Schnell, R. (2013). Privacy-Preserving Record Linkage and Privacy-Preserving Blocking for Large Files with Cryptographic Keys using Multibit Trees. Paper presented at the Joint Statistical Meeting, 3-8 Aug 2013, Montreal, Canada.

CITY UNIVERSITY LONDON

City Research Online

Privacy-Preserving Record Linkage and Privacy-Preserving Blocking for Large Files with Cryptographic Keys using Multibit Trees

Rainer Schnell*

Abstract

Increasingly, administrative data is being used for statistical purposes, for example registry based census taking. In practice, this usually requires linking separate files containing information on the same unit, without revealing the identity of the unit. If the linkage has to be done without a unique identification number, it is necessary to compare keys which are derived from unit identifiers and which are assumed to be similar. When dealing with large files like census data or population registries, comparing each possible pair of keys of two files is impossible. Therefore, special algorithms (blocking methods) have to be used to reduce the number of comparisons needed. If the identifiers have to be encrypted due to privacy concerns, the number of available algorithms for blocking is very limited. This paper describes the adoption of a recently introduced algorithm for this problem and its performance for large files.

Key Words: Indexing, Bloom-Filter, Census, PPRL, *q-gram* blocking

1. INTRODUCTION

Due to the increasing availability of administrative information, linking different databases to determine the overlap of the databases or to enhance the information available for a certain unit is a widely used strategy for statistical purposes. For example, of the 40 European censuses in 2010 only 21 were traditional censuses, while the rest was based on the linkage of registries (Valente 2010). Linking different databases using a set of common identifiers is trivial if a unique personal identification number (PID) can be used. In some countries (for example, the Scandinavian countries) a PID is available for all members of the population. In practice, however, most statistical linkage operations are based on personal identifiers such as the name or date of birth. Such identifiers must be combined to yield an identification code. However, the identifiers are usually neither stable nor recorded without errors (Winkler 2009). Therefore, the use of exact matching identifiers will only link a non-randomly selected subset of the records. Hence, methods allowing for small variations of identifiers are to be used. Unfortunately, encryption of identifiers usually restricts linking to exact matching identifiers only. Hence, methods for linking with encrypted identifiers allowing for small errors in identifiers have to be used. Suitable methods are called "privacy preserving record linkage techniques". A method for privacy preserving record linkage which has recently become popular, is the use of Bloom-Filters.

2. USING BLOOM-FILTERS FOR ENCODING IDENTIFIERS

In 2009, we suggested the use of Bloom-Filters for cryptographic encoding of identifiers (Schnell et al. 2009). Since then, this approach has been used in different countries by different research teams and compares favorably to other approaches (Vatsalan et al. 2013).

The basic principle is splitting the string representing each identifier (for example the name) into a set of unique subsets of length n (n-grams). For example, with $n = 2$ the bigram set of "PETER" is $\_P, PE, ET, TE, ER, R\_$. Each bigram of the set is mapped

*German Record Linkage Center, University of Duisburg-Essen, 47165 Duisburg, Germany

