so that clustering needs to be performed only once. For incremental linkage, however, records can continuously be added (or changed) so that the clusters need to be adapted accordingly. The entity clustering has as input an existing (possibly empty) set of clusters χ_{exist}, as well as a set of new records R. The objective is then to assign each record $r \in R$ either to an existing cluster $c \in \chi_{exist}$, or to create a new cluster for it. In the latter case, the new record r does not match any previously added record (and thus entity) and therefore is considered as a new singleton cluster. Finally, the



(a) Batch Matching (b) Incremental Matching

Figure 8.1: Linkage Modes. The steps of the linkage are indicated with numbers in circles.

output consists of the set of adapted clusters χ_{exists} and the set of new clusters χ_{new}, i. e., $\chi'_{exist} = \chi_{exist} \cup \chi_{new}$.

In general, this corresponds to finding an assignment between each record and a cluster. A single record can be added to at most one of the previously existing clusters. However, several records may be added to the same cluster. Finding an optimal assignment depends on the characteristics of the databases to be incrementally matched. Assuming that a database owner will not provide duplicate records (which corresponds to a duplicate-free database in the static case), each cluster can contain at most one record from the respective database. If, in contrast, a database owner can provide duplicate records (which corresponds to a dirty database in the static case), then each cluster can contain more than one record from the respective database.

8.2.4 Support for Multiple Database Owners

While most previous PPRL approaches focus on only two data sources, it is essential to support multi-party approaches with two or more data owners. In this case, the matching records should not only be linked but also clustered so that all records in a cluster match with each other. Incremental linkage has to determine whether a new record belongs to an existing cluster or whether it represents a new cluster.

8.2.5 Ease of Use

PPRL workflows should need minimal parameter tuning effort or deep knowledge of the underlying techniques. Therefore, a PPRL toolbox should provide appropriate default settings for methods and parameters, as well as guidance on how to adjust these settings to fit a particular use case. Moreover, the effort for integrating PPRL in existing system environments or applications should be minimized.

8.2.6 Evaluation

It is important for both practitioners and researchers to test and evaluate different PPRL methods and parameter settings to determine effective configurations and appropriately balance efficiency and privacy. For this purpose, tool support to generate and use realistic synthetic test data is highly beneficial to determine effective PPRL workflows. Furthermore, providing analysis and measurement facilities is helpful to evaluate different approaches and configurations under uniform conditions.

8.3 Related Work

Many PPRL approaches have been proposed in the last years, as summarized in [VCV13; Vat+17; Gko+21]. Recent approaches mostly rely on encoding techniques based on Bloom filters[SBR09; SBR11] and make use of a trusted linkage unit that centrally conducts the linkage of encoded records. While some PPRL tools have been implemented, they do not meet all requirements collected in Section 8.2. A detailed comparison of existing PPRL tools is given in Table 8.1.

Mainzelliste [LBÜ15] is a pseudonymization and identity management system designed for multi-site medical applications. While it has already been used in medical joint research projects in Germany, it focuses on field-level Bloom filter encodings and does not support current hardening techniques and therefore does not provide tunable privacy protection. The Mainzelliste supports incremental linkage, where a single record can be added at a time. There is no distinction made according to the source of the records and whether they are duplicate-free. However, as shown in [NR18], a joint consideration of multiple records or complete databases can improve the linkage quality over a record-by-record matching. Moreover, each cluster is represented by the oldest (first inserted) record which can lead to a low linkage quality as it cannot compensate for successive changes, for example, if a person first moves and then changes his/her name due to a marriage.

LSHDB [KGV16] is another prototype, that offers parallel and distributed processing, privacy-preserving blocking as well as both batch and incremental linkage. However, an incremental linkage is only supported with limits, e. g., there is no clustering mechanism for more than two records referring to the same entity. Besides, LSHDB does not provide any encoding, hardening, or evaluation facilities.

PRIVATEER [Kar+15] is a PPRL research prototype that provides different blocking and matching algorithms. It lacks support for state-of-the-art private blocking approaches and does not provide incremental linkage.

	PRIMAT	Mainzelliste [LBÜ15]	LSHDB [KGV16]	PRIVATEER [Kar+15]	SOEMPI [Tot+14]	Package 'PPRL' [SR22]
Open-source	✓	✓	✓	✓	✓	✓
Linkage Unit (LU) + SMC	LU	LU	LU	LU	LU	LU
Data cleaning	✓	(✓)	✗	✗	✗	✗
Flexible encoding	✓	FBF	✗	CLK	RBF	BF
Hardening support	✓	✗	✗	✗	✗	✓
Private blocking	✓	✓	✓	SB	✓	MBT
Distributed/parallel execution	(✓)	✗	✓	✗	✗	✗
Batch(B) + Inc.(I) Matching	✓	I	✓	B	B+DB	B
Match Cluster Management	✓	(✓)	✗	✗	✗	✗

Table 8.1: Comparison of PPRL Tools.

SOEMPI [Tot+14] provides PPRL functionality and is also designed for medical applications. SOEMPI offers a variety of methods, including current encoding and blocking techniques. However, it does not provide evaluation facilities, nor it is optimized for incremental linkage.

Package 'PPRL' [SR22] is a toolbox for deterministic, probabilistic, and privacy-preserving record linkage. It combines the functionality of the predecessor software MergeToolBox (MTB) [SBB04] with current privacy-preserving linkage techniques. However, Package 'PPRL' mainly focuses on encoding techniques while providing only limited support for linkage methods.

Overall, most existing tools fail to cover the entire PPRL process and lack any preprocessing support to weaken typical data quality problems. Moreover, the existing tools are mostly focusing on a certain set of encoding and linkage techniques, instead of providing different techniques to support individual use cases.

8.4 Description of Toolbox Implementation

In Figure 8.2 the overall architecture of PRIMAT is depicted. We differentiate between two main roles for participating in a PPRL process (see also Section 2.7):

- **Database owners (DOs):** manage sensitive data in the form of person-related database records, e. g., patient records, that should be linked to the other database

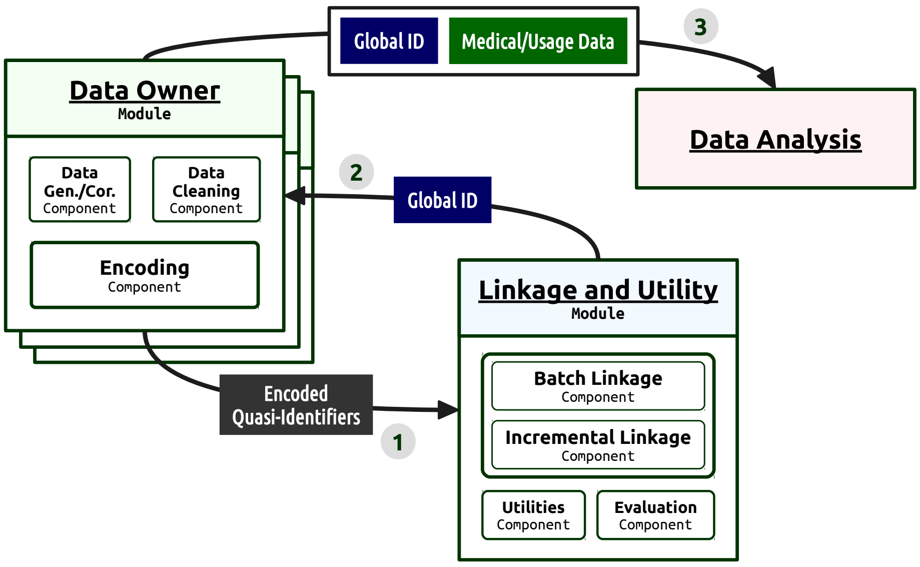


Figure 8.2: Architecture of PRIMAT.

owners' records in a privacy-preserving manner. Any database record consists of specific attributes and has a unique record identifier that is no entity identifier. We do not make any assumptions about the status of the databases to be linked regarding inconsistencies or their deduplication status, i. e., if there is more than one record per entity in a single database.

- **Trusted third parties:** can have multiple responsibilities within a linkage process. For PRIMAT, we assume that a trusted linkage unit (LU) performs the actual linkage of encoded records submitted by the database owners. Using a linkage unit is beneficial since it requires less complex protocols and low communication costs. Secondly, we assume a data analyst who wants to combine the database owners' data for analysis or research. Other third parties may be involved to coordinate communication between database owners or provide parameter recommendations.

From a high-level perspective, each PPRL process consists of three steps (see Figure 8.2):

1. Each database owner prepares their records for linkage and encodes the identifying attributes (quasi-identifiers) of each entity. The encoded quasi-identifiers are then sent to the linkage unit.
2. The linkage unit conducts the linkage and returns global IDs (pseudonyms) to the database owners. Thereby, every pair of records that has the same global ID is considered a match.
3. The database owners send the medical/usage data together with the entities' global ID to the data analyst, where the data for the same ID can be combined for enhanced analysis.

Primat consists of several key components that cover each step of the PPRL process pipeline. Based on the two aforementioned roles typically occurring in PPRL protocols, the components are separated into two modules, namely a *DO module* for pre-processing and encoding of database records and a *linkage and utility module* which mainly provides various linkage techniques for use at the linkage unit. Moreover, each module provides functions for sending/receiving and generating/parsing data and parameter files. In the following subsections, we give a description of each component in the two modules.

An overview of the components and functions offered by Primat can be found in Table 8.2. Currently, out of 21 planned functions, 14 are implemented and ready for use, including all essential functions required to perform a PPRL workflow. The missing functions are planned as extensions to further improve the usability and performance in some use cases. We have already implemented some of these planned extensions (4 out of 7), such as metric space filtering [SR16; Seh+21] our approaches for parallel and distributed matching [FSR18; Gla+18], but an integration into Primat is still pending.

8.4.1 DO Module

The DO module provides procedures that are required for database owners to appropriately prepare their records for linkage. It consists of three key components (D_1 - D_3) which we will describe in detail below.

8.4.1.1 Data Generation and Corruption Component

This component allows database owners to generate realistic synthetic datasets, possibly based on real-world data, which can be used for probing and balancing PPRL workflows. The linkage result obtained for a synthetic dataset can be analyzed, which supports the selection of appropriate methods and fine-tuning of parameter settings.

8.4.1.2 Data Cleaning Component

The data cleaning component (D_2) allows an extensive pre-processing of the database owners' data. In particular, it provides common operations to clean and standardize data, e. g., removing/replacing values or splitting/merging attributes. Primat also offers traditional record linkage techniques that can be used to detect intra-source duplicates. The data cleaning functions provided by Primat are listed in Table 8.3.

Component	**Function**	**Status**	**Reference**
Data generation (D_1)	Generation of realistic synthetic test data	⚙	⑂ [Dao22]
	Corruption functions to simulate dirty data	🕘	⑂ [TVC13]
Data cleaning (D_2)	Split/merge/remove attributes	✓	
	Replace/remove unwanted values and stopwords	✓	
	OCR transformation	✓	
Encoding (D_3)	Bloom filter encodings	✓	
	Bloom filter hardening techniques	✓	
	Support of alternative encoding schemes	✓	
	Blocking key generation	✓	
Utilities (L_1)	Pre-processing templates	🕘	🔗 [Nób+18]
	Private schema matching	🕘	🔗 [SCS15]
Matching (L_2, L_3)	Standard blocking	✓	
	LSH-based blocking	✓	
	Metric space filtering	⚙	⑂ [SR16; Seh+21]
	Threshold-based classification	✓	
	Post-processing methods	✓	
	Parallel and distributed matching	⚙	⑂ [FSR18; Gla+18]
	Multi-party support & match cluster management	✓	
	Incremental matching	✓	
Evaluation (L_4)	Measures for assessing quality, privacy and scalability	✓	
	Masked match result visualization	⚙	⑂ [Sch21]

Table 8.2: Functional overview of PRIMAT. The symbol ✓ denotes implemented features and the symbol ⚙ denotes features where an integration of an available implementation (⑂) is outstanding. In contrast, the symbol 🕘 indicates planned features where the corresponding approaches are described in the literature (🔗).

Data Cleaning Function	**Description**
AccentRemover	Removes diacritics.
CharacterReplacer	Replaces characters with strings based on a given replacement mapping.
DigitRemover	Removes all digits.
GenderNormalizer	Transforms all gender character sequences, such as male, female, m, f, w, 0, and 1, into a uniform representation.
LetterLowerCaseToNumber-Normalizer	Converts o → 0, l → 1, z → 2, q → 4, s → 5, and g → 9 to handle OCR errors.
LetterUpperCaseToNumber-Normalizer	Converts O → 0, L → 1, Z → 2, A → 4, S → 5, G → 6, and B → 8 to handle OCR errors.
LowerCaseNormalizer	Converts all letters to lowercase.
NonDigitRemover	Remove all characters that are no digits.
NullRemover	Removes strings representing null values.
NumberToLetterLowerCase-Normalizer	Converts 0 → o, 1 → l, 2 → z, 4 → q, 5 → s, 9 → g to handle OCR errors.
NumberToLetterUpperCase-Normalizer	Converts 0 → O, 1 → L, 2 → Z, 4 → A, 5 → S, 6 → G, and 8 → B to handle OCR errors.
PunctuationRemover	Removes punctuation like dots or commas.
RegexReplacer	Replaces each substring that matches the given regular expression with the given replacement.
SpecialCharacterRemover	Removes special characters, such as ?,!,$.
StandardStringNormalizer	Applies WhitespaceRemover, UmlautNormalizer, AccentRemover, PunctuationRemover, NumberToLetterLowerCaseNormalizer, and LowerCaseNormalizer.
StandardNumberNormalizer	Applies LetterLowerCaseToNumberNormalizer, LetterUpperCaseToNumberNormalizer, and NonDigitRemover.
StringReplacer	Replaces substrings with strings based on a given replacement mapping.
SubstringNormalizer(0,x)	Extracts the substring from the beginning to the x-th character.
TrimNormalizer	Removes trailing or leading whitespace.
UmlautNormalizer	Converts ä → ae, Ä → Ae, ö → oe, Ö → Oe, ü → ue, Ü → Ue, and ß → ss.
UpperCaseNormalizer	Converts all letters to upper case.
WhitespaceRemover	Removes all whitespace.

Table 8.3: Overview of data cleaning functions offered by PRIMAT.

8.4.1.3 Encoding Component

The encoding component (D_3) works as follows: At first, the identifying attributes of each entity are transformed into a set of relevant record features. An overview of the feature extraction functions that are provided by PRIMAT is given in Table 8.4.

Extractor Function	Data Type	Description
QGramExtractor(q)	String	Constructs all consecutive substrings of length q where typically $1 \leq q \leq 3$. Character padding can be enabled, which adds padding characters at the beginning and end of the string [Chr12b].
SubstringByPosition-Extractor(i,j)	String	Extracts a substring from position i to position j using an optimistic approach that returns an empty string or the maximum available substring if the attribute value is too short.
SubstringByRegex-Extractor(e)	String	Extracts substrings based on the specified regular expression e.
PhoneticCode-Extractor(f)	String	Encodes the attribute value by applying a phonetic encoding function f ∈ {Soundex, Metaphone, Double Methaphone, Cologne Phonetic, NYSIIS, Beider Morse} [OR18; HM02; Phi00; Pos69; Chr06].
JaccardLSHExtractor	BitSet	Applies a permutation p on the attribute value and returns the position of the first 1-bit [Dur12].
HammingLSHExtractor	BitSet	Extracts an LSH key by sampling bits from certain (randomly selected) bit positions (see Section 3.3).
IntegerRangeExtractor	Integer	Generates a list of integer values for a given integer value n by successively adding/subtracting 1 where values outside a given range are wrapped around the boundary of the range [VC16].
IdentityExtractor	All	Applies the identity function. Useful for attributes with short values (e. g., gender) or for error-free attributes.

Table 8.4: Overview of extractor functions offered by PRIMAT.

