# PRIVATEER: A Private Record Linkage Toolkit

Alexandros Karakasidis<sup>1</sup>, Georgia Koloniari<sup>2</sup>, and Vassilios S. Verykios<sup>1</sup>

<sup>1</sup> Hellenic Open University, School of Science & Technology, Patras, Greece {a.karakasidis, verykios}@eap.gr

<sup>2</sup> University of Macedonia, Department of Applied Informatics, Thessaloniki, Greece gkoloniari@uom.edu.gr

Abstract. Privacy preserving record linkage (PPRL) is the process of integrating data across multiple heterogeneous data sources without compromising their privacy. While many techniques have been developed for PPRL, there has not been, up to now, a universally accepted method providing both performance and quality of results in all cases. To this end, we present PRIVATEER, a toolkit which aims at enabling practitioners to compare various techniques involved in the PPRL process and determine the best for their needs. The toolkit is based on a simulator, designed to be highly configurable, modular and extensible, allowing the user to test different configurations by combining a number of privacy preserving blocking and matching methods with corresponding distance and similarity measures on her own or sample data. We showcase the usability of our toolkit by presenting experimental results measuring both quality and performance of state-of-the-art PPRL methods.

Keywords: Privacy, Record Linkage, Toolkit, Simulator

## 1 Introduction

Large volumes of data, describing individuals, are collected by public or private organizations. Being able to interconnect and analyze these independently stored data is beneficial for businesses, governments and research. For instance, being able to interconnect data of distinct health organizations, hospitals and physicians could prevent epidemic outbreaks or help us better understand the mechanisms behind certain diseases.

The task of linking such heterogeneous data, known as the *record linkage* problem, is not trivial [4]. Since data are stored by independent organizations, there is no unique identifier to determine common data. Also, such data suffer from low quality, exhibiting typos, abbreviations and misspellings, requiring approximate matching methods to link them. Additional challenges arise when we need to protect the privacy of the subjects described by the data. Therefore, *Privacy Preserving Record Linkage (PPRL)* has emerged as the problem where data from heterogeneous sources are integrated without revealing to the participating sources any extra knowledge not relating to their common records.

To privately link data with accuracy, *Privacy Preserving Matching (PPM)* methods focus on elaborately matching data, while maintaining privacy. Though accurate, matching methods often incur high computational costs. To increase their scalability, PPM methods are combined with *Privacy Preserving Blocking (PPB)* methods. PPB methods prune out unlikely to match candidate pairs of records, while also maintaining privacy. Even though they improve efficiency, PPB methods often compromise matching quality.

To the best of our knowledge, there is no single solution that successfully addresses the privacy preserving record linkage problem to its full extent. In particular, state-of-the-art PPB and PPM methods usually focus on a specific type of data. For instance, there are PPM approaches appropriate for string data [11, 16] and others for numerical data [9, 10]. Moreover, there are two party methods and others that require the deployment of a third party. As a result, each time that a PPRL solution has to be deployed, the most appropriate algorithms for the given application scenario have to be selected.

To this end, we introduce *PRIVATEER*, a PRIVATE REcord linkage toolkit designed to assist practitioners in this selection process, and also serve as a comparison tool for different methods. PRIVATEER is a modular simulator, constantly under development, implementing a set of state-of-the-art PPRL methods. Considering the various tasks involved in PPRL, the aim of PRIVATEER is to accommodate all this functionality into a single toolkit. It consists of clearly separate modules, each of them performing a separate task, and uses different components implementing various privacy preserving matching and blocking methods, such as different distance measures used in approximate matching and reference set generators, where a reference set is used as an intermediate point of comparison. Highly composable, PRIVATEER allows all compatible PPB methods to be combined with compatible PPM ones, and with corresponding auxiliary methods, so as to form a complete PPRL solution.

PRIVATEER is also configurable enabling the user to fine tune each of the deployed methods and provides a variety of performance measures. While the toolkit is equipped with sample data, it also enables the use of a user's own data. Towards extensibility, PRIVATEER's clear and comprehensive design provides an API for extending its functionality. Through the Java paradigm the user is able to add its own modules and algorithms.

