



Chi, Y., Hong, J., Jurek, A., Liu, W., & O'Reilly, D. (2017). Privacy preserving record linkage in the presence of missing values. *Information Systems*, 71, 199-210.
<https://doi.org/10.1016/j.is.2017.07.001>

Peer reviewed version

License (if available):
Unspecified

Link to published version (if available):
[10.1016/j.is.2017.07.001](https://doi.org/10.1016/j.is.2017.07.001)

[Link to publication record on the Bristol Research Portal](#)
PDF-document

This is the author accepted manuscript (AAM). The final published version (version of record) is available online via Elsevier at <http://www.sciencedirect.com/science/article/pii/S030643791630504X>. Please refer to any applicable terms of use of the publisher.

University of Bristol – Bristol Research Portal

General rights

This document is made available in accordance with publisher policies. Please cite only the published version using the reference above. Full terms of use are available:
<http://www.bristol.ac.uk/red/research-policy/pure/user-guides/brp-terms/>

Accepted Manuscript

Privacy Preserving Record Linkage in the Presence of Missing Values

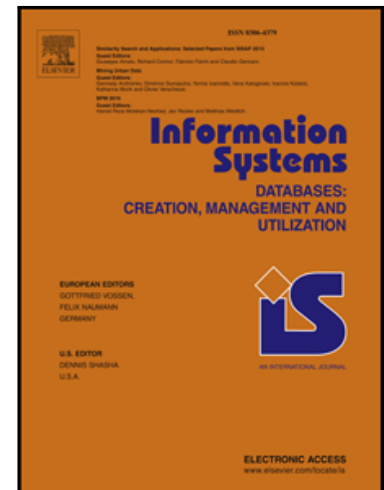
Yuan Chi, Jun Hong, Anna Jurek, Weiru Liu, Dermot O'Reilly

PII: S0306-4379(16)30504-X
DOI: [10.1016/j.is.2017.07.001](https://doi.org/10.1016/j.is.2017.07.001)
Reference: IS 1230

To appear in: *Information Systems*

Received date: 24 October 2016
Revised date: 19 June 2017
Accepted date: 5 July 2017

Please cite this article as: Yuan Chi, Jun Hong, Anna Jurek, Weiru Liu, Dermot O'Reilly, Privacy Preserving Record Linkage in the Presence of Missing Values, *Information Systems* (2017), doi: [10.1016/j.is.2017.07.001](https://doi.org/10.1016/j.is.2017.07.001)



This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

Highlights

- It is proposed that the missing value in a record is handled by utilising the values of the corresponding fields in the k-NNs of this record.
- The proposed method for dealing with missing values allows the use of the traditional blocking techniques to handle the scalability issue.
- The existing Bloom filter protocol has been adapted to address both issues of missing values and privacy preservation

ACCEPTED MANUSCRIPT

Privacy Preserving Record Linkage in the Presence of Missing Values

Yuan Chi^{a,*}, Jun Hong^b, Anna Jurek^a, Weiru Liu^c, Dermot O'Reilly^d

^a*School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, 18 Malone Road, BT9 6RT Belfast, United Kingdom*

^b*Department of Computer Science and Creative Technologies, University of the West of England, Coldharbour Lane, BS16 1QY Bristol, United Kingdom*

^c*Merchant Venturers School of Engineering, University of Bristol, 75 Woodland Road, BS8 1UB Bristol, United Kingdom*

^d*Centre for Public Health, Queen's University Belfast, Institute of Clinical Sciences, Block B, BT12 6BA Belfast, United Kingdom*

Abstract

The problem of record linkage is to identify records from two datasets, which refer to the same entities (e.g. patients). A particular issue of record linkage is the presence of missing values in records, which has not been fully addressed. Another issue is how privacy and confidentiality can be preserved in the process of record linkage. In this paper, we propose an approach for privacy preserving record linkage in the presence of missing values. For any missing value in a record, our approach imputes the similarity measure between the missing value and the value of the corresponding field in any of the possible matching records from another dataset. We use the k -NNs (k Nearest Neighbours in the same dataset) of the record with the missing value and their distances to the record for similarity imputation. For privacy preservation, our approach uses the Bloom filter protocol in the settings of both standard privacy preserving record linkage without missing values and privacy preserving record linkage with missing values. We have conducted an experimental evaluation using three pairs of synthetic datasets with different rates of missing values. Our experimental results

[☆]Fully documented templates are available in the elsarticle package on CTAN.

^{*}Corresponding author

Email addresses: y.chi@qub.ac.uk (Yuan Chi), jun.hong@uwe.ac.uk (Jun Hong), a.jurek@qub.ac.uk (Anna Jurek), weiru.liu@bristol.ac.uk (Weiru Liu), d.oreilly@qub.ac.uk (Dermot O'Reilly)

show the effectiveness and efficiency of our proposed approach.

Keywords: Record linkage, Probabilistic record linkage, Privacy preserving record linkage, Missing values, k -nearest neighbours, Data encryption

1. Introduction

Record linkage, also known as data matching, duplicate detection, or entity resolution, refers to the process of identifying and aggregating records from one or more datasets, which represent the same real-world entities [1, 2]. For example, Table 1 shows two datasets, R and S , which contain the personal information of individuals. To link the two datasets, the desirable output are the record pairs that refer to the same individuals, i.e. $(R2, S1)$.

Record linkage is often needed in tasks such as creating a linked dataset for further analysis [3]. As long as the data for the same entity is spread across more than one dataset, record linkage would be needed for the analysis of such data. For example, in medical and social sciences research, the disjunctive or additional data about the same individual must be obtained by combining two or more different datasets through record linkage, such that complete family trees over a period of time can be created [4].

Unfortunately, records to be linked across different datasets often lack unique identifiers for performing such an identifying and aggregating process [1]. To overcome this problem, many techniques have been developed for record linkage over the past decade [5] in various applications. For example, in the areas of national censuses and health, most of the linkage systems use the probabilistic record linkage technique [6, 7].

Most of the current techniques for record linkage are based on comparing the values of several partially unique fields in a pair of records, which are generally available (e.g. name, date of birth, and address), or even a combination of them to identify and link records about the same individual [5]. However, in various domains, missing values may be present in records due to a variety of reasons. For example, in the case of medical databases, patients may not wish

Table 1: Two example datasets for record linkage.

| <i>Dataset</i> | <i>#</i> | <i>Surname</i> | <i>GivenName</i> | <i>Gender</i> | <i>Street</i> | <i>City</i> | <i>Postcode</i> | <i>Telephone</i> |
|----------------|----------|----------------|------------------|---------------|-------------------------|---------------|-----------------|------------------|
| <i>R</i> | 1 | Green | Gabrielle | F | 435 s. la cienega blvd. | Los Angeles | 92603 | 426 341 2521 |
| | 2 | Lorigman | Vinn | | 8358 sunset blvd. west | Hollywood | 92613 | 421 908 1495 |
| | 3 | | Kte | F | 95 ave. a at 6th st. | | 9261e | 427 724 5566 |
| | 4 | Heron | Keely | | 3434 peachtree rd. | San Francisco | 92692 | |
| | 5 | | Zhane | F | | San Francisco | 92602 | 427 939 4314 |
| | 6 | Hron | David | M | 3434 peachtree rd. ne | | 9260q | |
| | 7 | Mulin | Jhane | F | | San Francisco | 92605 | |
| <i>S</i> | 1 | Lonkman | Finn | | 8358 sunset blvd. | W. Hollywood | 92612 | 421-908-1549 |
| | 2 | Henningj | Kade | F | 95 ave. a | New York City | 92614 | 427-729-5665 |
| | 3 | Dixon | Cherea | M | 99 e. 52nd st. | | 9260t | |
| | 4 | Heron | Shane | F | 3434 peachtree rd. | San Francisco | 92602 | 427-939-4314 |

to provide all the required information or clinical workflows cannot ensure that data collection and/or documentation are accurate and complete [8]. It was reported that for the electronic medical records of HIV patients, the median
30 missing data rate was about 10.9% [9]. For the problem of missing values in traditional databases for statistical analysis (e.g. statistical classification) [10], a variety of methods have been developed. In general, some of these approaches ignore either the missing values themselves or the records with missing values altogether while the other approaches impute the missing values instead.

35 Another major challenge in record linkage is how to protect the privacy and confidentiality of sensitive information (e.g. names and addresses of people), when datasets are linked between organisations [2]. In certain applications (e.g. linking large datasets about people), while such personal identifying fields are commonly used in the linkage process, they must be kept private and confidential
40 [11]. The problem of finding records that represent the same individual in separate datasets without revealing the identity of the individual is called privacy-preserving record linkage (PPRL). Although various techniques for PPRL have been developed for linking datasets between organisations, there is currently no work addressing the problem of missing values in record linkage while at the
45 same time addressing the issue of privacy and confidentiality.

In this paper, we propose an approach to record linkage in the presence of missing values, while simultaneously addressing the issue of privacy and confidentiality. For the issue of missing values, our approach imputes the similarity measure between the missing value and the value of the corresponding field in
50 any of the possible matching records in another dataset, using a collection of values for the corresponding field in its k -NNs (Nearest Neighbours) in the same dataset. Our observation is that it is very likely that the record with a missing value and its k -NNs (i.e. very similar records) have similar values for the corresponding field. This observation generally holds in large datasets (e.g. census
55 data). For example, there is a high probability that the people who live in the same address share the same last name or telephone number [12].

The k -NNs of a record with a missing value are selected based on the sim-

ilarity measures between these k -NNs and the record with the missing value. Each of such similarity measures is calculated between the values for each of the corresponding fields in a pair of records, which does not have any missing value. For the values of the corresponding field(s) in the k -NNs, an associated weight vector is constructed, with each weight in the vector representing the distance (i.e. similarity measure) between the respective k -NN and the record with the missing value. These weights will reflect the different levels of contributions that the different values of the corresponding field in the k -NNs make to the imputation of the similarity measure on the corresponding field between the record with the missing value and any of the possible matching records in another dataset.