Secondly, these record features are encoded. We focus on Bloom-filter-based encoding techniques as they are widely used in the PPRL domain [VCV13; Vat+17]. Since standard Bloom filters are vulnerable to certain types of attacks (see Section 2.8.2.2), PRIMAT also provides recent hardening techniques (discussed in Section 7.2) to enhance privacy and to limit the success of attacks. PRIMAT also provides the two-step hash encoding [RCS20] as an alternative encoding technique to encodings based on Bloom filters.

8.4.2 Linkage and Utility Module

The linkage and utility module mainly provides a wide range of methods and procedures required for linkage. It offers four components (L_1 - L_4) which we will describe in detail below.

8.4.2.1 Utilities Component

For database owners, it is challenging to select and agree on appropriate methods and parameters. Hence, the utility component (L_1) provides methods to automatically determine parameters based on masked sample data. In particular, private schema matching [Nób+18] and pre-processing templates [SCS15] are planned to support the database owners to consistently prepare their data for linkage.

8.4.2.2 Batch Linkage

The batch linkage component (L_2) is the main component of this module as it implements various linkage techniques. In particular, standard and LSH-based blocking (see Section 3.3) are provided to reduce the complexity of the linkage. We also plan to integrate metric space filtering approaches based on our previous work [SR16; Seh+21]. For comparing candidate record pairs, we provide frequently used similarity measures as listed in Table 8.5.

Similarity Function	Data Type	Description
BraunBlanquetSimilarity	Set, BitSet	$sim_{BraunBlanquet}(X,Y) = {\|X \cap Y\|}/{\max(\|X\|,\|Y\|)}$.
Dice similarity	Set, BitSet	See Equation 2.24.
Jaccard similarity	Set, BitSet	See Equation 2.23.
OverlapSimilarity	Set, BitSet	$sim_{Simpson}(X,Y) = {\|X \cap Y\|}/{\min(\|X\|,\|Y\|)}$.
Hamming similarity	BitSet	$sim_{Hamming}(X,Y) = 1 - {\|\|X \oplus Y\|\|_1}/{\max(\|\|X\|\|,\|\|Y\|\|)}$.
ContainmentSimilarity	String	Returns 1 if one string is contained in the other, and 0 otherwise.
JaroWinklerSimilarity	String	See [Chr12b].
LevenshteinSimilarity	String	See [Chr12b]
PrefixSimilarity	String	Returns 1 if one string has the other string as a prefix, and 0 otherwise.
SuffixSimilarity	String	Returns 1 if one string has the other string as a suffix, and 0 otherwise.
ExactSimilarity	All	Returns 1 if both values are equal and 0 otherwise.

Table 8.5: Overview of similarity functions offered by PRIMAT.

Furthermore, we included several post-processing methods in PRIMAT as shown in Table 8.6. As described in Chapter 5, post-processing can significantly increase linkage quality by selecting the best match candidates when there are multiple candidate records exceeding a given similarity threshold. This is beneficial if the databases to be linked are considered clean (i. e., duplicate-free) and thus a one-to-one link cardinality constraint can be enforced.

In order to determine the transitive closure, PRIMAT provides a function to calculate all connected components $\mathcal{CC}$ of a graph. In graph theory, a connected component CC is a maximal-connected subgraph of a graph G, i. e., CC is not part of any larger connected subgraph [Die17]. Each vertex and each edge belong to exactly one connected component. For each connected component $CC_i \in \mathcal{CC}$ it is ensured that all vertices in CC_i are pairwise connected by supplementing missing edges.

Post-Processing Method	**Link Constraint**	**Description**
Connected Components	Transitive Closure	Determines all connected components of a similarity graph and adds missing edges within each component.
Symmetric Best Match (Max1-Both)	1:1	Removes each edge (link) $e = (a, b) \in E$ of a similarity graph $SG = (V_A \cup V_B, E)$ with $a \in V_A, b \in V_B$ where vertex (record) a or b has another edge with a higher similarity than e.
Stable Matching (Stable Marriage)	1:1	Generates a stable matching, where no two records of the two sources both have a higher similarity to each other than to their matching record.
Maximum Weight Matching (Kuhn-Munkres Algorithm)	1:1	Generates a maximum weight matching, where the sum of the overall similarities between records in the final linkage result is maximized.
Asymmetric Best Match (Max1-Right/Left)	1:N/N:1	For any record $a \in V_A$ (or $b \in V_B$) only the best matching record of the other source is accepted. This approach does not result in a graph matching.

Table 8.6: Overview of post-processing methods offered by PRIMAT.

Finally, we plan to integrate our parallel and distributed PPRL approaches based on our previous work (see Chapter 3 and [Gla+18]). These approaches make it possible to efficiently link even tens of millions of records but generally require the commissioning of an HDFS computer cluster. Our distributed PPRL approaches are based on Apache Flink, a framework and distributed processing engine. Apache Flink also supports a local setup on a single machine utilizing available CPU cores, but this will likely introduce an overhead for small- to medium-scale linkage projects. However, even without these extensions, PRIMAT is able to efficiently link up to a million records on commodity hardware. Only the linkage of larger databases will require a high-performance server or an HDFS computer cluster.

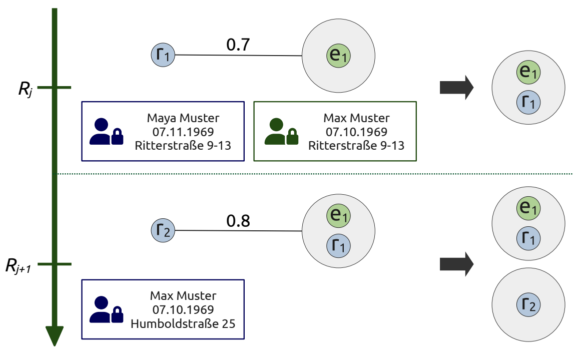
8.4.2.3 Incremental Linkage Component

The incremental linkage component (L_3) extends the batch linkage component (L_2) by providing database support to store, retrieve, and update previous match results, i. e., clusters of matching records (entity clusters). Hence, new records can be added continuously from multiple parties without repeating the complete linkage for every record.

A database is necessary to store all previously added records as well as partial linkage results, such as the resulting similarity graph and entity clusters. Also, information about the database owners and their databases needs to be stored, especially which databases are considered clean/dirty. In each linkage iteration j, the incremental linkage approach uses as input the set of new records R_j to be matched and stored, the set of existing records $R_{exist} = R_0 \cup \ldots \cup R_{j-1}$, the set of existing clusters $\chi_{exist} = \chi_0 \cup \ldots \cup \chi_{j-1}$ as well as a specific linkage configuration, including a blocking function $f_{blocking}$, a comparison method (set of similarity functions to be applied on the records' attribute values) and a decision model $\mathcal{DM}$ for classifying matches and non-matches. It is generally possible to insert only one record in an iteration, i. e., $|R_j| = 1$. However, inserting and matching multiple records (or even a complete new source database) at once can significantly improve linkage quality over record-wise matching, at least for duplicate-free source databases, as shown in [NR18].

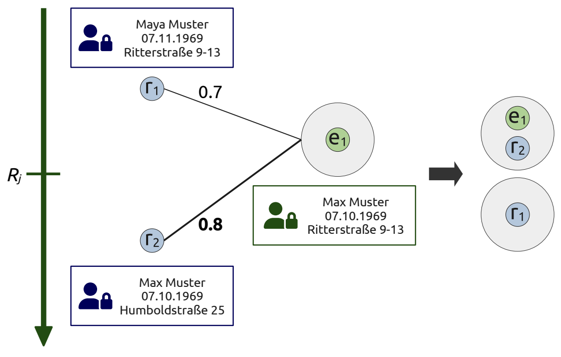
In Figure 8.3 we illustrate the drawback of a record-wise matching compared to a batch-wise matching of two records $r_1, r_2 \in D_A$ where D_A is considered as duplicate-free. If records r_1 and r_2 are inserted and matched separately, then r_1 is matched to record $e_1 \in D_B$ assuming that $sim(r_1, e_1) = 0.7 \geq t$. As a consequence, r_1 and e_1 are added to the same cluster. When r_2 is inserted, it has a higher similarity to e_1 than r_1 has to e_1. However, since the cluster of e_1 already contains a record of database D_A and D_A is considered clean, r_2 cannot be added to this cluster and therefore builds a new singleton cluster. In Figure 8.3b, records r_1 and r_2 are inserted and matched in the same linkage iteration (batch). In that case, the higher similarity of r_2 to e_1 can be taken into account, and therefore r_2 and e_1 are assigned to the same cluster, while r_1 builds a singleton cluster.

The incremental matching follows the general linkage process as depicted in Figure 2.1. We assume that the incremental linkage is conducted by a linkage unit. Therefore, the linkage unit maintains the aforementioned database, denoted as D_{LU}. The first step of the incremental linkage is blocking, which aims to limit the number of candidate record pairs. Each new record $r \in R_j$ is assigned to a set of blocks depending on the blocking function $f_{blocking}$. Without blocking, each new record needs to be compared to all previously added records, which would result in a linear complexity that grows



R_j
R_{j+1}
0.7
0.8
Maya Muster 07.11.1969 Ritterstraße 9-13
Max Muster 07.10.1969 Ritterstraße 9-13
Max Muster 07.10.1969 Humboldstraße 25

(a) Record-wise Matching



R_j
0.7
0.8
Maya Muster 07.11.1969 Ritterstraße 9-13
Max Muster 07.10.1969 Ritterstraße 9-13
Max Muster 07.10.1969 Humboldstraße 25

(b) Batch-wise Matching

Figure 8.3: Record-wise vs. batch-wise incremental matching.

with the number of records in the database D_{LU}. Each new record $r \in R_j$ as well as its blocking key values are stored in the database.

We then consider two incremental matching variants, namely record-based and cluster-based incremental matching, which we will describe in the following. Both approaches are also illustrated in Figure 8.4.

8.4.3 Record-based Incremental Matching

The record-based incremental matching approach is similar to batch matching. In the blocking step, each new record $r \in R_j$ is compared to all existing (previously added) records that have the same value for at least one blocking key. The resulting record pairs are considered candidate pairs and compared in detail (see Section 2.6.3.1).

In our example in Figure 8.4, the blocking key values for each record are indicated with two squares below each vertex. For instance, records r_3 and e_5 are considered as a candidate record pair because they have the same value for the first blocking key. In contrast, r_1 and e_5 are not compared in detail as they do not share any blocking key value.

We denote the set of candidate record pairs for iteration j as $\mathtt{C}_j$. In the next step, these candidate record pairs are compared in detail and classified based on a decision model $\mathcal{DM}$ which results in a set of (potential) matches $\mathtt{M}_j \subseteq \{(r, e) \,|\, r \in R_j, e \in R_{exist}\}$. Each record $r \in R_j$ will generally match to a specific subset of existing records, denoted as $\mathtt{M}_j(r) = \{e \,|\, (r, e) \in \mathtt{M}_i\}$. If $\mathtt{M}_j(r) = \emptyset$, then no matching record for r was found and thus r is assigned to a new (singleton) match cluster $c_r = \{r\}$ which is added to the set of new clusters χ_j, i. e., $\chi'_j = \chi_j \cup \{c_r\}$. If $|\mathtt{M}_j(r)| \geq 1$, then r matches to one or more existing records $e \in \mathtt{M}_j(r)$.

Depending on whether a source database is considered clean (duplicate-free) or dirty (containing intra-source duplicates), only one or multiple records from the same source

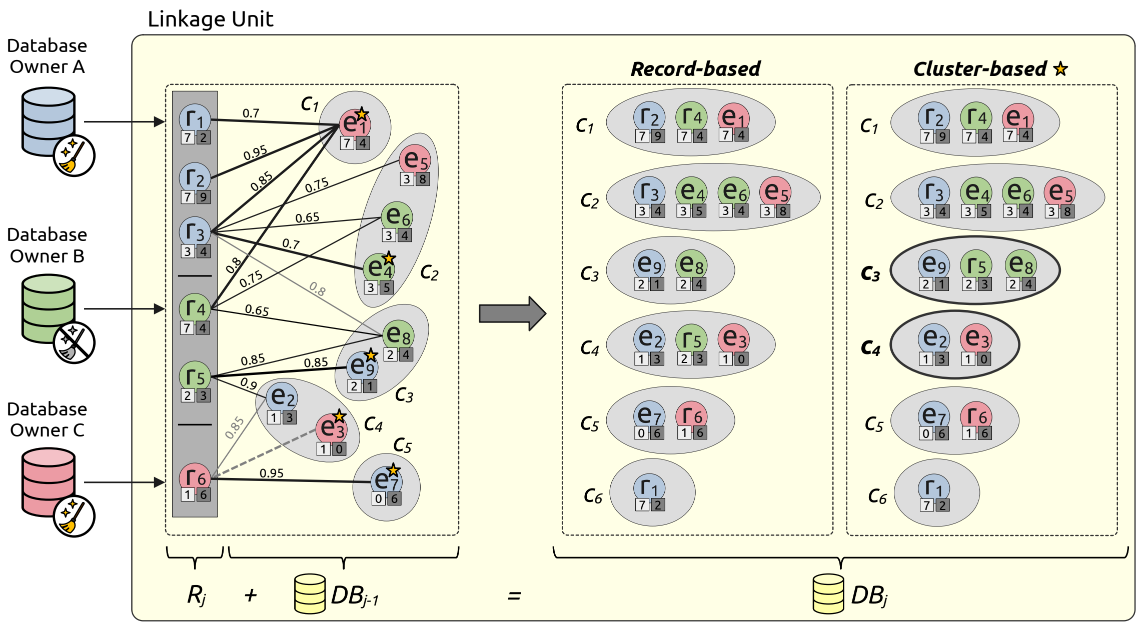


Figure 8.4: Record-based vs. cluster-based incremental matching of databases D_A, D_B, and D_C. While the databases D_A and D_C are considered clean, the database D_B is considered dirty (as indicated by the broom icon). Blocking is conducted using two blocking keys. The blocking key values for each record are indicated with two squares below each vertex. Gray edges indicate a link constraint violation and are therefore not considered as potential match candidates. In cluster-based incremental matching, a star denotes the representative of each cluster.

database can be assigned to the same match cluster. If a database is considered clean, then no two records of that database can be assigned to the same match cluster. On the other hand, if a database is considered dirty, then each match cluster may contain several records of that database. Therefore, if $r \in D_P$ and database D_P of database owner P is considered as clean, for each matching record $e \in \mathtt{M}_j(r)$ it needs to be checked, if its corresponding cluster $c(e) \in \chi_{exist}$ already contains another record from that database. In this case, e is no longer considered as a match, i. e., $\mathtt{M}'_j(r) = \mathtt{M}_j(r) \setminus \{e\}$. In our running example, databases D_A and D_C are considered as clean. Therefore, the candidate pair (r_3, e_8) is removed (as indicated with a gray edge) since the cluster $c(e_8) = c_3$ already contains record $e_9 \in D_A$. Also, the candidate pair (r_6, e_2) is removed as $c(e_2) = c_4$ already contains record $r_3 \in D_C$.

Finally, a post-processing step needs to be conducted in order to enforce the desired link constraints between clusters and records. In general, each new record can be assigned to at maximum one existing match cluster (since each match cluster represents a real-world entity). As default strategy, for each new record $r \in R_i$, therefore, we select the cluster $c(e)$ of record $e \in \mathtt{M}'_j(r)$ to which r has the highest similarity. However, there may be two (or more) records $r_1, r_2 \in R_j$ that were submitted by database owner P which corresponding database D_P is considered clean, and both, r_1 and r_2, have the highest similarity to a record $e \in R_{exist}$, or to records $e_1, e_2 \in R_{exist}$ which correspond to the same match cluster, i. e., $c(e_1) = c(e_2)$. In these cases, a one-to-one link constraint needs to be enforced (see Table 8.6). After the post-processing step, each record $r \in R_j$ is either assigned to an existing cluster $c \in \chi_{exist}$ or to a new (singleton) cluster $c' \in \chi_j$. Finally, the database D_{LU} is updated accordingly.

