Sorted Nearest Neighborhood Clustering for Efficient Private Blocking

Dinusha Vatsalan and Peter Christen

Research School of Computer Science, College of Engineering and Computer Science
The Australian National University, Canberra ACT 0200, Australia
{dinusha.vatsalan,peter.christen}@anu.edu.au

Abstract. Record linkage is an emerging research area which is required by various real-world applications to identify which records in different data sources refer to the same real-world entities. Often privacy concerns and restrictions prevent the use of traditional record linkage applications across different organizations. Linking records in situations where no private or confidential information can be revealed is known as privacy-preserving record linkage (PPRL). As with traditional record linkage applications, scalability is a main challenge in PPRL. This challenge is generally addressed by employing a blocking technique that aims to reduce the number of candidate record pairs by removing record pairs that likely refer to non-matches without comparing them in detail. This paper presents an efficient private blocking technique based on a sorted neighborhood approach that combines k-anonymous clustering and the use of public reference values. An empirical study conducted on real-world databases shows that this approach is scalable to large databases, and that it can provide effective blocking while preserving k-anonymous characteristics. The proposed approach can be up-to two orders of magnitude faster than two state-of-the-art private blocking techniques, k-nearest neighbor clustering and Hamming based locality sensitive hashing.

Keywords: sorted neighborhood, nearest neighbor clustering, locality sensitive hashing, k-anonymity, reference values, scalability.

1 Introduction

Integrating large volumes of data from different sources is an important data pre-processing step in many data mining applications [1]. Since unique entity identifiers are not always available in all the databases to be linked, common identifying attributes, such as names and addresses, are often used to identify records that need to be reconciled to the same real-world entities. The degraded quality of data residing in databases (due to transcription errors, missing values, or inconsistent formats), makes the task of integrating databases challenging [2]. Integrating such data in the presence of data quality errors requires approximate comparison functions to be employed to identify if two entities are the same [1]. These comparison functions are often expensive in terms of computational complexity. A naive pair-wise comparison of two databases is of quadratic

J. Pei et al. (Eds.): PAKDD 2013, Part II, LNAI 7819, pp. 341–352, 2013.

complexity in their sizes. This makes the record linkage process not scalable to large databases. The scalability problem has been studied by introducing two-step linkage algorithms that avoid all pair-wise comparisons by employing a blocking or indexing technique [3] in the first step, so that detailed comparisons using expensive similarity comparison functions are only made in the second step on a smaller number of candidate record pairs.

Linking data across databases from different organizations is more challenging due to privacy and confidentiality issues that often preclude exchanging sensitive information regarding the entities. Identifying which records in two databases have the same or approximately the same values for a set of attributes without revealing the actual values of these attributes is known as the problem of 'privacy-preserving record linkage' (PPRL) [4–6]. Blocking for PPRL needs to be conducted in such a way that no sensitive information that can be used to infer individual records and their attribute values is revealed to any party involved in the process, or to an external adversary. The scalability challenge of PPRL has been addressed by several recent approaches that adapt existing blocking techniques, such as standard blocking [7], mapping based blocking [8], clustering [9], and locality sensitive hashing [10], into a privacy-preserving context.

One popular blocking technique used in traditional record linkage is the sorted neighborhood approach [11, 12], where database tables are sorted according to a 'sorting key' over which a sliding window of fixed size is moved. Candidate record pairs are then generated from the records that are within the current window. This approach is very efficient compared to other blocking techniques in that its resulting number of candidate record pairs is $O((n_A + n_B)w)$, compared to $O((n_A \cdot n_B)/b)$ for other blocking techniques [3], where n_A and n_B are the number of records in the two databases to be linked, b is the number of blocks generated, and w is the size of the window. However, the use of sorted neighborhood methods for private blocking has so far not been studied.

We propose an efficient three-party blocking technique for PPRL based on the sorted neighborhood approach using a combination of two privacy techniques: k-anonymous clustering [13] and public reference values [14]. The aim of this approach is to efficiently create k-anonymous clusters represented by reference values from which candidate record pairs are generated, without revealing any information that can be used to infer individual records and their values.

The contributions of this paper are (1) an efficient blocking technique for PPRL based on the sorted neighborhood approach; (2) two variations to generate k-anonymous clusters; (3) an analysis of our solution regarding complexity, privacy, and quality; and (4) an empirical evaluation using real-world datasets. We compare our approach with two state-of-the-art three-party private blocking techniques, which are Karakasidis et al.'s [15] approach based on k-nearest neighbor clustering, and Durham's [16] approach based on Hamming based locality sensitive hashing.