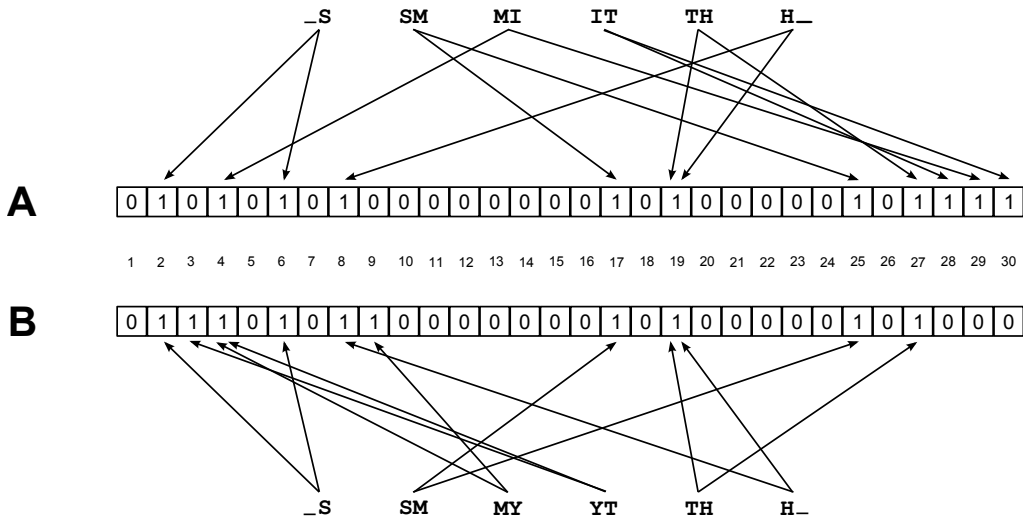

Figure 1: Example for the mapping of two names (SMITH, SMYTH) using bigrams and two hash functions to two Bloom-Filters (A, B) with 30 bits each.

with k different cryptographic one-way hash functions to a bit vector of length l (a Bloom-Filter). As hash functions, keyed hash functions (HMACs), usually MD-5 and SHA-1, are used (for details on HMACs, see Martin 2012). Figure 1 shows a simple example of mapping names to Bloom-Filters using bigrams. In the example, 8 identical bit positions are set to 1 in both Bloom-Filters. In total, 11 bits in A and 10 bits in B are set to 1. Using the Dice coefficient, the similarity of the two Bloom-Filters is $(2 * 8)/(10 + 11) \approx 0.762$.

In general, the similarity between two strings can be approximated by using the Dice-coefficient of their Bloom-Filter. In practice, the use of longer Bloom-Filters (500 or 1000 bits) and more hash functions (typically 15) has been found useful.

In the initial proposal, each identifier was mapped to a separate Bloom-Filter. For the use of record linkage, each identifier encoded in a Bloom-Filter could be used for computing the similarity of two records. However, if a random sample of identifiers in the population is available to the attacker, a cryptographic attack on Bloom-Filters might be successful for the most frequent names (see Kuzu et al. 2013). Therefore, the security of separate Bloom-Filter encodings had to be enhanced.

3. BLOOM-FILTER BASED PRIVACY PRESERVING RECORD LINKAGE: CRYPTOGRAPHIC LONG TERM KEYS (CLK)

If a PID is not available, the number of possible identifiers is quite limited in most administrative databases. Some administrative databases contain unique identifiers. For example, birth registries usually have their own PID, hour of birth, minute of birth, sequence number in case of twins, birth weight and Apgar-Score. But in general, these special identifiers are not available in other databases. Therefore, a key for linking must be based on those identifiers common to all administrative databases. This set is, of course, specific to local regulations, but in general the basic set of identifiers (BSID) consists of the first name, surname (at birth), sex, date of birth, country of birth and place of birth. Additional identifiers are usually not given or even more volatile than those within the BSID. A cryptographic key based on BSIDs first requires the standardization of the identifiers (uppercase, transforming special characters, removing titles and blanks etc.). As a next step, the set of unique n-grams of each identifier is formed. Numerical data such as date of birth is also treated as string and splitted into n-grams. Usually, each element of date of birth (day, month, year)

is used separately as one string variable and handled independently. Finally, this unique set of each identifier is mapped with a different number of hash-functions and a different password to the same bit-array. The resulting binary vector (typically 500-1000 elements) is a cryptographic long-term key (CLK), which can be used for linking databases (Schnell et al. 2011). One advantage of CLKs is the fact that a given bit set to 1 may be due to different identifiers. This property makes attacks much more difficult than attacks on separate Bloom-Filters.

A further increase in the difficulty of attacks can be achieved by limiting the number of bigrams per identifiers. This can be done with different methods. Additionally, random bits may be added. No successful attack on CLKs has been reported up to now. Given the way CLKs are constructed, the kind of attacks used for Bloom-Filters will not work for CLKs. Therefore, the main problem for the use of CLKs in practical applications is the size of databases used for registry-based research.

4. LINKING LARGE DATABASES WITH CLKs

Finding two very similar CLKs may also be seen as the problem of finding nearest neighbors in a high dimensional binary space. If we have a database similar to the size of a census, we have to search the nearest neighbor among more than 100 million candidates. Therefore, a direct comparison of similarity among all pairs of CLKs is practically impossible. The number of comparisons has to be reduced to the range usually considered suitable for similarity computations, such as cluster analysis. Hence, groupings of cases to smaller subsets are needed. Algorithms for generating these kind of groupings are called blocking methods.