In our running example, records $r_1, r_2, r_3 \in D_A$ have their highest similarity to record $e_1 \in D_C$. However, since D_A is considered as clean, at maximum one record can be assigned to the corresponding cluster of e_1 which is c_1. In our example, we obtain a stable matching and therefore assign r_2 to cluster c_1, r_3 to cluster c_2, and r_1 to a new singleton cluster c_6. By using Max1-Both instead, record r_2 would be assigned to cluster c_1, while r_2 and r_3 would build two new singleton clusters.

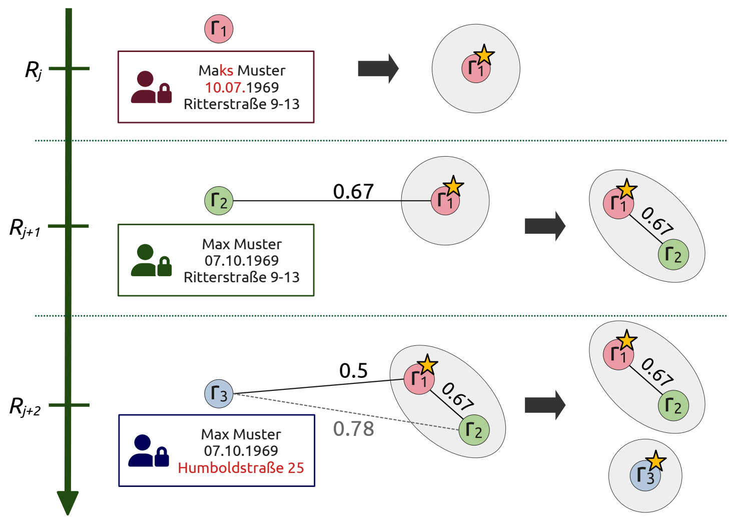
8.4.4 Cluster-based Incremental Matching

In our cluster-based incremental matching approach, each cluster is assigned a cluster representative. Similar to clustering approaches [LRU14], we distinguish two types of cluster representatives: medoids (also known as clustroids) and centroids. In general, the medoid of a cluster is the most centrally located actual data point in that cluster. In contrast, the centroid of a cluster constitutes the center of the cluster, which is not necessarily an actual data point. In the clustering literature, medoids and centroids are generally defined by using a certain measure of distance. Here, we do not use these strict definitions but rather use the terms to differentiate whether the cluster representatives are actual data points (records) or not. We denote the respective strategies as $\texttt{REPR}_{\texttt{Medoid}}$ and $\texttt{REPR}_{\texttt{Centroid}}$.

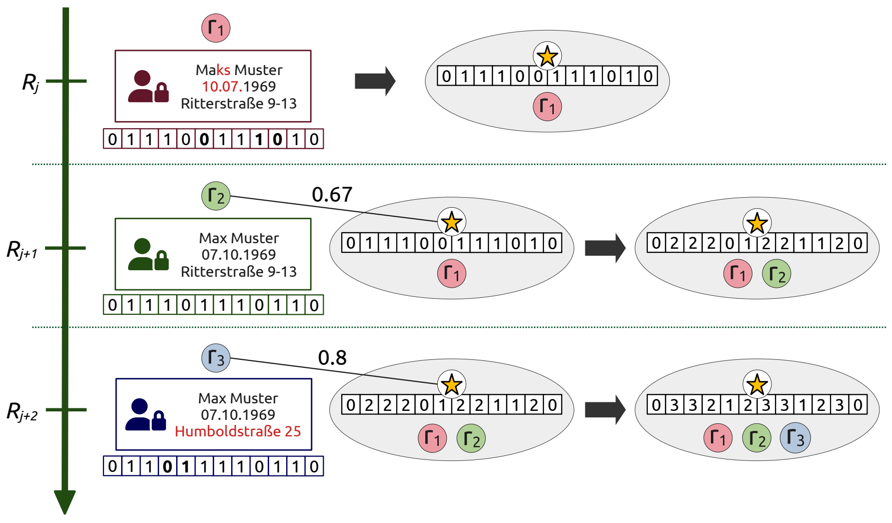
For both types of cluster representatives, we consider several selection strategies. Currently, as medoid either the oldest (first inserted) or the newest (last inserted) record of the cluster can be selected. In addition, the record with the highest average similarity to the other cluster members could be chosen. How to select the centroid of a cluster depends on the privacy technique that is used to encode the records. In this work, we focus on encodings based on Bloom filters and therefore consider the centroid of a cluster by aggregating the Bloom filters of the cluster members. A trivial approach is to combine the Bloom filters of the cluster members into a single Bloom filter by using the bitwise AND- or bitwise OR-operation. However, this approach does not seem promising, since the number of 1-bits in the combined Bloom filter is likely to become quite low (AND) or high (OR) as the cluster size increases. As a more sophisticated approach, we therefore consider combining the Bloom filters of the cluster members into an integer array by counting the number of 1-bits for each bit position. This data structure is similar to a counting Bloom filter (CBF) [Fan+00; Mit02].

In Figure 8.5, we illustrate two different strategies for selecting cluster representatives. In Figure 8.5a, the record that was assigned first to that cluster is chosen as cluster representative ($\texttt{REPR}^{\texttt{1st}}_{\texttt{Medoid}}$). The drawback of this approach is that it depends on the quality of the record that was inserted first and initially builds the cluster. In the example, r_1 is inserted first and contains some errors. While r_2 can still be matched to r_1 with a similarity of 0.67, the similarity between r_3 and r_1 is too low, and therefore r_3 is (wrongly) assigned to a singleton cluster. In Figure 8.5b, a counting Bloom filter (CBF) is maintained to represent the cluster ($\texttt{REPR}^{\texttt{CBF}}_{\texttt{Centroid}}$). The cluster representative is dynamically updated after adding a new record. As a consequence, r_3 now reaches a similarity of 0.8 (using Equation 8.2) to the cluster representative and is therefore (correctly) assigned to the same cluster as r_1 and r_2. However, these strategies require a comparative evaluation, especially considering increasing cluster sizes and different sequences for inserting records.

In our cluster-based incremental matching, also each cluster is assigned a set of blocking key values and thus blocks. For assigning blocking key values to clusters, we consider two strategies, $\texttt{BLOCK}_{\texttt{MBRS}}$ and $\texttt{BLOCK}_{\texttt{REPR}}$. The $\texttt{BLOCK}_{\texttt{MBRS}}$ strategy determines the blocking key values of a cluster depending on the cluster members, namely either by union or by intersection of the blocking key values of all records in the cluster. The $\texttt{BLOCK}_{\texttt{REPR}}$ strategy instead takes the blocking key values of the cluster representative also for the cluster itself. The blocking step results in a set of record-cluster candidate pairs (r, c)



(a) Medoid – Oldest



(b) Centroid – CBF

Figure 8.5: Strategies for selecting cluster representatives

that are built for each record $r \in R_j$ that is assigned to the same block as a cluster $c \in \chi_{exist}$.

To perform the comparison on record-cluster candidate pairs, we also consider two strategies, namely $\texttt{COMP}_{\texttt{MBRS}}$ and $\texttt{COMP}_{\texttt{REPR}}$. Currently, PRIMAT applies strategy $\texttt{COMP}_{\texttt{REPR}}$, where each record is compared to the cluster representative only, which can be either a medoid or centroid. While medoids are of the same type of data (as they are an actual cluster member), centroids may be of a different type. This is particularly the case when a cluster of records encoded as Bloom filters, i. e., bit arrays, is represented by an integer array (counting Bloom filter). Let $x \in \mathbb{B}^m$ be a bit array of size m and $y \in \mathbb{N}^m$ an integer array of size m. We can calculate the similarity between x and y by simply considering each element $y[i] > 0$ as 1 and then calculating any binary similarity measure (see Definition 2.8.1.2). For the Jaccard similarity, the similarity function is then defined as:

$$sim_{SimpleJaccard}(x, y) = \frac{\sum_{i=0}^{m-1} x[i] \wedge \operatorname{sgn}(y[i])}{\sum_{i=0}^{m-1} x[i] \vee \operatorname{sgn}(y[i])} \tag{8.1}$$

To take into account how many cluster members set a certain position in y, we consider the following similarity function, which is also based on Jaccard similarity:

$$sim(x, y)_{WeightedJaccard} = \frac{\sum_{i=0}^{m-1} x[i] \cdot \frac{y[i]}{|c(y)|}}{\sum_{i=0}^{m-1} \frac{y[i]}{|c(y)|}} = \frac{\sum_{i=0}^{m-1} x[i] \cdot y[i]}{\sum_{i=0}^{m-1} y[i]} \tag{8.2}$$

Here $|c(y)|$ denotes the size of the cluster which y represents, i. e., the number of records (members) in the cluster. Moreover, we consider the cosine similarity that is frequently used in data mining applications to compare real-valued vectors [LRU14]:

$$sim_{\cos(x,y)} = \frac{\sum_{i=0}^{m-1} x[i] \cdot y[i]}{\sqrt{\sum_{i=0}^{m-1} x[i]^2 \cdot \sum_{i=0}^{m-1} y[i]^2}} \tag{8.3}$$

For example, let $x = [0, 1, 1, 0, 0, 1]$ and $y = [1, 3, 2, 0, 0, 2]$ with $|c(y)| = 3$. Then, $sim_{SimpleJaccard}(x, y) = \frac{3}{4}$, $sim_{WeightedJaccard}(x, y) = \frac{3+2+2}{1+3+2+2} = \frac{7}{8}$, and $sim_{\cos(x,y)} = \frac{7}{\sqrt{3} \cdot \sqrt{18}} \approx 0.9526$.

Record r is then added to cluster c, where r has the highest similarity to the cluster representative. Yet, if r originates from a database that is considered clean, then c must not contain another record of this database.

When using strategy $\texttt{COMP}_{\texttt{REPR}}$, a new record is compared only with the cluster representatives. This strategy is very efficient because the comparisons are performed only on a subset of the cluster representatives. However, depending on the strategy to select the cluster representative and the quality of the cluster representative, this strategy might lead to misclassifications.

In our running example shown in Figure 8.4, we illustrate the cluster-based incremental using the strategies $\texttt{BLOCK}_{\texttt{REPR}}$ and $\texttt{COMP}_{\texttt{REPR}}$. This approach generates only 7 candidate pairs (indicated with thick lines), while the record-based incremental matching approach generates 15 candidate pairs. However, r_5 and e_3 have different values for both blocking keys and r_5 is therefore not compared to any member of cluster c_4. As a consequence, r_5 is assigned to cluster c_3 instead of cluster c_4 like in the record-based incremental matching approach.

In contrast to strategy $\texttt{COMP}_{\texttt{REPR}}$, the $\texttt{COMP}_{\texttt{MBRS}}$ strategy (additionally) considers the similarities between a record r and the members of a candidate cluster. As a consequence, this strategy requires additional record pair comparisons and is therefore less efficient than the COMP$_{\texttt{REPR}}$ strategy. To determine the optimal cluster for a new record r, different heuristics can be considered that utilize features obtained from the resulting similarity (sub)graph. In particular, we consider selecting the cluster where (1) record r has the highest average similarity to the cluster members; or (2) where r has the largest number of links (above a given threshold) to members of the cluster.

Similar to the record-based incremental linkage approach, a post-processing step is conducted to enforce the link constraints between records and clusters depending on the characteristics of the source databases. Then, each record $r \in R_j$ is either assigned to an existing or new cluster. Finally, the cluster representatives and cluster blocking key values are updated accordingly.

8.4.4.1 Evaluation Component

It is important for both practitioners and researchers to evaluate PPRL methods and workflows. For researchers, such evaluation is often straightforward since ground truth data, i. e., the correct linkage result, is usually available. However, quality metrics like precision, recall, and F-measure need to be determined to comparatively evaluate different PPRL methods and parameters. In practice, in contrast, ground truth data is generally not available and, in addition, a manual inspection of records or linkage results is often prohibited due to privacy constraints. Nevertheless, some metrics, such as runtime, reduction ratio, or the number and average size of blocks, can still indicate bad parameter or method choices. Primat allows calculating various performance measures, including our unsupervised approaches to estimating linkage quality in the absence of ground truth data (see Chapter 6). We also plan to add privacy-preserving visualization approaches that can be utilized to grant authorized experts insights into linkage results without degrading privacy [Kum+14; Rag+18]. An implementation already exists [Sch21], but an integration into Primat is currently pending.

8.5 Conclusion

In this chapter, we presented our PPRL toolbox Primat that allows a flexible definition, execution, and evaluation of PPRL workflows. Primat provides various state-of-the-art encoding and linkage techniques covering the entire PPRL process, and thus drastically reduces the effort to deploy PPRL in practice. For future work, we plan to integrate additional features in Primat, such as the planned functions for generating realistic test datasets as well as approaches for masked match result visualization. We also plan to gradually add new methods related to ongoing research in the field of PPRL. Finally, we plan to comprehensively evaluate Primat's capabilities regarding incremental matching to determine which of our proposed methods performs best.

9 Conclusion and Outlook

In this thesis, we have presented comprehensive research in scalable and accurate privacy-preserving record linkage (PPRL). In Section 9.1, we summarize our research problems and contributions of this thesis. Finally, in Section 9.2, we discuss directions for future research.

9.1 Conclusion

Privacy-preserving record linkage (PPRL) is confronted with three main challenges which are quality, scalability, and privacy. The linkage process must be effective and provide high-quality linkage results that allow the integration of data from different databases for accurate and extensive data analysis. PPRL also needs to be scalable, so that even several million records from multiple databases can be linked efficiently. Finally, PPRL needs to protect the privacy of individuals whose corresponding records are part of the linkage. Therefore, no personal data should be revealed in the linkage process. In this thesis, we tackled all three key challenges of linking sensitive data. We addressed several shortcomings identified in the area of PPRL by the following contributions:

- We developed parallel and distributed PPRL approaches using blocking techniques based on locality-sensitive hashing. Our approaches utilize Apache Flink as a modern distributed processing framework and can thus be executed on large shared-nothing computer clusters to parallelize and speed up computations. Our parallel PPRL approaches showed high efficiency and effectiveness for large synthetic and real-world datasets with up to 16 million records.
- We extended LSH-based blocking to support attribute-level encodings. We implemented our extensions within the identity management software named Mainzelliste. The Mainzelliste is widely used in the medical domain and focuses on

attribute-level encodings as they generally provide higher linkage quality at the expense of lower security properties. This compromise is necessary in some biomedical applications to ensure a very high linkage quality. Our extensions for LSH-based blocking showed drastically improved runtimes without reductions in terms of linkage quality.

- Simple threshold-based classification approaches are widely used in PPRL scenarios but can lead to low linkage quality, in particular, if the threshold is not chosen appropriately. In addition, depending on the characteristics of the source databases, certain link constraints arise that are not taken into account by a pairwise classification of record pairs. To enforce such link constraints and to reduce the impact of thresholds that have been set too low, we proposed different post-processing strategies to resolve multiple match candidates. We showed that post-processing raises the overall linkage quality, in particular in PPRL scenarios that deal with dense or dirty data. Moreover, the linkage quality is far more stable for lower thresholds when using post-processing. This becomes highly beneficial in practical applications where an optimal threshold value is usually hard to define.
- Another major contribution of our work relates to the evaluation of PPRL algorithms. Assessing linkage quality is difficult in real-world applications, as ground truth data is often not available and manual verification of match results is not possible due to privacy concerns. Therefore, we proposed novel unsupervised approaches for privately estimating the linkage quality. Based on these estimates, methods, and parameters can be adjusted and optimized. Our evaluation using several real-world datasets from different domains showed that our estimates are close to the actual linkage quality.
- We have also conducted a comprehensive and comparative evaluation of different hardening techniques for encodings based on Bloom filters. In this context, we have proposed a novel set of privacy measures that assess the uniformity of the frequency distribution of 1-bits within the encoded records. Previous approaches for measuring privacy often depend on a reference dataset or a certain type of attack. In contrast, our proposed privacy measures solely depend on a Bloom filter dataset. The evaluation showed that multiple hardening techniques lead to poor linkage quality and therefore cannot be used in practice. A few hardening techniques, however, are able to maintain a high linkage quality while reducing any frequency information, making frequency-based cryptanalysis unlikely to be successful.
- We have developed an open-source PPRL toolbox called PRIMAT that combines several state-of-the-art encoding and linkage methods into a single framework that allows both practitioners and researchers to easily set up and comparatively

evaluate different PPRL pipelines. Primat offers several modules for data owners and the linkage unit that can be integrated as a library (dependency) into existing projects to provide PPRL functions. In addition, Primat offers two linkage modes, namely batch matching and incremental matching. To the best of our knowledge, Primat is the only open-source PPRL tool that covers the entire PPRL process and provides full support for both linkage modes.