The remainder of this paper is structured as follows. In the following section, we provide an overview of related work in private blocking. In Sect. 3 we describe our protocol using two small example datasets. In Sect. 4 we analyse the protocol

and in Sect. 5 we validate these analyses through an empirical study. Finally we summarize our findings and provide directions for future research in Sect. 6.

2 Related Work

The use of a blocking technique is crucial in PPRL applications to make the linkage across large databases scalable [3]. Several approaches have been proposed for private blocking. Most work in PPRL that has investigated scalability has employed the basic standard blocking approach [17–20]. In traditional standard blocking, all records that have the same blocking key value (the value of a single attribute or a combination of attributes) are inserted into the same block, and only the records within the same block are compared in detail with each other in the comparison step. Each record is inserted into one block only [3]. Al-Lawati et al. [17] explored three methods of token blocking to group hash signatures of the TF-IDF distances of attribute values. Karakasidis et al. [20] proposed phonetic based private blocking where an encoding function, such as Soundex or NYSIIS [1], is used to group records that have similar (sounding) values into the same block. Generalization techniques, such as value generalization hierarchies [18], k-anonymity [13] and binning [21], have been used for private blocking to generalize records with similar characteristics into the same blocks.

Mapping based blocking [8] is another technique that has been employed in PPRL. Scannapieco et al. [22] and Yakout et al. [23] used a multi-dimensional embedded space into which attribute values are mapped while preserving the distances between these values. A clustering or nearest neighbor approach is then applied on these multi-dimensional objects to extract candidate record pairs.

Karakasidis et al. [15] used the k-nearest neighbor clustering algorithm to group records such that similar records are put into the same clusters, and each cluster consists of at least k elements to provide a k-anonymous privacy guarantee. Initially clusters are generated for a set of reference values that are shared by the database owners, such that each cluster consists of at least k reference values. Each database owner assigns their records into these clusters based on their similarity. These clusters are sent to a third party that merges the corresponding clusters to generate candidate record pairs.

Locality sensitive hashing (LSH) has recently been investigated as an efficient technique for scalable record linkage [10, 16]. LSH allows hashing of values in such ways that the likelihood that two similar values are hashed into the same block can be specified through the use of certain hashing functions. Durham [16] investigated how LSH can be applied in Bloom filter based PPRL to reduce the number of record pair comparisons. A Bloom filter is a bit array data structure where hash functions are used to map a set of elements (q-grams extracted from attribute values) into the bit array. For private blocking, an iterative pruning approach is employed, where random bits are sampled at each iteration from the Bloom filters and sent to a third party. The third party then uses Hamming based LSH functions to compute the Hamming distance that allows efficient generation of candidate record pairs.

D^A

RecID	Surname	GivenName	SKV
RA1	millar	robert	millarrobert
RA2	marten	amyas	martenamyas
RA3	melar	gail	melargail
RA4	miller	robart	millerrobart
RA5	morley	william	morleywilliam
RA6	philips	colin	philipscolin
RA7	smith	alisen	smithalisen
RA8	sampson	taylor	sampsontaylor

Reference values

R'

1	*millar*
2	*myler*
3	*robinson*
4	*smith*

D^B

RecID	Surname	GivenName	SKV
RB1	millar	robert	millarrobert
RB2	marris	roberto	marrisroberto
RB3	malar	gayle	malargayle
RB4	perris	charles	perrischarles
RB5	robbins	william	robbinswilliam
RB6	robertson	amy	robertsonamy
RB7	samuell	tailor	samuelltailor
RB8	smeeth	rupert	smeethrupert
RB9	smith	alison	smithalison

Fig. 1. Example databases held by Alice ($\mathbf{D^A}$) and Bob ($\mathbf{D^B}$) with surname and given name attributes and their sorting key values (SKVs), and a list of reference values ($\mathbf{R'}$) along with their position values, used to illustrate the protocol described in Sect. 3

3 Sorted Neighborhood Based Private Blocking

In this section we describe the steps of our sorted neighborhood clustering (SNC) based private blocking method, and illustrate it with an example consisting of two small databases with given names and surnames, as shown in Fig. 1. Assume *Alice* and *Bob* are the two owners of their respective databases $\mathbf{D^A}$ and $\mathbf{D^B}$, and *Carol* is the trusted third party. Alice and Bob share the sorted reference list $\mathbf{R'}$ containing n_R reference values selected from the publicly available reference dataset $\mathbf{R}$. Fig. 2 illustrates the three-party setting for private blocking.