4.1 Blocking methods for CLKs

There are a variety of blocking methods for reducing the number of record pairs which need to be compared for the use in record linkage (Christen 2012). However, in regard to data structures similar to CLKs, the choice of possible methods is more limited. This paper is limited to suitable candidates from the set of best performing methods in the comparison study of Christen (2012).

Two obvious methods are the use of external blocks and sorting. External blocks are formed by encrypting one element of the BSIDs with a different cryptographic hash function and using this code as a block. Examples for blocks are postcodes for residence, year or decade of birth or phonetic encodings of names. External blocking is the application of the widely used *Standard Blocking* (Herzog et al. 2007) to CLKs with an external encrypted key. But if the resulting block identifiers of two records of the same person differ by just one bit, external blocking will not be able to recover this pair.

The next obvious method would be sorting. Hernandez and Stolfo (1998) introduced the *Sorted Neighbourhood Method*. Both input files are pooled and sorted according to a blocking key. A window of a fixed size is then slided over the records. Two records from different input files form a candidate pair if they are covered by the window at the same time.

Canopy Clustering as suggested by McCallum et al. (2000) form candidate pairs from those records placed in the same canopy. All records from both input files are pooled. The first canopy is created by choosing a record at random from this pool. This randomly chosen record constitutes the central point of the first canopy. All records within a certain loosely defined distance l from the central point are added to the canopy. Then, the central point and any records in the canopy within a certain more closely defined distance t from the

former are removed from the record pool. Further canopies are built in the very same way as the first one until there are no more remaining records. The result is a set of potentially overlapping canopies. Pairs which can be formed from the records of the same canopy constitute the set of candidate pairs.

However, no blocking method shows optimal performance in all settings (Christen 2012). Therefore, the search for blocking methods still continues.

4.2 A new blocking method

In January 2013, I suggested the use of Multibit Trees for similarity filtering in record linkage in general without reference to privacy preserving record linkage (Bachteler et al. 2013). The method described in that paper is called *q-gram Blocking*. The basic idea of *q-gram Blocking* is the transformation of all identifiers in a standard non privacy preserving record linkage problem to a bit array like a CLK as a first step, followed by the use of a Multibit Tree to find nearest neighbors. The transformation to a CLK data structure allows the application of any method for finding nearest neighbors in high dimensional binary space to the problem of finding nearest neighbors to unencrypted nominal data (or at least data treated as nominal). Therefore, this approach can be used for blocking or similarity filtering in *all* record linkage applications.

However, the suggested searching method can also be used for blocking in privacy preserving record linkage with CLKs. If two files of CLKs are available, the so called q-gram blocking described by us consists of the query of each CLK in the smaller file within the Multibit Tree built from the CLKs of the larger file. Only CLKs in the resulting set form candidate pairs.

Before the results of a simulation are reported, some details on Multibit Trees have to be explained. Kristensen et al. (2010) introduced the Multibit Tree to search huge databases of structural information about chemical molecules. If the query vectors are all binary, we search a query $\vec{A}$ in a database of binary vectors $\vec{B}$. We want to find all records in the database with a similarity to $\vec{A}$ above a certain threshold t.

Multibit Trees work in three steps. In the first step, the vectors of the larger file are grouped into bins, which are formed by the size of the vector (denoted by $|\vec{B}|$), which here is the number of bits set to 1 in $\vec{B}$. Therefore, all bins satisfying

$$|\vec{B}| \leq t\,|\vec{A}| \;\text{ or }\; t|\vec{B}| \geq |\vec{A}| \tag{1}$$

can be ignored in the searching step, because $\frac{min(|\vec{B}|,|\vec{A}|)}{max(|\vec{B}|,|\vec{A}|)}$ constitutes an upper bound of $S_J(\vec{A},\vec{B})$.