Overall, our approaches improve the applicability of PPRL approaches in real-world applications where large datasets need to be matched accurately and efficiently. However, there are still several open research questions in the area of PPRL that need to be addressed.

9.2 Outlook

Finally, we discuss several open research questions that are left for future work.

Enhanced privacy measures for PPRL: The aim of privacy measures is to quantify the degree of privacy and the amount of protection that is offered by a specific privacy-preserving technology. Such measures are therefore necessary to objectively compare different competing approaches in a standardized way. While different privacy measures have been proposed in the literature [WE18], there is currently no single all-purpose privacy measure that is commonly accepted [CRS20]. Due to the diversity and complexity of privacy measures, choosing an appropriate privacy measure is challenging [WE18]. Moreover, such measures should ideally be applicable to different encoding techniques developed for PPRL. They should also take various factors into account, such as certain characteristics of the databases to be linked (e. g., size of databases or frequency distribution of values), as well as the adversary's capabilities and goals. In addition, privacy measures are required that do not only provide a single measure for each database but also provide assessment for individual, particularly vulnerable subsets of records. This indicates the need for novel multidimensional privacy measures.

As discussed in Chapter 7, most existing privacy measures for PPRL rely on a global reference dataset or depend on the re-identification rate of a specific attack and its assumptions. Therefore, we proposed a novel set of privacy measures that solely depend on the encoded (Bloom filter) database and measure the uniformity (evenness) of the 1-bit distribution. The more uniform the 1-bit distribution is, the less likely any frequency-based cryptanalysis will be successful. However, a uniform 1-bit distribution does not necessarily imply a fully patternless outcome that is indistinguishable from a randomly generated output. Therefore, more attention towards this direction is required and novel measures to quantify privacy in terms of randomness are needed. To assess the extent of correlation between different encoding fragments (e. g., bit positions) that could be exploited in attack scenarios, correlation measures such as the Phi coefficient are interesting to investigate.

Selective hardening for vulnerable values: Hardening techniques aim to improve the privacy properties of encoding techniques in order to prevent certain types of attacks. Until now, typically only a single hardening technique is applied on the entire input databases to be linked. Therefore, the vulnerability of individuals or groups of encoded records is not considered. Since hardening techniques generally show a trade-off between linkage quality and privacy, a compelling research direction is to analyze which minimum combination of techniques is required to minimize the likelihood of success regarding different attacks while maintaining a high linkage quality. Ideally, an approach would automatically identify all vulnerable subsets of records in the input databases and then determine a minimal set of techniques that are required to protect these subsets against known attacks.

Overcoming similarity vulnerabilities: To address errors and inconsistencies in real-world data, PPRL encoding techniques need to support approximate matching and thus the calculation of similarities between encoded records. As a consequence, the similarities between encoded records can be compared to the similarities of plaintext records, e. g., by using publicly available datasets. Since the encodings are similarity-preserving, there remains a relationship between the plaintext and the encoded records. Most attacks on PPRL developed so far exploit the length and frequency information of encoded values. However, recent graph-based similarity attacks exploit this relationship by analyzing and matching the similarity graphs of the encoded and the plaintext records. So far, only a few works have focused on hardening or novel encoding techniques that aim at avoiding attacks that exploit the similarity neighborhood of encodings. Therefore, new encoding techniques and protocols are needed to hide or perturb the similarities between encoded records to prevent such attacks.

Evaluation of different encoding schemes: Encodings based on Bloom filters [SBR11] have become the quasi-standard in recent PPRL approaches in both research and practical applications [Vat+17; Gko+21]. At the same time, several vulnerabilities were identified, and successful attacks were mounted on such encodings [Vid+23]. Different hardening techniques and several alternative encoding techniques, such as the use of tabulation min-hash [Smi17], two-step hash encoding [RCS20], and autoencoders [Chr+22], have therefore been proposed. Currently, however, a comprehensive evaluation of these different encoding schemes in terms of linkage quality and privacy guarantees is missing. It also remains unclear how alternative encoding techniques can withstand novel pattern mining or graph-based attacks.

Advanced classification techniques: Another important research question is how to employ advanced classification techniques in a privacy-preserving context. In particular, the recent focus on the use of record-level encodings due to their lower vulnerability against attacks limits the choice of classification methods. As a consequence, more sophisticated classification techniques developed for traditional record linkage often cannot be applied in PPRL scenarios. This also restricts the handling of null values, swapped attributes, or heterogeneous databases with different database schemes. More efforts are needed in this direction of research to achieve high-quality linkage results in real-world PPRL applications.

Protocols without a linkage unit: Most existing PPRL approaches that have been proposed in the literature rely on a trusted third party, the linkage unit, that conducts the actual linkage of encoded records [Vat+17; Gko+21]. In general, protocols that utilize a linkage unit are straightforward to deploy and efficient in terms of communication and computation costs. In contrast, if the linkage is solely conducted among the database owners, the protocols are more complex and typically require high communication and computation costs to ensure that no linkage participant can learn something about another party's private data [CRS20]. In practice, however, determining a unit or organization that can act as a linkage unit is often challenging. This is because such an organization must be trustworthy, as well as independent of the database owners involved in the linkage. In this regard, further research is needed that investigates efficient protocols without a linkage unit, such as scalable techniques based on secure multi-party computation (SMC) or homomorphic encryption approaches.

PPRL for other adversarial models: Most PPRL approaches and attack scenarios assume an honest-but-curious (HBC) adversarial model [Vat+17; Gko+21]. In practice, however, this assumption may be insufficient, in particular in multi-party PPRL scenarios where potentially many parties participate in the linkage, and the parties may not know and trust each other. Such multi-party linkage scenarios also increase the risk of privacy breaches due to possible collusion between a subset of parties. Future work is required to explore threats and countermeasures for a range of malicious adversarial models. For example, it would be interesting to investigate how honest linkage participants can detect different types of misbehavior by dishonest parties.

Combination of PPRL and privacy-preserving machine learning: The goal of privacy-preserving machine learning (PPML) is to allow the analysis of person-related data with machine learning methods while guaranteeing a high degree of privacy to prevent the identity of individuals to be revealed [AC19]. A main focus of PPML has been to perturb database records or analysis results, for instance, by generalization of values or by introducing noise with differential privacy techniques [Liu+21]. While there has been a substantial degree of research on PPML and PPRL, the combination of both has achieved almost no attention so far. State-of-the-art PPML methods [Pap+18] train machine learning models at different database owners, combine the learned models for improved classification, and then apply differential privacy [Dwo06] to the learned results for improved privacy. However, such approaches are unable to utilize the combined and complementing information about individuals from the different databases as made possible by the additional use of PPRL. A task for future work is to investigate novel combinations of PPRL and PPML to support the enhanced use of PPML in applications that benefit from combined personal data from multiple databases.

List of Figures

List of Tables

References

[AC19] Mohammad Al-Rubaie and J. Morris Chang. In: *IEEE Security & Privacy* 17.2 (2019), pp. 49–58. DOI: 10.1109/MSEC.2018.2888775.

[AES03] Rakesh Agrawal, Alexandre V. Evfimievski, and Ramakrishnan Srikant. “Information Sharing Across Private Databases.” In: *ACM SIGMOD International Conference on Management of Data*. ACM, 2003, pp. 86–97. DOI: 10.1145/872757.872771.

[AGK12] Mohammad Alaggan, Sébastien Gambs, and Anne-Marie Kermarrec. “BLIP: Non-interactive Differentially-Private Similarity Computation on Bloom filters.” In: *Stabilization, Safety and Security of Distributed Systems*. 2012, pp. 202–216. DOI: 10.1007/978-3-642-33536-5_20.

[AHS23] Frederik Armknecht, Youzhe Heng, and Rainer Schnell. “Strengthening Privacy-Preserving Record Linkage Using Diffusion.” In: *Proceedings on Privacy Enhancing Technologies Symposium (POPETS)*. Vol. 2023. 2. 2023, pp. 298–311. DOI: 10.56553/popets-2023-0054.

[AKM13] Hotham Altwaijry, Dmitri V. Kalashnikov, and Sharad Mehrotra. “Query-Driven Approach to Entity Resolution.” In: *Proceedings of the VLDB Endowment*. Vol. 6. 14. 2013, pp. 1846–1857. DOI: 10.14778/2556549.2556567.

[Arp+18] Daniel Arp, Erwin Quiring, Tammo Krueger, Stanimir Dragiev, and Konrad Rieck. “Privacy-Enhanced Fraud Detection with Bloom Filters.” In: *Security and Privacy in Communication Networks*. Springer, 2018, pp. 396–415. DOI: 10.1007/978-3-030-01701-9_22.

[AS94] Rakesh Agrawal and Ramakrishnan Srikant. “Fast Algorithms for Mining Association Rules in Large Databases.” In: *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB)*. Morgan Kaufmann Publishers Inc., 1994, pp. 487–499. ISBN: 1-55860-153-8.

[BAQ01] F. Borst, F.-A. Allaert, and C. Quantin. “The Swiss solution for anonymously chaining patient files.” In: *Studies in Health Technology and Informatics* (2001), pp. 1239–1241. DOI: 10.3233/978-1-60750-928-8-1239.

[BBR11] Zohra Bellahsene, Angela Bonifati, and Erhard Rahm, eds. *Schema Matching and Mapping*. Springer, 2011. ISBN: 978-3-642-16518-4. DOI: 10.1007/978-3-642-16518-4.

[Ber+16] Inga Bernemann, Markus Kersting, Jana Prokein, Michael Hummel, Norman Klopp, and Thomas Illig. "Zentralisierte Biobanken als Grundlage für die medizinische Forschung." In: *Bundesgesundheitsblatt* 59 (2016), pp. 336–343. DOI: 10.1007/s00103-015-2295-2.

[Bey04] Paul Beynon-Davies. *Database Systems*. 3. Palgrave Macmillan, 2004. ISBN: 978-1403916013.

[BG02] J. Bass and C. Garfield. "Statistical linkage keys: How effective are they?" In: *Symposium on Health Data Linkage*. 2002, pp. 40–45. URL: https://phidu.torrens.edu.au/pdf/1999-2004/symposium-proceedings-2003/bass.pdf (visited on 07/05/2023).

[BGH11] Guilherme Dal Bianco, Renata Galante, and Carlos A. Heuser. "A fast approach for parallel deduplication on multicore processors." In: *Proceedings of the 2011 ACM Symposium on Applied Computing (SAC)*. ACM, 2011, pp. 1027–1032. DOI: 10.1145/1982185.1982411.

[Bin+22] Olivier Binette, Sokhna A York, Emma Hickerson, Youngsoo Baek, Sarvo Madhavan, and Christina Jones. *Estimating the Performance of Entity Resolution Algorithms: Lessons Learned Through PatentsView.org*. 2022. DOI: 10.48550/ARXIV.2210.01230.

[Blo70] Burton Bloom. "Space/Time Trade-offs in Hash Coding with Allowable Errors." In: *Communications of the ACM (CACM)* 13.7 (1970), pp. 422–426. DOI: 10.1145/362686.362692.

[BM04] Andrei Broder and Michael Mitzenmacher. "Network Applications of Bloom Filters: A Survey." In: *Internet Mathematics* 1.4 (2004), pp. 485–509. DOI: 10.1080/15427951.2004.10129096.

[Böh+12] Christoph Böhm, Gerard de Melo, Felix Naumann, and Gerhard Weikum. "LINDA: Distributed Web-of-Data-Scale Entity Matching." In: *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM)*. ACM, 2012, pp. 2104–2108. DOI: 10.1145/2396761.2398582.

[Boy+16] James H. Boyd, Tenniel Guiver, Sean M. Randall, Anna M. Ferrante, J. B. Semmens, Phil Anderson, and Teresa Dickinson. "A simple sampling method for estimating the accuracy of large scale record linkage projects." In: *Methods of Information in Medicine* 55.03 (2016), pp. 276–283. DOI: 10.3414/ME15-01-0152.

[BRF15] James H. Boyd, Sean M. Randall, and Anna M. Ferrante. "Application of Privacy-Preserving Techniques in Operational Record Linkage Centres." In: *Medical Data Privacy Handbook*. Springer, 2015, pp. 267–287. ISBN: 978-3-319-23633-9. DOI: 10.1007/978-3-319-23633-9_11.

[Bro+17] Adrian P. Brown, Christian Borgs, Sean M. Randall, and Rainer Schnell. "Evaluating privacy-preserving record linkage using cryptographic long-term keys and multibit trees on large medical datasets." In: *BMC Medical Informatics and Decision Making* 17.83 (2017). DOI: 10.1186/s12911-017-0478-5.

[Bro97] Andrei Z. Broder. "On the resemblance and containment of documents." In: *Compression and Complexity of Sequences Proceedings*. IEEE, 1997, pp. 21–29. DOI: 10.1109/SEQUEN.1997.666900.

[BS22] Olivier Binette and Rebecca C. Steorts. "(Almost) All of Entity Resolution." In: *Science Advances* 8.12 (2022). DOI: 10.1126/sciadv.abi8021.

[Buc12] Johannes Buchmann. *Internet Privacy: Eine multidisziplinäre Bestandsaufnahme / A multidisciplinary analysis*. Springer, 2012. DOI: 10.1007/978-3-642-31943-3.

[Bun23] Bundesministerium für Gesundheit (BMG). *Gesetzliche Krankenversicherung – Mitglieder, mitversicherte Angehörige und Krankenstand (Monatswerte Januar-Dezember 2022)*. 2023. URL: https://www.bundesgesundheitsministerium.de/fileadmin/Dateien/3_Downloads/Statistiken/GKV/Mitglieder_Versicherte/Januar_bis_Dezember_2022_bf.pdf (visited on 06/28/2023).

[BW15] Manuel Burkhart and Birgitt Wiese. *Deutsches Mukoviszidose-Register – Berichtsband 2015*. 2015. URL: https://www.muko.info/fileadmin/user_upload/angebote/qualitaetsmanagement/register/berichtsbaende/berichtsband_2015.pdf (visited on 03/03/2020).

[Can+18] Rémi Canillas, Rania Talbi, Sara Bouchenak, Omar Hasan, Lionel Brunie, and Laurent Sarrat. "Exploratory Study of Privacy Preserving Fraud Detection." In: *Proceedings of the 19th International Middleware Conference Industry*. ACM, 2018, pp. 25–31. DOI: 10.1145/3284028.3284032.

[Car+15] Paris Carbone, Asterios Katsifodimos, Stephan Ewen, Volker Markl, Seif Haridi, and Kostas Tzoumas. "Apache Flink: Stream and Batch Processing in a Single Engine." In: *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering (TCDE)* 38.4 (2015).

[Cau15] Jörg Caumanns. *100% Standards: CDA, FHIR, CTS-2 und EFA für elektronische Fragebögen*. 2015. URL: https://cdn3.scrivito.com/fokus/57a537e2ec27cb7b/0a3a0655dcc079f58890e39dbdca4781/E-HEALTH_Standards_PB_03-2015_v03.pdf (visited on 03/03/2020).

[CCT10] Seung-Seok Choi, Sung-Hyuk Cha, and Charles C Tappert. "A survey of binary similarity and distance measures." In: *Journal of Systemics, Cybernetics and Informatics* 8.1 (2010), pp. 43–48. DOI: 10.1.1.352.6123.