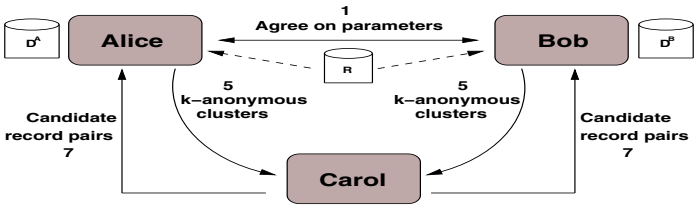

Fig. 2. Three-party private blocking. Numbers given correspond to the steps described in Sect. 3 that involve an exchange of data between parties

The two database owners Alice and Bob perform the following steps:

1. Agree upon the list of attributes to be used as the sorting keys, the reference dataset $\mathbf{R}$, the number of reference values to be used n_R, the minimum number of elements in a cluster k, a similarity function $sim(\cdot, \cdot)$ to compare reference values, and the minimum similarity threshold s_t used to decide if two clusters are to be merged in the SNC-Sim approach described in Step 4.
2. Alice and Bob each selects and sorts n_R ($n_R \leq |\mathbf{R}|$) reference values. The value for n_R can be chosen as $n_R = \min(|\mathbf{D^A}|, |\mathbf{D^B}|)/k$, so that each cluster will contain roughly around k database records. It is important to note that both Alice and Bob have the same list of sorted reference values $\mathbf{R'}$ at the end of this step. A secret random seed shared by Alice and Bob (but not known to Carol) can be used to select the same set of values from $\mathbf{R}$ into $\mathbf{R'}$ by both Alice and Bob.

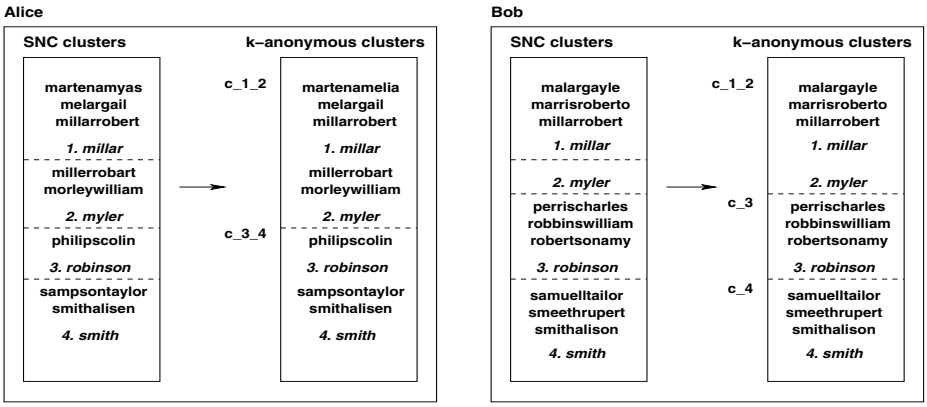

Fig. 3. Insertion of SKVs into the sorted list of reference values where each cluster is represented by one reference value (shown in italics font), and merging of clusters to create k-anonymous clusters (clusters that contain at least k SKVs) where each cluster is represented by one or more reference values (in this example, $k = 3$).

3. Alice and Bob individually insert their records based on the records' sorting key values (SKVs) into the sorted list of reference values to create SNC clusters, as shown in Fig. 3. An inverted index data structure can be used to efficiently insert records where the keys are the reference values and the corresponding values contain a list of SKVs which are lexicographically sorted before the reference value.
4. The next step is to create k-anonymous clusters. After inserting records into the sorted reference values there will be n_R clusters each represented by one reference value. However, to provide a k-anonymous privacy guarantee, each database owner has to merge their clusters in such a way that each cluster contains at least k database records. Cluster IDs are assigned to the merged clusters such that they consist of the position values of the reference values that reside in the merged clusters. We use cluster IDs of the form '$c\_x\_(x+1)\cdots\_(x+y)$', where x is the position value of the first reference value of the sorted reference values in the corresponding cluster, and $(y+1)$ is the number of reference values in that cluster. The merging of clusters to create k-anonymous clusters is shown in Fig. 3. This merging process can be done in two different ways.
   (a) SNC-Sim: Clusters are merged until the number of elements in them becomes greater than or equal to k and the similarity between reference values of adjacent clusters becomes less than the threshold s_t. This approach follows recent work on adaptive sorted neighborhood for duplicate detection [12]. Algo. 1 shows the main steps involved in this method. The clusters are merged until their size becomes greater than or equal to k (line 5 in Algo. 1). If the size of the (merged) cluster is greater than k, we compute the similarity of the next cluster's reference value r_{i+j+1} with the current cluster's reference value r_{i+j}, and if this similarity value $sim(r_{i+j+1}, r_{i+j})$