The reason that a missing value is dealt with in this way, rather than simply taking the value in the 1-NN record as the imputed missing value or deciding on a value by majority voting out of the values in the k -NNs, is to avoid any situation in which the 1-NN record holds a totally different value from the actual missing value, or these records have several slightly different values for the field (e.g. due to typographical errors). For example, as shown in Table 1, for record $R5$ with a missing value of *Surname*, the first of its k -NNs is $R7$, which, however, holds a wrong value of *Surname* for $R5$, whilst the second and third of its k -NNs and (i.e. $R4$ and $R6$ respectively) hold the correct value of *Surname* for $R5$.

Our approach uses both the k -NNs of the record with a missing value and the corresponding weight vector of the k -NNs to impute the similarity measure between the missing value and the value of the corresponding field in any of the possible matching records in another dataset. As a result, as long as the majority of the k -NNs of the record with the missing value hold the same value as or similar values (e.g. due to a typographical error) to the actual missing value of the corresponding field, the imputed similarity measure would be reasonable. Since our proposed approach imputes the similarity measure between the values of the same field in two records, traditional blocking techniques [13, 14], which primarily rely on similarity measures on some of the fields in different records,

can be simply applied to address the scalability issue. For privacy preservation,
90 our approach to dealing with missing values enables us to adapt the Bloom
filter protocol [15, 16] in our approach to privacy preserving record linkage in
the presence of missing values.

This paper makes three novel contributions: First, the missing values in
a record are handled by utilising the values of the corresponding fields in the
95 k -NNs of the record. Second, our proposed technique for dealing with missing
values allows use of the traditional blocking techniques for dealing with the
scalability issue. Finally, we have adapted the Bloom filter protocol in our
approach to address both issues of missing values and privacy preservation.
The remainder of the paper is organised as follows: Section 2 describes related
100 work. Section 3 presents the formulation of the research problem we solve in
this paper and briefly introduces the fundamentals of the techniques proposed
for the solutions to the problem. Section 4 describes our proposed approach
to privacy preserving record linkage in the presence of missing values. Section
5 presents our experimental evaluation of the proposed approach. Section 6
105 concludes the paper.

2. Related Work

Record linkage typically uses a set of non-unique identifying fields [17]. Cur-
rent approaches to record linkage can be divided into three categories. The
first category is called deterministic record linkage, in which whether a pair of
110 records match is determined by the exact agreement or disagreement between
the corresponding values of each of the identifying fields [18]. The methods
of this category have the advantages of being simple, transparent, and easy to
accept [18]. However, their common drawback is that they do not tolerate the
presence of any errors (e.g. typographical or phonetic errors) in records [8].
115 The second category are probabilistic record linkage methods, which estimate
the likelihood that two records match [8]. They allow the presence of some er-
rors in records by considering the similarity measure between the values of the

corresponding field in the records. The last category of methods are machine learning based, where a variety of machine learning techniques are applied to train a classifier to decide on whether a pair of records matches. Such machine learning approaches can often achieve higher accuracy rates [19].

In some specific domains (e.g. patient records), no private or confidential information can be revealed. As a result, various PPRL techniques have been developed [2]. Although they operate differently, they share the same principle: the records in the datasets to be linked are encoded at the sources while record linkage is carried out based on the encoded records only, such that no sensitive information is ever revealed during the process of record linkage. The existing techniques for PPRL can be generally classified into three generations [2]. The first generation of methods only allow exact matching [20]. They generally encode field values into hash codes, through using some one-way hash functions, and then decide on whether their hash values match in an exact fashion. However, these methods have a major drawback: any small difference (e.g. a difference in a single character) between field values would result in completely different hash values, which makes them only work for exactly matching field values. To overcome this drawback, the second generation of methods use approximate matching. They aim to encrypt records in an appropriate manner such that the similarity measure between the corresponding field values in a pair of records can be calculated on the basis of their encrypted values [16]. A variety of robust techniques have been developed, for example, Bloom filter protocol [16], phonetic encoding [21], random and public reference values [22], and secure multi-party computation [23]. In addition to approximate matching, the third generation of methods also consider their scalability to large datasets. A number of approaches have been developed by combining existing blocking techniques with some encoding, perturbation, or cryptographic methods [24, 2].

To handle missing values in statistical analysis, there are two major classes of methods: one is simply ignoring the missing values, and the other is estimating the missing values based on the corresponding values of the other records in the same dataset [10]. In the former class, two approaches have been used: either

completely removing the records with missing values or ignoring the fields with
150 missing values in the process of record linkage. However, both approaches may
result in valid record pairs being missed, since removing any records or any
fields of a record is equivalent to removing data that is originally used for record
linkage. In addition, because an eliminated record will never be matched, these
methods will always increase the number of non-match record pairs. While
155 for the class of estimation based methods, only rarely have such methods been
applied to deal with missing values in record linkage. This is due to the fact
that majority of fields in records are strings, the values of which are difficult to
impute.

In addition to the statistical approaches to dealing with missing values, ma-
160 chine learning based methods have also been developed, including clustering
based techniques [25, 26], autoassociative neural networks [27], decision tree
imputation [28] and so on. In particular, the clustering based methods rely on
the idea of dividing records in a dataset into clusters, and then replacing the
missing values of records in a cluster by some statistical values (e.g. mean or
165 mode values) in the same cluster. There are two different approaches: k -means
clustering based and k -NN based. The k -means clustering based imputation
methods [26] first apply k -means clustering to divide records into clusters, and
then impute the missing values based on the records in a cluster. The k -NN
based imputation techniques [25] first decide on the k -NNs of a record with
170 missing values based on a similarity metric, and then use the corresponding
values of the k -NNs to impute the missing values in the record.

Recently, three new approaches have been proposed in [8], based on adapting
the solutions for dealing with missing values in standard classification to the
problem of record linkage with missing values. The first method redistributes
175 field weights associated with the fields with missing values to the other fields,
and assigns zero weights to the fields with missing values. The second method
imputes the similarity measure between two corresponding fields in a record
pair, when either of them has a missing value, rather than imputing the missing
value of a field. The last method adds previously considered non-identifying

180 fields to the set of identifying fields to compensate for the missing values in the existing identifying fields. However, none of the above three methods has either explored solutions to PPRL with missing values or addressed the scalability issue of linking large-scale datasets.

3. Preliminary

185 Given two datasets, R and S , the problem of record linkage is typically formulated as a classification problem, i.e. whether a pair of records, $(r, s) \in R \times S$, is classified as match, non-match or possible match [6]. A pair of records match if they refer to the same entity (e.g. a patient). The classification algorithm for record linkage uses a similarity vector for each pair of records. Each element in the similarity vector represents the similarity measure between the two
190 corresponding values of an identifying field in the record pair.

Definition 1 (Similarity Vector). Given t similarity metrics $F = [F_1, F_2, \dots, F_t]$ and a record pair $(r, s) \in R \times S$, a t -dimensional similarity vector, $f(r, s)$, is defined as $[f_1(r_1, s_1), f_2(r_2, s_2), \dots, f_t(r_t, s_t)] \in [0, 1]^t$, where $f_i(r_i, s_i)$ ($1 \leq i \leq t$) represents the similarity measure between the the pair of values in the corresponding fields, $r_i \in r$ and $s_i \in s$, as follows:

$$f_i(r_i, s_i) = \begin{cases} 1 & \text{if } F_i(r_i, s_i) \geq \gamma_i \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where r_i and s_i are the i th identifying fields of r and s respectively, and γ_i is the threshold for the i th identifying field, which determines the level of similarity measure between r_i and s_i for them to be treated as the same (i.e. the similarity value of 1). For example, given the two datasets R and S shown in Table 1,
195 a string similarity metrics, such as Jaccard distance, can be applied to field *Surname*.

For the linkage of the two datasets shown in Table 1, a record linkage algorithm should return the following classifications: $R2$ and $S1$ is a match; $R3$
200 and $S2$ as well as $R5$ and $S4$ are possible matches hence require clerical review,

while the other pairs of records are non-matches. Clerical review involves manually determining whether a pair of records matches when it cannot be decided automatically whether such a pair of records match.

Privacy Preserving Record Linkage (PPRL) aims to classify record pairs into two disjoint classes only: matches (M) and non-matches (U). In addition, only the values of those fields in a matching pair of records, which have been agreed between two parties, are revealed. Also, no clerical review is allowed due to privacy. For example, for the two datasets shown in Table 1, a PPRL approach should produce the following output: $R2$ and $S1$ is a match (i.e. in M) with the values of the agreed fields revealed only.

For two large datasets R and S , it is not practical to enumerate all record pairs $(r, s) \in R \times S$, and classify them accordingly. Blocking techniques [13, 14] typically use a blocking scheme to quickly identify a relatively small subset of record pairs, which are more likely to be matched, for the subsequent classification.

Definition 2 (Blocking Scheme). A blocking scheme is defined as a binary function $B : R \times S \rightarrow \{true, false\}$, which has the property that the set of every record pair $(r, s) \in R \times S$, where $B(r, s) = true$, can be computed efficiently.

For example, a blocking scheme could be a string similarity measure between a pair of records on a selected field, such as:

$$Jaccard(r_i, s_i) \geq \theta$$

where *Jaccard* refers to the Jaccard similarity coefficient, r_i and s_i are the i th fields of the records r and s respectively, while θ is a predefined threshold, where θ can be set to a level so that only the set of record pairs that can be compared in detail efficiently are selected. A more generalised blocking scheme can be defined, which can consist of a combination of similarity measures between two records on one or more selected fields.