[Cha+21] Panagiotis Charalampopoulos, Huiping Chen, Peter Christen, Grigorios Loukides, Nadia Pisanti, Solon P. Pissis, and Jakub Radoszewski. "Pattern Masking for Dictionary Matching." In: *32nd International Symposium on Algorithms and Computation (ISAAC 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik GmbH, 2021, pp. 1–19. DOI: 10.4230/LIPIcs.ISAAC.2021.65.

[Che+05] Jonathan Chen, S. Joshua Swamidass, Yimeng Dou, Jocelyne Bruand, and Pierre Baldi. "ChemDB: A Public Database of Small Molecules and Related Chemoinformatics Resources." In: *Bioinformatics* 21.22 (2005), pp. 4133–4139. DOI: 10.1093/bioinformatics/bti683.

[Che76] Peter Pin-Shan Chen. "The Entity-Relationship Model—toward a Unified View of Data." In: *ACM Transactions on Database Systems (TODS)* 1.1 (1976), pp. 9–36. DOI: 10.1145/320434.320440.

[ChI15] ChILD-EU Research Consortium. *Ethics/Data Safety*. 2015. URL: http://www.klinikum.uni-muenchen.de/Child-EU/en/child-eu-register/register/ethics_data_safety/index.html (visited on 06/27/2023).

[CHK23] Peter Christen, David J. Hand, and Nishadi Kirielle. "A Review of the F-Measure: Its History, Properties, Criticism, and Alternatives." In: *ACM Computing Surveys* (2023). DOI: 10.1145/3606367.

[Chr+18a] Peter Christen, Thilina Ranbaduge, Dinusha Vatsalan, and Rainer Schnell. "Precise and Fast Cryptanalysis for Bloom Filter Based Privacy-Preserving Record Linkage." In: *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 31.11 (2018), pp. 2164–2177. DOI: 10.1109/TKDE.2018.2874004.

[Chr+18b] Peter Christen, Anushka Vidanage, Thilina Ranbaduge, and Rainer Schnell. "Pattern-Mining Based Cryptanalysis of Bloom Filters for Privacy-Preserving Record Linkage." In: *Advances in Knowledge Discovery and Data Mining (PAKDD 2018)*. Vol. 10939. Springer, 2018, pp. 530–542. DOI: 10.1007/978-3-319-93040-4_42.

[Chr+22] Victor Christen, Tim Häntschel, Peter Christen, and Erhard Rahm. “Privacy-Preserving Record Linkage Using Autoencoders.” In: *International Journal of Data Science and Analytics* 15.4 (2022), pp. 347–357. DOI: 10.1007/s41060-022-00377-2.

[Chr06] Peter Christen. “A Comparison of Personal Name Matching: Techniques and Practical Issues.” In: *Data Mining Workshops, 2006. ICDM Workshops 2006. Sixth IEEE International Conference on*. IEEE, 2006, pp. 290–294.

[Chr12a] Peter Christen. “A Survey of Indexing Techniques for Scalable Record Linkage and Deduplication.” In: *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 24.9 (2012), pp. 1537–1555. DOI: 10.1109/TKDE.2011.127.

[Chr12b] Peter Christen. *Data Matching. Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Springer, 2012. DOI: 10.1007/978-3-642-31164-2.

[Con+05] Paolo Contiero, A. Tittarelli, G. Tagliabue, A. Maghini, S. Fabiano, P. Crosignani, and R. Tessandori. “The EpiLink Record Linkage Software: Presentation and Results of Linkage Test on Cancer Registry Files.” In: *Methods of information in medicine* 44.01 (2005), pp. 66–71. DOI: 10.1055/s-0038-1633924.

[Cou+21] T. G. Coulson, M. Bailey, C. Reid, G. Shardey, Williams-Spence. J., S. Huckson, S. Chavan, and D. Pilcher. “Linkage of Australian national registry data using a statistical linkage key.” In: *BMC Medical Informatics and Decision Making* 21.37 (2021). DOI: 10.1186/s12911-021-01393-1.

[Cou50] Council of Europe. *Convention for the Protection of Human Rights and Fundamental Freedoms*. 1950. URL: https://www.echr.coe.int/documents/d/echr/convention_ENG (visited on 10/13/2023).

[CRS20] Peter Christen, Thilina Ranbaduge, and Rainer Schnell. *Linking Sensitive Data. Methods and Techniques for Practical Privacy-Preserving Information Sharing*. Springer, 2020. DOI: 10.1007/978-3-030-59706-1.

[CSS18] Xiao Chen, Eike Schallehn, and Gunter Saake. “Cloud-Scale Entity Resolution: Current State and Open Challenges.” In: *Open Journal of Big Data (OJBD)* 4.1 (2018), pp. 30–51. URL: http://nbn-resolving.de/urn:nbn:de:101:1-201804155766 (visited on 06/30/2023).

[CV12] Lidia Ceriani and Paolo Verme. “The Origins of the Gini Index: Extracts from Variabilità e Mutabilità (1912) by Corrado Gini.” In: *The Journal of Economic Inequality* 10.3 (2012), pp. 421–443. DOI: 10.1007/s10888-011-9188-x.

[CV13] Peter Christen and Dinusha Vatsalan. “Flexible and extensible generation and corruption of personal data.” In: *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management (CIKM)*. ACM, 2013, pp. 1165–1168. DOI: 10.1145/2505515.2507815.

[CZ18] Lianhua Chi and Xingquan Zhu. “Hashing Techniques: A Survey and Taxonomy.” In: *ACM Computing Surveys* 50.1 (2018), pp. 1–36. DOI: 10.1145/3047307.

[Dao22] Duc Dung Dao. “Konzeption und Realisierung einer Anwendung zur Generierung von personenbezogenen Daten.” Bachelor’s Thesis. Universität Leipzig, 2022.

[Dat04] Christopher John Date. *An Introduction to Database Systems*. 8. Addison-Wesley, 2004. ISBN: 0-321-18956-6.

[Deu23] Deutsche Telekom AG. *Deutsche Telekom Company Presentation*. 2023. URL: https://www.telekom.com/resource/blob/326540/f23a7bf1dd5b37cdc4af9ad02f2b4c35/dl-presentation-deutsche-telekom-data.pdf (visited on 06/28/2023).

[DG08] Jeffrey Dean and Sanjay Ghemawat. “MapReduce: Simplified Data Processing on Large Clusters.” In: *Communications of the ACM (CACM)* 51.1 (2008).

[DH19] James C. Doidge and Katie L. Harron. “Reflections on modern methods: linkage error bias.” In: *International Journal of Epidemiology* 48.6 (2019), pp. 2050–2060. DOI: 10.1093/ije/dyz203.

[Dic45] Lee R. Dice. “Measures of the Amount of Ecologic Association Between Species.” In: *Ecology* 26.3 (1945), pp. 297–302. DOI: 10.2307/1932409.

[Die17] Reinhard Diestel. *Graph Theory*. 5. Springer, 2017. DOI: 10.1007/978-3-662-53622-3.

[DQB95] L. Dusserre, C. Quantin, and H. Bouzelat. “A One Way Public Key Cryptosystem for the Linkage of Nominal Files in Epidemiological Studies.” In: *Medinfo* 8 (1995), pp. 644–647. URL: https://pubmed.ncbi.nlm.nih.gov/8591288/ (visited on 06/28/2023).

[DR02] Hong-Hai Do and Erhard Rahm. “COMA - A System for Flexible Combination of Schema Matching Approaches.” In: *Proceedings of the 28th international conference on Very Large Data Bases (VLDB)*. VLDB Endowment, 2002, pp. 610–621. DOI: 10.5555/1287369.1287422.

[Dun46] Halbert L. Dunn. “Record linkage.” In: *American Journal of Public Health and the Nations Health* 36.12 (1946), pp. 1412–1416. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1624512/ (visited on 06/28/2023).

[Dur+14] Elizabeth A. Durham, Murat Kantarcioglu, Yuan Xue, Csaba Toth, Mehmet Kuzu, and Bradley Malin. “Composite Bloom Filters for Secure Record Linkage.” In: *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 26.12 (2014), pp. 2956–2968. DOI: 10.1109/TKDE.2013.91.

[Dur12] Elizabeth Ashley Durham. “A Framework for Accurate, Efficient Private Record Linkage.” PhD thesis. Vanderbilt University, 2012. URL: https://etd.library.vanderbilt.edu/etd-03262012-144837 (visited on 06/27/2023).

[Dwo06] Cynthia Dwork. “Differential Privacy.” In: *Automata, Languages and Programming*. Springer, 2006, pp. 1–12. DOI: 10.1007/11787006_1.

[Eft+17] Vasilis Efthymiou, George Papadakis, George Papastefanatos, Kostas Stefanidis, and Themis Palpanas. “Parallel Meta-Blocking for Scaling Entity Resolution over Big Heterogeneous Data.” In: *Information Systems* 65.C (2017), pp. 137–157. DOI: 10.1016/j.is.2016.12.001.

[Ege+15] Rolf Egert, Marc Fischlin, David Gens, Sven Jacob, Matthias Senker, and Jörn Tillmanns. “Privately Computing Set-Union and Set-Intersection Cardinality via Bloom Filters.” In: *Information Security and Privacy*. Vol. 9144. 2015, pp. 413–430. DOI: 10.1007/978-3-319-19962-7_24.

[EIV07] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. “Duplicate Record Detection: A survey.” In: *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 19.1 (2007), pp. 1–16. DOI: 10.1109/TKDE.2007.250581.

[ES03] D. M. Endres and J. E. Schindelin. “A New Metric for Probability Distributions.” In: *IEEE Transactions on Information Theory* 49.7 (2003), pp. 1858–1860. DOI: 10.1109/TIT.2003.813506.

[Eur12] European Parliament, Council & Commission. “Charter of Fundamental Rights of the European Union.” In: *Official Journal of the European Union (OJ C 326)* (2012), pp. 391–407. DOI: 10.3000/1977091X.C_2012.326.eng.

[Eur16a] European Parliament & Council. “Directive (EU) 2016/681 of the European Parliament and of the Council of 27 April 2016 on the use of passenger name record (PNR) data for the prevention, detection, investigation and prosecution of terrorist offences and serious crime.” In: *Official Journal of* *the European Union Legislation series 119 (OJ L 119)* 59 (2016), pp. 132–149. ISSN: 1977-0677.

[Eur16b] European Parliament & Council. "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)." In: *Official Journal of the European Union Legislation series 119 (OJ L 119)* 59 (2016), pp. 1–88. ISSN: 1977-0677.

[EVE02] Mohamed G. Elfeky, Vassilios S. Verykios, and Ahmed K. Elmagarmid. "TAILOR: A Record Linkage Toolbox." In: *Proceedings of the 18th International Conference on Data Engineering (ICDE)*. IEEE, 2002, pp. 17–28. DOI: 10.1109/ICDE.2002.994694.

[Fan+00] Li Fan, Pei Cao, Jussara Almeida, and Andrei Z. Broder. "Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol." In: *Proceedings of IEEE/ACM Transactions on Networking (TON)* 8.3 (2000), pp. 281–293. DOI: 10.1109/90.851975.

[Fis+15] Jeffrey Fisher, Peter Christen, Qing Wang, and Erhard Rahm. "A Clustering-Based Framework to Control Block Sizes for Entity Resolution." In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2015. DOI: 10.1145/2783258.2783396.

[FL83] Wendy R. Fox and Gabriel W. Lasker. "The Distribution of Surname Frequencies." In: *International Statistical Review* 51.1 (1983), pp. 81–87. DOI: 10.2307/1402733.

[FNP04] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. "Efficient Private Matching and Set Intersection." In: *Advances in Cryptology - EUROCRYPT 2004*. Vol. 3027. Springer, 2004, pp. 1–19. DOI: 10.1007/978-3-540-24676-3_1.

[For+13] Benedikt Forchhammer, Thorsten Papenbrock, Thomas Stening, Sven Viehmeier, Uwe Draisbach, and Felix Naumann. "Duplicate Detection on GPUs." In: *Proceedings Datenbanksysteme für Business, Technologie und Web (BTW)*. Gesellschaft für Informatik, 2013. ISBN: 978-3-88579-608-4. URL: https://dl.gi.de/items/83d5091b-b2fc-4c28-bb09-5a55d95b6d80 (visited on 06/27/2023).

[Fra+18] Martin Franke, Ziad Sehili, Marcel Gladbach, and Erhard Rahm. “Post-Processing Methods for High Quality Privacy-Preserving Record Linkage.” In: *Data Privacy Management, Cryptocurrencies and Blockchain Technology (DPM, CBT)*. Springer, 2018, pp. 263–278. DOI: 10.1007/978-3-030-00305-0_19.

[Fra+19] Martin Franke, Marcel Gladbach, Ziad Sehili, Florens Rohde, and Erhard Rahm. “ScaDS Research on Scalable Privacy-preserving Record Linkage.” In: *Datenbank-Spektrum* 19.1 (2019), pp. 31–40. DOI: 10.1007/s13222-019-00305-y.

[Fra+21] Martin Franke, Ziad Sehili, Florens Rohde, and Erhard Rahm. “Evaluation of Hardening Techniques for Privacy-Preserving Record Linkage.” In: *Proceedings of the 24th International Conference on Extending Database Technology (EDBT)*. OpenProceedings.org, 2021. DOI: 10.5441/002/EDBT.2021.26.

[Fra+24] Martin Franke, Victor Christen, Peter Christen, Florens Rohde, and Erhard Rahm. “(Privately) Estimating Linkage Quality for Record Linkage.” In: *Proceedings of the 27th International Conference on Extending Database Technology (EDBT)*. OpenProceedings.org, 2024.

[Fra23] Martin Franke. *PRIMAT: Private Matching Toolbox*. 2023. URL: https://git.informatik.uni-leipzig.de/dbs/pprl/primat (visited on 06/28/2023).

[FS69] Ivan P. Fellegi and Alan B. Sunter. “A Theory for Record Linkage.” In: *Journal of the American Statistical Association (JASA)* 64.328 (1969), pp. 1183–1210. DOI: 10.1080/01621459.1969.10501049.

[FSR18] Martin Franke, Ziad Sehili, and Erhard Rahm. “Parallel Privacy-preserving Record Linkage using LSH-based Blocking.” In: *Proceedings of the 3rd International Conference on Internet of Things, Big Data and Security (IoTBDS)*. SCITEPRESS - Science and Technology Publications, 2018, pp. 195–203. DOI: 10.5220/0006682701950203.

[FSR19] Martin Franke, Ziad Sehili, and Erhard Rahm. “PRIMAT: A Toolbox for Fast Privacy-Preserving Matching.” In: *Proceedings of the VLDB Endowment*. Vol. 12. 12. 2019, pp. 1826–1829. DOI: 10.14778/3352063.3352076.

[FT04] B. Fuglede and F. Topsoe. “Jensen-Shannon Divergence and Hilbert Space Embedding.” In: *International Symposium on Information Theory (ISIT) Proceedings*. 2004. DOI: 10.1109/ISIT.2004.1365067.

[Gas72] Joseph L. Gastwirth. “The Estimation of the Lorenz Curve and Gini Index.” In: *The Review of Economics and Statistics* 54.3 (1972), p. 306. DOI: 10.2307/1937992.

[GH21] Leonardo Gazzarri and Melanie Herschel. “End-to-End Task Based Parallelization for Entity Resolution on Dynamic Data.” In: *Proceedings of the 2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 1248–1259. DOI: 10.1109/ICDE51399.2021.00112.

[GI89] Dan Gusfield and Robert W. Irving. *The Stable Marriage Problem: Structure and Algorithms*. MIT Press, 1989. ISBN: 978-0262515528. DOI: 10.5555/68392.

[Gib+16] Alison Gibberd, Rajah Supramaniam, Anthony Dillon, Bruce K. Armstrong, and Dianne L. O’Connel. “Lung cancer treatment and mortality for Aboriginal people in New South Wales, Australia: results from a population-based record linkage study and medical record audit.” In: *BMC Cancer* 16.1 (2016), p. 289. DOI: 10.1186/s12885-016-2322-1.