Algorithm 1. Merging Clusters using Sim

Input:
- $\mathbf{R}'$: List of sorted reference values $(r_1, \ldots, r_{n_R})$
- $\mathbf{S}$: Set of clusters $(c : [v_1, \ldots, v_l])$
- Minimum number of elements in a cluster k
- Minimum similarity threshold s_t
- Similarity comparison function $sim(\cdot, \cdot)$

Output:
- $\mathbf{O}$: Set of k-anonymous clusters $(\text{`}c\_x \ldots \_(x+y)\text{'} : [v_1, \ldots, v_k])$

```
1:  i = 0
2:  while i < n_R do
3:    clus_vals = []; clus_id = 'c_'
4:    num_vals = 0; sim_val = 0.0; j = 0
5:    while (num_vals ≤ k and i + j < n_R)
      or (sim_val ≥ s_t and i + j < n_R) do
6:      r_i = R'[i + j]; b = S[r_{i+j}]
7:      num_vals += len(b)
8:      clus_vals += b
9:      sim_val = sim(r_{i+j}, r_{i+j+1})
10:     clus_id += str(i+j)+'_'
11:     j += 1
12:   O[clus_id] = clus_vals
13:   i += j
```

Algorithm 2. Merging Clusters using Size

Input:
- $\mathbf{S}$: Set of clusters $(c : [v_1, \ldots, v_l])$
- Minimum number of elements in a cluster k

Output:
- $\mathbf{O}$: Set of k-anonymous clusters $(\text{`}c\_x \ldots \_(x+y)\text{'} : [v_1, \ldots, v_k])$

```
1:    ids = []; sizes = []
2:    for (r, b) ∈ S do
3:      ids += [r]; sizes += [len(b)]
4:    min_size = min(sizes)
5:    while min_size < k and len(ids) > 1 do
6:      min_size_clus=S.getID(len(b)=min_size)
7:      clus_vals=S[min_size_clus]
8:      i = ids.getindex(min_size_clus)
9:      prev_clus = ids[i-1]; next_clus = ids[i+1]
10:     if len(S[prev_clus]) < len(S[next_clus])
11:       clus_id = min_size_clus + prev_clus
12:       clus_vals += S[prev_clus]
13:     else
14:       clus_id = min_size_clus + next_clus
15:       clus_vals += S[next_clus]
16:     update(sizes); min_size = min(sizes)
17:     O[clus_id] = clus_vals
```

is greater than or equal to s_t, then we continue to merge the next cluster with the current cluster. Lines 5-11 show this loop of merging. In line 12, the values in the merged cluster are stored in the output set of k-anonymous clusters with its cluster ID. This method inserts similar values into one cluster, at the cost that the resulting larger clusters will generate more candidate record pairs.

(b) SNC-Size: Clusters are merged until the minimum size of the clusters becomes greater than or equal to k by iteratively identifying the smallest cluster. Algo. 2 provides an overview of this method. In lines 2-4, we find the cluster with the smallest number of elements, and if this number is less than k (line 5) we merge it with the smaller of its two neighboring clusters. We repeat this merging (lines 5-17) until the minimum cluster size is at least k. The values in the merged cluster are stored in the output set of k-anonymous clusters in line 17 along with its cluster ID. Compared to SNC-Sim, this method results in a smaller number of records in most clusters. However, true matches might be missed depending on the value for k, because the similarity between values is not considered.

5. Once the k-anonymous clusters are created, they need to be sent to a third party, Carol, to generate candidate record pairs. The values in the clusters are replaced by their (encrypted) record IDs.

Carol receives the k-anonymous clusters from Alice and Bob and performs the following steps:

6. Find corresponding clusters from Alice and Bob based on the reference position values in the cluster IDs to generate candidate record pairs as is illustrated in Fig. 4.

