Privacy Preserving Record Linkage

Rainer Schnell

German Record Linkage Center, University of Duisburg-Essen

1.9. 2014

1 Introduction

Record linkage is an increasing popular research strategy in medicine, the social sciences, commercial data mining and official statistics. Most often, the purpose of the linkage operation is the production of a dataset containing information on individual persons or organisations. With such *microdata*, two different kinds of privacy concerns aris The first problem is, that datasets containing more information on the same unit can be re-identified more easily. This problem is the central topic of *statistical disclosure control* (Hundepool et al. 2012). Since this problem is not specific to record linkage, it will not be discussed here.

The second problem concerns the attributes that can be used to identify an individual entity. In most modern societies, protecting such identifiers in administrative databases is required by law. Furthermore, releasing these identifiers for research is usually limited to a few, highly regulated special cases. Therefore, linking records of different databases by identifiers even within the same public administration may be prohibited by law. If such linkages are allowed, usually special techniques protecting the identifiers have to be used. The set of techniques for record linkage without revealing identifiers is called *Privacy Preserving Record Linkage* or *PPRL*.

Privacy preserving analysis techniques in general and specifically PPRL have become active fields of research in Computer Science, Statistics and some application fields as Epidemiology, Health Service Research and Survey Methodology. As a result of this intensive research, the list of available privacy preserving techniques is growing fast.[1] In total, the list of available privacy preserving analysis and linkage techniques is impressive. To limit the scope of chapter, the focus here will be privacy preserving techniques, which have been used on databases covering national populations to generate general purpose statistical microdata.

[1]The keyword "privacy-preserving" currently (2014) yield about 4.070 citations in CiteSeer^X, in Google Scholar about 31.800, in Pubmed about 90 and ScienceDirect about 1.200 publications.

2 Chapter outline

The number of PPRL techniques actually used for data production of research microdata is small. This chapter will shortly describe the current available techniques which are suitable for large population databases such as census datasets, social security administration data or cancer registries.

Section 3 begins with a description of the most basic and widely used standard scenario for PPRL: The use of a trusted third party with error-free identification numbers. Some refinements are described and finally the application scenario for many medical research settings is introduced: Linking decentralized databases using PPRL by a trustee for the generation of microdata files (section 3.4). The chapter focuses on methods most appropriate for such applications.

A short description of most widely discussed PPRL techniques is given in section 4. The remaining chapter concentrates on one of the most frequently used techniques: Bloom filter based PPRL.

Files beyond about 50,000 records each need special techniques (*blocking*) since not all possible pairs in a record linkage process can be computed. The set of available techniques is described, a new technique introduced and some comparisons to older techniques reported (section 5).

The central problem of PPRL is privacy protection (section 6). Therefore, currently known attacks on PPRL protocols are summarized (section 6.3). Based on this research, counter measures against this attacks are described (section 7). Finally, research needs in PPRL are delineated (section 8) and policy implementation issues mentioned (section 9).

Although some sections contain standard material covered in reviews previously published, some of the techniques and results are presented here for the first time. This new material is contained in sections 5.2, 6.3, and 7 .

3 Linking with and without Personal identification Numbers

A distinction between *(direct) identifiers* (for example, personal identification numbers) and *quasi-identifiers* (for example, names and date of birth) is useful in practice. If a one-to-one mapping of individuals and personal identification codes is possible, we call this code a direct identifier.[2]

If a unique direct identifier is available, it is most often a national identification number (or personal identification number, *PID*). In such settings, record linkage is in principle a technically a trivial file merge operation.[3] However, few countries have universal national

[2]Of course, the definition is neither clear-cut nor universally accepted. For example, the American *HIPAA/HITECH* act defines as direct identifiers of an individual 18 items, such as names, geographic subdivisions smaller than a state, including zip codes beyond the third digit, all elements of dates (except year), phone numbers, electronic addresses, social security numbers, account data, finger prints, face images and more. For details see Trinckes (2013).

[3]In practice, PIDs do contain errors (Newman & Brown 1997; Grannis et al. 2002). Most but not all PID systems have checksums, so that most simple data processing errors can be identified easily. The

PIDs (for example: Denmark, Finland, Norway, Sweden). In most other countries *quasi-identifiers* such as names, dates of birth and addresses must be used. These characteristics are neither unique, nor stable over time and often recorded with errors. Record linkage only using exact matching quasi-identifiers will therefore miss many matches. The amount of errors in quasi-identifiers is often remarkable high, if the data quality of the identifier is not important for the organizational purposes for which the data were collected. Therefore, identifiers concerning financial transactions such as bank account numbers usually have lower error rates than quasi-identifiers, where error rates often exceeds 10-15% per field. In accordance with this, Winkler (2009) reports that 25 percent of matches in a census would have been missed if only exact matching identifiers had been used. In many applications, the remaining matching subset is not a random sample of all true matches in the databases. Here the resulting missing information can not statistically be considered as missing at random.[4] An example are neonatal databases: Children of mothers with changes in their names might be different from children of mothers with no change in their names, or different from mothers whose names are missing. In such applications, additional techniques have to be used. Some of the available options will be discussed later.

3.1 Linking using a trusted third party

The basic model for PPRL is the use of a trusted third party (figure 1). Database holders A and B submit the personal identifiers and the corresponding identification codes of the persons in their database to the trustee. The trustee identifies identical persons in both databases and delivers the list of corresponding identification codes to the research group, which then links the separate databases A and B, which no longer contain personal identifiers, based on the linkage list of the trustee. Therefore only the trustee sees the identification codes and personal identifiers, whereas the research group sees only the identification code. However, the trustee also learns who is a member of which database.