Similar toolkits, such as TAILOR [6], have been developed for the classical record linkage problem. Related tools for PPRL include data generators [3], corrupters [2], and tools that combine both functionalities [18]. While these tools aim at developing test data for PPRL, PRIVATEER aims at testing different PPRL methods. A first version of our toolkit is presented in [14], but without offering composability or a graphical user interface.

The rest of this paper is organized as follows. In Section 2, we present the architectural design of PRIVATEER and briefly describe the PPM and PPB algorithms, distance measures, and reference set generation methods currently implemented in the toolkit. Section 3 provides experimental results derived from PRIVATEER and we conclude in Section 4.



Fig. 1. Architecture of PRIVATEER

## 2 **PRIVATEER's Architecture**

The primary goals of PRIVATEER's design is composability and extensibility. We wanted PRIVATEER to easily enable the combination of different methods to compose a complete PPRL solution, and the incorporation of new ones as state-of-the-art PPM and PPB methods are constantly developed. To this end, an object-oriented design is appropriate, thus, we used Java to build the simulator comprising PRIVATEER. Also, Java has the property of *Write Once Run Anywhere*, providing machine independence, and there are many out-of-the-box libraries and data structures one can exploit.

We have made a particular effort to limit the assumptions when designing the architectural components of the simulator. In particular, we assume that all operations are executed sequentially. In many PPB and PPM algorithms, there are functions that require the parallel operation of the matching parties. PRI-VATEER simulates this functionality by sequentially executing separate objects. Also, we only focus on the processing cost and do not examine communication costs and data propagation delays.

## 2.1 PRIVATEER's Modules

We followed a modular design, separating data, execution functionality and implemented methods. This approach improves PRIVATEER's extensibility, as it is very easy through its API to add new features and modules. Moreover, since data is retrieved in a transparent way, new data sources may be added as well. Next, we present the architectural elements that comprise PRIVATEER.

The Base Module. The base module holds PRIVATEER's main functionality organizing the execution of each evaluation experiment. It coordinates the different modules and enables us to execute each experiment multiple times. As illustrated in Fig. 1(1), it hosts the configuration module, a PPB and a PPM module, the results processing and the visualization module.

The Configuration Module. The configuration module is invoked by the base module and is responsible for setting an experiment's execution parameters. It consists of a GUI implemented in JavaFX which receives input from the user and a mechanism that maps user input to specific parameters. This mapping mechanism is important, since each of PRIVATEER's modules has to be separately configured using separate parameters provided by the same user interface. **The Nodes.** The nodes implement the functionality of the data sources and a third party, when one is needed. While a data source node has direct access to its data, the third party neither holds nor has direct access to any data. Instead, it is fed with data by the source nodes. Nodes are deployed by the PPB or PPM module and implement their part of the respective algorithm (Fig. 1(2)).

**The Data Module.** The data module is used to provide direct access to external data. It is invoked by each data source node. The used data may be stored in different databases and the configuration module enables the user to select her own input database, thanks to a transparent connectivity mechanism. Currently, MySQL and SQLite databases are supported.

**The Measure Module.** This module implements the distance or similarity measure to be used by the PPB and later PPM algorithm, in a transparent way. Thus, it allows us to deploy any compatible measure with any PPB and PPM method using the GUI. This way, the user has the ability to test the available algorithms and assess their behavior with different configurations.

The Reference Set Generator Module. Similarly to the measure module, the reference set generator provides different methods that can be used interchangeably with our PPB and PPM methods.

The PPB and PPM Modules. The privacy preserving blocking (PPB) and the privacy preserving matching (PPM) modules implement the corresponding type of protocols. Depending on the type of the protocol, as shown in Fig. 1(1), each module initiates two nodes and, when necessary, a third party node. All these nodes implement the same algorithm, which is chosen through the GUI and provided to the corresponding module by the base module. For a PPB or PPM algorithm to operate, the PPB or PPM module load a measure module, which implements the measure used by the corresponding algorithm. If it is required by the implemented algorithm, a reference set generator module is also loaded.

PPRL protocols may vary regarding their implementation and the number of participating nodes. Generally, however, they are either based on some type of data transformation or they may exhibit a multiparty computational behavior. In both cases, a third party may be required and all options may be implemented in PRIVATEER. Moreover, to offer the ability of experimenting with a no blocking setup, a mockup No Blocking method has also been implemented.