225 3.1. Probabilistic Record Linkage

Probabilistic record linkage is based on the probabilistic decision rule formalised in [6]. It assigns a comparison weight score to a pair of records based on a number of similarity measures between the records on the corresponding fields, and treats a pair of records with a comparison weight score above a given threshold as a match. For a record pair $(r, s) \in R \times S$, a comparison weight score represents the likelihood ratio defined as follows:

$$\mathbb{R}(f(r, s)) = \frac{m(f(r, s))}{u(f(r, s))} \quad (2)$$

where

$$m(f(r, s)) = P(f(r, s) \mid (r, s) \in M) \quad (3)$$

and

$$u(f(r, s)) = P(f(r, s) \mid (r, s) \in U) \quad (4)$$

are the conditional probabilities of $f(r, s)$ given records r and s are a match (i.e. in M) and a non-match (i.e. in U), respectively. Under the assumption of conditional independence [6], both Eqs. (3) and (4) can be simplified as:

$$\begin{aligned} m(f(r, s)) = & m_1(f_1(r_1, s_1)) \times m_2(f_2(r_2, s_2)) \times \dots \\ & \times m_t(f_t(r_t, s_t)) \end{aligned} \quad (5)$$

and

$$\begin{aligned} u(f(r, s)) = & u_1(f_1(r_1, s_1)) \times u_2(f_2(r_2, s_2)) \times \dots \\ & \times u_t(f_t(r_t, s_t)) \end{aligned} \quad (6)$$

where for $i = 1, 2, \dots, t$

$$m_i(f_i(r_i, s_i)) = P(f_i(r_i, s_i) \mid (r, s) \in M) \quad (7)$$

and

$$u_i(f_i(r_i, s_i)) = P(f_i(r_i, s_i) \mid (r, s) \in U) \quad (8)$$

are the conditional probabilities of $f_i(r_i, s_i)$, given records r and s are a match (i.e. in M) and non-match (i.e. in U), respectively. To further simplify the

computation of the likelihood ratio in Eq. (2), a computationally convenient function \log_2 is used, such that an increase of 1 unit in $\log_2 \mathbb{R}(f(r, s))$ corresponds to an increase of 2 units in $\mathbb{R}(f(r, s))$. Therefore, the comparison weight score for a pair of records r and s can be expressed as:

$$\begin{aligned} \mathbb{W}(f(r, s)) &= \log_2 \frac{m(f(r, s))}{u(f(r, s))} \\ &= \mathbb{W}_1(f_1(r_1, s_1)) + \mathbb{W}_2(f_2(r_2, s_2)) + \dots \\ &\quad + \mathbb{W}_t(f_t(r_t, s_t)) \end{aligned} \quad (9)$$

where $\mathbb{W}_i(f_i(r_i, s_i))$ ($i = 1, 2, \dots, t$) is the comparison weight score for the i th identifying fields of r and s , obtained as follows:

$$\mathbb{W}_i(f_i(r_i, s_i)) = \begin{cases} \mathbb{W}_i^a & \text{if } f_i(r_i, s_i) = 1 \\ \mathbb{W}_i^d & \text{if } f_i(r_i, s_i) = 0 \end{cases} \quad (10)$$

where \mathbb{W}_i^a and \mathbb{W}_i^d are the agreement and disagreement weights for the i th identifying field respectively, calculated as follows:

$$\mathbb{W}_i^a = \log_2 \left(\frac{m_i(f_i(r_i, s_i) = 1)}{u_i(f_i(r_i, s_i) = 1)} \right) \quad (11)$$

and

$$\mathbb{W}_i^d = \log_2 \left(\frac{1 - m_i(f_i(r_i, s_i) = 1)}{1 - u_i(f_i(r_i, s_i) = 1)} \right) \quad (12)$$

respectively. Both $m_i(f_i(r_i, s_i))$ and $u_i(f_i(r_i, s_i))$ ($i = 1, 2, \dots, t$) can be estimated either using a given training dataset or by the EM algorithm as described in [29]. Finally, whether a pair of records r and s is a match or non-match is classified as follows:

$$(r, s) = \begin{cases} \text{match} & \text{if } \mathbb{W}(f(r, s)) \geq \lambda \\ \text{non-match} & \text{otherwise} \end{cases} \quad (13)$$

where λ is a pre-defined threshold, and its value can be obtained by balancing between an acceptable recall and precision.

3.2. Bloom Filter Protocol

The Bloom filter protocol has been proposed for encryption in record linkage [16]. Under this protocol, the dataset owners involving in the linkage process

should first agree on: A bit array of length l , in which every bit is initially set to 0, and m independent hash functions $\{h_1, h_2, \dots, h_m\}$, each of which should be one way only, such that it is impossible to determine the original input from a hash output. In addition, each hash function should always generate the same output for the same input. An input x is encrypted by each of the hash functions h_p ($p = 1, 2, \dots, m$), which produces an output $h_p(x)$. In order to store each output $h_p(x)$ in a shared Bloom filter with fixed length l , the remainder q of the division of the output $h_p(x)$ by the length of Bloom filter l is obtained by the modulo operation, mod, as:

$$q = h_p(x) \bmod l \quad (14)$$

where $0 \leq q \leq l - 1$. Then, the $(q + 1)$ th bit in the Bloom filter is set to 1. As
 230 a result, for each output $h_p(x)$, there is always a bit in the Bloom filter to be set to 1. To reduce the possibility that two different input values x and y are mapped onto the same bit in a Bloom filter, each input value is hashed m times by m hash functions (i.e. by setting a large number for m), so that it is more likely that each hash output uniquely corresponds to a bit in the Bloom filter.
 235 If the corresponding bit has already been set to 1, no change is made.

The similarity measure between two input values can therefore be compared in a privacy preserving manner once they are mapped onto two respective Bloom filters. The similarity measure between two Bloom filters can be calculated using Dice coefficient as follows:

$$F^{DC}(\mathbb{B}(r_i), \mathbb{B}(s_i)) = \frac{2 \times n_{\mathbb{B}(r_i)\mathbb{B}(s_i)}}{n_{\mathbb{B}(r_i)} + n_{\mathbb{B}(s_i)}} \quad (15)$$

where $\mathbb{B}(r_i)$ and $\mathbb{B}(s_i)$ are the Bloom filters of the values for the i th identifying fields of records r and s respectively, $n_{\mathbb{B}(r_i)\mathbb{B}(s_i)}$ represents the number of the corresponding bits in both Bloom filters $\mathbb{B}(r_i)$ and $\mathbb{B}(s_i)$, which have been set to 1, while $n_{\mathbb{B}(r_i)}$ and $n_{\mathbb{B}(s_i)}$ represents the number of bits in Bloom filters $\mathbb{B}(r_i)$
 240 and $\mathbb{B}(s_i)$ respectively, which have been set to 1.

Dice coefficient is the most common way of measuring the similarity between two Bloom filters. Compared to token-based similarity metrics (e.g. Jaccard

index, overlap coefficient, and Hamming distance), Dice coefficient is insensitive to the number of 0 bits in Bloom filters. In addition, it has been reported [15] that Dice coefficient is more suitable for large-scale record linkage, compared
 245 to an alternate metric for comparing Bloom filters (i.e. secure edit distance [30]). Secure edit distance [30] is computationally more expensive as it requires a specific way of creating Bloom filters and the password used to encrypt data needs to be known to the third party.

We now illustrate an example of encryption using the Bloom filter protocol. We need to calculate the similarity measure between $r_i = SMITH$ and $s_i = SMYTH$ in a privacy preserving manner. As shown in Table 2, we assume that the length of Bloom filter l and the number of independent hash functions m are set to 100 and 3, respectively. The output of each of the independent hash functions $h_p(x)$ ($p = 1, \dots, m$) can be efficiently computed on the basis of two independent hash functions \hat{h}_1 and \hat{h}_2 [31], as:

$$h_p(x) = \hat{h}_1(x) + p\hat{h}_2(x) \quad (16)$$

In this example, we use two well known cryptographic hash functions SHA1 and MD5 for \hat{h}_1 and \hat{h}_2 , respectively. As we can see in Table 2, 13 bits are set to 1 in both $\mathbb{B}(r_i)$ and $\mathbb{B}(s_i)$, and 17 bits are set to 1 in $\mathbb{B}(r_i)$ while 16 bits are set to 1 in $\mathbb{B}(s_i)$. Using Dice coefficient, the similarity measure between $\mathbb{B}(s_i)$ and $\mathbb{B}(r_i)$ can be calculated as follows:

$$\begin{aligned} F^{DC}(\mathbb{B}(r_i), \mathbb{B}(s_i)) &= \frac{2 \times n_{\mathbb{B}(r_i)\mathbb{B}(s_i)}}{n_{\mathbb{B}(r_i)} + n_{\mathbb{B}(s_i)}} \\ &= \frac{2 \times 13}{16 + 17} \\ &\approx 0.788 \end{aligned} \quad (17)$$

250 4. Proposed Approach

In this section, we first describe a k -NN based approach for dealing with missing values in the standard probabilistic record linkage setting. We then de-

scribe how such an approach can be used in the probabilistic privacy preserving record linkage in the presence of missing values.

255 *4.1. A k -NN Based Approach to Probabilistic Record Linkage in the Presence of Missing Values*

For each record in a dataset with a missing value in any of its identifying fields, we first find a set of records in the same dataset, which are the k -NNs of the record. This set of records will then be used to impute the similarity
260 measure between the missing value and a value of the corresponding field in any of the possible matching records in another dataset.

4.1.1. k -NN Graph Construction

Given each record in a dataset, we first generate a k -NNG (Nearest Neighbour Graph) in which a node represents the record, k other nodes represent k
265 records in the same dataset, which are most similar to the given record by a given similarity measure, with each of the k nodes connected to the node for the given record.