In the second step the vectors of equal size are stored in a binary tree structure for each bin. The storing algorithm starts with all fingerprints of the bin assigned to the root node of the tree. At each parent node, the algorithm then recursively assigns the fingerprints $\vec{B}_i$ having a 0 at a fixed bit position to the left subtree and all fingerprints having a 1 at that bit position to the right subtree. The determining bit position is chosen at each parent node in the way that the tree is kept as balanced as possible. Two additional lists are stored at each node: List O contains all bit positions with constant value 0 and list I all bit positions with constant value 1 in all remaining vectors below that node. The recursion stops if the number of fingerprints at a node falls below a previously defined threshold s.

Actually, searching a vector $\vec{A}$ is done in three phases. First, all bins satisfying equation 1 are eliminated. Second, at each node of the tree currently searched, the recorded lists O and I allow the computation of an upper bound of the Jaccard similarity for all fingerprints below the current node. If the bounds fall below the similarity threshold t, all nodes below

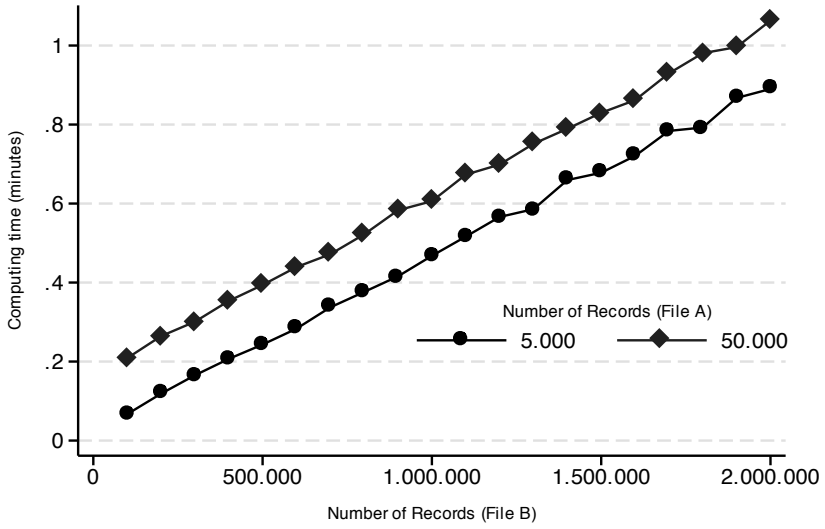

Figure 2: Computing time (in minutes) for finding 5.000 and 50.000 CLK records (File A) within a larger database (File B). Exact matches only.

this one can be eliminated from the search. Finally, the Jaccard similarity between the query vector $\vec{A}$ and all $\vec{B}_i$ at any node not previously eliminated is computed.

For the application of Multibit Trees on files of CLKs, the tree structure is built for the first file and the records of the other file are queried sequentially. Therefore, the time required for querying the records of the second file is more important for the overall running time than the time for building the tree. In general, the average query time in Multibit Trees show an almost linear increase with the number of records in the second file. This is an attractive scaling property of the method.

4.3 A simulation study of linking with CLKs

We studied the performance of Multibit Trees in a series of simulations. In the initial publication of q-gram blocking (Bachteler et al. 2013), we reported comparisons inter alia between Multibit Trees, canopy clustering, sorted neighborhood and standard blocking. In most situations, Multibit Trees outperformed the methods performing best in other comparison studies.

In subsequent simulations, longer CLKs (1320 bit, number of hash functions $k = 15$ per identificator field) of BSIDs (name, surname, sex, day/month/year of birth, place of birth, country of birth) were simulated with 5.000 and 50.000 records for the small file (A) and 100.000 to 2.000.000 records for the larger file (B). File A was a random sample of file B, but 10% of records in A were simulated with errors in the BSIDs. Figure 2 shows the computing time in minutes for exact matching CLKs by the use of Multibit Trees. Even with 50.000 records in A and 2 million records in B, the match needed only slightly more than a minute on a standard server (16GB RAM, 2· Quadcore CPU, 2.8GHZ, Ubuntu 10.0.4). Of more interest is a similarity match with a similarity threshold of .95 (see figure 3). In this test, 5.000 records are still matched in less than 4 minutes, but for matching 50.000 records to 2 million records about 27 minutes are needed.

