THE JOURNAL OF DIGITAL FORENSICS, SECURITY AND LAW

Journal of Digital Forensics, Security and Law

Volume 17 Article 6

March 2022

A Combined Approach For Private Indexing Mechanism

Pranita Maruti Desai Ms.
University of Mumbai, pranita.m.desai@gmail.com

Vijay Maruti Shelake Mr.
University of Mumbai, vijay_sakec@yahoo.co.in

Recommended Citation

Desai, Pranita Maruti Ms. and Shelake, Vijay Maruti Mr. (2022) "A Combined Approach For Private Indexing Mechanism," *Journal of Digital Forensics, Security and Law*: Vol. 17 , Article 6.
DOI: https://doi.org/10.15394/jdfsl.2022.1790
Available at: https://commons.erau.edu/jdfsl/vol17/iss1/6

A COMBINED APPROACH FOR PRIVATE INDEXING MECHANISM

Ms. Pranita Maruti Desai and Mr. Vijay Maruti Shelake
University of Mumbai
Yadavrao Tasgaonkar College of Engineering and Management
Dr. N.Y. Tasgaonkar Technical Education Complex, Chandai, Bhivpuri Road Station, Tal. Karjat
Raigad, Maharashtra - 410201, India
pranita.m.desai@gmail.com, vijay_sakec@yahoo.co.in

ABSTRACT

Private indexing is a set of approaches for analyzing research data that are similar or resemble similar ones. This is used in the database to keep track of the keys and their values. The main subject of this research is private indexing in record linkage to secure the data. Because unique personal identification numbers or social security numbers are not accessible in most countries or databases, data linkage is limited to attributes such as date of birth and names to distinguish between the number of records and the real-life entities they represent. For security reasons, the encryption of these identifiers is required. Privacy-preserving record linkage, frequently used to link private data within several databases from different companies, prevents sensitive information from being exposed to other companies. This research used a combined method to evaluate the data, using classic and new indexing methods. A combined approach is more secure than typical standard indexing in terms of privacy. Multibit tree indexing, which groups comparable data in many ways, creates a scalable tree-like structure that is both space and time flexible, as it avoids the need for redundant block structures. Because the record pair numbers to compare are the Cartesian product of both the file record numbers, the work required grows with the number of records to compare in the files. The evaluation findings of this research showed that combined method is scalable in terms of the number of databases to be linked, the database size, and the time required.

Keywords: Indexing, Deduplication, Record linkage, Data preprocessing, Fingerprint, Query

1. INTRODUCTION

With the advancement in technology, a large amount of data is being collected by both private and public-sector companies and individuals. Many of these records are about people involved in financial transactions, transactions related to shopping, transactions with travel facilities, health records, and electronic data. This also includes records of census, tax, social security, blog entries, tweets, emails, and SMS. Businesses and governments use this information for their advantage. The information gathered is saved in the database as records. Each record has a key field that allows it to be distinguished from others. This record key is important in finding the data whose key is already known to the user. Indexing is a technique for quickly retrieving records from database files that have some attributes on which it has been performed. It is used in databases to keep track of the number of records. For example, in the health care industry, maintaining patient records up to date. Frequently, data from diverse sources must be combined and linked. When databases are linked across businesses, data security, preserving privacy and confidentiality is vital to protecting sensitive data used for analysis.

If indexing were not used, every record in one dataset would need to be compared to every other record in another dataset. This results in a substantially larger number of comparisons and would rapidly lower system performance. Suppose each record in one dataset must be compared to each record in another dataset. In that case, the number of record pair comparisons increases as the number of records to be matched increases. For example, the number of record comparisons with a rising number of records is illustrated in Table 1. In the second row, 1000 records from one dataset need

Table 1. Record comparison

Number of Records	Number of Comparisons
1,000	1,000,000
10,000	100,000,000
1,000,000	1,000,000,000,000

to be compared with 1000 records from another dataset. For any number of datasets, this strategy is computationally inefficient, which will consume extra time and space. It limits comparison pairings to those that are most likely to match.