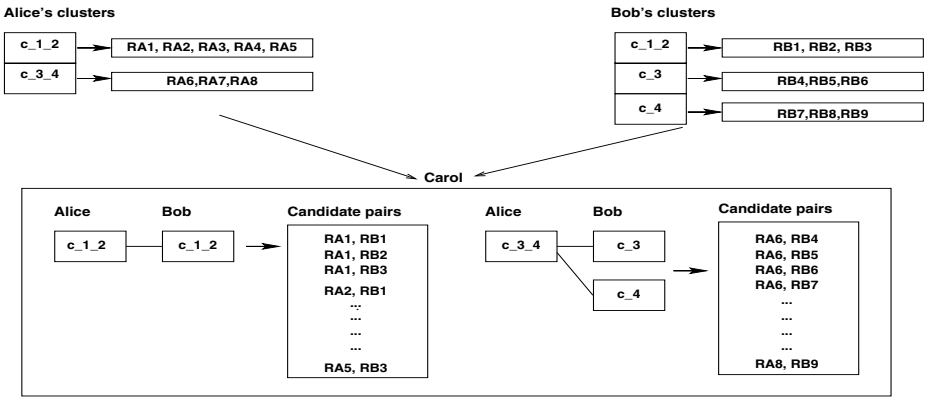

Fig. 4. The merging of corresponding clusters from Alice and Bob to generate candidate record pairs (made of record IDs) as conducted by Carol in Step 6 of the protocol

Algorithm 3. Generating Candidate Record Pairs

Input:
- $\mathbf{O^A}$: Alice's set of k-anonymous clusters
- $\mathbf{O^B}$: Bob's set of k-anonymous clusters

Output:
- $\mathbf{C}$: Set of candidate record pairs

```
1:  for (i^A, b^A) ∈ O^A do
2:    ref_pos_vals_alice = get_ref_pos_vals(i^A)
3:    bob_clusters = []; b^B = []
4:    for ref_pos ∈ ref_pos_vals_alice do
5:      bob_clusters += O^B.getIDs(ref_pos ∈ ID)
6:      b^B += O^B[bob_clusters]
7:    for alice_rec_ID ∈ b^A do
8:      for bob_rec_ID ∈ b^B do
9:        C += [alice_rec_ID,bob_rec_ID]
```

This process is explained in Algo. 3. From the cluster IDs, Carol extracts the position values of the reference values that reside in the corresponding clusters (line 2). In lines 3-6, Carol finds for each of Alice's clusters all of Bob's clusters that need to be merged by extracting the position values from Alice's cluster IDs. Carol then performs a nested loop (lines 7-9) over Alice's clusters and Bob's corresponding clusters, and stores the record pairs from Alice's and Bob's records in the output set of candidate record pairs.

7. Carol sends the record IDs of the candidate pairs $\mathbf{C}$ back to Alice and Bob which then employ a PPRL protocol on each block individually [14, 21, 24].

4 Analysis of the Protocol

In this section we analyse our SNC private blocking approach in terms of complexity, privacy, and quality.

1. **Complexity:** Assuming both databases contain n records ($n = n_A = n_B$) and n_R reference values are selected from the reference dataset, sorting

these n_R reference values is of $O(n_R\ log\ n_R)$ complexity, and inserting the n database records into the sorted list of reference values is of $O(n)$ complexity. At the end of the insertion of records into the sorted list, there will be n_R clusters each represented by one reference value. Merging clusters to create k-anonymous clusters requires a loop over n_R clusters, which is of $O(n_R)$ complexity, and results in less than or equal to n_R merged clusters.

Sending these clusters to Carol is of $O(n)$ communication complexity. Carol performs a loop over the clusters (a maximum of n_R clusters) to merge and generate candidate record pairs from Alice's and Bob's records. The computation complexity of this step is $O(n_R^2)$.

The overall complexity of our approach is linear in the size of the databases n and quadratic in the number of reference values n_R. The number of resulting clusters will be on average n/k. Assuming each cluster contains k records, the number of candidate record pairs generated by our approach is $\frac{n}{k} \times k^2 = n\ k$.

2. **Privacy:** We assume that all parties that participate in the protocol follow the 'honest but curious' (HBC) adversary model [4, 5], in that they try to find out as much as possible about the data from other parties while following the protocol. Since each cluster consists of at least k elements, it is difficult for Carol to perform an attack to infer individual records. The value for k has to be chosen carefully. A higher value for k provides stronger privacy guarantees but more candidate record pairs will be generated. The variance between the cluster sizes generated with our approach is very low compared to other private blocking techniques. Therefore, Carol cannot learn the frequency distribution of the clusters generated to conduct a frequency attack. We present the cluster sizes generated on some real databases in Sect. 5.

   Merging the blocks to create k-anonymous clusters makes the protocol more secure and harder for a frequency attack. Further, Carol does not know how the reference values $\mathbf{R}'$ were selected from the reference dataset $\mathbf{R}$ ($\mathbf{R}' \in \mathbf{R}$), and therefore she cannot learn the clustering details. However, as with other three-party protocols, collusion between the third party and one of the database owners with the aim to identify the other database owner's data, is a privacy risk in this approach as well [4, 5].