The naive way of generating all the k -NNGs for all the records in a large dataset is computationally expensive[32]. As a result, much research has been
270 focused on generating k -NNGs in an efficient manner [33]. For example, techniques have been proposed for generating exact k -NNGs [34], and approximate k -NNGs using space partition trees [35, 36], local search [37] and locality sensitive hashing [38]. In record linkage, blocking techniques [24] have been proposed for reducing the number of record pairs to be compared. In particular, those
275 blocking techniques, referred as intra-blocking schemes can be used to split a dataset into non-overlapping blocks, where records within the same block are more similar to each other than to those in a different block. For a record with a missing value, we find its k -NNs in the same block.

An intra-blocking scheme is based on comparing either the values of a single
280 record field, or the concatenation of values from several record fields. As a result, an intra-blocking scheme would require at most $n_t \times (n_t - 1)/2$ computations,

which corresponds to a computational complexity of $O(n_t^2/2)$, where n_t refers to the total number of records in a dataset. Usually, such a scheme is chosen to be cheap to operate, so that it can be run in a reasonable time for moderately sized datasets [39]. For an intra-blocking scheme, if there is any missing value in the record field(s) for intra-blocking, it would ignore the missing value and simply return a *false*.

Given a t -dimensional similarity metrics $\rho = [\rho_1, \rho_2, \dots, \rho_t]$ for t identifying fields, for each record in a dataset R with a missing value in any of its identifying fields, we need to calculate the similarity measure between the given record and each of the other records in R in order to find the k -NNs of the given record.

We use the probabilistic record linkage framework described in Section 3.1. For a pair of records r^a and r^b in the same dataset, its comparison weight score $\mathbb{W}^\rho(\rho(r^a, r^b))$ can be calculated as:

$$\begin{aligned} \mathbb{W}^\rho(\rho(r^a, r^b)) = & \mathbb{W}_1^\rho(\rho_1(r_1^a, r_1^b)) + \mathbb{W}_2^\rho(\rho_2(r_2^a, r_2^b)) + \dots \\ & + \mathbb{W}_t^\rho(\rho_t(r_t^a, r_t^b)) \end{aligned} \quad (18)$$

where $[r_1^a, r_2^a, \dots, r_t^a]$ and $[r_1^b, r_2^b, \dots, r_t^b]$ are the t -dimensional vectors of records r^a and r^b . If there is a missing value of any identifying field in either $[r_1^a, r_2^a, \dots, r_t^a]$ or $[r_1^b, r_2^b, \dots, r_t^b]$, we set $\rho_i(r_i^a, r_i^b) = 0$, for $i = 1, 2, \dots, t$.

Finally we set a threshold, ε , so that only those records that have a comparison weight score above the threshold are selected and ranked as the possible k -NNs of the record. Formally, a record r^c is labelled as a k -NN of record r^a , if $\mathbb{W}^\rho(\rho(r^a, r^c)) \geq \varepsilon$ and r^c is among the k -NNs of r^a . The pseudocode for finding the k -NNs of a record with a missing value is presented in Algorithm 1.

In Algorithm 1, there are two iterations: the main part of the computational cost is in the first iteration where the comparison weight score between the record with a missing value and each of the other records in the same dataset is calculated. The second iteration involves filtering through a set of similarity measures with a pre-defined threshold only to select the set of k -NNs. The total computational complexity is in the order of $O(n_k \times t)$, where n_k is the number of records in the same block, and t is the dimensionality of the record vector.

Algorithm 1 Finding the k -NNs of a record with a missing value

Input: record r^u , n_k records $\{r^{v_1}, r^{v_2}, \dots, r^{v_n}\}$ in the same block of r^u , number of k -NNs k , threshold ε

Output: k -NN set K^u

```

1:  $G \leftarrow \emptyset$ 
2:  $K^u \leftarrow \emptyset$ 
3: for  $i = 1$  to  $n_k$  do
4:   Compute  $\mathbb{W}^\rho(\rho(r^u, r^{v_i}))$  with (18)
5:    $G \leftarrow G \cup \{\mathbb{W}^\rho(\rho(r^u, r^{v_i}))\}$ 
6: end for
7: Select  $k$  records in  $G$ ,  $\{r^{k_1}, r^{k_2}, \dots, r^{k_k}\}$ 
8: for  $i = 1$  to  $k$  do
9:   if  $\mathbb{W}^\rho(\rho(r^u, r^{k_i})) \geq \varepsilon$  then
10:     $K^u \leftarrow K^u \cup \{r^{k_i}\}$ 
11:   end if
12: end for
13: return  $K^u$ 

```

4.1.2. Calculating Weights of k -NNs

We generate a weight vector for the k -NNs of each record with a missing value. Each element in the weight vector is assigned on the basis of the comparison weight score between the given record and each of its corresponding k -NNs. Given a record r^u in dataset R , and its k -NNs $K^u = \{r^{k_1}, r^{k_2}, \dots, r^{k_k}\}$ in R , the corresponding weight vector w^u is computed as follows:

$$w^u = [\mathbb{W}^\rho(\rho(r^u, r^{k_1})), \mathbb{W}^\rho(\rho(r^u, r^{k_2})), \dots, \mathbb{W}^\rho(\rho(r^u, r^{k_k}))] \quad (19)$$

where $\mathbb{W}^\rho(\rho(\cdot, \cdot))$ is the comparison weight score calculated using Eq. (18). Furthermore, the weight vector w is normalised to a unit vector $\overline{w^u}$, as:

$$\overline{w^u} = \frac{w^u}{\|w^u\|_2} \quad (20)$$

where $\|w^u\|_2$ is the 2-norm of w^u and computed as:

$$\|w^u\|_2 = \sqrt{\sum_{i=1}^k \mathbb{W}\rho(\rho(r^u, r^{k_i}))^2} \quad (21)$$

4.1.3. Dealing with Missing Values in Record Pairs

In order to classify a pair of records as match or non-match, the classifier
 310 needs to take as input a similarity vector for the pair. Each element in the
 similarity vector represents the similarity measure between the values of the
 two corresponding identifying fields in the records. However, when either of the
 values is missing, instead of imputing the missing value, we impute the similarity
 measure using the k -NNs of the record with the missing value. The similarity
 315 measure $F_i(r_i, s_i)$ on the i th identifying fields of records r and s , when either of
 the identifying fields has a missing value, is computed as follows:

$$F_i(r_i, s_i) = \begin{cases} F_i(\sim, s_i) & \text{if } r_i \text{ has a missing value} \\ F_i(r_i, \sim) & \text{if } s_i \text{ has a missing value} \\ 0 & \text{if both } r_i \text{ and } s_i \text{ have} \\ & \text{missing values} \end{cases} \quad (22)$$

where $F_i(\sim, s_i)$ represents the imputed similarity measure given r_i has a missing
 value, calculated as follows:

$$F_i(\sim, s_i) = F_i(r_i^{x_1}, s_i) \times \overline{w^{x_1}} + F_i(r_i^{x_2}, s_i) \times \overline{w^{x_2}} + \dots \\ + F_i(r_i^{x_k}, s_i) \times \overline{w^{x_k}} \quad (23)$$

where $\{r_i^{x_1}, r_i^{x_2}, \dots, r_i^{x_k}\}$ are the k values of the i th identifying fields in the k -
 NNs of the record r , and $\overline{w^x} = [\overline{w^{x_1}}, \overline{w^{x_2}}, \dots, \overline{w^{x_k}}]$ is the corresponding weight
 vector of the k -NNs.

Similarly, $F_i(r_i, \sim)$ represents the imputed similarity measure given s_i has a
 missing value, calculated as follows:

$$F_i(r_i, \sim) = F_i(r_i, s_i^{y_1}) \times \overline{w^{y_1}} + F_i(r_i, s_i^{y_2}) \times \overline{w^{y_2}} + \dots \\ + F_i(r_i, s_i^{y_k}) \times \overline{w^{y_k}} \quad (24)$$

320 where $\{s^{y_1}, s^{y_2}, \dots, s^{y_k}\}$ are the k values of the i identifying fields in the k -NNs of record s , and $\bar{w}^y = [\bar{w}^{y_1}, \bar{w}^{y_2}, \dots, \bar{w}^{y_k}]$ is the corresponding weight vector of the k -NNs.

To illustrate, assume that the similarity measure between the i th identifying fields r_i and s_i (e.g. $r_i = SMITH$ and $s_i = SMYTH$) of the records r and s needs to be computed. We assume that an appropriate similarity metrics can be used for measuring the similarity between the i th fields in the records. For purpose of illustration, we use the combination of 2-gram (i.e. bigram) and Dice coefficient. It has been shown in [40] that using the combination of 2-gram (i.e. bigram) and Dice coefficient could achieve a higher accuracy rate, compared to other string similarity measures. Using the 2-gram technique, each of r_i and s_i is split into a set of two adjacent letters, with blanks first padded on both sides of the string to make the first and last letters their own bigrams (i.e. $\{-S, SM, MI, IT, TH, H-\}$ and $\{-S, SM, MY, YT, TH, H-\}$ for r_i and s_i , respectively). In both bigrams of r_i and s_i , there are 6 members, 4 of which are in common. The Dice coefficient between r_i and s_i is:

$$\begin{aligned} F^{DC}(r_i, s_i) &= \frac{2 \times n_{r_i s_i}}{n_{r_i} + n_{s_i}} \\ &= \frac{2 \times 4}{6 + 6} \\ &\approx 0.667 \end{aligned} \quad (25)$$

The value of Dice coefficient ranges between 0 and 1, with a higher value representing a higher degree of similarity. Assume that we still have $r_i = SMITH$ but the value of s_i is missing. In this case, we have three values SMY and two $SMYTH$ s that are the corresponding values of the i th identifying fields in the 3-NNs of record s , along with their corresponding weight vector $\bar{w} = [0.35, 0.33, 0.32]$. First, we would need to generate three corresponding bi-gram sets for the three different values (i.e. $SMITH$, SMY and $SMYTH$): $\{-S, SM, MI, IT, TH, H-\}$, $\{-S, SM, MY, Y-\}$ and $\{-S, SM, MY, YT, TH, H-\}$, respectively. It can be seen that there are 6 and 4 members in the bi-gram sets of $SMITH$ and SMY respectively, 2 of which are in common. Then, using the

same similarity metrics (i.e. Dice coefficient), we can impute the similarity measure between r_i and s_i with SMY , the two $SMYTH$, and their corresponding weights as follows:

$$F^{DC}(r_i, \sim) = \frac{2 \times 2}{6 + 4} \times 0.35 + \frac{2 \times 4}{6 + 6} \times (0.33 + 0.32) \approx 0.573 \quad (26)$$

It can be seen that the difference between the similarity values calculated using Eqs. (25) and (26) is fairly small, which shows the promise of the proposed approach for imputing similarity measures.