[Gko+21] Aris Gkoulalas-Divanis, Dinusha Vatsalan, Dimitrios Karapiperis, and Murat Kantarcioglu. “Modern Privacy-Preserving Record Linkage Techniques: An Overview.” In: *IEEE Transactions on Information Forensics and Security (TIFS)* 16 (2021), pp. 4966–4987. DOI: 10.1109/TIFS.2021.3114026.

[Gla+18] Marcel Gladbach, Ziad Sehili, Thomas Kudrass, Peter Christen, and Erhard Rahm. “Distributed Privacy-Preserving Record Linkage Using Pivot-Based Filter Techniques.” In: *IEEE 34th International Conference on Data Engineering Workshops (ICDEW) Proceedings*. IEEE, 2018, pp. 33–38. DOI: 10.1109/ICDEW.2018.00013.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. “How to Play ANY Mental Game.” In: *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing (STOC)*. ACM, 1987, pp. 218–229. DOI: 10.1145/28395.28420.

[God17] Michelle Goddard. “The EU General Data Protection Regulation (GDPR): European Regulation That Has a Global Impact.” In: *International Journal of Market Research* 59.6 (2017), pp. 703–705. DOI: 10.2501/IJMR-2017-050.

[Gre18] Samuel Greengard. “Weighing the Impact of GDPR.” In: *Communications of the ACM* 61.11 (2018), pp. 16–18. DOI: 10.1145/3276744.

[GS62] David Gale and Lloyd S. Shapley. “College Admissions and the Stability of Marriage.” In: *The American Mathematical Monthly* 69.1 (1962), pp. 9–15. DOI: 10.2307/2312726.

[Hav+14] C. Havemann, T. Bahls, M. Bialke, W. Hoffmann, M. Quade, and T. Mauß. *Verfahrensbeschreibung und Datenschutzkonzept des Zentralen Datenmanagements des Deutschen Zentrums für Herz-Kreislauf-Forschung.* 2014. URL: https://dzhk.de/fileadmin/user_upload/Datenschutzkonzept_des_DZHK.pdf (visited on 06/07/2023).

[HC18] David Hand and Peter Christen. "A Note on Using the F-Measure for Evaluating Record Linkage Algorithms." In: *Statistics and Computing* 28.3 (2018), pp. 539–547. DOI: 10.1007/s11222-017-9746-6.

[HCK21] David J. Hand, Peter Christen, and Nishadi Kirielle. "F*: an interpretable transformation of the F-measure." In: *Machine Learning* 110.3 (2021), pp. 451–456. DOI: 10.1007/s10994-021-05964-1.

[HDG20] Katie Harron, James C. Doidge, and Harvey Goldstein. "Assessing data linkage quality in cohort studies." In: *Annals of Human Biology* 47.2 (2020), pp. 218–226. DOI: 10.1080/03014460.2020.1742379.

[HF10] Rob Hall and Stephen E. Fienberg. "Privacy-Preserving Record Linkage." In: *International Conference on Privacy in Statistical Databases (PSD).* Springer, 2010, pp. 269–283. DOI: 10.1007/978-3-642-15838-4_24.

[Hil+20] Kai Hildebrandt, Fabian Panse, Niklas Wilcke, and Norbert Ritter. "Large-Scale Data Pollution with Apache Spark." In: *IEEE Transactions on Big Data* 6.2 (2020), pp. 396–411. DOI: 10.1109/TBDATA.2016.2637378.

[HKN14] Arvid Heise, Gjergji Kasneci, and Felix Naumann. "Estimating the number and sizes of fuzzy-duplicate clusters." In: *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM).* 2014, pp. 959–968. DOI: 10.1145/2661829.2661885.

[HKP12] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques.* 3. Morgan Kaufmann, 2012. DOI: 10.1016/C2009-0-61819-5.

[HM02] David Holmes and M. Catherine McCabe. "Improving Precision and Recall for Soundex Retrieval." In: *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC).* IEEE, 2002, pp. 22–26. DOI: 10.1109/ITCC.2002.1000354.

[HR15] Wilko Henecka and Matthew Roughan. "Privacy-Preserving Fraud Detection Across Multiple Phone Record Databases." In: *IEEE Transactions on Dependable and Secure Computing (TDSC)* 12.6 (2015), pp. 640–651. DOI: 10.1109/TDSC.2014.2382573.

[HS98] Mauricio A. Hernández and Salvatore J. Stolfo. "Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem." In: *Data Mining and Knowledge Discovery* 2.1 (1998), pp. 9–37. DOI: `10.1023/A:1009761603038`.

[HSW07] Thomas N. Herzog, Fritz J. Scheuren, and William E. Winkler. *Data Quality and Record Linkage Techniques*. Springer, 2007. DOI: `10.1007/0-387-69505-2`.

[IM08] Kazuo Iwama and Shuichi Miyazaki. "A Survey of the Stable Marriage Problem and Its Variants." In: *Proceedings of the International Conference on Informatics Education and Research for Knowledge-Circulating Society (ICKS)*. IEEE, 2008, pp. 131–136. DOI: `10.1109/ICKS.2008.7`.

[IM98] Piotr Indyk and Rajeev Motwani. "Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality." In: *Proceedings of the thirtieth annual ACM symposium on Theory of computing (STOC)*. ACM, 1998, pp. 604–613. DOI: `10.1145/276698.276876`.

[Irv94] Robert W. Irving. "Stable Marriage and Indifference." In: *Discrete Applied Mathematics* 48.3 (1994), pp. 261–272. DOI: `10.1016/0166-218X(92)00179-P`.

[Jac12] Paul Jaccard. "The distribution of the flora in the alpine zone." In: *New Phytologist* 11.2 (1912), pp. 37–50. DOI: `10.1111/j.1469-8137.1912.tb05611.x`.

[JH06] Jeff Jonas and Jim Harper. "Effective Counterterrorism and the Limited Role of Predictive Data Mining." In: *Policy Analysis* 584 (2006). URL: `http://www.jstor.org/stable/resrep04886` (visited on 06/27/2023).

[Jia+14] Yu Jiang, Guoliang Li, Jianhua Feng, and Wen-Syan Li. "String Similarity Joins: An Experimental Evaluation." In: *Proceedings of the VLDB Endowment* 7.8 (2014), pp. 625–636. DOI: `10.14778/2732296.2732299`.

[Kar+15] Alexandros Karakasidis et al. "PRIVATEER: A Private Record Linkage Toolkit." In: *CAiSe Forum*. 2015, pp. 197–204. URL: `http://ceur-ws.org/Vol-1367/#paper-26` (visited on 06/27/2023).

[Kar05] Rosemary Karmel. *Data linkage protocols using a statistical linkage key*. Tech. rep. Australian Institute of Health and Welfare, 2005.

[Kas+20] Saffija Kasem-Madani, Timo Malderle, Felix Boes, and Michael Meier. "Privacy-Preserving Warning Management for an Identity Leakage Warning Network." In: *Proceedings of the 2020 European Interdisciplinary Cybersecurity Conference (EICC)*. ACM, 2020, pp. 1–6. DOI: `10.1145/3424954.3424955`.

[KGV16] Dimitrios Karapiperis, Aris Gkoulalas-Divanis, and Vassilios S. Verykios. “LSHDB: A parallel and distributed engine for record linkage and similarity search.” In: *Proceedings of the IEEE 16th International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2016, pp. 1–4. DOI: 10.1109/ICDMW.2016.7867099.

[KGV17] Dimitrios Karapiperis, Aris Gkoulalas-Divanis, and Vassilios S. Verykios. “Distance-Aware Encoding of Numerical Values for Privacy-Preserving Record Linkage.” In: *Proceedings of the 2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. IEEE, 2017, pp. 135–138. DOI: 10.1109/ICDE.2017.58.

[KGV18] Dimitrios Karapiperis, Aris Gkoulalas-Divanis, and Vassilios S. Verykios. “FEDERAL: A Framework for Distance-Aware Privacy-Preserving Record Linkage.” In: *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 30.2 (2018), pp. 292–304. DOI: 10.1109/TKDE.2017.2761759.

[Kho+15] Abel N. Kho, John P. Cashy, Kathryn L. Jackson, Adam R. Pah, Satyender Goel, Jörn Boehnke, John Eric Humphries, Scott Duke Kominers, Bala N Hota, Shannon A Sims, et al. “Design and implementation of a privacy preserving electronic health record linkage tool in Chicago.” In: *Journal of the American Medical Informatics Association (JAMIA)* 22.5 (2015), pp. 1072–1080. DOI: 10.1093/jamia/ocv038.

[KK23] Alexandros Karakasidis and Georgia Koloniari. “More Sparking Soundex-Based Privacy-Preserving Record Linkage.” In: *Algorithmic Aspects of Cloud Computing (ALGOCLOUD 2022), LNCS*. Vol. 13799. Springer, 2023, pp. 73–93. DOI: 10.1007/978-3-031-33437-5_5.

[KL51] S. Kullback and R. A. Leibler. “On Information and Sufficiency.” In: *The Annals of Mathematical Statistics* 22.1 (1951), pp. 79–86. DOI: 10.1214/aoms/1177729694.

[KR10] Hanna Köpcke and Erhard Rahm. “Frameworks for entity matching: A comparison.” In: *Data & Knowledge Engineering* 69.2 (2010), pp. 197–210. DOI: 10.1016/j.datak.2009.10.003.

[Kra91] Mark A. Kramer. “Nonlinear principal component analysis using autoassociative neural networks.” In: *AIChE Journal* 37.2 (1991), pp. 233–243. DOI: 10.1002/aic.690370209.

[Kra92] Mark A. Kramer. “Autoassociative neural networks.” In: *Computers & Chemical Engineering* 16.4 (1992), pp. 313–328. DOI: 10.1016/0098-1354(92)80051-A.

[KS14] Martin Kroll and Simone Steinmetzer. “Automated Cryptanalysis of Bloom Filter Encryptions of Health Records.” In: *Proceedings of the International Conference on Health Informatics (ICHI)* (2014). DOI: 10.5220/0005176000050013.

[KTR12] Lars Kolb, Andreas Thor, and Erhard Rahm. “Dedoop: Efficient Deduplication with Hadoop.” In: *Proceedings of the VLDB Endowment* 5.12 (2012), pp. 1878–1881. DOI: 10.14778/2367502.2367527.

[KTR13] Lars Kolb, Andreas Thor, and Erhard Rahm. “Don’t Match Twice: Redundancy-free Similarity Computation with MapReduce.” In: *Proceedings of the Second Workshop on Data Analytics in the Cloud (DanaC)*. ACM, 2013, pp. 1–5. DOI: 10.1145/2486767.2486768.

[Kue+12] Claudia E. Kuehni, Corina S. Rueegg, Gisela Michel, Cornelia E. Rebholz, Marie-Pierre F. Strippoli, Felix K. Niggli, Matthias Egger, Nicolas X. von der Weid, and Swiss Paediatric Oncology Group (SPOG). “Cohort Profile: The Swiss Childhood Cancer Survivor Study.” In: *International Journal of Epidemiology* 41.6 (2012), pp. 1553–1564. DOI: 10.1093/ije/dyr142.

[Kum+14] Hye-Chung Kum, Ashok Krishnamurthy, Ashwin Machanavajjhala, Michael K Reiter, and Stanley Ahalt. “Privacy preserving interactive record linkage (PPIRL).” In: *Journal of the American Medical Informatics Association (JAMIA)* 21.2 (2014), pp. 212–220. DOI: 10.1136/amiajnl-2013-002165.

[Kum+21] Sumit Kumar Debnath, Pantelimon Stănică, Nibedita Kundu, and Tanmay Choudhury. “Secure and Efficient Multiparty Private Set Intersection Cardinality.” In: *Adv. Math. Commun.* 15.2 (2021), pp. 365–386. DOI: 10.3934/amc.2020071.

[Kuz+11] Mehmet Kuzu, Murat Kantarcioglu, Elizabeth Durham, and Bradley Malin. “A Constraint Satisfaction Cryptanalysis of Bloom Filters in Private Record Linkage.” In: *Privacy Enhancing Technologies (PETS)*. Springer, 2011, pp. 226–245. DOI: 10.1007/978-3-642-22263-4_13.

[KV09] Alexandros Karakasidis and Vassilios S. Verykios. “Privacy Preserving Record Linkage Using Phonetic Codes.” In: *Proceedings of the 2009 Fourth Balkan Conference in Informatics (BCI)*. IEEE, 2009. DOI: 10.1109/BCI.2009.29.

[KV13] Dimitrios Karapiperis and Vassilios S. Verykios. “A Distributed Framework For Scaling Up LSH-Based Computations in Privacy Preserving Record Linkage.” In: *Proceedings of the 6th Balkan Conference in Informatics (BCI)*. ACM, 2013. DOI: 10.1145/2490257.2490258.

[KV14] Dimitrios Karapiperis and Vassilios S. Verykios. “A Distributed Near-Optimal LSH-based Framework for Privacy-Preserving Record Linkage.” In: *Computer Science and Information Systems* 11.2 (2014), pp. 745–763. DOI: 10.2298/CSIS140215040K.

[KV15] Dimitrios Karapiperis and Vassilios S. Verykios. “An LSH-Based Blocking Approach with a Homomorphic Matching Technique for Privacy-Preserving Record Linkage.” In: *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 27.4 (2015), pp. 909–921. DOI: 10.1109/TKDE.2014.2349916.

[KV16] Dimitrios Karapiperis and Vassilios S. Verykios. “A fast and efficient Hamming LSH-based scheme for accurate linkage.” In: *Knowledge and Information Systems (KAIS)* 49.3 (2016), pp. 861–884. DOI: 10.1007/s10115-016-0919-y.

[KVC12] Alexandros Karakasidis, Vassilios S. Verykios, and Peter Christen. “Fake Injection Strategies for Private Phonetic Matching.” In: *Data Privacy Management and Autonomous Spontaneus Security (DPM, SETOP)*. Springer, 2012, pp. 9–24. DOI: 10.1007/978-3-642-28879-1_2.

[Lam93] Diane Lambert. “Measures of disclosure risk and harm.” In: *Journal of Official Statistics* 9 (1993), pp. 313–313.

[LBÜ15] Martin Lablans, Andreas Borg, and Frank Ückert. “A RESTful interface to pseudonymization services in modern web applications.” In: *BMC Medical Informatics and Decision Making* 15.1 (2015), p. 2. DOI: 10.1186/s12911-014-0123-5.

[LBÜ23] Martin Lablans, Andreas Borg, and Frank Ückert. *Mainzelliste (Bitbucket Repository)*. 2023. URL: https://bitbucket.org/medicalinformatics/mainzelliste (visited on 06/28/2023).

[Lee+18] Junghye Lee, Jimeng Sun, Fei Wang, Shuang Wang, Chi-Hyuck Jun, and Xiaoqian Jiang. “Privacy-Preserving Patient Similarity Learning in a Federated Environment: Development and Analysis.” In: *JMIR Medical Informatics* 6.2 (2018), e20. DOI: 10.2196/medinform.7744.

[Len06] Rainer Lenz. “Measuring the Disclosure Protection of Micro Aggregated Business Microdata. An Analysis Taking as An Example the German Structure of Costs Survey.” In: (2006). URL: https://www.statistischebibliothek.de/mir/receive/DEMonografie_mods_00000698 (visited on 06/27/2023).

[Liu+21] Bo Liu, Ming Ding, Sina Shaham, Wenny Rahayu, Farhad Farokhi, and Zihuai Lin. “When Machine Learning Meets Privacy: A Survey and Outlook.” In: *ACM Computing Surveys* 54.2 (2021). DOI: 10.1145/3436755.

[LP09] Yehuda Lindell and Benny Pinkas. “Secure Multiparty Computation for Privacy-Preserving Data Mining.” In: *Journal of Privacy and Confidentiality* 1.1 (2009), pp. 59–98.

[LR96] A. J. Lait and Brian Randell. “An Assessment of Name Matching Algorithms.” In: *Technical Report Series-University of Newcastle Upon Tyne Computing Science* (1996). URL: https://citeseerx.ist.psu.edu/document?doi=f0c1613a85244f6386ed8c6e4eae2880208ed675 (visited on 06/27/2023).