So the computing time increases linear with the number of records in both files within the simulated range. The computing time will increase with decreasing similarity thresholds, since the number of pairwise comparisons will increase. However, the exceptional performance of Multibit Trees for this task even for large files is surprising: The running

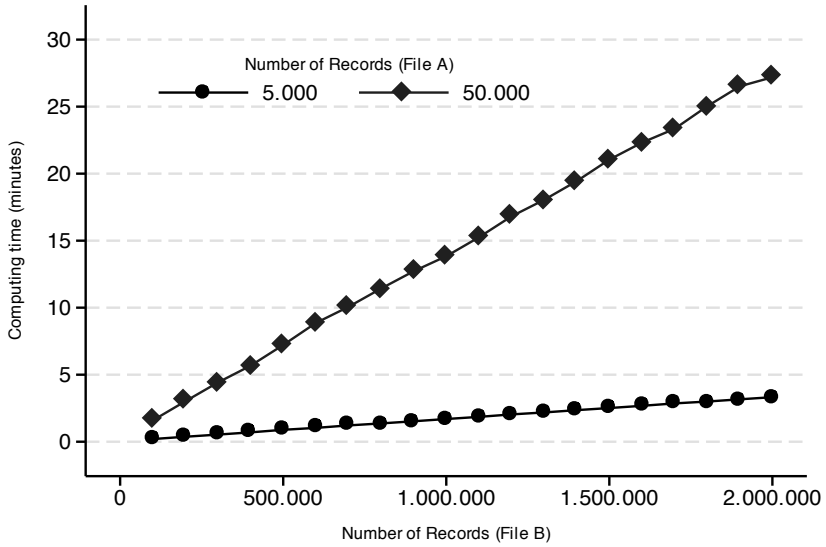

Figure 3: Computing time (in minutes) for finding 5.000 and 50.000 CLK records (File A) within a larger database (File B). Similarity threshold 0.95.

time for matching two files with 1 million CLKs each is less than 5 minutes for an exact match and for a similarity match (.95) less than 5:40 hours.

After the initial simulations had shown promising results, we implemented Multibit Trees as a R-library using C++. With this version, we observed a decrease in computing time by a factor of more than 5 (see figure 4).

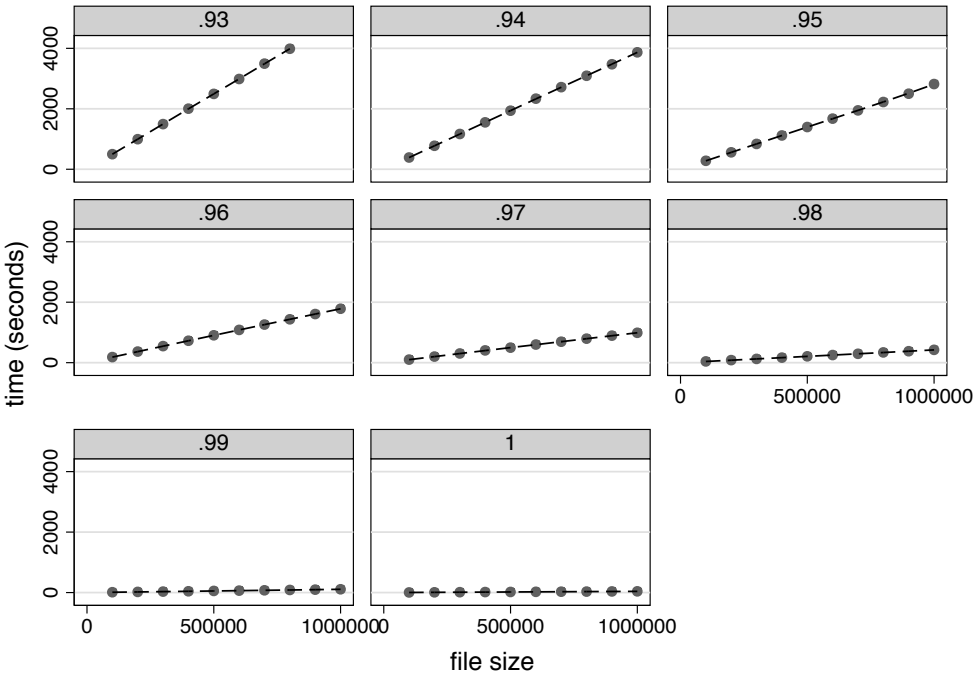

Figure 4: Query time in seconds for finding each nearest neighbor by a Multibit Tree (C++ version) in a file with 1 million records for varying file sizes (200.000 – 1.000.000) depending on the similarity threshold (.93 – 1).