4.2. Privacy Preserving Record Linkage with Missing Values

Privacy Preserving Record Linkage (PPRL) requires that the fields about personal information in each of the two datasets to be encrypted by their corresponding owner: First, the two owners agree on a password or pass phrase for the purpose of encryption; Then they encrypt these fields using an encryption software; Finally, the encrypted fields are used as encrypted identifying fields for record linkage.

4.2.1. Privacy Preservation in the Presence of Missing Values

In order to deal with missing values in privacy preserving record linkage, we propose to adapt the Bloom filter approach to PPRL developed in [16]. There are several reasons why we have chosen to adapt this approach over other methods for privacy preserving record linkage in the presence of missing values. First, the Bloom filter approach allows the similarity measure between the two values of the corresponding identifying fields in two records to be calculated even after the two values have been encrypted, which is required in probabilistic record linkage. Second, it has shown quality improvements over other privacy preserving protocols [16] like the Swiss anonymous linkage code [41], which implements an identifier based on the phonetic codes of some identifying fields. Third, it appears robust, well-developed, and adaptable for large-scale record linkage [15]. Finally, and most importantly, the Bloom filter approach can be

easily adapted to work with the k -NNs of a record with a missing value and the weight vector of the k -NNs.

The Bloom filter approach to PPRL [16] uses a three-party protocol, where the linkage is done by a (trusted) third party (i.e. the linkage unit) in a Honest-
 350 But-Curious (HBC) model [2]. The two dataset owners encrypt their data records, and transfer the encrypted data securely to the linkage unit. The HBC model assumes that each of the three parties correctly follows the protocol, while being curious about whether they are able to find out as much information as possible from any received data [3]. The Bloom filter approach [16] can stop
 355 any dictionary attack as long as the third-party linkage unit does not collude with either of the dataset owners. Although there are still some other types of potential attacks (e.g. frequency attacks), a number of additions to the original Bloom filter approach have enhanced its level of security [42]. For each record with a missing value, we need to encrypt the values of the corresponding
 360 identifying fields in the k -NNs of the record, along with their weight vector. We propose a method to encrypt each of these values onto a Bloom filter, along with the weight of the corresponding NN. To achieve this, instead of simply setting the bit in the Bloom filter, which corresponds to the hash output of the bigram set of such a value, to 1, it is set to the weight of the corresponding NN. When
 365 the hash output of more than one bigram set is mapped to the same bit in the Bloom filter, the bit is set to the sum of all the corresponding weights.

In the previous example, we have $r_i = SMITH$ in record r while the value of s_i is missing in record s . We also have three values SMY and two $SMYTH$ s that are the corresponding values of the i th identifying fields in the 3-NNs of record s , along with their weight vector $\bar{w} = [0.35, 0.33, 0.32]$. We now need to map the bigrams of these three values onto two separate Bloom filters: one for $SMITH$, while the other for both SMY and two $SMYTH$. We can then calculate the Dice coefficient between the two Bloom filters, which imputes the similarity measure between $r_i = SMITH$ and s_i with a missing value. Since we now have real numbers instead of binary numbers in the Bloom filter for SMY and the two $SMYTH$, the original Dice coefficient formula as shown in Eq.

(15) needs to be adapted as follows:

$$\hat{F}^{DC}(\mathbb{B}(r_i), \mathbb{B}(s_i)) = \frac{2 \times \text{sum}(\mathbb{B}(r_i) \cdot \mathbb{B}(s_i))}{\text{sum}(\mathbb{B}(r_i)) + \text{sum}(\mathbb{B}(s_i))} \quad (27)$$

where the two Bloom filters $\mathbb{B}(r_i)$ and $\mathbb{B}(s_i)$ are treated as vectors, and \cdot represents the dot product, and $\text{sum}(\cdot)$ corresponds to the sum of every value in a vector. It can be seen that, for two Bloom filters with binary values, the original
 370 Dice coefficient formula (as in Eq. (15)) is a special case of the adapted Dice coefficient formula (as in Eq. (27)).

With the Bloom filter length l set to the exactly same as before, and the bigrams in Table 2, the calculation of the Dice coefficient between $r_i = SMITH$ and s_i with a missing value is done in a privacy preserving manner, as follows:

$$\begin{aligned} \hat{F}^{DC}(\mathbb{B}(r_i), \mathbb{B}(\sim)) &= \frac{2 \times \text{sum}(\mathbb{B}(r_i) \cdot \mathbb{B}(\sim))}{\text{sum}(\mathbb{B}(r_i)) + \text{sum}(\mathbb{B}(\sim))} \\ &= \frac{2 \times (0.65 + 1 \times 4 + 0.65 \times 3 + 1 \times 3 + 0.65 + 1)}{17 + 14.250} \\ &= \frac{2 \times 11.250}{17 + 14.250} \\ &= 0.720 \end{aligned} \quad (28)$$

where the Bloom filter $\mathbb{B}(\sim)$ is calculated with SMY , and $SMYTHs$, along with the weight vector \bar{w} of the corresponding 3-NNs, as shown in the last column of Table 2. Again, comparing the similarity measures calculated by Dice coefficient
 375 in Eqs. (17) and (28), we can see that the difference between the two measures is also fairly small. This shows that the proposed approach for dealing with missing values can work well in the context of privacy preserving record linkage using the Bloom filter protocol.

4.2.2. Computational Complexity

380 In this section, we analyse the computational complexity of different stages of our proposed approach to privacy preserving record linkage in the presence of missing values. Our approach starts with the intra-blocking process in each of the datasets R and S . Assume that the numbers of records in R and S are n_R and n_S , and their corresponding numbers of missing values are m_R and m_S .

385 For the two intra-blocking processes in R and S respectively, the correspond-
 ing computational complexities are $O(n_R^2/2)$ and $O(n_S^2/2)$, respectively. In the
 next stage, our approach searches for the k -NNs of each record with missing
 values in the same block. For both datasets, R and S , this search process has
 the computational complexity of $O((m_R + m_S) \times n_{kA} \times t)$, where n_{kA} refers to
 390 the average number of records within each block, and t is the dimensionality of
 the record vector. For the next imputation process, the corresponding computa-
 tional complexity would be $O((m_R + m_S) \times k \times l)$, where k corresponds to
 k -NN and l is the length of the Bloom filter. For the following blocking process,
 it would require a hash operation with a complexity of $O((n_R + n_S) \times q_B \times p)$, a
 395 communication cost with a complexity of $O((n_R + n_S) \times l)$, and a bit comparison
 with a complexity of $O(n_R \times n_S \times l^2)$, where q_B is the average number of n -grams
 in each record field for blocking, and p is the number of hash functions used to
 map n -grams into a Bloom filter. For the final matching stage, there would be
 another hash operation with a complexity of $O(n_B \times (n_{RL} + n_{SL}) \times q_L \times p)$,
 400 another communication cost with a complexity of $O(n_B \times (n_{RL} + n_{SL}) \times l)$, and
 another bit comparison with a complexity of $O(n_B \times n_{RM} \times n_{SM} \times l^2)$, where
 n_B refers to the number of blocks after blocking, n_{RL} and n_{SL} represent the
 average number of records in each block from R and S respectively, q_L is the
 average number of n -grams in each record field for linkage, and n_{RM} and n_{SM}
 405 correspond to the maximum numbers of records in each block from R and S
 respectively.

5. Experimental Evaluation

In this section, we present the experimental results of our proposed approach.

We have compared our approach with five existing algorithms for record linkage
 410 in both the absence (i.e. PRL and BF) and presence (i.e. FRIL-0, FRIL-100
 and FLE) of missing values: the standard Probabilistic Record Linkage (PRL)
 method [6], the Bloom Filter (BF) approach [15, 16], the two methods available
 in the Fine-grained Record Integration and Linkage (FRIL) tool FRIL-0 and

FRIL-100 [43], as well as the Full Linkage Expansion (FLE) technique proposed
 415 in [8]. We conducted all the experiments using MATLAB R2014b on a machine
 with 3.6-GHz Intel Core i7 CPU and 16-GB DDR3 RAM running the Windows
 7 operating system. The objective of our evaluation is to show that our proposed
 method outperforms the three existing techniques (i.e. FRIL-0, FRIL-100, and
 FLE) for record linkage in the presence of missing values, while the other two
 420 existing methods (PRL and BF) are used as the baselines for standard record
 linkage and privacy preserving recording linkage without missing values.