3.2 Linking with encrypted PIDs

Knowledge gain of the trustee can be prevented by using encrypted identifiers. Encryption is usually done with cryptographic hash functions (*Hash Message Authentication Codes, HMACs*). These are one-way functions, so that two different inputs will be mapped to two different outcomes, but there is no way of finding the input given only the outcome. In practice, HMACs with a password are used (*keyed HMACs*). Currently, *MD-5* or *SHA-1* are most often used as HMAC (Martin 2012; Stallings 2011). In countries with PID covering the whole population, usually an ID is generated by hashing the PID with a keyed HMAC: The resulting linkage ID is specific to this linkage project and

main problems in practice are missing identifiers and multiple PIDs for the same person. Despite this, linking with PIDs is much easier than without.

[4] Identifiers seem to be missing or erroneous more often for vulnerable populations, therefore mortality may be underestimated in such linkages (Zingmond et al. 2004; Ford et al. 2006).

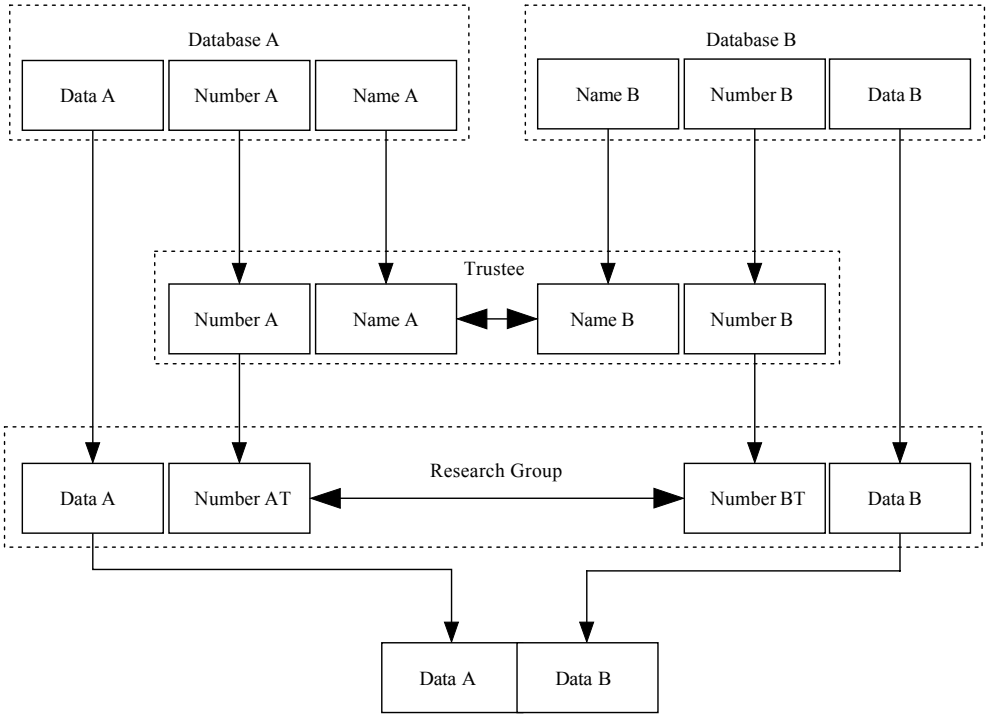

Figure 1: Standard model for linking two databases with a trusted third party. Adopted from Schnell et al. (2013).

can not be used for other linkages.

3.3 Linking with encrypted quasi-identifiers

If no universal and unique PID is available, PPRL with a trusted third party uses encrypted quasi-identifiers. These are always prone to errors, for example by typing or optical character recognition errors or by changes in values such as name changes after marriage or changes of addresses. Slightly different ways of spelling names, for example the inclusion or exclusion of academic or generational titles or other name suffixes, usually do not strongly affect record linkage procedures with unencrypted identifiers (for an example, see Randall et al. 2013). However, if the identifiers are encrypted with HMACs, even small variations will result in completely different encodings after encryption. Therefore, standardization (Christen 2012a) of quasi- identifiers before encryption is highly recommended. The standard technique to make HMAC encrypted quasi-identifiers resilient to small variations in spelling is the use of phonetic encodings for names. A phonetic code maps similarly pronounced strings to the same key, so that very similar variants of a name form a common group. For almost 100 years, the most popular phonetic function has been *Soundex* (Christen 2012a), initially developed for census operations. For example, the names Engel, Engall, Engehl, Ehngel, Ehngehl, Enngeel, Ehnkiehl and Ehenekiehl all produce the same code (in this example: E524).

For record linkage, phonetic codes are normally additionally encrypted with a cryptographic function such as SHA-1. Using such a combination of standardization, phonetic encoding and encryption of the phonetic code will result in acceptable linkage results for many applications. This combination is widely used in medical applications Borst et al. (2001). However, persons appearing with entirely different names in the two databases will be lost, therefore in practical applications fall-back strategies for potential record-pairs with non-matching phonetic-codes are commonly in use. The effect of these strategies on overall linkage performance is not well studied.