Regarding its operation, the PPB passes on the parameters received from the configuration module to the respective modules it hosts and executes the selected PPB algorithm. When the execution concludes, it produces lists of blocks of record ids which are then used by the PPM module. The PPM module receives the blocks of record ids from the PPB module and applies the selected PPM algorithm on each of them. Upon the conclusion of the execution of the PPM module, a list with matching record ids is produced which is then passed to the results processing module for processing.

**The Results Processing Module.** The results processing module processes the matches list received from the PPM module and calculates the number of total matches, the number of true and false matches, precision, recall and the execution times of the PPM and PPB modules. If the input data have the same unique identifiers (i.e., corrupted versions of the same dataset), the module automatically calculates the measurements based on the common primary key. Otherwise, a matching csv file with the matches between different primary keys is used.

**The Visualization Module.** The role of the visualization module is to visualize the results of each experiment by collecting the statistics produced by the results processing module and produce plots and tables of these statistics.

#### 2.2 Implemented Algorithms and Measures

The modules of the architecture enable the selection of a method for each of the different tasks in the PPRL process. Thus, the PRIVATEER is equipped with a pool of implemented state-of-the-art approaches for each such task, i.e., PPB and PPM algorithms, distance and similarity measures and reference set generators (Tables 1-3).

The PPB algorithms currently implemented in PRIVATEER are illustrated in the upper part of Table 1. The first column holds the name of the algorithm, the second a short description, the third shows the type of data each algorithm is designed for and the forth indicates whether a third party node is required. Similarly, PPM algorithms are included in the lower part of Table 1.

Many of the PPRL algorithms that have been suggested, use a distance or similarity measure [4]. We designed PRIVATEER so as to offer the ability to use in the PPRL algorithms alternative measures, other than the default ones. Possible combinations are illustrated in the third column of Table 2. Default algorithms that use the measure are indicated by bold.

Reference sets (RS) are used in PPRL as an intermediate point of comparison to provide privacy. A reference set may be derived from a publicly available database or be arbitrarily generated. Table 3 summarizes PRIVATEER's currently available methods for RS generation. All methods apart from *Arbitrary String Generation* [16] use a user defined publicly available dataset.

#### **3** Execution Results

To illustrate PRIVATEER's usability, we present sample experiments exploring alternative combinations of PPB and PPM methods, beyond their default setup, so as to assess their behavior and performance in different configurations.

All experiments are conducted using an Intel i3 PC with 8GB of RAM. We use a sample dataset, which originates from the North Carolina voters database<sup>3</sup>. We assume that the attributes: last name, first name, middle name, city of residence and precinct description comprise a candidate key, and use them for both matching and blocking. After deduplicating the dataset based on the selected candidate key, we generate two databases of 50 000 records each, so that the 25% of their records is common. Finally, we use the "master" table of Lahman's baseball database<sup>4</sup>, as a source for reference set creation. Since we are interested