3. **Quality:** A good blocking technique should have two properties [3]: (1) effectiveness - all similar records should be grouped into the same cluster, and (2) efficiency - the number candidate record pairs should be as small as possible while including all true matching record pairs. SNC-Sim retrieves more similar records compared to SNC-Size as similarity is used in SNC-Sim to determine the maximum size of a cluster. However, SNC-Sim is more likely to group more records into one cluster. This results in higher effectiveness and lower efficiency for the SNC-Sim method comparatively.

   The value of k also determines the effectiveness and efficiency of blocking. A higher value for k results in a more effective but less efficient blocking. An optimal value for k needs to be set such that high values for both effectiveness and efficiency are achieved while k guarantees sufficient privacy as well.

Table 1. The number of records in the datasets used for experiments, and the number of records that occur in both datasets of a pair (i.e. the number of true matches)

Dataset sizes	25% overlap	50% overlap	75% overlap
1730 / 1730	446	897	1310
17,294 / 17,294	4365	8611	12,973
172,938 / 172,938	42,980	86,363	129,542
1,729,379 / 1,729,379	432,538	864,487	1,297,029

5 Experimental Evaluation

We conducted experiments using a real Australian telephone database containing 6,917,514 records. We extracted four attributes commonly used for record linkage: Given name (with 78,336 unique values), Surname (with 404,651 unique values), Suburb (town) name (13,109 unique values), and Postcode (2,632 unique values). To generate datasets of different sizes, we sampled 0.1%, 1%, 10% and 100% of records in the full database twice each, and stored them into pairs of files such that 25%, 50% or 75% of records appeared in both files of a pair. Table 1 provides an overview of the twelve pairs of datasets we generated.

The record pairs that occur in both datasets are exact matches. To investigate the performance of our protocol in the context of 'dirty data' (where attribute values contain errors and variations), we generated another series of datasets where we modified each attribute value by applying one randomly selected character edit operation (insert, delete, substitute, or transposition) [25]. As it turns out, there is no difference in the performance of our protocol with modified and not modified datasets, and we therefore only report averaged results.

We prototyped the protocol using the Python programming language (version 2.7.3). We also implemented prototypes of Karakasidis et al. [15]'s k-anonymous nearest neighbor clustering and Durham [16]'s Hamming based locality sensitive hashing for comparative analysis and evaluation. We name these two techniques as k-NN and HLSH in the results, respectively. We used parameter settings for the k-NN and HLSH methods in a similar range as used by the authors of these two state-of-the-art private blocking methods. For k-NN, k is set to 3 and the similarity threshold is set as $s_t = 0.6$. In the HLSH method, the number of iterations is set to $\mu = 20$, the number of hash functions is 30, and the number of bits to be sampled from the Bloom filters at each iteration is $\phi = 24$. The parameters for the SNC approaches were set as $k = 100$ and $s_t = 0.9$. All tests were run on a compute server with 64 bit Intel Xeon (2.4 GHz) CPUs, 128 GBytes of main memory and running Ubuntu 11.04. The prototype and test datasets are available from the authors.

Fig. 5 shows the total time required for private blocking of the four approaches. As can be seen from the figure, the SNC approach (both variations of SNC-Sim and SNC-Size) requires nearly two magnitudes less time by the database owners than the other two approaches (i.e. around 100 times faster) and is also linear in the size of the databases. We were unable to conduct experiments for the HLSH and k-NN approaches on the 1,729,379 datasets due to their memory requirements. The k-NN approach requires less time for the third party than

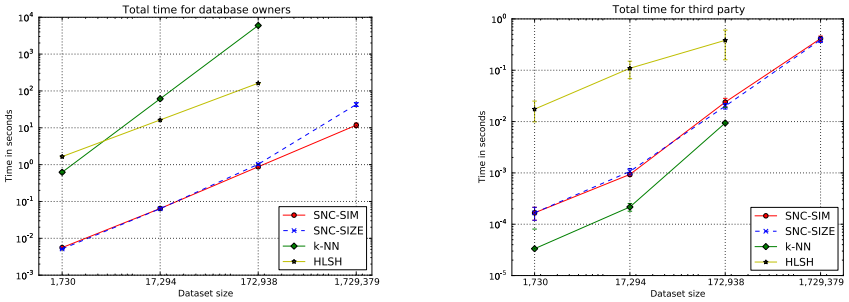

Fig. 5. Total time of the four blocking approaches required by database owners (left) and third party (right) averaged over the results of all variations of each dataset