3.4 PPRL in decentralized organisations

Organisations covering nation states are likely to be decentralized. An extreme example might be the health care system in Germany. There, currently about 2000 hospitals use diverse hospital information systems without a PID or SSN. Individual financial transactions between more than 170 health care insurance companies and hospitals are based on company specific identifiers and names. Information exchange on individual patients between hospitals is usually based on name, sex, date of birth and insurance company. Only statistical information exchange is based on encrypted quasi-identifiers. For such statistical data collection processes, PPRL procedures are legally required. For example, the cancer registries use encrypted phonetic codes for PPRL.[5]

In large decentralized organizations as this, legal requirements restrict the number of usable PPRL approaches severely. Most protocols with strong privacy guarantees

[5]The actual organisation of German cancer registries is quite complex. For the technical details, see Hentschel & Katalinic (2008).

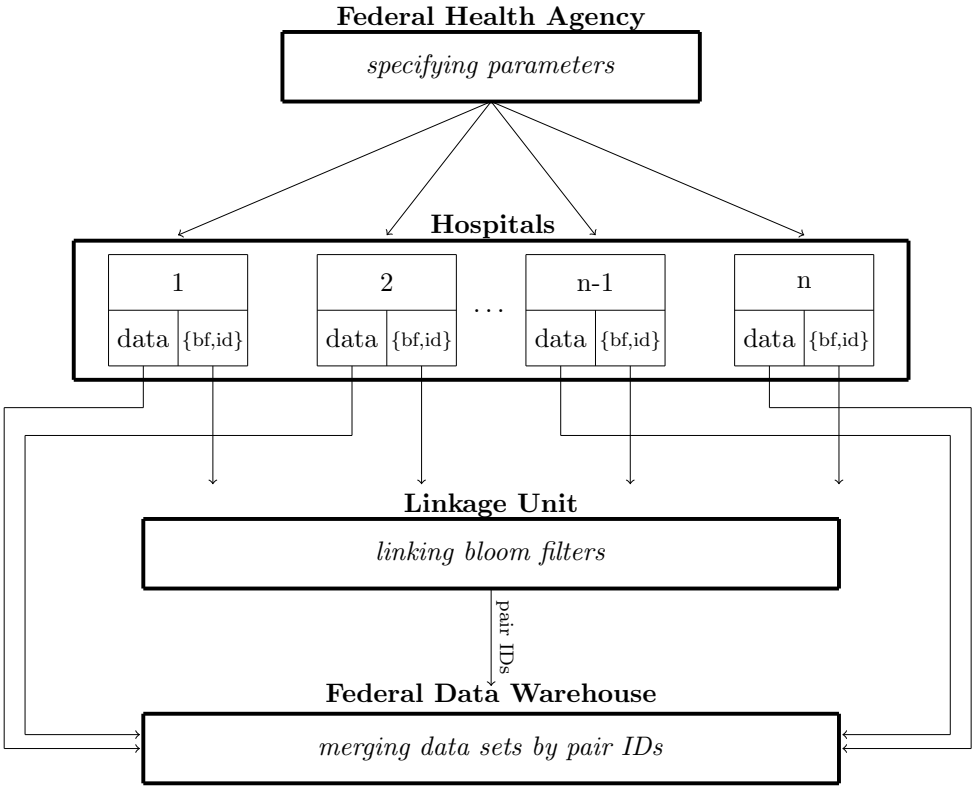

Figure 2: PPRL in a decentralized health care system

seem to require repeated interactions between database owners (for an example, see Attallah et al. 2003). Although often technically simple and even in some protocols requiring only few interactions or little network traffic, such protocols are not suited for linkage operations across different organizations. For example, medical data, crime or tax records are distributed among many different agencies for different purposes in most European countries. It is unlikely that such agencies will allow any external internet connection for their databases. Protocols requiring repeated access to external servers are no option for such secure environments.

The standard scenario in settings like these is shown in fig 2. Here, a federal agency (*FA*) will set all required parameters for a linkage operation and send them to the data holding organizations, for example hospitals. The hospitals split their databases in two sets: {IDs, data} and {IDs, quasi-identifiers}. They send the datasets to a federal data warehouse (DW). The hospitals encrypt the identifiers according to the parameters specified by the federal agency *FA*. The resulting encrypted identifiers (here denoted as bf) and the corresponding IDs are send to a linkage unit which acts as a trustee. The trustee links all records by using the bf alone. The list of all linked pairs of IDs is send to the data warehouse which merges all data sets according to the list of matching IDs.

In such setting, the agency *FA* does not receive any data, therefore it can not gain knowledge on any sensitive data. The database holders have their own data and do not receive anything new. Therefore, they don't learn anything by the execution of such protocols. The data warehouse DW has access to all non-sensitive data but to no identifiers at all. Nevertheless, in such settings the non-sensitive data could be linked to external information containing identifiers. Since this kind of attack is always possible if micro-data is available, such attacks can only be limited by statistical disclosure protection measures (Hundepool et al. 2012).

All technical interesting things happen within the linkage unit. In most security analysis in PPRL, the linkage unit acting as trustee is the attacker. Since the unit receives only encrypted identifiers, the trustee might try to re-identify the encrypted keys. Most of the relevant literature in PPRL discusses ways to prevent re-identification by the trustee. It should be noted that nearly all protocols intended for applications in PPRL assume *semi-honest* or *honest-but-curious* parties. Semi-honest parties act according to the protocol, but try to learn secret information of the other parties. They may use additional information or use information they gain during the execution of the protocol. For the scenario described in figure 2, this is widely regarded as a realistic assumption.