5.1. Datasets

For both algorithms PRL and BF, there is no missing value in the datasets for
 evaluation. For the other three algorithms FRIL-0, FRIL-100 and FLE, as well
 425 as our proposed approach, there are a certain percentage of missing values in the
 datasets. We have generated and used a collection of synthetic datasets based on
 GeCo [44, 45]. There are several benefits from choosing synthetic datasets over
 real ones: First, the dataset size can be set with a computational cost estimate.
 Second, we can define the record fields in the dataset. For example, for our
 430 experimental evaluation, we have chosen six fields (i.e. *GivenName*, *Surname*,
Postcode, *Telephone*, *Gender*, and *City*) for linkage. These are often available
 in real datasets. Third, the ground truth of the record linkage results is known,
 to facilitate the quality assessment for benchmarking. Finally, the percentage
 of missing values in datasets can be controlled, such that benchmarking can be
 435 carried out at different levels. Each pair of synthetic datasets consists of two
 individual datasets, R and S , as well as the two corresponding datasets R' and
 S' with a certain percentage of missing values. Table 3 shows the characteristics
 of datasets R and S , including the sizes (i.e. the number of records), the first
 six fields used as identifying fields, the similarity metrics used for each of the six
 440 identifying fields, the blocking and intra-blocking schemes used. There is also
 an additional 7th ID field in the datasets for the purpose of labelling matching
 pairs of records only. For brevity, the names of the first six fields are abbreviated
 by their first letters.

20% of the records in datasets R and S are the same (i.e. those records
 445 with the same IDs in both datasets). To simulate various types of errors (e.g.
 typographical errors) in records to reflect the quality of typical linked datasets, a
 variety of the corruption methods in GeCo [44, 45] have been used (e.g. phonetic
 variation, keyboard mistake, name misspelling), to randomly corrupt any four
 of the first six fields of each record. Each pair of datasets R' and S' have been
 450 generated from the same pair of datasets R and S respectively with a certain
 percentage of records in R' and S' with a missing value in one of the first six
 fields. To generalise the characteristics of synthetic datasets, three pairs of
 synthetic datasets R and S have been generated, along with three pairs of R'
 and S' with missing values, which have 10%, 20%, and 25% of records with
 455 missing values. We have used two similarity metrics (i.e. Dice coefficient (\mathbb{DC})
 and string equality (\mathbb{SE})) for calculating the similarity measures between the
 corresponding values for each of the first six fields, as shown in Table 3. Both
 similarity metrics can be computed efficiently [1]. In Table 3, for example,
 $\mathbb{DC}(GN)$ denotes that Dice coefficient has been used on field *GivenName*, and
 460 $\mathbb{DC}(P,C)$ denotes that Dice coefficient has been used on the concatenation of
 fields *Postcode* and *City*.

5.2. Algorithms

PRL and BF are the two algorithms for standard record linkage and privacy
 preserving record linkage respectively. We run them on the three pairs
 465 of datasets R and S respectively, to benchmark the performances of standard
 record linkage and privacy preserving record linkage on each pair of datasets.
 FRIL-0 and FRIL-100 are two of the existing algorithms for record linkage with
 missing values: FRIL-0 assumes that each of the missing values is completely
 different from the value of the corresponding field in the matching record from
 470 another dataset (i.e. the similarity measure for the corresponding field in the
 similarity vector is 0). In contrast, FRIL-100 assumes that each of the missing
 values is identical to the value of the corresponding field in the matching record
 from another dataset (i.e. the similarity measure for the corresponding field in

the similarity vector is 1). FLE is one of the three algorithms proposed in [8]
 475 for record linkage with missing values, which is reported to have achieved the
 best performance among the three algorithms. For a record with missing val-
 ues, FLE first redistributes the weights associated with the fields with missing
 values to the other fields without missing values, where the weight redistribu-
 tion is based on relative proportions across the remaining fields; It then assigns
 480 zero weight to those fields with missing values. We have run FRIL-0, FRIL-100
 and FLE for record linkage with missing values on each pair of datasets R' and
 S' to benchmark the performance of record linkage with missing values. Their
 performances have been compared with the performances of our algorithms for
 privacy preserving record linkage with missing values.

485 5.3. Evaluation

For the PRL, FRIL-0, FRIL-100 and FLE algorithms, no field in the datasets
 was encrypted. For the algorithm BF and our algorithm for privacy preserving
 record linkage with missing values, all the fields in the datasets that contain
 personal information were encrypted using the Bloom filter protocol. The sim-
 490 ilarity measure for each of the fields in the similarity vector for each pair of
 records was calculated using a combination of bigrams and Dice coefficient. For
 encryption using the Bloom filter protocol, the implementation described in [16]
 was used for creating Bloom filters with some adaptations: the length of the
 Bloom filters l and the number of independent hash functions m were set to 100
 495 and 3, respectively. The settings were set the same as for the method in [15],
 with the same ratio of l to m as in [16] (i.e. the ratios of l to m set to 1000
 and 30, respectively). Although these settings may result in a slightly higher
 false positive rate, compared to those in [16], they helped dramatically reduce
 the sizes of the Bloom filters while achieving the same performance.

500 For each pair of datasets, the performances of the five existing algorithms
 PRL, FRIL-0, FRIL-100, FLE and BF were compared with the performances
 of our own algorithms respectively. In all the existing algorithms and our own
 algorithms, we used the probabilistic record linkage framework, where the agree-

ment and disagreement weights were calculated using the EM method described
 505 in [29]. The performances of each algorithm with different thresholds were measured, with the highest performance used for comparison.

Two blocking schemes, as shown in Table 3, were used for blocking: one is Dice coefficient between the bigrams of the concatenation *Surname* and the first initial of *GivenName*, and the other is Dice coefficient between the special
 510 bigrams of *Telephone*. Since the values of field *Telephone* are in numeric string format, the order of occurrences of individual digits needs to be kept. In addition, there are smaller variations among different string positions (i.e. digits are from 0 to 9 only), which makes the standard bigram technique less effective for distinguishing them from each other. To tolerate small typographical errors
 515 in *Telephone* (e.g. due to typing errors), special n -grams were generated for *Telephone*. The special n -grams of a string are the combinations of individual n -grams of the string and the order of their occurrences. For example, for a value 1234567 of *Telephone*, its special 2-gram set is {112, 223, 334, 445, 556, 667} (with the first digits in each bigram indicating its order). Similarly, three intra-
 520 blocking schemes, as shown in Table 3, were defined for the efficient search of the k -NNs of a record with a missing value in a dataset.

5.4. Linkage Results

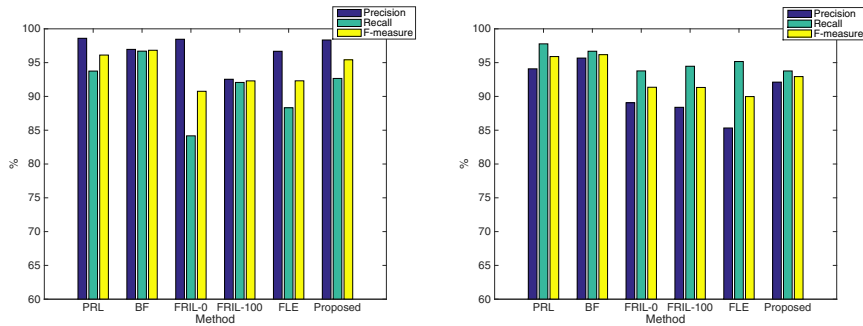
Figure 1 shows the comparison of performances across different algorithms. In each of the subfigures, a vertical dashed line is used to separate those algo-
 525 rithms that do not deal with missing values and the other algorithms that deal with missing values. The quality of each record linkage algorithm was evaluated on precision, recall and F-measure respectively. F-measure is the harmonic mean of precision and recall. For each of the six algorithms, a threshold λ for determining whether a pair of records is a match or non-match was set to the level
 530 that maximised the corresponding F-measure. As shown in Figure 1, for the three different rates of missing values, our algorithm outperforms both FRIL-100 and FLE on both precision and recall. Though the precision of our algorithm is slightly lower than that of FRIL-0, FRIL-0 takes the most cautious approach

by assuming that a missing value is completely different from the value of the
 535 corresponding field in the matching record from another dataset (i.e. similarity
 measure is 0). On the other hand, our algorithm does not have this assumption,
 hence handling missing values better. Compared with the other two algorithms
 PRL and BF, our algorithm achieved a similar precision but a lower recall. This
 was because that some records with missing values in the datasets do not have
 540 any k -NNs or some of their k -NNs also have missing values in the same fields.
 As the rate of missing values increases, the recall of our algorithms decrease.

To show that our proposed approach works well with the traditional blocking
 techniques to address the scalability issue in record linkage, the percentages of
 record pairs (to be matched) rejected and true matches retained after blocking
 545 by each of the six methods for three different rates of missing values are shown
 in Figures 2 and 3, respectively. It can be seen that our proposed approach
 outperformed the other three algorithms for record linkage with missing values,
 especially at a higher rate of missing values. This is because that the proposed
 approach imputes similarity measures between two records on the fields with
 550 missing values, which can then be used for blocking.