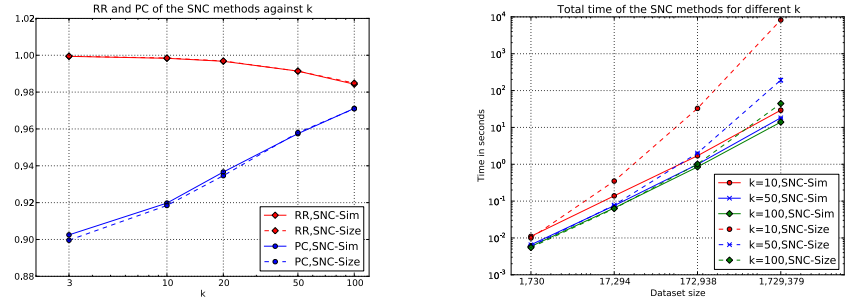

Fig. 6. RR and PC values for the k-anonymous SNC blocking approaches with the 17,294 datasets (left plot) and total time of the SNC approaches for different k against the dataset size (right plot) averaged over the results of all variations of each dataset

the SNC approaches, because a smaller number of candidate record pairs are generated with the k-NN approach at the cost of much reduced number of true matched record pairs (as is illustrated in the right plot in Fig. 7).

Reduction Ratio (RR) and Pairs Completeness (PC) can be used to assess the efficiency and effectiveness of blocking, respectively [3]. RR is the fraction of record pairs that are removed by a blocking technique and PC is the fraction of true matching record pairs that are included in the candidate record pairs generated by a blocking technique. The SNC approaches achieve high values for both PC and RR even when $k = 100$, which gives a strong privacy guarantee (Fig. 6 left plot). As discussed in Sect. 4, the effectiveness of blocking increases with k while efficiency decreases. As expected, the SNC-Sim approach provides a relatively higher PC than the SNC-Size. The right plot in Fig. 6 shows the scalability of our approach with different values for k.

The size of the clusters generated by the four approaches and the resulting PC and RR values are presented in Fig. 7. The SNC approaches have lower variances between the cluster sizes which makes a frequency attack by Carol harder. As illustrated in the left plot, the SNC approaches generate nearly uniform distributions of clusters. The k-NN approach has a higher RR but lower PC, while

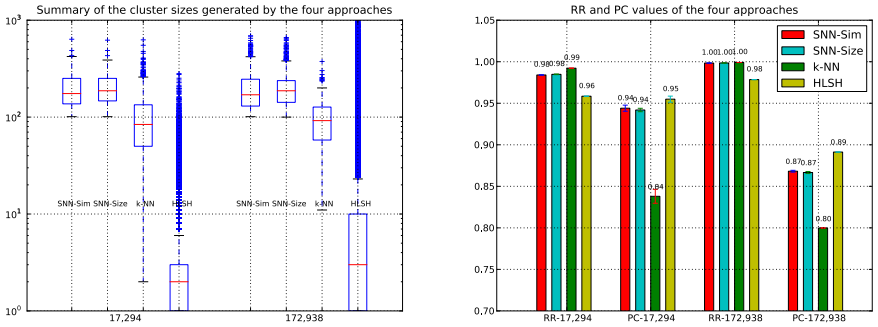

Fig. 7. Sizes of the clusters generated by the four approaches - with the bottom and top of the boxes representing the lower and upper quartiles, the band near the middle of the boxes the median, and the two ends of the whiskers the standard deviation above and below the mean of the distribution (left plot); and RR and PC values of the four approaches (right plot) using the 17,294 and 172,938 datasets

a lower RR and higher PC are achieved with the HLSH approach. The SNC approach performs superior compared to the other two approaches by achieving higher results for both RR and PC.

6 Conclusion

In this paper, we proposed an efficient private blocking technique that can be used to make privacy-preserving record linkage applications scalable to large databases. Our method is based on the sorted nearest neighborhood clustering approach, and uses a combination of the privacy techniques reference values and k-anonymous clustering. Experiments conducted on real-world large databases, each containing nearly 2 million records, validate that our approach is scalable and effective in generating candidate record pairs while preserving k-anonymity privacy characteristics. Our approach also outperforms two existing state-of-the-art private blocking techniques in terms of speed, efficiency, and effectiveness.

As discussed earlier, three-party solutions are often susceptible to collusion between parties. As future work, we aim to study how the sorted neighborhood clustering can be used for private blocking in a two-party context. Theoretically analyzing and modelling the privacy, complexity, and quality of our solution is another direction for future research.