4 PPRL approaches

The literature on PPRL approaches is widely scattered across academic fields between Statistics, Computer Science and Cryptography. A number of reviews, tutorials and taxonomies on available techniques have been published since 2010, for example Hall & Fienberg (2010); Navarro-Arribas & Torra (2012); Christen et al. (2013); Vatsalan et al. (2013) and Verykios & Christen (2013). Therefore, only the basic approaches will be

described shortly. A large part of the recent literature on PPRL consists of combinations and extensions of these basic approaches.

4.1 Phonetic Codes

The idea of using phonetic codes for PPRL have been described above in section 3.3. For most real world settings, variations of these techniques are the standard method for PPRL. After a long neglect in research, recently new developments for linkage techniques using phonetic codes have been reported. For example, Karakasidis & Verykios (2009) and Karakasidis et al. (2012) suggested the inclusion of non-existing records to prevent frequency attacks (see section 6) on encrypted databases with phonetic codes.

For practical applications, many variants of phonetic codes have been used for record linkage in general. However, very few of them have been tested in comparison to new techniques. Recent simulations and a test with a regional cancer registry show linkage qualities which are rarely exceeded by modern techniques (Schnell et al. 2014). However, this result may be due to extensive preprocessing and needs independent replication.

4.2 High-dimensional embeddings

The idea of distance preserving mapping of strings into a high dimensional euclidean space has been suggested by Scannapieco et al. (2007). Two database holders build an embedding space from shared random reference strings and embed their strings within using a distance preserving embedding method (*SparseMap*, see Hjaltason & Samet 2003). Then the distances of the embedded strings and the reference strings are sent to a trusted third party, which computes their similarity. The protocol has been modified (Bachteler et al. 2011) and extended in many different ways, for example as double-embedding (Adly 2009) or as secure computation protocol (Yakout et al. 2009). Although the idea has generated many variants, it seems to be rarely used in practical applications.

4.3 Reference tables

A protocol based on a public list of reference strings was suggested by Pang & Hansen (2006) and Pang et al. (2009). For a given identifier, both database holders compute a distance between the identifier string and all reference strings in the set. For distances lower than a threshold, the reference string is encrypted using a key known to the database holders. For each identifier string, the resulting set of encrypted reference strings and the computed distances are sent to a third party. This party computes an approximation to the distances between the plain text identifiers.

In a simulation study Bachteler et al. (2010) compared the reference string method (Pang et al. 2009) with the high dimensional embedding (Scannapieco et al. 2007) and the Bloom filter approach (see section 4.5). The reference string method performs inferior to the other approaches. However, due to privacy considerations, the reference string method has generated a series of variants and combinations, for example the protocol suggested by Vatsalan et al. (2011).

4.4 Secure Multiparty Computations for PPRL

The field of *Secure Multiparty Computation (SMC)* protocols is concerned with joint computations of a function by a set of parties with private inputs. The parties should learn only the intended output of the protocol. Even if some of the parties collude, it should be impossible to obtain more information (Lindell & Pinkas 2009). Data mining is an obvious application for such protocols. Hence, a large number of SMC protocols to solve specific data mining problems have been suggested, for example, SMC k-means clustering Vaidya & Clifton (2003).[6] Naturally, the idea can be applied to the secure computation of similarities of quasi-identifiers. Since this is the core of a PPRL problem, there are suggestions for SMC protocols in PPRL. For example Atallah et al. (2003) proposed a SMC protocol for the computation of *edit distances*, the number of edit operations such as inserting, deleting, substituting characters to transform one string into another string. Another SMC protocol for the computation of other string distances has been suggested by Ravikumar et al. (2004). However, all currently known SMC protocols in PPRL can hardly be applied for large datasets, since the required computing time is infeasible. Furthermore, SMC protocols require network based interactions between database holders. In highly secure environments such as Social Security Administrations or medical registries, this is rarely an acceptable proposal. Therefore, in real world decentralized PPRL applications, SMC protocols are currently not used.

4.5 Bloom filter based PPRL

The idea of Bloom filter based PPRL was suggested by Schnell et al. (2009). The method is based on the idea of splitting an identifier into substrings of length 2 (*bigrams*) and mapping the bigrams by HMACs to a binary vector. Only these vectors are used for linkage.

For example, the set of bigrams of the name *SMITH* is the set[7]

$$\{\text{␣S}, SM, MI, IT, TH, \text{ H␣}\}$$

This set of bigrams is mapped with a hash function to a binary vector (see figure 3). For the mapping of each bigram Schnell et al. (2009) proposed the use of several different HMACs for each bigram. The combination of a bit-vector with hash functions is called Bloom filters in computer science (Bloom 1970). Hereby, the similarity of two strings can be computed by using the Bloom filters only.

The procedure is best explained using an example. If we want to compare the similarity of the names "Smyth" and "Smith" using bigrams. We can use a standard string similarity measure like the *Dice coefficient*

$$D_{a,b} = \frac{2h}{(|a| + |b|)}$$

[6] It should be noted, that SMC protocols are not suitable for exploratory work during the initial development of a statistical model. Since diagnostics based on residuals are essential for the development of a statistical model, SMCs are of limited use in such applications.

[7] The bigrams ␣S and H␣ are due to *padding*: start and end of a string are marked by adding a blank on the left respectively on the right end of the string.