Indexing separates the data into subsets or blocks based on the premise that no matches exist between the blocks that are different. Areas like business names, family names, and dates of birth are most used to create blocks. Since blocks can lead to typographical or spelling problems, they are frequently standardized. Certain matches may occur within the blocks. For example, records for a woman who has changed her surname may not be related or linked in a block based on the surname because it changes after marriage. However, if the date of birth is used, the records may be linked. As a result, several indexing variables are frequently used, increasing the likelihood that a linkage missed by one run may be identified by a subsequent indexing pass, reducing future errors.

The indexing method filters out similar or approximately similar records from the number of record comparisons. This is the most important step in the privacy-preserving record linkage (PPRL), which is a type of record linkage (A./M. Mitzenmacher, Kirsch (2006)) process. After data preprocessing and deduplication, which reduces duplicates and errors in data, the proper indexing method increases the chances of getting the more accurate number of matching pairs used in the final linkage process.

The following is the paper's structure: the background knowledge of the methodologies used in this research is elaborated in Section 2. The proposed research technique is described in Section 3, together with its system requirements and architecture. The results of the experiments we have described in Section 4. The conclusion and future scope are addressed in Section 5.

2. RELATED WORK

By bringing potentially linkable record pairs together, indexing or searching reduces the number of comparative record pairs. A good indexing variable attribute should have a high number of attribute values that are fairly and evenly distributed. Also, it should have a low probability of reporting an error. Linkable record pairs can be broken due to errors in the characteristics used for indexing. Many phonetic codes have been developed for text properties to prevent the effects of spelling and aural problems when recording names. New York State Identification and Intelligence System (NYSIIS) and Soundex are two common phonetic codes. These codes were used to represent various types of names and English pronunciations.

Before indexing, data preprocessing, identifier selection, and data duplication are all essential procedures. Real-world data from various data holders will typically store the same information in various ways. How one maintains a date of birth, for example, will differ per country due to differences in the placement and year separation of days and months. Other issues emerge when months are stored using their names or even abbreviations rather than their numeric values in the intended language. Only the final two digits (Baxter R, GuL (2004)) are used when storing birth years, eliminating the century. For this reason, data preparation before linkage is an important aspect of every record linkage application. This stage includes data cleaning for inconsistencies, foreign character sets, erroneous characters caused by encoding issues, and normalizing variables. Unaffectedly, the record linkage variables provided in both datasets are limited to the same set of identifiers.

Other things to consider are the number of values that are missing in both selected datasets. If there

are few entries (Cohen WW, Richman J. (2002)), the variable is not a good fit. The quantity of information attained by including a variable to identify an entity in the data is also important. For example, an address is a better attribute to identify various entities than a selected person's gender. Finally, the number of errors that can be expected, such as address fields changing over time and resulting in missing linkages, is critical.

Further, deduplication is the process of matching a data file with itself or another file to detect representations of duplicate data of the same thing in a set of data. Multiple matches for a single representation indicate data duplication. Because duplicate data values add no information to the data but increase the size of the available database, they are frequently deleted before the data is gathered or separated as part of the preprocessing. The major goal of the privacy-preserving record linkage (Holmes, D., McCabe, C. M. (2002)) process is to link similar records. PPRL is a method of record linkage that uses encrypted identifiers and may become more popular in the future due to these new protection standards for data. Indexing is a phase in the process of record linkage that is important for reducing the number of needless comparisons.

Standard, sorted neighborhood (Kristensen, T. G./J. Nielsen/C. N. S. Pedersen (2010)), q-gram based (Latanya Sweeney (2002)), canopy clustering (Lifang Gu and Rohan Baxter (2006)), locality sensitive hashing and multibit trees (M. Hernandez and S. Stolfo (1998)) are some of the methods available. Some are traditional, while others are more recent procedures with improved features over the previous ones. To create the appropriate indexes for the comparison, we will be using both old and recent methodologies with their security features in this research. Our major goal is to combine standard indexing, which uses Soundex (Peter Christen (2012)) and multibit tree indexing, which uses Cryptographic Long-term Key (CLK) to encode the data.