For example, a comparison of two files with 1 million cases each was done without additional blocking. The tree was built within 48 seconds. The nearest neighbor was found in 40 seconds with exact matches. If a more reasonable similarity threshold of 0.95 is used the search takes about 48 minutes. Even with an unrealistic low similarity threshold of 0.9 the comparison takes just about 4 hours.

5. CONCLUSION

Privacy preserving record linkage for very large files requires blocking methods to reduce the number of comparisons. Here, the use of Multibit Trees has been suggested. Different simulations on large datasets showed superior performance compared to previously used methods even for large datasets as used in practical applications. The suggested method shows a linear increase in computing time with increasing filesize. Therefore, for most statistical applications, the speed and accuracy of Multibit trees will be more than sufficient. Only for very large datasets such as a population census with CLKs additional techniques are required. The most simple option would be external blocking. For large scale problems like census operations, an obvious external block would be year of birth. If the year of birth is encrypted with an HMAC such as MD-5 or SHA-1, the resulting code for a year of birth would form a block and within each block CLKs with Multibit Trees could be used. Given just four machines with current standard hardware and current implementations of CLKs and Multibit Trees the privacy preserving record linkages for each European census can be done within 24 hours. So for the first time, the combination of CLKs, Multibit Trees and external blocking on year of birth would allow PPRL even with datasets as large as a European Census.

References

Bachteler, T., Reiher, J., and Schnell, R. (2013), "Similarity Filtering with Multibit Trees for Record Linkage," Working Paper WP-GRLC-2013-02, German Record Linkage Center, Nuremberg.

Christen, P. (2012), "A Survey of Indexing Techniques for Scalable Record Linkage and Deduplication," *IEEE Transactions on Knowledge and Data Engineering*, 24, 1537–1555.

Hernandez, M. A. and Stolfo, S. S. (1998), "Real-world Data Is Dirty: Data Cleansing and the Merge/purge Problem," *Data Mining and Knowledge Discovery*, 2, 9–37.

Herzog, T. N., Scheuren, F. J., and Winkler, W. E. (2007), *Data Quality and Record Linkage Techniques*, New York: Springer.

Kristensen, T. G., Nielsen, J., and Pedersen, C. N. S. (2010), "A Tree-based Method for the Rapid Screening of Chemical Fingerprints," *Algorithms for Molecular Biology*, 5.

Kuzu, M., Kantarcioglu, M., Durham, E. A., Toth, C., and Malin, B. (2013), "A Practical Approach to Achieve Private Medical Record Linkage in Light of Public Resources," *Journal of the American Medical Informatics Association*, 20, 285–292.

Martin, K. M. (2012), *Everyday Cryptography. Fundamental Principles and Applications*, Oxford: Oxford University Press.

McCallum, A., Nigam, K., and Ungar, L. H. (2000), "Efficient Clustering of High-dimensional Data Sets with Application to Reference Matching," in *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining: 20–23 August 2000; Boston*, eds. Ramakrishnan, R., Stolfo, S., Bayardo, R., and Parsa, I., New York: ACM, pp. 169–178.

Schnell, R., Bachteler, T., and Reiher, J. (2009), "Privacy-preserving Record Linkage Using Bloom Filters," *BMC Medical Informatics and Decision Making*, 9, 1–11.

— (2011), "A Novel Error-Tolerant Anonymous Linking Code," Working Paper WP-GRLC-2011-02, German Record Linkage Center, Nuremberg.

Valente, P. (2010), "Census Taking in Europe: How Are Populations Counted in 2010?" *Population & Societies*, 467, 1–4.

Vatsalan, D., Christen, P., and Verykios, V. S. (2013), "A Taxonomy of Privacy-preserving Record Linkage Techniques," *Information Systems*, 38, 946–969.

Winkler, W. E. (2009), "Record Linkage," in *Handbook of Statistics Vol. 29A, Sample Surveys: Design, Methods and Applications*, eds. Pfeffermann, D. and Rao, C., Amsterdam: Elsevier, North-Holland, pp. 351–380.