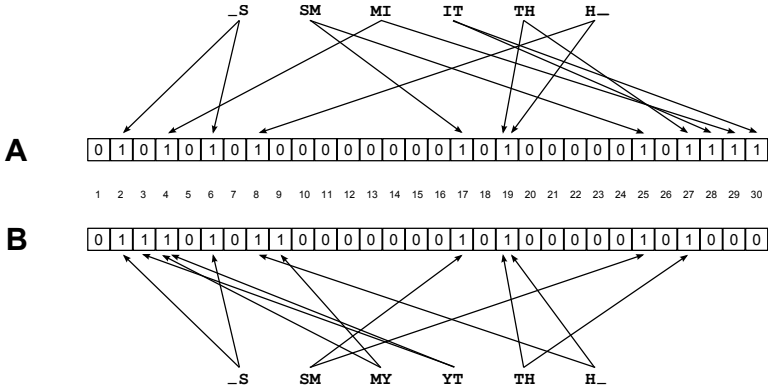

Figure 3: Example of the encoding of SMITH and SMYTH with two different cryptographic functions into 30-bit Bloom filters A and B. Taken from Schnell et al. (2009).

where h is the number of shared bigrams and $|a|$, $|b|$ is the number of bigrams in the strings a, b. For example, the bigram similarity of "Smith" and "Smyth" can be computed by splitting the names into 2 sets of 6 bigrams each ($\{ \_s, sm, mi, it, th, h\_ \}$ and $\{\_s, sm, my, yt, th, h\_\}$), counting the shared bigrams $\{\_s, sm, th, h\_\}$ and computing the Dice coefficient as $\frac{2\times 4}{(6+6)} \approx 0.67$.

If we want to compare the two strings with a Bloom filter encoding, we could for example use bigrams and Bloom filters with 30 bits and two HMACs only (in practice, Bloom filters with 500 or 1000 bits and 15 to 50 hash functions are used). Figure 3 shows the encoding for this example. In both Bloom filters 8 identical bits are set to 1. Overall $1 + 10$ bits are set to 1. Using the Dice-coefficient, the similarity of the two Bloom filters is computed as $2 * 8/(11 + 10) \approx .76$. In general, the similarity of two Bloom filters approximates the similarity of the unencrypted strings. Since numerical identifiers such as date of birth can be represented as strings, also numerical data be encoded in Bloom filters. Therefore, all available information on quasi-identifiers can be used for privacy preserving record linkage with Bloom filters.

In the initial proposal, each quasi-identifier is mapped to one Bloom filter. Therefore, this approach can be used with little or no modification with most record linkage programs. However, in some legal environments, record linkage has to be done with exactly one identifier, for example a PID may be required by law. For such settings Schnell et al. (2013) suggested the use of the encoding described above, but with one common Bloom filter (called a *cryptographic long term key* or *CLK* for all identifiers. So first name, last name, addresses, sex, date of birth etc. are all mapped to the same Bloom filter (but with different hash functions and different passwords for each identifier). The authors suggested varying the number of hash-functions k between identifiers i, for example to reflect the assumed discriminating power of identifiers for possible pairs. This can be approximated by selecting k_i proportional to the entropy H_i of the identifier.

Simulations and real world applications consistently showed consistently acceptable

performance of CLKs. For example, in an application of a German cancer registry Richter (2013), a CLK based on first name, last name, date of birth, zip-code and place of residence was used to link two files with 138,142 and 198,475 records. Considering only exact matches on the CLK, 18 pairs were found which had been not matched before using the procedures of German cancer registries (for details, see Hentschel & Katalinic 2008) and clerical editing on unencrypted names. In general, record linkage based on separately encoded Bloom filters should perform slightly better than linkages based on CLKs of the same identifiers. However, CLKs have an obvious advantage over separate identifiers: An attack on CLKs seem to be more difficult than an attack on separate Bloom filters (for details, see section 6.3). Since strings encoded in binary vectors are a flexible data structure for PPRL string similarity computations, several modifications of the basic idea have been suggested (see section 7).

Bloom filter-based record linkage has been used in real-world medical applications, such as in Brazil (Napoleão Rocha 2013), Germany (Schnell 2014) and Switzerland (Kuehni et al. 2011). The largest application so far has been an Australian study (Randall et al. 2014). Here, healthcare data with more than 26 million records have been used. PPRL with Bloom filters showed no difference in linkage quality compared with traditional probabilistic methods using fully unencrypted personal identifiers.

5 PPRL for very large databases: Blocking

Comparing each record of a file with each record of a second file requires $n * m$ comparisons. With a national database with millions of records computing the similarity of all pairs is inconceivable. In practice, record linkage is therefore usually based on comparisons within small subsets of all possible pairs. Techniques for selecting these subsets are called *blocking* or *indexing* methods (Christen 2012a;b). In most PPRL projects blocking also has to be done in a privacy preserving way (*privacy preserving blocking*). Privacy preserving blocking has become a field of research on its own. Recent proposals include Karakasidis & Verykios (2011; 2012a;b); Vatsalan et al. (2013) and Karapiperis & Verykios (2014). However, only a subset of the proposed blocking methods can be used with Bloom filter based PPRL.

5.1 Blocking for PPRL with Bloom filters

Linking two databases consisting of Bloom filters implies the search for pairs of similar binary vectors. This may be seen as a problem of finding nearest neighbors in a high dimensional binary space. In the case of population databases, a nearest neighbor among more than 100 million candidates has to be found. With regard to Bloom filter based PPRL, the number of suitable candidates for blocking is quite limited. Five promising methods will be described here.