For decades, many classic indexing strategies have been used in deduplication and data matching. The identifier for each record is simply entered into one block, which makes this approach unique. Soundex, phonix, double metaphone, phonex, NYSIIS, and other methods generate indexing keys. Other indexing methods split a single record into many blocks. An indexing key value (IKV) is generated for each record in the input database. This IKV determines where a record is placed in the database. All records with the same IKV are grouped into the same block. For database matching, pairs of candidate records are constructed from all the records in both databases with the same IKV. If an IKV occurs exclusively in records from one of the databases, no record pairings will be created from this block because the matching block in the other database has no records. All similar pairs of record identifiers inside a block are used to generate candidate record pairs for deduplication. To prevent superfluous pairs, each similar record pair only has to be compared once because the comparison of two records is symmetric. For example, as illustrated in Table 2, with the three record identifiers' Rid1', 'Rid2', and 'Rid3' in a block, the resulting record pairs for deduplication would be (Rid1, Rid3), (Rid2, Rid3), and (Rid1, Rid2) but not (Rid3, Rid2) or (Rid3, Rid1), (Rid2, Rid1).

The Soundex algorithm is one of the most extensively used and oldest methods of phonetic encoding. It encodes name strings based on the pronunciation of the American-English language by preserving the first letter of the string and transforming the other letters into integers according to the transformation rule. All the zeroes that correspond to the letter's "w", "y", "h" and vowels are removed from the encoded string because they are all repeats of the same number. For example, a converted encoding of "t0440555" is translated to "t45" and a transformed encoding of "k770399051" is changed to "k7391". Moreover, if the first digit of the encoding is less than three digits, the code is extended with 0's to a total length of 3 digits, resulting in "t45" becoming "t450". In contrast, codes with more than three digits are truncated to 3 digits alone, resulting in "k7391" becoming "k739". For example, we have shown in Table 2 the indexing key values of these records, like Robert becomes R163 and Ashcroft becomes A261.

Multibit trees work with bit vectors, as suggested by Kristensen for cheminformatics and adapted by Schnell for PPRL. This is used to quickly search large databases of molecular fingerprints. In a bit vector of length l, a molecular fingerprint describes structural information about molecules. To locate

Table 2. Lastname values with their Soundex encodes

RecId	Lastname	IKVs
Rid1	Robert	R163
Rid2	Ashcraft	A261
Rid3	Rubin	R150
Rid4	Ashcroft	A261
Rid5	Rupert	R163

A261	R150	R163
↓	↓	↓
Rid2	Rid3	Rid1
Rid4		Rid5

these pairs, molecular fingerprints are used to look for structurally related compounds.

Assume one needs to look up a query (Pyle D (1999)) in a database of molecular fingerprints, Yi. To put it another way, the goal is to locate all fingerprints in the database that are similar to X above particular threshold t. Multibit trees locate all fingerprints, Yi where (X, Yi) ≥ t, or filter out any fingerprints lower or equal to t.

Multibit trees follow a three-step process: The fingerprint database is partitioned into groups of fingerprints for future use in the first step. In the second step, an actual tree is formed within each created group. The trees built earlier are searched for fingerprints in the third step. All the fingerprints, Yi, are divided into bins of identical size during the partition process. The bit numbers set to 1 in Y determine the size of a fingerprint, which is indicated by |Y|. Because $\frac{\min(|Y|,|X|)}{\max(|Y|,|X|}$ is an upper bound on S_j(X, Yi), all bins satisfying t|Y| ≥ |X| or |Y| ≤ t|X| can be disregarded in the searching step.