<sup>&</sup>lt;sup>3</sup> Available at ftp://alt.ncsbe.gov/data/

<sup>&</sup>lt;sup>4</sup> Available at http://www.seanlahman.com

	Name	Description	Datatype	3P		
PPB Algorithms						
В1	No Blocking	Option when the use of blocking is not desired. Records to be linked are organized in a single block for each of the two data sources.	N/A			
B2	SNEF [13]	SNEF is a meta-blocking algorithm specifically designed to shift the Sorted Neighborhood approach [8] to the PPRL paradigm. This algorithm uses reference set clustering for cre- ating blocks.	Strings, Numbers	~		
В3	TPPB [19]	A single attribute is utilized for blocking. Each data source cre- ates a separate reference set from a publicly available database. Then, it merges its data with the reference set and lexicograph- ically sorts the result.	Strings			
Β4	Simple Blocking [1]	Introduces hash signature transformations to securely repre- sent TF/IDF weight vectors. Blocks consist of records which share common strings called tokens, with a token being an intact word. The Jaccard metric is used for comparing hash signatures.	Strings	۲		
В5	Phonetic Blocking [12]	Uses Soundex's inherent privacy characteristics, which are based on the common information suppression property of all phonetic encoding algorithms. Records are organized in blocks based on their phonetic encoding similarity.	Strings	~		
		PPM Algorithms				
M1	Bloom Bigrams [17]	Bigrams of strings are inserted into a Bloom Filter. Then, a bitwise comparison is performed on Bloom Filters using the Dice Coefficient.	Strings	~		
M2	Embedding Spaces [16]	Based on the edit distances between actual and reference set data, a transformation of the data, called embedding, is derived and used for matching.	Strings	~		
М3	Secure Edit Distance [12]	Secures the use of edit distance through the use of Bloom fil- ters. Each character of a matching field is extracted and after appending its position in the string, it is inserted in a Bloom filter which is encrypted using a secure hash function.	Strings			
M4	Encryption [9, 10]	Homomorphic encryption manages to calculate differences se- curely based on the properties of specific cryptographic algo- rithms, such as RSA.	Numbers			
M5	Soundex Matching [11]	Transforms attributes into Soundex codes and exploit its in- herent information suppression properties to provide privacy.	Strings	~		
M6	Hash Signatures [1]	Calculates the TF/IDF weights of all words, called tokens, within all fields. Then, a hash function is used to create arrays of these weights, one for each record. This method is not suit- able for approximate matching, since the entire token (word) is encoded. Table 1. Supported BDP and BDM algorithms.	Strings	~		

## Table 1. Supported PPB and PPM algorithms.

Measure	Description	Compatibility			
	Distance measures				
Levenshtein	Calculates the minimum number of operations (character insertion, deletion and substitution) required to transform one string to another.	B2, B3, M2			
Damerau - Levenshtein	Alternative to Levenshtein distance that also takes into ac- count the transposition of two adjacent characters.	B2, B3, M2			
	Similarity measures				
Jaro - Winkler	Focused on short strings such as names. Produces a nor- malized score between 0, when there is no similarity, and 1, where there is an exact match between two strings.	B2, B3, M2			
Jaccard coefficient	Used to measure set similarity. In the context of PRIVA- TEER, it is used with bit vector representations of strings, as in [1].	<b>B4</b> , M1			
Dice's coefficient	Also used with bit vector representations of strings. Used as a default similarity measure in [17].	B4, <b>M1</b>			
Q-grams Dice's coefficient	Dice's coefficient for plain text comparisons in combination with q-grams, as in [7].	B2, B3, M2			

 Table 2. Supported measures.

Method	Description	Compatibility
NN-Clustering	Tries to produce the most representative reference set from the entire data corpus.	M2, B2, B3
K-Medoids Clustering	Elects medoids from the most representative reference set from the entire data corpus.	M2, <b>B2</b> , B3
Arbitrary String Generation	Generates random strings	$\mathbf{M2},\mathbf{B2},\mathbf{B3}$
Durstenfeld Shuffle	Generates a reference set of specific size. Given an array of elements the first $n$ of them are randomly exchanged with others in the array and selected out of a data corpus.	

 Table 3. Reference set generators.



**Fig. 2.** Results illustrated by PRIVATEER's visualization module: (left) PPM, (center) PPB and (right) RS generation.

in linking low quality data, we corrupt one of the two databases using the corrupter described in [2]. The corrupted records contain one error per attribute so that a join with the original ones returns no matches. With equal probability, an error might either be a character insertion, deletion or substitution.

Figure 2(left) illustrates how PRIVATEER allows us to test a blocking method with different matching methods. In particular, we combine the TPPB method [19], with three PPM methods: Bloom Bigrams [17], originally used in [19] as well, Hash Signatures [1], and Phonetics Matching [11]. We compare the three combinations in terms of precision and recall. As we observe in Fig. 2(left), Hash Signatures have zero recall, since this method is not designed for approximate matching. On the other hand, the Bloom Bigrams' high performance in terms of recall justifies its overall popularity [10].

In the second experiment, we combine a matching method with different blocking methods. In particular, Bloom Bigrams is combined with Simple Blocking [1], SNEF [13] and TPPB [19]. As for blocking, efficiency is important, we measure time to determine the most efficient blocking method for Bloom Bigrams matching. Figure 2(center) illustrates the time required for PPB, for PPM and for the overall PPRL process. We observe how the matching time varies depending on the blocking method used. While Simple Blocking is the fastest blocking approach, it does not manage to significantly reduce the matching time. On the other hand, while SNEF is slower while blocking, it achieves the best overall time as it significantly reduces matching time. Finally, TPPB has the highest blocking time, but achieves the best matching time. It clearly creates the optimal blocks, but unfortunately the time blocking requires is greater than the overall time for SNEF.