The first obvious method is the use of external blocks. External blocks are formed by encrypting a quasi-identifier with an HMAC and using the hash value as a block. In practice, very often regional identifiers or birth cohorts are used as blocks. This kind of blocking (called *standard blocking*, Herzog et al. 2007) relies on blocking variables

without errors. Errors in block identifiers of two records for the same person will usually result in a missed link for this pair. Therefore, in practice most often different blocking variables (for example postcodes, sex, date of birth, phonetic codes of names) are used sequentially.[8]

The second obvious method is sorting. The approach introduced by Hernández & Stolfo (1998) as *Sorted Neighbourhood Method* has become the standard for handling large files with encrypted identifiers. Both input files are pooled and sorted according to a blocking key. For binary vectors, this can, for example, be the number of bits set to one (the *Hamming weight*). A window of a fixed size is then slid 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 (McCallum et al. 2000) forms candidate pairs from 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. Additional canopies are built in the same way as the first 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.

LSH-Blocking is based on *Locality Sensitive Hashing, LSH* (Indyk & Motwani 1998), a general technique for the search of approximate nearest neighbors. For the search on binary data, very often *MinHash* (Broder 1997) has been used as hash function for LSH (for details, see Leskovec et al. 2014). However, LSH for Bloom filters simply samples bits from the binary vectors and uses similar patterns of samples as blocks (Durham 2012). Many variations of the LSH-Blocking have been suggested recently (Karapiperis & Verykios 2014; Steorts et al. 2014). For the setting described in 3.4, multi party protocols or protocols requiring many iterations between database owners can not be used. However, in general the performance of simple LSH-Blocking with very large datasets seems to be disappointing (Bachteler et al. 2013).

Finally, the use of *Multibit trees* (Kristensen et al. 2010) for blocking in general record linkage, without reference to privacy preserving record linkage, was suggested recently as *q-gram blocking* by Bachteler et al. (2013). However, the method can also be used for blocking binary vectors such as Bloom filters in PPRL (Schnell 2013; 2014). With deduplicated datasets, after blocking Bloom filters with Multibit trees no further processing is needed, since the search already yields the nearest neighbor. Since the technique is new, it will be explained in more detail.

[8] This practice is common in cancer registries or during census operations. Given the ease of re-identification by unique combinations of encrypted demographic information, this is a questionable routine.

5.2 Blocking Bloom filters with Multbit trees

Multibit trees were introduced to search huge databases of structural information about chemical molecules (Kristensen et al. 2010). If the query vectors are all binary, a query A is searched in a database of binary vectors B. All records in the database with a similarity to A above a certain threshold t should be retrieved. In chemoinformatics, very often the *Tanimoto coefficient* $\frac{A \cap B}{A \cup B}$ is used for measuring similarity of binary vectors.[9] The coefficient is simply the ratio of the number of 1's the vectors have in common to the number of 1's where either vector has a 1. The algorithm uses the fact stated by Swamidass & Baldi (2007) that given the number of 1's in A and B (denoted by $|A|$ and $|B|$), the upper bound of the Tanimoto coefficient is $T_{max} = \frac{min(|A|,|B|)}{max(|A|,|B|)}$. If all vectors are stored in database indexed by $|B|$, searching for the vector A can be limited to those entries for which $|B| \leq t|A|$ or $|B| \geq |A|/t$. The increased speed of the algorithm is primarily due to those eliminations and the use of some special data structures.[10]

For the application of Multibit trees on Bloom filters, 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 needed to build the tree. In general, the average query time in Multibit trees shows a linear increase with the number of records in the second file. This is an attractive scaling property of the method.

5.3 Empirical comparison of Blocking techniques for Bloom filters

In the initial publication of q-gram blocking by Bachteler et al. (2013), comparisons inter alia between Multibit trees, canopy clustering, sorted neighborhood and standard blocking were reported. In most situations, Multibit trees outperformed the other methods, even those that had performed best in other comparison studies Christen (2012b). After the promising results of the initial simulations, the original Java implementation of the Multibit tree algorithm was implemented as an R-library using C++. With this version, computing time was further decreased by a factor of more than five.

The performance of that implementation of Multibit trees was studied in a series of simulations (Schnell 2014). For the simulation reported here, files were generated with Febrl (Christen 2008). Files with 1,000, 5,000, 10,000, 50,000, 100,000, 500,000 and 1 million records were prepared. CLKs with 1,000 bits based on name, surname, sex and day/month/year of birth as identifiers and $k = 20$ hash functions for each of the 6 fields were generated with an additional Python script. The second file was a copy of the first file, but with 0, 5, 10, 15 and 20% records containing errors.

Figure 4 shows the F-measure for Canopy Clustering (CC), Sorted Neighborhood (SN) and Multibit trees (MBT). The performance of all blocking methods decreases with increasing error rates, but more sharply for the Sorted Neighborhood technique. The results for Canopy Clustering and Multibit trees are nearly identical. In general,

[9] Both Dice and Tanimoto coefficients have a range of 0–1, they are monotone related.

[10] Details on the Multibit tree algorithm and an implementation in Java can be found in Kristensen et al. (2010).