In the next tree building step, fingerprints of similar size are actually stored in a binary tree structure (Schnell, R./C. Borgs (2017)) with one single tree for each bin. The method starts by assigning all of the fingerprints from the bin to the tree's root node. The algorithm then recursively allocates all the fingerprints, Yi, with 0 at a fixed bit location to the subtree on the left side and all other fingerprints with a 1 at the bit position to the subtree on the right side at each parent node. At each parent node, the deciding bit position is determined so that the tree remains as balanced as possible. At each node, there are also two lists of match bits. List O contains all positions of bit with constant value 0 in all remaining fingerprints below that node, and list I contains all positions of bit with constant value 1. The recursion ends when the number of fingerprints at a node falls below a previously determined threshold.

The search procedure for query fingerprint A is divided into three sections. All bins that met the conditions indicated in the partition step were discarded in the first phase of searching. In the second phase of the search, each remaining tree is subjected to a depth first search. For each of the tree nodes currently explored, the recorded lists of O's and I allow the computation of an upper bound of the Jaccard Similarity for all fingerprints below the currently visited node. During the search, the algorithm determines the bit position numbers assigned to 1 in X as well as the number of bit positions set to 0.

The Cryptographic Long-term Key (CLK), an extension of the bloom filter, is used in the multibit tree approach. This is a type of composite bloom filter (S. Joshua Swamidass and Pierre Baldi (2007)), which is the result of an OR operation on each identifier attribute's bloom filter. In Figure 1,

we have shown the CLKs construction with 0 and 1 values, which gives the Dice coefficient (Tobias Bachteler, Rainer Schnell and J¨org Reiher (2011)), which is the sum of the unique bi-grams in both sets divided by the doubled intersect of the two sets of bi-grams. To approximate the dice coefficient using CLKs, each unique element in a collection of q-grams sets the k number of different bit positions in the CLK to a value of 1. A hash function value is one of the k mappings of an element to a selected bit location.

Figure 1 depicts how a CLK is built from two separate names using a hash function with k=1 and l=18 bits for each bi-gram. For visualization purposes, the bit locations are set to 1 and in the same sequence as the bigrams. However, this may vary depending on the string. We have two strings, FREDDIE and FREDDEE, that differ by one character, resulting in an edit distance of one. The clear-text bi-grams' dice coefficient would be:

$$D = \frac{2.4}{6+6} = 0.667$$

FREDDIE – FR RE ED DD DI IE

FR			**RE**	**ED**			**DD**				**DI**			**IE**			
1	0	0	**1**	**1**	0	0	**1**	0	0	0	**1**	0	0	**1**	0	0	0

1	0	0	**1**	**1**	0	0	**1**	0	0	**1**	0	0	0	0	0	0	**1**
FR			**RE**	**ED**			**DD**			**DE**							**EE**

FREDDEE – FR RE ED DD DE EE

Figure 1. CLK construction (C. Borgs (2019))

The basic method for implementing any CLK-based encoding usually follows the pattern outlined in the following subsections:

1. Input strings are standardized depending on the sort of data they contain.
2. Blank space is sometimes added to standardized strings at the end and beginning as padding, which gives the string's last and first character additional weight. The resulting string is then broken into q-grams by dividing it into subsets of length q, where q is one of the user defined parameters.
3. In the CLK, each element formed in step (2) corresponds to numerous k bit places. The technique determines these bit places, which is dependent on the implementation details. Currently, using random hashing to hash components into the CLK is strongly suggested overusing double hashing, which is considered outdated.
4. The resulting bit positions of value 1 in the initially empty CLK with all bit positions set to 0 are dependent on the parameter choices.

For each element formed in step (2), steps (3) and (4) are repeated. A standard CLK is a bit vector that contains only the elements of a single identity.

Some parameters must be selected to construct a multibit tree, such as the length of the CLK within which values of 0 and 1 are alternately inserted, the minimal threshold which is Tanimoto coefficient and the number of functions which is k. There are three phases of indexing with Q-gram fingerprints using multibit trees:

1. Both the file records are converted into Q-gram fingerprints.
2. Larger file fingerprints are stored in a multibit tree.
3. A smaller file is compared to each fingerprint's multibit tree.