[LRU14] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of Massive Datasets*. Cambridge University Press, 2014. ISBN: 978-1108476348.

[LS11] Bart Lamiroy and Tao Sun. “Computing precision and recall with missing or uncertain ground truth.” In: *International Workshop on Graphics Recognition (GREC)*. Springer, 2011, pp. 149–162. DOI: 10.1007/978-3-642-36824-0_15.

[LSR21] Stefan Lerm, Alieh Saeedi, and Erhard Rahm. “Extended Affinity Propagation Clustering for Multi-source Entity Resolution.” In: *Proceedings Datenbanksysteme für Business, Technologie und Web (BTW)*. Vol. P-311. LNI. Gesellschaft für Informatik, 2021, pp. 217–236. DOI: 10.18420/btw2021-11.

[LSÜ18] Martin Lablans, Esther Erika Schmidt, and Frank Ückert. “An Architecture for Translational Cancer Research As Exemplified by the German Cancer Consortium.” In: *JCO Clinical Cancer Informatics* 2 (2018), pp. 1–8. DOI: 10.1200/CCI.17.00062.

[LT05] Hsiao-Ying Lin and Wen-Guey Tzeng. “An Efficient Solution to the Millionaires’ Problem Based on Homomorphic Encryption.” In: *Applied Cryptography and Network Security*. Springer, 2005, pp. 456–466. DOI: 10.1007/11496137_31.

[Luo+17] Qingwei Luo, Xue Qin Yu, David Paul Smith, David Eamon Goldsbury, Claire Cooke-Yarborough, Manish Indravadan Patel, and Dianne Lesley O’Connell. “Cancer-related hospitalisations and ’unknown’ stage prostate cancer: a population-based record linkage study.” In: *BMJ open* 7.1 (2017). DOI: 10.1136/bmjopen-2016-014259.

[MC16] Leigh Metcalf and William Casey. *Cybersecurity and Applied Mathematics*. Syngress, 2016. DOI: 10.1016/C2015-0-01807-X.

[MG07] Anan Marie and Avigdor Gal. "On the Stable Marriage of Maximum Weight Royal Couples." In: *AAAI Workshop on Information Integration on the Web*. 2007. URL: https://citeseerx.ist.psu.edu/document?doi=70535747fd11d4f6a6cb7feca7fd50f59eeae89c (visited on 06/27/2023).

[MGR02] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. "Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching." In: *Proceedings of the 18th International Conference on Data Engineering (ICDE)*. 2002, pp. 117–128. DOI: 10.1109/ICDE.2002.994702.

[Mit+16] William Mitchell, Rinku Dewri, Ramakrishna Thurimella, and Max Roschke. "A Graph Traversal Attack on Bloom Filter Based Medical Data Aggregation." In: *International Journal of Big Data Intelligence* 4.4 (2016), p. 217. DOI: 10.1504/IJBDI.2017.086956.

[Mit02] M. Mitzenmacher. "Compressed Bloom filters." In: *Proceedings of IEEE/ACM Transactions on Networking (TON* 10.5 (2002), pp. 604–612. DOI: 10.1109/TNET.2002.803864.

[Moo+14] Cecilia L. Moore, Janaki Amin, Heather F. Gidding, and Matthew G. Law. "A New Method for Assessing How Sensitivity and Specificity of Linkage Studies Affects Estimation." In: *PLOS ONE* 9.7 (2014), pp. 1–6. DOI: 10.1371/journal.pone.0103690.

[MR17] Neil G. Marchant and Benjamin I. P. Rubinstein. "In Search of an Entity Resolution OASIS: Optimal Asymptotic Sequential Importance Sampling." In: *Proceedings of the VLDB Endowment*. Vol. 10. 11. 2017, p. 12. DOI: 10.14778/3137628.3137642.

[MS07] Christian Meilicke and Heiner Stuckenschmidt. "Analyzing Mapping Extraction Approaches." In: *Proceedings of the 2nd International Conference on Ontology Matching (OM)*. 2007, pp. 25–36. URL: https://ceur-ws.org/Vol-304/paper3.pdf (visited on 06/27/2023).

[MT79] Robert Morris and Ken Thompson. "Password Security: A Case History." In: *Communications of the ACM (CACM)* 22.11 (1979), pp. 594–597. DOI: 10.1145/359168.359172.

[MU17] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis*. Cambridge University Press, 2017. ISBN: 978-1-107-15488-9.

[Mun57] James Munkres. "Algorithms for the Assignment and Transportation Problems." In: *Journal of the Society for Industrial and Applied Mathematics* 5.1 (1957), pp. 32–38. URL: https://www.jstor.org/stable/2098689 (visited on 06/20/2023).

[Mus+14] Marita Muscholl, Martin Lablans, Thomas OF Wagner, and Frank Ückert. “OSSE – open source registry software solution.” In: *Orphanet Journal of Rare Diseases* 9.1 (2014). DOI: 10.1186/1750-1172-9-S1-O9.

[MW70] David G. McVitie and Leslie B. Wilson. “Stable Marriage Assignment for Unequal Sets.” In: *BIT Numerical Mathematics* 10.3 (1970), pp. 295–309. DOI: 10.1007/BF01934199.

[Nat08] National Institute of Standards and Technology. *The Keyed-Hash Message Authentication Code (HMAC)*. Tech. rep. NIST FIPS 198-1. 2008. DOI: 10.6028/NIST.FIPS.198-1.

[NdM12] Andriy Nikolov, Mathieu d’Aquin, and Enrico Motta. “Unsupervised Learning of Link Discovery Configuration.” In: *The Semantic Web: Research and Applications (ESWC)*. Springer, 2012, pp. 119–133. DOI: 10.1007/978-3-642-30284-8_15.

[New+59] Howard B. Newcombe, James M. Kennedy, S. J. Axford, and Allison P. James. “Automatic Linkage of Vital Records.” In: *Science* 130.3381 (1959), pp. 954–959. DOI: 10.1126/science.130.3381.954.

[New67] Howard B. Newcombe. “Record Linking: The Design of Efficient Systems for Linking Records into Individual and Family Histories.” In: *The American Journal of Human Genetics* 19.3 Pt 1 (1967), p. 335. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1706275/ (visited on 06/28/2023).

[Ngo+13] Axel-Cyrille Ngonga Ngomo, Lars Kolb, Norman Heino, Michael Hartung, Sören Auer, and Erhard Rahm. “When to Reach for the Cloud: Using Parallel Hardware for Link Discovery.” In: *The Semantic Web: Semantics and Big Data (ESWC)*. Springer, 2013, pp. 275–289. DOI: 10.1007/978-3-642-38288-8_19.

[Nie+14] Frank Niedermeyer, Simone Steinmetzer, Martin Kroll, and Rainer Schnell. “Cryptanalysis of Basic Bloom Filters Used for Privacy Preserving Record Linkage.” In: *German Record Linkage Center, Working Paper Series, No. WP-GRLC-2014-04* (2014). DOI: 10.2139/ssrn.3530867.

[NK62] Howard B. Newcombe and James M. Kennedy. “Record Linkage: Making Maximum Use of the Discriminating Power of Identifying Information.” In: *Communications ACM (CACM)* 5.11 (1962), pp. 563–566. DOI: 10.1145/368996.369026.

[NL13] Axel-Cyrille Ngonga Ngomo and Klaus Lyko. “Unsupervised Learning of Link Specifications: Deterministic vs. Non-Deterministic.” In: *Proceedings of the Ontology Matching Workshop*. 2013, p. 12. DOI: 10.5555/2874493.2874496.

[Nób+18] Thiago Pereira da Nóbrega, Carlos Eduardo S. Pires, Tiago Brasileiro Araújo, and Demetrio Gomes Mestre. “Blind attribute pairing for privacy-preserving record linkage.” In: *Proceedings of ACM Symposium on Applied Computing (SAC)*. 2018, pp. 557–564. ISBN: 978-1-4503-5191-1. DOI: 10.1145/3167132.3167193.

[Nor23] North Carolina State Board of Elections (NCSBE). *Voter Registration Data*. 2023. URL: https://www.ncsbe.gov/results-data/voter-registration-data (visited on 06/22/2023).

[NR18] Markus Nentwig and Erhard Rahm. “Incremental clustering on linked data.” In: *IEEE International Conference on Data Mining Workshops (ICDMW)*. 2018. DOI: 10.1109/ICDMW.2018.00084.

[OR18] M. Odell and R. Russell. “The soundex coding system.” In: *US Patents* 1261167 (1918).

[OSR19] Daniel Obraczka, Alieh Saeedi, and Erhard Rahm. “Knowledge Graph Completion with FAMER.” In: *Proceedings of ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD), DI2KG Workshop*. 2019. URL: http://ceur-ws.org/Vol-2512/paper1.pdf (visited on 05/05/2023).

[Pag+99] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. “The PageRank Citation Ranking: Bringing Order to the Web.” In: *The Web Conference*. 1999.

[Pan+21] Fabian Panse, André Düjon, Wolfram Wingerath, and Benjamin Wollmer. “Generating Realistic Test Datasets for Duplicate Detection at Scale Using Historical Voter Data.” In: *Proceedings of the 24th International Conference on Extending Database Technology (EDBT)*. 2021, pp. 570–581. DOI: 10.5441/002/edbt.2021.67.

[Pap+17] George Papadakis, Konstantina Bereta, Themis Palpanas, and Manolis Koubarakis. “Multi-core Meta-blocking for Big Linked Data.” In: ACM, 2017, pp. 33–40. DOI: 10.1145/3132218.3132230.

[Pap+18] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. “Scalable Private Learning with PATE.” In: *Sixth International Conference on Learning Representations (ICLR)*. 2018. URL: https://openreview.net/forum?id=rkZB1XbRZ (visited on 07/26/2023).

[Pap+21] George Papadakis, Ekaterini Ioannou, Emanouil Thanos, and Themis Palpanas. *The Four Generations of Entity Resolution*. Vol. 16. Synthesis Lectures on Data Management 2. Springer, 2021, pp. 1–170. DOI: `10.1007/978-3-031-01878-7`.

[Pap+22] George Papadakis, Vasilis Efthymiou, Emmanouil Thanos, and Oktie Hassanzadeh. "Bipartite Graph Matching Algorithms for Clean-Clean Entity Resolution: An Empirical Evaluation." In: *Proceedings of the 25th International Conference on Extending Database Technology (EDBT)*. 2022. DOI: `10.48786/edbt.2022.41`.

[PBM14] Emmanouil Antonios Platanios, Avrim Blum, and Tom Mitchell. "Estimating accuracy from unlabeled data." In: *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence (UAI)*. 2014. DOI: `10.1184/R1/6605273.v1`.

[Phi00] Lawrence Philips. "The double metaphone search algorithm." In: *C/C++ users journal* 18.6 (2000), pp. 38–43.

[Phu+12] Clifton Phua, Kate Smith-Miles, Vincent Lee, and Ross Gayler. "Resilient Identity Crime Detection." In: *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 24.3 (2012), pp. 533–546. DOI: `10.1109/TKDE.2010.262`.

[Pin+19] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. "SpOT-Light: Lightweight Private Set Intersection from Sparse OT Extension." In: *Advances in Cryptology*. Vol. 11694. Springer, 2019, pp. 401–431. DOI: `10.1007/978-3-030-26954-8_13`.

[Pit+18] Robespierre Pita, Clicia Pinto, Samila Sena, Rosemeire Fiaccone, Leila Amorim, Sandra Reis, Mauricio L. Barreto, Spiros Denaxas, and Marcos Ennes Barreto. "On the Accuracy and Scalability of Probabilistic Data Linkage Over the Brazilian 114 Million Cohort." In: *IEEE Journal of Biomedical and Health Informatics (JBHI)* 22.2 (2018), pp. 346–353. DOI: `10.1109/JBHI.2018.2796941`.

[Pos69] Hans Joachim Postel. "Die Kölner Phonetik. Ein Verfahren zur Identifizierung von Personennamen auf der Grundlage der Gestaltanalyse." In: *IBM-Nachrichten* 19 (1969), pp. 925–931.

[Pow+17] Conrad Pow, Karey Iron, James Boyd, Adrian Brown, Simon Thompson, Nelson Chong, and Charlotte Ma. "Privacy-Preserving Record Linkage: An International Collaboration between Canada, Australia and Wales." In: *International Journal of Population Data Science* 1.1 (2017). DOI: `10.23889/ijpds.v1i1.101`.

[Qua+98] Catherine Quantin, H. Bouzelat, F. A. A Allaert, A. M. Benhamiche, J. Faivre, and L. Dusserre. "How to ensure data security of an epidemiological follow-up:quality assessment of an anonymous record linkage procedure." In: *International Journal of Medical Informatics* 49.1 (1998), pp. 117–122. DOI: `https://doi.org/10.1016/S1386-5056(98)00019-7`.

[Rag+18] Eric D. Ragan, Hye-Chung Kum, Gurudev Ilangovan, and Han Wang. "Balancing Privacy and Information Disclosure in Interactive Record Linkage with Visual Masking." In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 2018. DOI: `10.1145/3173574.3173900`.

[Ran+14] Sean M. Randall, Anna M. Ferrante, James H. Boyd, Jacqueline K. Bauer, and James B. Semmens. "Privacy-preserving record linkage on large real world datasets." In: *Journal of Biomedical Informatics* 50 (2014), pp. 205–212. DOI: `10.1016/j.jbi.2013.12.003`.

[Ran+16] Sean M. Randall, Anna M. Ferrante, James 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?" In: *Health Information Management Journal* 45.2 (2016), pp. 71–79. DOI: `10.1177/1833358316647587`.

[Ran+19] Sean M. Randall, Adrian P. Brown, Anna M. Ferrante, and James H. Boyd. "Privacy Preserving Linkage Using Multiple Dynamic Match Keys." In: *International Journal of Population Data Science* 4.1 (2019). DOI: `10.23889/ijpds.v4i1.1094`.

[RCS20] Thilina Ranbaduge, Peter Christen, and Rainer Schnell. "Secure and Accurate Two-Step Hash Encoding for Privacy-Preserving Record Linkage." In: *Advances in Knowledge Discovery and Data Mining*. Vol. 12085. Springer, 2020, pp. 139–151. DOI: `10.1007/978-3-030-47436-2_11`.

[RCS21] Thilina Ranbaduge, Peter Christen, and Rainer Schnell. "Large Scale Record Linkage in the Presence of Missing Data." In: *arXiv preprint* (2021). DOI: `10.48550/arXiv.2104.09677`.

[RD00] Erhard Rahm and Hong Hai Do. "Data Cleaning: Problems and Current Approaches." In: *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering* 23.4 (2000), pp. 3–13.

[Roh+21] Florens Rohde, Martin Franke, Ziad Sehili, Martin Lablans, and Erhard Rahm. "Optimization of the Mainzelliste software for fast privacy-preserving record linkage." In: *Journal of Translational Medicine* 19.33 (2021). DOI: `10.1186/s12967-020-02678-1`.

[Roh+23] Florens Rohde, Martin Franke, Victor Christen, and Erhard Rahm. “Value-specific Weighting for Record-level Encodings in Privacy-Preserving Record Linkage.” In: *Proceedings Datenbanksysteme für Business, Technologie und Web (BTW)*. Gesellschaft für Informatik, 2023. DOI: 10.18420/BTW2023-21.

[Ros10] Sheldon M. Ross. *Introduction to Probability Models*. 10th. Elsevier, 2010. ISBN: 978-0-12-375686-2.

[Rou09] Vassil Roussev. “Hashing and Data Fingerprinting in Digital Forensics.” In: *IEEE Security & Privacy* 7.2 (2009), pp. 49–55. DOI: 10.1109/MSP.2009.40.

[Rou10] V. Roussev. “Data fingerprinting with similarity digests.” In: *Advances in Digital Forensics*. Vol. 337. Springer, 2010, pp. 207–226. DOI: 10.1007/978-3-642-15506-2_15.