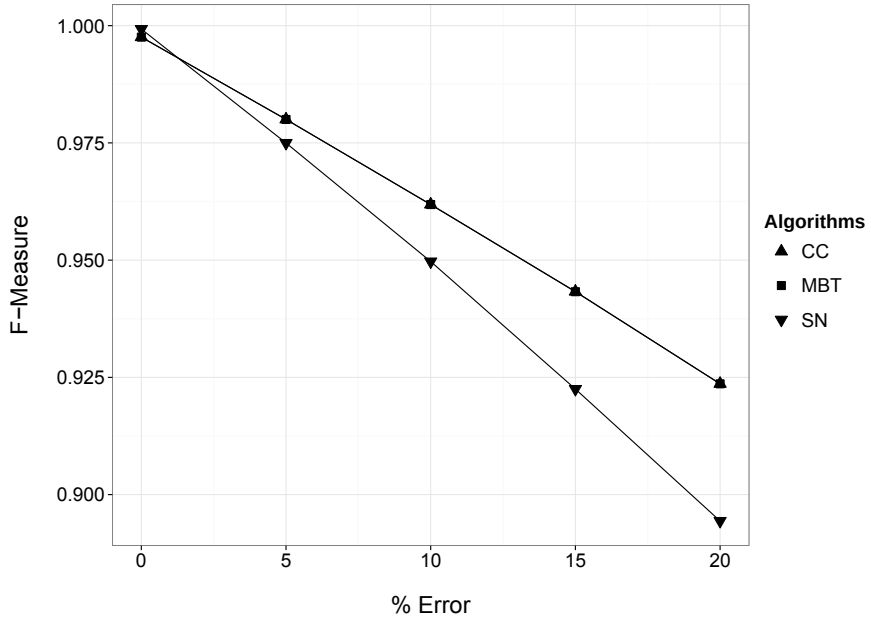

Figure 4: F- measure for Canopy Clustering (CC), Sorted Neighborhood (SN) and Multibit Trees (MBT). Errors in 5%, 10%, 15% and 20% of the records. Results for CC and MBT are nearly identical. Taken from Schnell (2014).

the precision achieved by MBT is independent of error and independent of file size. The observed recall for MBT decreases linearly with error but is independent of file size. Overall, MBT performs as well as the best known traditional technique.

It must be kept in mind that all blocking methods need careful specification of parameters, for example the window size in Sorted Neighborhood or the thresholds l and t in Canopy Clustering. For the use of Multibit trees this parameter is the Tanimoto threshold as described in section 5.2. If this threshold is selected too high, the algorithm will be very fast but will miss many possible pairs. Selecting the threshold too low will increase the computing time. Hence, the threshold has to be selected carefully. The optimal threshold depends among other parameters on the number of errors in the quasi-identifiers.

Therefore, to achieve sufficient recall, the Tanimoto threshold must be lowered with increasing number of errors. In the simulations reported in Schnell (2014), a threshold of 0.85 gives a minimum recall of 0.85 even with 20% errors. This threshold shows a minimum precision of 0.95. For most applications, the 0.85-threshold therefore seems to be a reasonable start value. If the application has higher demands on precision and recall, using a threshold of 0.8 with blocks of at most 100,000 records will increase recall beyond 0.93 despite 20% errors.

For Multibit trees the computing time will increase with decreasing similarity thresholds, since the number of pairwise comparisons will increase. However, even with a low

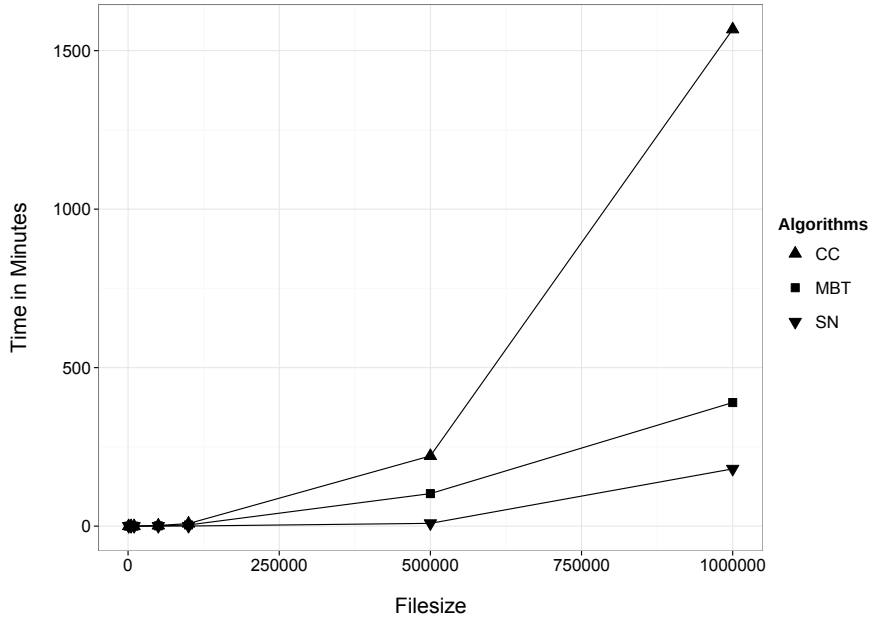

Figure 5: Time in minutes for finding best matching pairs (100,000 – 1,000,000 records) for Canopy Clustering (CC), Sorted Neighborhood (SN) and Multibit Trees (MBT), 10% errors. Taken from Schnell (2014).

similarity threshold of .85, two files of 500,000 records with 10% errors could be matched in 5902 seconds. For this combination of parameters, a recall of 0.931 and a precision of 0.977 was observed: exactly the same performance as Canopy Clustering in about 40% of the time (see figure 5).