3. PROPOSED WORK

When the number of parties grows, it becomes computationally expensive to generate candidate record sets for myriads of databases. In such cases, methods for reducing the space of comparison are required. In the record linkage process, these strategies are known as indexing methods. Such algorithms identify reduced sets of candidate records for comparison and categorization by retaining true matching records in sets of candidate records while deleting as many genuine non-matching record sets as possible.

As the number of parties increases, more complex indexing systems are required. More comparisons are required regardless of whether there are large block numbers with small record numbers or small block numbers with large numbers of records. This challenge demonstrates the number of candidate record sets generated for various parties using various block sizes and datasets.

The generated candidate number record sets grow in large volumes with the increasing number of

parties involved in a multiparty protocol. The number of candidates generated record sets becomes so large that it is practically impossible to handle, even with very small-sized blocks. One of the primary issues is the lack of control over block sizes in currently available indexing systems. The comparison procedure becomes significantly more arduous and time-consuming when a large number of blocks are generated in various sizes. We created a method that constructs blocks of entries in a balanced tree data structure using an indexing technique and phonetic encoding to address this issue. Each party will have a tree data structure with leaf nodes holding blocks of records. The tree is constructed securely, with no information about each party's records being shared.

We have developed a technique that uses both classic indexing and more contemporary tree-based indexing to incorporate the benefits of both processes while also overcoming the shortcomings that can emerge when using them alone. This research employed a combination of strategies that decreased both space and time. The normal indexing method takes more time to create blocks than the multibit tree. The multibit tree produces good results, it still produces a lot of false-positive pairs, which we can eliminate with this combined technique.

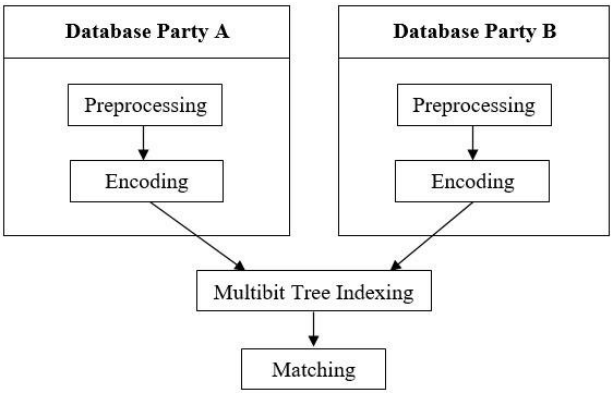

Figure 2. Indexing step in comparing two databases

By using more than one database for indexing and with the help of an efficient method, we created a collection of pairs and sent them to be compared further. Figure 2 depicts two database parties preprocessing their data at their respective sites. Both sides provide their preprocessed data to be indexed, where pair blocks are formed and matched. We encode our data after preprocessing. We encode the characteristics in a multibit tree as CLKs, which we have decided to use as index keys. This proposed methodology encoded attributes using phonetic encoding, a conventional indexing method and then translated the encoded data into a cryptographic long-term key, another method to secure the data. We used the multibit tree approach to obtain approximate matches after converting them into CLKs.

The layered design in Figure 3 depicts a step-by-step breakdown of the indexing approach that leads to potential matches among a vast number of entries. This process we have discussed earlier. This is the actual bottom-to-top approach to get the proper matching fields. Standardization is the most important step because error-free data gives the most appropriate and true matches.

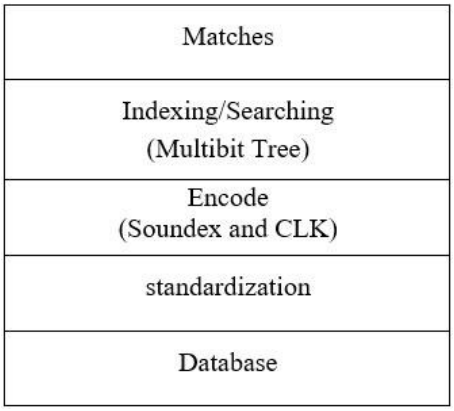

Figure 3. Layered design with Indexing step

The information flow diagram in Figure 4 depicts how record data flows through the indexing process, resulting in record match pairs.