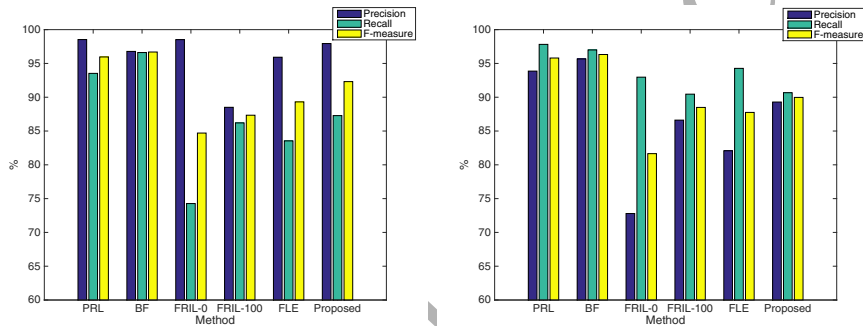
5.5. Computational Costs

To show the scalability of the proposed approach, the runtime of each of the
 methods on each of the datasets was recorded to produce the average runtimes,
 as shown in Table 4. All the six methods were run independently in Matlab:
 555 for both the proposed method and BF, the similarity calculation was done on
 vectors (i.e. Bloom filters); while for the other four methods, the similarity
 calculation was done on sets of string tokens. For each experiment for each of
 the six methods, the number of computational thread in Matlab was set to 1.
 From Table 4, it can be seen that our proposed approach has a relatively low
 560 computational cost at the intra-blocking stage. The additional pay of the pro-
 posed method on runtime is mainly in the stage of intra-blocking (i.e. handling
 missing values). Since the missing values in a record are handled within its own
 dataset, the process of handling missing values can be carried out in parallel



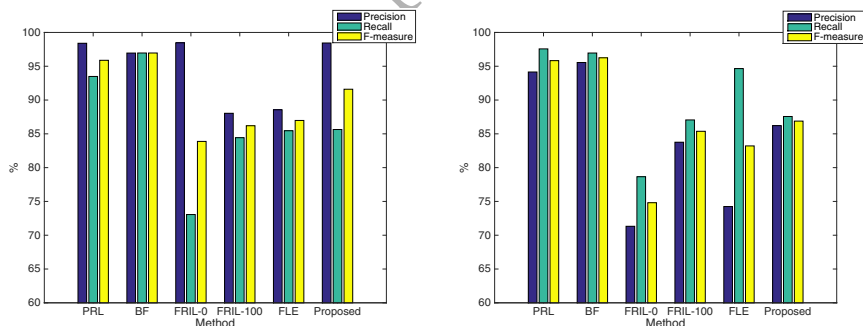
(a) Pair 1 (10% missing)

(b) Pair 1 (10% missing)



(c) Pair 2 (20% missing)

(d) Pair 2 (20% missing)

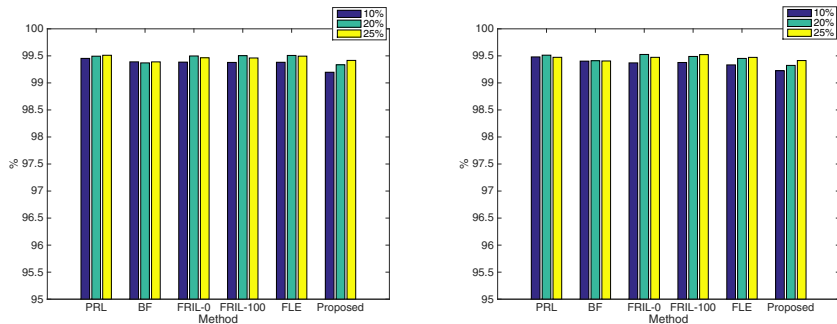


(e) Pair 3 (25% missing)

(f) Pair 3 (25% missing)

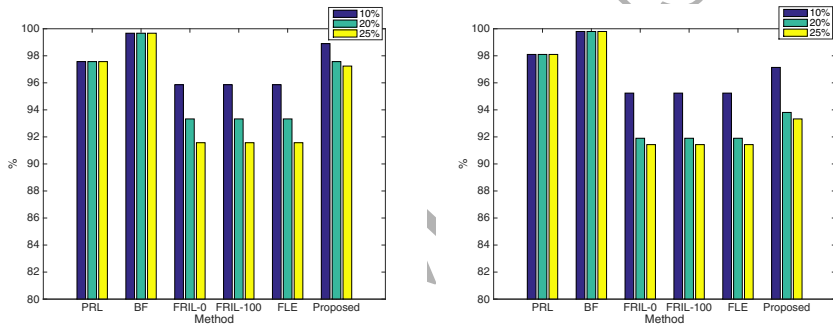
Figure 1: Comparison of linkage qualities across different methods: at most one missing value per record (the left column), and at most two missing values per record (the right column).

by each dataset owner before the process of record linkage. At the stage of
 565 blocking, both BF and our proposed approach outperform the other methods,



(a) at most one missing value per record (b) at most two missing values per record

Figure 2: Comparison of percentages of record pairs (to be matched) rejected after blocking.



(a) at most one missing value per record (b) at most two missing values per record

Figure 3: Comparison of percentages of true matches retained after blocking.

as both methods utilise Bloom filters (i.e. vectors) while Matlab works more efficiently with vectors than with string tokens. At the last stage, the linkage stage, the two methods are a little bit slower than the other methods. There are two reasons: One is that the encryption through Bloom filters are computationally quite expensive, at the linkage stage, because all the fields need to be encrypted; while at the blocking stage, only the fields used as blocking keys need to be encrypted. The other is through blocking, the number of record pairs for comparison is significantly reduced, as a result, the advantage of Matlab on computing vectors is not too obvious.

575 6. Conclusions

In this paper, we have proposed a new approach to privacy preserving record linkage in the presence of missing values, while at the same time addressing the scalability issue. Our approach is based on the assumption that the k -NNs of a record with a missing value in a large dataset would provide a set of similar values to the missing value, which can be used to impute the similarity measure between the record with the missing value and a record from another dataset. With the imputed similarity measures, an existing blocking technique can be directly used to deal with the scalability of record linkage. We have also adapted the Bloom filter approach so that the k -NNs of a record with a missing value can be encrypted together with the record, such that record linkage in the presence of missing values can be carried out in a privacy preserving manner. Compared with the existing algorithms for record linkage using three pairs of simulated datasets with different rates of missing values, we have shown that our proposed approach has achieved reasonably good linkage performances in the presence of missing values in a privacy preserving manner. In future work, we plan to apply the proposed approach to the real-world large-scale datasets.

Acknowledgment

This research is supported by the UK ESRC project, Administrative Data Research Centre - Northern Ireland (ADRC-NI, ES/L007509/1).

595 References

- [1] A. Elmagarmid, P. Ipeirotis, V. Verykios, Duplicate record detection: A survey, *IEEE Trans. Knowl. Data Eng.* 19 (1) (2007) 1–16.
- [2] D. Vatsalan, P. Christen, V. Verykios, A taxonomy of privacy-preserving record linkage techniques, *Inf. Syst.* 38 (6) (2013) 946–969.

- 600 [3] R. Hall, S. Fienberg, Privacy-preserving record linkage, in: Privacy in Statistical Databases - UNESCO Chair in Data Privacy, International Conference, PSD 2010. Proceedings, 2010, pp. 269–283.
- [4] L. Antonie, K. Inwood, D. Lizotte, J. Ross, Tracking people over time in
19th century canada for longitudinal analysis, *Machine Learning* 95 (1)
605 (2014) 129–146.
- [5] P. Christen, *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*, Springer Science & Business Media, 2012.
- [6] I. Fellegi, A. Sunter, A theory for record linkage, *Journal of the American*
610 *Statistical Association* 64 (328) (1969) 1183–1210.
- [7] H. Newcombe, J. Kennedy, Record linkage: making maximum use of the discriminating power of identifying information, *Commun. ACM* 5 (11) (1962) 563–566.
- [8] T. Ong, M. Mannino, L. Schilling, M. Kahn, Improving record linkage
615 performance in the presence of missing linkage data, *Journal of Biomedical Informatics* 52 (2014) 43–54.
- [9] M. Forster, C. Bailey, M. Brinkhof, C. Graber, A. Boulle, M. Spohr, E. Balestre, M. May, O. Keiser, A. Jahn, M. Egge, Electronic medical record systems, data quality and loss to follow-up: survey of antiretroviral therapy programmes in resource-limited settings, *Bulletin of the World*
620 *Health Organization* 86 (2008) 939–947.
- [10] J. Hair, *Multivariate Data Analysis*, 2nd Edition, Pearson Prentice Hall, 2006.
- [11] C. Clifton, M. Kantarcioglu, A. Doan, G. Schadow, J. Vaidya, A. Elmagarmid, D. Suci, Privacy-preserving data integration and sharing, in: Proceedings of the 9th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, DMKD 2004, 2004, pp. 19–26.
- 625

- [12] P. Christen, D. Vatsalan, Z. Fu, Advanced record linkage methods and privacy aspects for population reconstruction - A survey and case studies, in: Population Reconstruction, 2015, pp. 87–110.
- [13] M. Bilenko, B. Kamath, R. Mooney, Adaptive blocking: Learning to scale up record linkage, in: Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006), 2006, pp. 87–96.
- [14] A. McCallum, K. Nigam, L. H. Ungar, Efficient clustering of high-dimensional data sets with application to reference matching, in: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, 2000, pp. 169–178.
- [15] S. Randall, A. Ferrante, J. Boyd, J. Bauer, J. Semmens, Privacy-preserving record linkage on large real world datasets, *Journal of Biomedical Informatics* 50 (2014) 205–212.
- [16] R. Schnell, T. Bachteler, J. Reiher, Privacy-preserving record linkage using bloom filters, *BMC Med. Inf. & Decision Making* 9 (2009) 41.
- [17] T. Herzog, F. Scheuren, W. Winkler, *Data Quality and Record Linkage Techniques*, Springer Science - Business Media, 2007.
- [18] S. Grannis, J. Overhage, C. McDonald, Analysis of identifier performance using a deterministic linkage algorithm, in: AMIA 2002, American Medical Informatics Association Annual Symposium, 2002.
- [19] D. Wilson, Beyond probabilistic record linkage: Using neural networks and complex features to improve genealogical record linkage, in: The 2011 International Joint Conference on Neural Networks, IJCNN 2011, 2011, pp. 9–14.
- [20] C. Quantin, H. Bouzelat, F. Allaert, A. Benhamiche, J. Faivre, L. Dusserre, How to ensure data security of an epidemiological follow-up: quality assessment of an anonymous record linkage procedure, *Int J Med Inform* 49 (1) (1998) 117–122.