Next, we evaluate three reference set generation methods with SNEF[13] and use Bloom Bigrams for matching [17]. The first one is the arbitrary string generation method [16], the second is a modification of the Durstenfeld shuffle algorithm [5], while the third uses the K-medoids algorithm [15], which was originally used in [13]. Figure 2(right) shows that with regards to precision and recall, the Durstenfeld Shuffle, and not the K-medoids used in the default setup of SNEF, is the best choice.

#### 4 Conclusions

We have presented PRIVATEER, a modular, highly configurable privacy preserving record linkage toolkit that enables combining and comparing different PPRL solutions. We showed the usability of our toolkit by including evaluation results derived from experiments run by the PRIVATEER on sample data.

Acknowledgment. Attendance to this conference was partially supported by the Hellenic Artificial Intelligence Society (EETN).

## References

- 1. Al-Lawati, A., Lee, D., McDaniel, P.: Blocking aware private record linkage. In: IQIS (2005)
- 2. Bachteler, T., Reiher, J.: A test data generator for evaluating record linkage methods. Tech. rep., German RLC Work. Paper No. wp-grlc-2012-01 (2012)
- 3. Christen, P.: Febrl-: an open source data cleaning, deduplication and record linkage system with a graphical user interface. In: ACM SIGKDD (2008)
- Christen, P.: Data Matching. Data-Centric Systems and Applications, Springer (2012)
- 5. Durstenfeld, R.: Algorithm 235: Random permutation. Communications of the ACM 7(7), 420– (Jul 1964)
- Elfeky, M.G., Verykios, V.S., Elmagarmid, A.K.: Tailor: a record linkage toolbox. In: IEEE ICDE (2002)
- Gravano, L., Ipeirotis, P.G., Jagadish, H.V., Koudas, N., Muthukrishnan, S., Pietarinen, L., Srivastava, D.: Using q-grams in a dbms for approximate string processing. IEEE Data Engineering Bulletin 24(4), 28–34 (2001)
- Hernández, M.A., Stolfo, S.J.: Real-world data is dirty: Data cleansing and the merge/purge problem. Data Mining & Knowledge Discovery 2(1), 9–37 (1998)
- 9. Inan, A., Kantarcioglu, M., Bertino, E., Scannapieco, M.: A hybrid approach to private record linkage. In: IEEE ICDE (2008)
- Inan, A., Kantarcioglu, M., Ghinita, G., Bertino, E.: Private record matching using differential privacy. In: ACM EDBT (2010)
- 11. Karakasidis, A., Verykios, V.S.: Privacy preserving record linkage using phonetic codes. In: BCI (2009)
- Karakasidis, A., Verykios, V.S.: Secure blocking + secure matching = secure record linkage. JCSE 5(3), 223–235 (2011)
- 13. Karakasidis, A., Verykios, V.S.: A sorted neighborhood approach to multidimensional privacy preserving blocking. In: IEEE ICDMW (2012)
- Karakasidis, A., Verykios, V.S.: A simulator for privacy preserving record linkage. In: EANN, Part II (2013)
- Kaufman, L., Rousseeuw, P.: Clustering by Means of Medoids. Reports of the Faculty of Mathematics and Informatics. Delft Univ. of Technology (1987)
- 16. Scannapieco, M., Figotin, I., Bertino, E., Elmagarmid, A.K.: Privacy preserving schema and data matching. In: ACM SIGMOD (2007)
- 17. Schnell, R., Bachteler, T., Reiher, J.: Privacy preserving record linkage using bloom filters. BMC Medical Informatics & Decision Making 9(1), 41+ (2009)
- Tran, K., Vatsalan, D., Christen, P.: Geco: an online personal data generator and corruptor. In: ACM CIKM (2013)
- 19. Vatsalan, D., Christen, P., Verykios, V.S.: Efficient two-party private blocking based on sorted nearest neighborhood clustering. In: ACM CIKM (2013)