[RS20] Thilina Ranbaduge and Rainer Schnell. “Securing Bloom Filters for Privacy-preserving Record Linkage.” In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management (CIKM)*. ACM, 2020, pp. 2185–2188. DOI: 10.1145/3340531.3412105.

[San+07] Walter Santos, Thiago Teixeira, Carla Machado, Wagner Meira Jr., Renato Ferreira, Dorgival Guedes, and Altigran S. Da Silva. “A Scalable Parallel Deduplication Algorithm.” In: *Computer Architecture and High Performance Computing*. IEEE, 2007, pp. 79–86. DOI: 10.1109/SBAC-PAD.2007.32.

[SB16a] Rainer Schnell and Christian Borgs. “Randomized Response and Balanced Bloom Filters for Privacy Preserving Record Linkage.” In: *International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2016, pp. 218–224. DOI: 10.1109/ICDMW.2016.29.

[SB16b] Rainer Schnell and Christian Borgs. “XOR-Folding for Hardening Bloom Filter-Based Encryptions for Privacy-Preserving Record Linkage.” In: *German Record Linkage Center, Working Paper Series, NO. WP-GRLC-2016-03* (2016). DOI: 10.2139/ssrn.3527984.

[SB18] Rainer Schnell and Christian Borgs. “Hardening Encrypted Patient Names Against Cryptographic Attacks Using Cellular Automata.” In: *Proceedings of the 2018 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2018, pp. 518–522. DOI: 10.1109/ICDMW.2018.00082.

[SBB04] Rainer Schnell, Tobias Bachteler, and Stefan Bender. "A Toolbox for record linkage." In: *Austrian Journal of Statistics* 33.1-2 (2004), pp. 125–133. URL: https://nbn-resolving.org/urn:nbn:de:0168-ssoar-131815 (visited on 06/27/2023).

[SBR09] Rainer Schnell, Tobias Bachteler, and Jörg Reiher. "Privacy-preserving record linkage using Bloom filters." In: *BMC Medical Informatics and Decision Making* 9.1 (2009), p. 41. DOI: 10.1186/1472-6947-9-41.

[SBR11] Rainer Schnell, Tobias Bachteler, and Jörg Reiher. "A Novel Error-Tolerant Anonymous Linking Code." In: *German Record Linkage Center, No. WP-GRLC-2011-02* (2011). DOI: 10.2139/ssrn.3549247.

[Sca+07] Monica Scannapieco, Ilya Figotin, Elisa Bertino, and Ahmed K. Elmagarmid. "Privacy Preserving Schema and Data Matching." In: *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*. ACM, 2007, pp. 653–664. DOI: 10.1145/1247480.1247553.

[SCF10] Araken M. Santos, Anne M. P. Canuto, and Antonino Feitosa Neto. "Evaluating classification methods applied to multi-label tasks in different domains." In: *Proceedings of the 10th International Conference on Hybrid Intelligent Systems (HIS)*. 2010, pp. 61–66. DOI: 10.1109/HIS.2010.5600014.

[Sch15] Rainer Schnell. "Privacy-preserving record linkage." In: *Methodological Developments in Data Linkage*. John Wiley & Sons, 2015, pp. 201–225. URL: https://openaccess.city.ac.uk/id/eprint/14393/ (visited on 05/24/2023).

[Sch21] Max Schrodt. "Konzeption und Implementierung eines Tools zur visuellen Maskierung beim interaktiven Record Linkage." Bachelor's Thesis. Universität Leipzig, 2021.

[SCS15] Kurt Schmidlin, Kerri M. Clough-Gorr, and Adrian Spoerr. "Privacy Preserving Probabilistic Record Linkage (P3RL): a novel method for linking existing health-related data and maintaining participant confidentiality." In: *BMC Medical Research Methodology* 15.1 (2015). DOI: 10.1186/s12874-015-0038-6.

[SDR21] Alieh Saeedi, Lucie David, and Erhard Rahm. "Matching Entities from Multiple Sources with Hierarchical Agglomerative Clustering." In: *Proceedings of the 13th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management IC3K*. SCITEPRESS, 2021, pp. 40–50. DOI: 10.5220/0010649600003064.

[Seh+15] Ziad Sehili, Lars Kolb, Christian Borgs, Rainer Schnell, and Erhard Rahm. "Privacy Preserving Record Linkage with PPJoin." In: *Proceedings Datenbanksysteme für Business, Technologie und Web (BTW)*. Vol. P-241. LNI. Gesellschaft für Informatik, 2015, pp. 85–104. DOI: 20.500.12116/2453.

[Seh+21] Ziad Sehili, Florens Rohde, Martin Franke, and Erhard Rahm. "Multi-Party Privacy Preserving Record Linkage in Dynamic Metric Space." In: *Proceedings Datenbanksysteme für Business, Technologie und Web (BTW)*. Vol. P-311. LNI. Gesellschaft für Informatik, 2021, pp. 257–278. DOI: 10.18420/btw2021-13.

[Skr+16] T. Skripcak, U. Just, M. Simon, D. Büttner, A. Lühr, M. Baumann, and M. Krause. "Toward Distributed Conduction of Large-Scale Studies in Radiation Therapy and Oncology: Open-Source System Integration Approach." In: *IEEE Journal of Biomedical and Health Informatics* 20.5 (2016), pp. 1397–1403. DOI: 10.1109/JBHI.2015.2450833.

[Smi17] Duncan Smith. "Secure Pseudonymisation for Privacy-Preserving Probabilistic Record Linkage." In: *Journal of Information Security and Applications* 34 (2017), pp. 271–279. DOI: 10.1016/j.jisa.2017.01.002.

[SMR15] S. Joshua Swamidass, Matthew Matlock, and Leon Rozenblit. "Securely Measuring the Overlap between Private Datasets with Cryptosets." In: *PLOS ONE* 10.2 (2015). DOI: 10.1371/journal.pone.0117898.

[Sna07] Chakkrit Snae. "A Comparison and Analysis of Name Matching Algorithms." In: *International Journal of Computer and Information Engineering* 1.1 (2007), pp. 107–112.

[SPR18] Alieh Saeedi, Eric Peukert, and Erhard Rahm. "Using Link Features for Entity Clustering in Knowledge Graphs." In: *Proceedings of the European Semantic Web Conference (ESWC) 2018*. Vol. 10843. Lecture Notes in Computer Science. Springer, 2018, pp. 576–592. DOI: 10.1007/978-3-319-93417-4_37.

[SPR20] Alieh Saeedi, Eric Peukert, and Erhard Rahm. "Incremental Multi-source Entity Resolution for Knowledge Graph Completion." In: *Proceedings of the European Semantic Web Conference (ESWC) 2012*. Vol. 12123. Lecture Notes in Computer Science. Springer, 2020, pp. 393–408. DOI: 10.1007/978-3-030-49461-2_23.

[SR16] Ziad Sehili and Erhard Rahm. "Speeding up Privacy Preserving Record Linkage for Metric Space Similarity Measures." In: *Datenbank-Spektrum* 16.3 (2016), pp. 227–236. DOI: 10.1007/s13222-016-0222-9.

[SR22] Rainer Schnell and Dorothea Rukasz. *Package 'PPRL'*. 2022. URL: https://cran.r-project.org/package=PPRL (visited on 06/22/2023).

[SRB17] Rainer Schnell, Anke Richter, and Christian Borgs. "A Comparison of Statistical Linkage Keys with Bloom Filter-based Encryptions for Privacy-preserving Record Linkage Using Real-world Mammography Data:" in: *Proceedings of the 10th International Joint Conference on Biomedical Engineering Systems and Technologies*. SCITEPRESS - Science and Technology Publications, 2017, pp. 276–283. DOI: 10.5220/0006140302760283.

[Sta11] William Stallings. *Cryptography and Network Security: Principles and Practice*. 5. Prentice Hall, 2011. ISBN: 978-0-13-609704-4.

[Sta23a] Statistisches Bundesamt (Destatis). *Genesis-Online: Die Datenbank des Statistischen Bundesamtes*. 2023. URL: https://www-genesis.destatis.de/genesis/online (visited on 06/28/2023).

[Sta23b] Statistisches Bundesamt (Destatis). *Krankenhäuser in Deutschland*. 2023. URL: https://www.destatis.de/DE/Themen/Gesellschaft-Umwelt/Gesundheit/Krankenhaeuser/_inhalt.html (visited on 06/28/2023).

[Sto+17] Holger Storf, Jannik Schaaf, Dennis Kadioglu, Jens Göbel, Thomas O. F. Wagner, and Frank Ückert. "Register für seltene Erkrankungen." In: *Bundesgesundheitsblatt* 60.5 (2017), pp. 523–531. DOI: 10.1007/s00103-017-2536-7. URL: https://doi.org/10.1007/s00103-017-2536-7.

[The23a] The Apache Software Foundation. *Apache Flink - Stateful Computations over Data Streams*. 2023. URL: https://flink.apache.org/ (visited on 06/28/2023).

[The23b] The Apache Software Foundation. *Apache Spark - Unified engine for large-scale data analytics*. 2023. URL: https://spark.apache.org/ (visited on 06/28/2023).

[TMF21] TMF - Technologie- und Methodenplattform für die vernetzte medizinische Forschung e.V. *Mainzelliste | Toolpool Gesundheitsforschung*. 2021. URL: https://www.toolpool-gesundheitsforschung.de/produkte/mainzelliste (visited on 06/28/2023).

[Tot+14] Csaba Toth et al. "SOEMPI: A Secure Open Enterprise Master Patient Index Software Toolkit for Private Record Linkage." In: *AMIA Annual Symposium Proceedings* (2014), pp. 1105–1114. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4419976/ (visited on 06/27/2023).

[TVC13] Khoi-Nguyen Tran, Dinusha Vatsalan, and Peter Christen. “GeCo – An online personal data Generator and Corruptor.” In: *Proceedings of the 22nd ACM international conference on Information & Knowledge Management (CIKM)*. ACM, 2013, pp. 2473–2476. DOI: `10.1145/2505515.2508207`.

[Uni48] United Nations General Assembly. *Universal Declaration of Human Rights (UDHR)*. 1948.

[Vat+14] Dinusha Vatsalan, Peter Christen, Christine M. O’Keefe, and Vassilios S. Verykios. “An Evaluation Framework for Privacy-Preserving Record Linkage.” In: *Journal of Privacy and Confidentiality (JPC)* 6.1 (2014), p. 3. DOI: `10.29012/jpc.v6i1.636`.

[Vat+17] Dinusha Vatsalan, Ziad Sehili, Peter Christen, and Erhard Rahm. “Privacy-Preserving Record Linkage for Big Data: Current Approaches and Research Challenges.” In: *Handbook of Big Data Technologies*. Springer, 2017, pp. 851–895. DOI: `10.1007/978-3-319-49340-4_25`.

[VC14] Dinusha Vatsalan and Peter Christen. “Scalable Privacy-Preserving Record Linkage for Multiple Databases.” In: *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM)*. ACM, 2014, pp. 1795–1798. DOI: `10.1145/2661829.2661875`.

[VC16] Dinusha Vatsalan and Peter Christen. “Privacy-preserving matching of similar patients.” In: *Journal of Biomedical Informatics* 59 (2016), pp. 285–298. DOI: `10.1016/j.jbi.2015.12.004`.

[VCV13] Dinusha Vatsalan, Peter Christen, and Vassilios S. Verykios. “A taxonomy of privacy-preserving record linkage techniques.” In: *Information Systems* 38.6 (2013), pp. 946–969. DOI: `10.1016/j.is.2012.11.005`.

[Vid+20a] Anushka Vidanage, Peter Christen, Thilina Ranbaduge, and Rainer Schnell. “A Graph Matching Attack on Privacy-Preserving Record Linkage.” In: *International Conference on Information & Knowledge Management (CIKM)*. ACM, 2020, pp. 1485–1494. DOI: `10.1145/3340531.3411931`.

[Vid+20b] Anushka Vidanage, Thilina Ranbaduge, Peter Christen, and Sean Randall. “Privacy Attack on Multiple Dynamic Match-key Based Privacy-Preserving Record Linkage.” In: *International Journal of Population Data Science* 5.1 (2020). DOI: `10.23889/ijpds.v5i1.1345`.

[Vid+22] Anushka Vidanage, Thilina Ranbaduge, Peter Christen, and Rainer Schnell. “A Taxonomy of Attacks on Privacy-Preserving Record Linkage.” In: *Journal of Privacy and Confidentiality (JPC)* 12.1 (2022). DOI: `https://doi.org/10.29012/jpc.764`.

[Vid+23] Anushka Vidanage, Peter Christen, Thilina Ranbaduge, and Rainer Schnell. “A Vulnerability Assessment Framework for Privacy-Preserving Record Linkage.” In: *ACM Transactions on Privacy and Security* (2023). DOI: 10.1145/3589641.

[VRC19] Sirintra Vaiwsri, Thilina Ranbaduge, and Peter Christen. “Reference Values Based Hardening for Bloom Filters Based Privacy-Preserving Record Linkage.” In: *Proceedings of the 2018 Australasian Conference on Data Mining*. Vol. 996. Springer, 2019, pp. 189–202. DOI: 10.1007/978-981-13-6661-1_15.

[Wan+10] Chaokun Wang, Jianmin Wang, Xuemin Lin, Wei Wang, Haixun Wang, Hongsong Li, Wanpeng Tian, Jun Xu, and Rui Li. “MapDupReducer: Detecting Near Duplicates over Massiv Datasets.” In: *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. ACM, 2010. DOI: 10.1145/1807167.1807296.

[WCA04] Gang Wang, Hsinchun Chen, and Homa Atabakhsh. “Automatically Detecting Deceptive Criminal Identities.” In: *Communications of the ACM (CACM)* 47.3 (2004), pp. 70–76. DOI: 10.1145/971617.971618.

[WE18] Isabel Wagner and David Eckhoff. “Technical Privacy Metrics: A Systematic Survey.” In: *ACM Computing Surveys* 51.3 (2018), pp. 1–38. DOI: 10.1145/3168389.

[Web+12] S. C. Weber, H. Lowe, A. Das, and T. Ferris. “A simple heuristic for blindfolded record linkage.” In: *Journal of the American Medical Informatics Association (JAMIA)* 19 (2012), e157–e161. DOI: 10.1136/amiajnl-2011-000329.

[Wes01] Douglas Brent West. *Introduction to Graph Theory*. Vol. 2. Prentice Hall, 2001. ISBN: 978-0130144003.

[Xu+20] Jie Xu, Zhenxing Xu, Peter Walker, and Fei Wang. “Federated Patient Hashing.” In: *AAAI Conference on Artificial Intelligence*. Vol. 34. 04. 2020, pp. 6486–6493. DOI: 10.1609/aaai.v34i04.6121.

[XW05] Rui Xu and Donald C. Wunsch. “Survey of Clustering Algorithms.” In: *IEEE Transactions on Neural Networks* 16.3 (2005), pp. 645–678. DOI: 10.1109/TNN.2005.845141.

[Yao82] Andrew C. Yao. “Protocols for secure computations.” In: *23rd Annual Symposium on Foundations of Computer Science (SFCS) Proceedings*. 1982, pp. 160–164. DOI: 10.1109/SFCS.1982.38.

[YW20] Yun William Yu and Griffin M Weber. "Balancing Accuracy and Privacy in Federated Queries of Clinical Data Repositories: Algorithm Development and Validation." In: *Journal of Medical Internet Research* (2020). DOI: 10.2196/18735.

Selbständigkeitserklärung

Hiermit erkläre ich, die vorliegende Dissertation selbständig und ohne unzulässige fremde Hilfe angefertigt zu haben. Ich habe keine anderen als die angeführten Quellen und Hilfsmittel benutzt und sämtliche Textstellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen wurden, und alle Angaben, die auf mündlichen Auskünften beruhen, als solche kenntlich gemacht. Ebenfalls sind alle von anderen Personen bereitgestellten Materialien oder erbrachten Dienstleistungen als solche gekennzeichnet.

Leipzig, 25.10.2023 Martin Franke