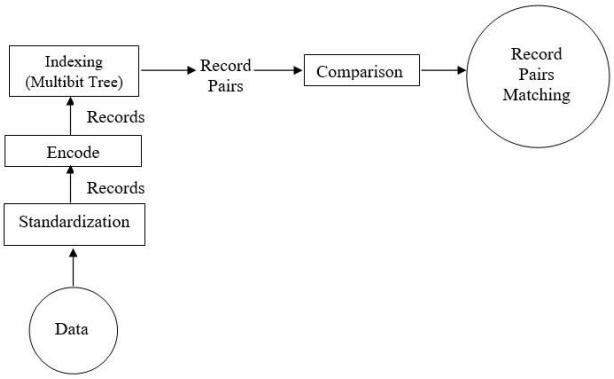

Figure 4. Information flow with Indexing step

Individual techniques, such as typical blocking Soundex approaches, are commonly used to encode properties. These keys are then compared to other database keys, and matching indexes are discovered. We used both phonetics and CLKs to encode the data in the combined technique and then combined their results to locate the matched pairs. Data matching, which is the latter part of the record linkage process, requires true or false matches to check the efficiency and quality of the indexing method.

4. RESULTS AND DISCUSSIONS

This experiment requires Rstudio, the R version software, and Rtools, which is used to install packages directly from the internet. This software is available as freeware and may be downloaded quickly. The R version is available depending on the type of operating system, along with many useful packages. These built-in packages are written in C or C++. We have used the R programming language, which is frequently used by statisticians and data miners to create statistical tools and for data analysis.

The experiment was conducted on a laptop computer with an Intel Core i3-7100U processor, a 64-bit version of windows 10 operating system, 4 GB of RAM. To run the data, a variety of software is available. The PPRL package provides the necessary functionality for using PPRL methods in R. It is available for free on CRAN. It can use PPRL methods to encrypt, preprocess, and link data, allowing us to complete the entire record linkage process in R. For the tree's creation, the Multibit Tree package is used. The gmodels package was used to plot the graphs. We tested our dataset and calculated comparison pairings using built-in R language tools. We have included both tabular and graphical representations of our findings.

We can observe from Table 3 that the combined technique offers better comparative values than the multibit tree alone. We measured the number of functions used, the time taken for each loop, and the bit positions set to one for the graph analysis. In this experiment, CLKs have a length of 256, which implies we will have 0 or 1 values within 256-bit vectors. Information is securely saved within these values. As the hash function number grows, so do the bit values, which can lead to false positives. This necessitates the use of a correct k function to avoid erroneous matches. We have also looked at the time it takes to execute each loop, which will help us estimate the total time required for the entire process. By calculating the amount of time, we can determine whether one way is better than the other.

Figure 5 shows the percentage of bits position set to 1 and the number of used hash functions using the combined approach.

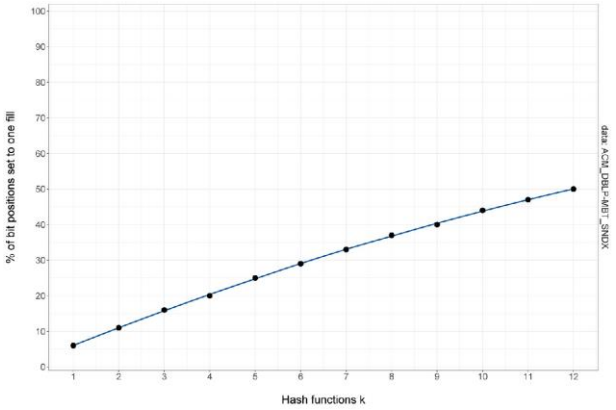

Figure 5. The percentage of bits position set to 1 and No. of used hash functions with Combined Method

Figure 6 shows the percent of bits position set to 1 and the number of used hash functions with MBT.