References

1. Christen, P.: Data Matching. Data-Centric Systems and Appl. Springer (2012)
2. Batini, C., Scannapieca, M.: Data quality: Concepts, methodologies and techniques. In: Data-Centric Systems and Appl. Springer (2006)

3. Christen, P.: A survey of indexing techniques for scalable record linkage and deduplication. IEEE Transactions on Knowledge and Data Engineering 12(9) (2012)
4. Vatsalan, D., Christen, P., Verykios, V.: A taxonomy of privacy-preserving record linkage techniques. Information Systems (2013)
5. Hall, R., Fienberg, S.: Privacy-preserving record linkage. In: Domingo-Ferrer, J., Magkos, E. (eds.) PSD 2010. LNCS, vol. 6344, pp. 269–283. Springer, Heidelberg (2010)
6. Churches, T., Christen, P.: Blind data linkage using n-gram similarity comparisons. In: Dai, H., Srikant, R., Zhang, C. (eds.) PAKDD 2004. LNCS (LNAI), vol. 3056, pp. 121–126. Springer, Heidelberg (2004)
7. Fellegi, I.P., Sunter, A.B.: A theory for record linkage. Journal of the American Statistical Society 64(328), 1183–1210 (1969)
8. Jin, L., Li, C., Mehrotra, S.: Efficient record linkage in large data sets. In: DASFAA 2003, pp. 137–146 (2003)
9. Cohen, W.W., Richman, J.: Learning to match and cluster large high-dimensional data sets for data integration. In: ACM SIGKDD, pp. 475–480 (2002)
10. Kim, H., Lee, D.: Harra: fast iterative hashed record linkage for large-scale data collections. In: EDBT, Lausanne, Switzerland, pp. 525–536 (2010)
11. Hernandez, M.A., Stolfo, S.J.: Real-world data is dirty: Data cleansing and the merge/purge problem. Data Mining and Knowledge Discovery 2(1), 9–37 (1998)
12. Draisbach, U., Naumann, F., Szott, S., Wonneberg, O.: Adaptive windows for duplicate detection. In: ICDE, pp. 1073–1083 (2012)
13. Sweeney, L.: K-anonymity: A model for protecting privacy. International Journal of Uncertainty Fuzziness and Knowledge Based Systems 10(5), 557–570 (2002)
14. Pang, C., Gu, L., Hansen, D., Maeder, A.: Privacy-preserving fuzzy matching using a public reference table. In: McClean, S., Millard, P., El-Darzi, E., Nugent, C. (eds.) Intelligent Patient Management. Studies in Computational Intelligence, vol. 189, pp. 71–89. Springer, Heidelberg (2009)
15. Karakasidis, A., Verykios, V.: Reference table based k-anonymous private blocking. In: ACM Symposium on Applied Computing, Riva del Garda, Italy (2012)
16. Durham, E.: A framework for accurate, efficient private record linkage. PhD thesis, Vanderbilt University (2012)
17. Al-Lawati, A., Lee, D., McDaniel, P.: Blocking-aware private record linkage. In: IQIS, pp. 59–68 (2005)
18. Inan, A., Kantarcioglu, M., Bertino, E., Scannapieco, M.: A hybrid approach to private record linkage. In: IEEE ICDE, Cancun, Mexico, pp. 496–505 (2008)
19. Inan, A., Kantarcioglu, M., Ghinita, G., Bertino, E.: Private record matching using differential privacy. In: EDBT (2010)
20. Karakasidis, A., Verykios, V., Christen, P.: Fake injection strategies for private phonetic matching. In: DPM, Leuven, Belgium (2011)
21. Vatsalan, D., Christen, P., Verykios, V.: An efficient two-party protocol for approximate matching in private record linkage. In: AusDM, CRPIT 121 (2011)
22. Scannapieco, M., Figotin, I., Bertino, E., Elmagarmid, A.: Privacy preserving schema and data matching. In: ACM SIGMOD, pp. 653–664 (2007)
23. Yakout, M., Atallah, M., Elmagarmid, A.: Efficient private record linkage. In: IEEE ICDE, Shanghai, pp. 1283–1286 (2009)
24. Schnell, R., Bachteler, T., Reiher, J.: Privacy-preserving record linkage using Bloom filters. BMC Medical Informatics and Decision Making 9(1) (2009)
25. Christen, P., Pudjijono, A.: Accurate synthetic generation of realistic personal information. In: Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T.-B. (eds.) PAKDD 2009. LNCS (LNAI), vol. 5476, pp. 507–514. Springer, Heidelberg (2009)