- [21] A. Karakasidis, V. Verykios, Privacy preserving record linkage using phonetic codes, in: 2009 Fourth Balkan Conference in Informatics, BCI 2009, 2009, pp. 101–106.
- [22] C. Pang, L. Gu, D. Hansen, A. Maeder, Privacy-preserving fuzzy matching
660 using a public reference table, *Intelligent Patient Management* 189 (2009)
71–89.
- [23] M. Atallah, F. Kerschbaum, W. Du, Secure and private sequence comparisons, in: *Proceedings of the 2003 ACM Workshop on Privacy in the Electronic Society, WPES 2003*, 2003, pp. 39–44.
- [24] P. Christen, A survey of indexing techniques for scalable record linkage and
665 deduplication, *IEEE Trans. Knowl. Data Eng.* 24 (9) (2012) 1537–1555.
- [25] E. Acuna, C. Rodriguez, The treatment of missing values and its effect on classifier accuracy, in: *Classification, Clustering, and Data Mining Applications*, 2004, pp. 639–647.
- [26] K. Wagstaff, Clustering with missing values: No imputation required, in:
670 *Classification, Clustering, and Data Mining Applications*, 2004, pp. 649–
658.
- [27] D. Pyle, *Data Preparation for Data Mining*, Morgan Kaufmann, 1999.
- [28] J. Quinlan, Induction of decision trees, *Machine Learning* 1 (1) (1986) 81–
675 106.
- [29] W. Winkler, Using the EM algorithm for weight computation in the fellegi-sunter model of record linkage, Tech. Rep. RR2000/05, Statistical Research Division, Methodology and Standards Directorate, U.S. Bureau of the Census (2000).
- [30] A. Karakasidis, V. Verykios, Secure blocking + secure matching = secure
680 record linkage, *JCSE* 5 (3) (2011) 223–235.

- [31] A. Kirsch, M. Mitzenmacher, Less hashing, same performance: Building a better bloom filter, *Random Struct. Algorithms* 33 (2) (2008) 187–218.
- [32] M. Connor, P. Kumar, Fast construction of k-nearest neighbor graphs for point clouds, *IEEE Trans. Vis. Comput. Graphics* 16 (4) (2010) 599–608.
- [33] U. von Luxburg, A tutorial on spectral clustering, *Statistics and Computing* 17 (4) (2007) 395–416.
- [34] P. Vaidya, An $o(n \log n)$ algorithm for the all-nearest-neighbors problem, *Discrete & Computational Geometry* 4 (1989) 101–115.
- [35] J. Chen, H. Fang, Y. Saad, Fast approximate k nn graph construction for high dimensional data via recursive lanczos bisection, *Journal of Machine Learning Research* 10 (2009) 1989–2012.
- [36] J. Wang, J. Wang, G. Zeng, Z. Tu, R. Gan, S. Li, Scalable k-nn graph construction for visual descriptors, in: *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 1106–1113.
- [37] W. Dong, M. Charikar, K. Li, Efficient k-nearest neighbor graph construction for generic similarity measures, in: *Proceedings of the 20th International Conference on World Wide Web, WWW 2011*, 2011, pp. 577–586.
- [38] A. Gionis, P. Indyk, R. Motwani, Similarity search in high dimensions via hashing, in: *VLDB’99, Proceedings of 25th International Conference on Very Large Data Bases*, 1999, pp. 518–529.
- [39] R. Steorts, S. Ventura, M. Sadinle, S. Fienberg, A comparison of blocking methods for record linkage, in: *Privacy in Statistical Databases - UNESCO Chair in Data Privacy, International Conference, PSD 2014, Ibiza, Spain, September 17-19, 2014. Proceedings*, 2014, pp. 253–268.
- [40] E. Durham, Y. Xue, M. Kantarcioglu, B. Malin, Quantifying the correctness, computational complexity, and security of privacy-preserving string comparators for record linkage, *Inf Fusion* 13 (4) (2012) 245–259.

- [41] F. Borst, F. Allaert, C. Quantin, Empirical aspects of record linkage across
710 multiple data sets using statistical linkage keys: the experience of the piac
cohort study, *Stud Health Technol Inform* 84 (Pt 2) (2001) 1239–1241.
- [42] M. Kuzu, M. Kantarcioglu, E. Durham, B. Malin, A constraint satisfac-
tion cryptanalysis of bloom filters in private record linkage, in: *Privacy
Enhancing Technologies*, 2011, pp. 226–245.
- 715 [43] P. Jurczyk, J. Lu, L. Xiong, J. Cragan, A. Correa, FRIL: A tool for com-
parative record linkage, in: *AMIA 2008, American Medical Informatics
Association Annual Symposium*, 2008.
- [44] P. Christen, D. Vatsalan, Flexible and extensible generation and corruption
of personal data, in: *22nd ACM International Conference on Information
720 and Knowledge Management, CIKM'13*, 2013, pp. 1165–1168.
- [45] K. Tran, D. Vatsalan, P. Christen, Geco: an online personal data generator
and corruptor, in: *22nd ACM International Conference on Information and
Knowledge Management, CIKM'13*, 2013, pp. 2473–2476.

Table 2: Example of using the Bloom filter protocol to encrypt strings for computing the similarity measure between strings in a privacy preserving manner.

| $h_p(\cdot)$ | <i>SMITH</i> | | <i>SMYTH</i> | | <i>SMY</i> | | <i>calculated</i> |
|--------------|--------------|-------------------|--------------|-------------------|------------|-------------------|-------------------|
| mod l | <i>bit</i> | <i>bigram (p)</i> | <i>bit</i> | <i>bigram (p)</i> | <i>bit</i> | <i>bigram (p)</i> | <i>bit</i> |
| 6 | 1 | <i>H_ (3)</i> | 1 | <i>H_ (3)</i> | 0 | | 0.65 |
| 11 | 0 | | 0 | | 1 | <i>Y_ (2)</i> | 0.35 |
| 12 | 1 | <i>SM (3)</i> | 1 | <i>SM (3)</i> | 1 | <i>SM (3)</i> | 1 |
| 13 | 1 | <i>MI (2)</i> | 1 | <i>MY (1)</i> | 1 | <i>MY (1)</i> | 1 |
| 17 | 1 | <i>IT (1)</i> | 0 | | 0 | | 0 |
| 21 | 1 | <i>_S (2)</i> | 1 | <i>_S (2)</i> | 1 | <i>_S (2)</i> | 1 |
| 32 | 0 | | 1 | <i>YT (2)</i> | 0 | | 0.65 |
| 39 | 0 | | 0 | | 1 | <i>Y_ (3)</i> | 0.35 |
| 48 | 1 | <i>H_ (1)</i> | 1 | <i>MY (2)</i> | 1 | <i>MY (2)</i> | 1 |
| 56 | 1 | <i>TH (1)</i> | 1 | <i>H_ (1)</i> | 0 | | 0.65 |
| 58 | 1 | <i>MI (1)</i> | 1 | <i>TH (2)</i> | 0 | | 0.65 |
| 60 | 1 | <i>TH (2)</i> | 1 | <i>TH (3)</i> | 0 | | 0.65 |
| 61 | 1 | <i>IT (3)</i> | 0 | | 0 | | 0 |
| 67 | 1 | <i>_S (3)</i> | 1 | <i>_S (3)</i> | 1 | <i>_S (3)</i> | 1 |
| 68 | 1 | <i>MI (3)</i> | 0 | | 0 | | 0 |
| 75 | 1 | <i>_S (1)</i> | 1 | <i>_S (1)</i> | 1 | <i>_S (1)</i> | 1 |
| 76 | 1 | <i>SM (1)</i> | 1 | <i>SM (1)</i> | 1 | <i>SM (1)</i> | 1 |
| 77 | 1 | <i>H_ (2)</i> | 1 | <i>H_ (2)</i> | 0 | | 0.65 |
| 81 | 0 | | 1 | <i>YT (3)</i> | 0 | | 0.65 |
| 83 | 0 | | 1 | <i>MY (3)</i> | 1 | <i>MY (3)</i> | 1 |
| | | | | <i>YT (1)</i> | | <i>Y_ (1)</i> | |
| 89 | 1 | <i>IT (2)</i> | 0 | | 0 | | 0 |
| 94 | 1 | <i>SM (2)</i> | 1 | <i>SM (2)</i> | 1 | <i>SM (2)</i> | 1 |

Table 3: Characteristics of the synthetic datasets R and S

| | |
|-----------------|---|
| Size: | $ R = 10^5$, $ S = 1.05 \times 10^5$ records |
| Columns: | <i>GivenName (GN)</i> , <i>Surname (S)</i> , <i>Postcode (P)</i> , <i>Telephone (T)</i> , <i>Gender (G)</i> , <i>City (C)</i> |
| Similarity: | $\mathbb{DC}(GN)$, $\mathbb{DC}(S)$, $\mathbb{DC}(P)$, $\mathbb{DC}(P, C)$ $\mathbb{DC}(T)$, $\mathbb{SE}(G)$, $\mathbb{DC}(C)$ |
| Blocking: | $\mathbb{DC}(\text{1st of } N, S) \geq 0.7$, $\mathbb{DC}(T) \geq 0.85$ |
| Intra-Blocking: | $\mathbb{DC}(S) \geq 0.85$, $\mathbb{DC}(P, C) \geq 0.85$, $\mathbb{DC}(T) \geq 0.85$ |

Table 4: Comparison of runtime at different computational stages (in hours).

| Method | Computation Stage | | | Total |
|----------|-------------------|----------|---------|--------|
| | Intra-Blocking | Blocking | Linkage | |
| PRL | – | 14.468 | 1.073 | 15.541 |
| BF | – | 4.712 | 1.671 | 6.383 |
| FRIL-0 | – | 13.523 | 0.946 | 14.469 |
| FRIL-100 | – | 13.248 | 1.128 | 14.376 |
| FLE | – | 14.002 | 1.045 | 15.047 |
| Proposed | 1.526 | 4.902 | 1.913 | 8.341 |