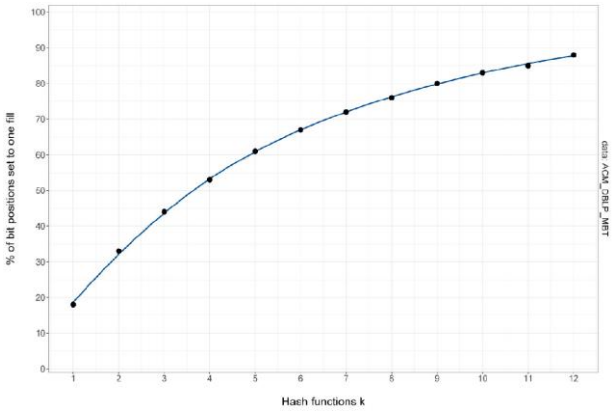

Figure 6. The percentage of bits position set to 1 and No. of used hash functions with MBT

Figure 7 shows the running time and the number of used hash functions with the combined approach.

Table 3. Result comparison of both methods

Indexing Methods / Evaluation Metrics	MBT	Combined Approach
Reduction ratio	0.9953	0.9996
Pairs completeness	0.7514	0.84
F-score	0.8563	0.9129
Running time (sec)	6.9535	4.9463

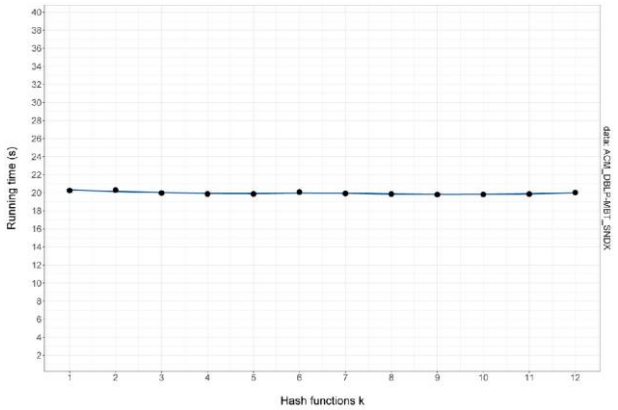

Figure 7. Running time and No. of used hash functions with Combined Method

Figure 8 shows the running time and the number of used hash functions with the combined approach.

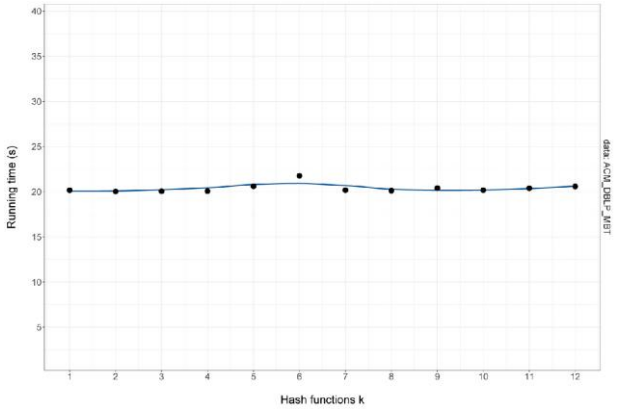

Figure 8. Running time and No. of used hash functions with MBT

We used two real datasets that contained information about articles that had been published earlier in the corresponding publication or conference. ID, title, author, publication, and year are all the fields in one dataset. Other fields in the collection include reference information, title, author, publication, and year. It will be simple to match data if datasets include common attributes used as index keys. Using a combined method, we were able to isolate matched record sets that are most likely equal.

The indexing keys have been encoded. We have calculated evaluation metrics such as reduction ratio, pair completeness, f-score, and running time depending on the number of records. Table 3 compares the two techniques using these metrics.

RR= 1- $\frac{s}{N}$ for reduction ratio, which is the possible number of record pairs in the total datasets and s is the number of record pairs produced by the indexing method for comparison.

PC= 1- $\frac{sM}{NM}$ for pair completeness, where NM is the total number of true match pairs in the entire dataset and sM is the number of true match record pairs in the set of record pairs produced for comparison by the indexing method.

F-score = $\frac{2*PC*RR}{PC+RR}$ for f-score value, which uses a harmonic mean to combine RR and PC.

5. CONCLUSION AND FUTURE WORK

We presented a combined indexing method for PPRL that uses traditional indexing and multibit trees. Each party creates a multibit tree structure based on the CLKs created by the parties, and their datasets collaborate to determine the optimum bit positions to use. An experimental evaluation of the suggested methodology was conducted, in which we tested it on two separate datasets with common values and varying record sizes.

The evaluation findings revealed that this methodology is scalable in terms of both the number of databases to be linked and the database size. This technique outperforms the traditional standard indexing strategy in terms of privacy and indexing quality. We used to employ phonetic encoding for index keys in classical indexing, which allowed us to encode data in a basic method, such as a single letter and integers up to a few digits. In the previous technique, we were directly encoding index keys in CLKs in a multibit tree, but in this combined technique, we are applying CLKs over phonetic encoded data, which is more secure and less vulnerable to attack. Finally, using private comparison and classification techniques, the blocks formed by a multibit tree can be compared to determine related record sets in different databases.

We intend to expand this method with other tree structures that can minimize the number of trees and split the tree nodes into more bits. We will also look into the best and most efficient parameter selection for encoding the data. This system has a disadvantage in that privacy can be jeopardized when some of the parties are not genuine, necessitating more protective safety measures and communication. We want to further improve this approach so that it can be used in applications in the real world of PPRL.

REFERENCES

A./M. Mitzenmacher, Kirsch. (2006). Less Hashing Same Performance: Building a Better CLK, 456-467, In Azar, Y./T. Erlebach (Eds.), Algorithms-ESA 2006, Proceedings of the 14th Annual European Symposium.

Baxter R, GuL. (2004). Adaptive filtering for efficient record linkage, In SIAM international conference on data mining, Orlando.

C. Borgs. (2019). Optimal Parameter Choice for Bloom Filter-based Privacy-preserving Record Linkage.

Cohen WW, Richman J. (2002). Learning to match and cluster large high dimensional datasets for data integration, In Proceedings of ACM SIGKDD, Edmonton.

Dice, L. R. (1945). Measures of the Amount of Ecologic Association between Species, In: Ecology.

Holmes, D., McCabe, C.M. (2002). Improving precision and recall for Soundex retrieval, In: Proceedings of the IEEE International Conference on Information Technology-Coding and Computing, Las Vegas.

Kristensen, T. G./J. Nielsen/C. N. S. Pedersen. (2010). A Tree-based Method for the Rapid Screening of Chemical Fingerprints, In: Algorithms for Molecular Biology.

Latanya Sweeney. (2002). k-anonymity: A model for protecting privacy, In: International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10.05.

Lifang Gu and Rohan Baxter. (2006). Decision models for record linkage, In: Data mining, Springer.

M. Hernandez and S. Stolfo. (1998). Real world data is dirty: data cleansing and the merge/purge problem, Journal of Data Mining and Knowledge Discovery, 1(2).

Peter Christen. (2012). Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection, Springer Science and Business Media.

Pyle D. (1999). Data preparation for data mining, Morgan Kaufmann Publishers, San Francisco.

Schnell, R./C. Borgs. (2017). State of the Art Privacy-preserving Record Linkage of Large Administrative Datasets, New Techniques and Technologies for Statistics.

S. Joshua Swamidass and Pierre Baldi. (2007). Bounds and Algorithms for Fast Exact Searches of Chemical Fingerprints in Linear and Sublinear Time, In: Journal of Chemical Information and Modeling.

Tobias Bachteler, Rainer Schnell and J¨org Reiher. (2011). A novel error-tolerant anonymous linking code, Working Paper WP-GRLC-2011-02, German Record Linkage Center.