5.4 Current recommendations for linking very large datasets with Bloom filters

Given current hardware and currently available algorithms for very large datasets such as a population census, very few techniques are suitable for Bloom filter based PPRL. The simplest option would be external blocking, for example by encrypted year of birth. This key would form a block and within each block, other techniques could be used. Sorted neighborhood is a robust option in such settings. Multibit trees would be another option. Both techniques seem to perform well. Restricting the blocks to about 1 million records, privacy preserving record linkage based on Bloomfilters could be done with a small cluster of servers within 24 hours. Given other restrictions of PPRL, this is already an acceptable time for most research applications.

6 Privacy considerations

Currently, from a mathematical point of view, it seems impossible to achieve the goal of a dataset, which can not be re-identified at all. This is attributed by Dwork & Pottenger (2013) to "...the fact that the privacy-preserving database is useful". Therefore, the current focus in computer science is the notion of *differential privacy*, usually targeting the quasi- identifiers.[11] Interestingly, the consequences for lawyers seem to be different. Ohm (2010) argues (in a chapter called "From Math to Sociology") that regulators need to ask questions that help reveal the risk of re- identification and threat of harm: "Because easy reidentification [sic!, RS] has taken away purely technological solutions that worked irrespective of these messier, human considerations, it follows that new solutions must explore, at least in part, the messiness." Correspondingly, German jurisdiction defines *de facto anonymity* as modification of identifiers such as only a disproportionate investment of time, cost and labor would lead to re-identification. Therefore, the analysis of a cryptosystem for an epidemiological cancer registry might depend on the assumed motives of an attacker. Some facts on reported privacy breaches will be summarized, before the technical details on preventing attacks will be given.

6.1 Probability of attacks

The American ITRC (`www.idtheftcenter.org`), a non-profit organization, collects information on privacy breaches in the US. According to their yearly breach report 2013 (IRTC 2014), 614 breaches with 91,982,172 exposed records were reported in the US. About 269 breaches with about 8.8 million records concerned medical or health care databases. Hence, more than 40% of all breaches pertain to medical or health care databases. In 2014, one incident alone concerned 4.5 million records in a hospital information system BBC-News (2014). In the US, medical databases seem to be attractive targets for attacks. However, it must be considered that American medical and health care databases very often contain financial information, such as credit card information. Furthermore, there may be a market for social security numbers or health care identification numbers with the aim to access medical services. For countries with national health services or differing health insurance systems, this is not necessarily true. Therefore, medical or health databases might be of lesser interest to external attacks outside the US. For example, the reports of the Federal German Data Protection Agencies for 2012 and 2013 do list a couple of incidents concerning medical databases (for example, loss of backup tapes) but not a single case of an outsider attack. In accordance with that, US data also shows that most privacy breaches are due to human error. In a study of 1046 total privacy breach incidents in the US between 2005 and 2008, Liginlal et al. (2009) reported that human error caused 67% of the incidents and malicious acts 33%. Finally, it must be kept in mind that from the attacker point of view, a social engineering attack on a human is usually easier than a mathematical attack on an encryption scheme.

Finally, the overall success probability of an attack on a medical database should be

[11] However, Dwork & Pottenger (2013) clearly states, that "...preventing re-identification does not ensure privacy ...").

considered. De-anonymization attacks on databases for research purposes have been rarely reported in the international literature. In a systematic review Emam et al. (2011) reported that only 14 attacks could be found in the literature. About 26% of the records could be re-identified in all studies. Only one of the attacks was aimed at a medical database anonymized according to current standards. In this study, 2 of 15,000 records could be correctly re-identified. Therefore, the overall probability of an attack on a research database and the success probability of such an attack seem to be entirely different from the "proof-of-concept"-attacks considered in computer science and statistics. For actual real implementations, this difference should be kept in mind (see section 9).

6.2 Kind of attacks

Recently, Vatsalan et al. (2014) classified the attacks most frequently discussed in the PPRL literature in five groups:

1. Dictionary attack: If the encryption function is known, an attacker can encrypt a large dictionary of possible quasi-identifiers to find matching encrypted identifiers.

2. Frequency attack: If a suitable population distribution of quasi-identifiers is available, the frequency distribution of encrypted quasi-identifiers can be used for assigning records to quasi-identifiers even if the encryption function is not known.

3. Cryptanalysis: Some encoding techniques, for example Bloom filters, are prone to cryptanalysis. Using the bit patterns of frequent names or frequent q-grams, some records might be correctly re-identified.

4. Composition attack: Using auxiliary information on the databases or at least some records, information might be combined to learn sensitive values of certain records.

5. Collusion: In multi-party protocols, a subset of database owners and the third party violates the protocol and try to learn the other database owners data.

Since either keyed HMACs or seeded random number generators are used for encryption, dictionary attacks on PPRL procedures don't seem promising. It should be noted, that collusion is not excluded by the honest-but-curious assumption. Most of the PPRL schemes in practical use today seem to be susceptible to collusion.[12] Therefore, in practice, the implementation of a PPRL protocol has to rely on the enforcement of the corresponding jurisdiction. Composition attacks have been defined by Ganta et al. (2008) as attacks, in which an adversary uses independent anonymized releases to breach privacy (for a textbook example with hospital data, see Chen et al. 2009). These kind of attacks have received little attention in the PPRL literature, since most protocols minimize the amount of information available for an attacker. In accordance with this reasoning, the published attacks on Bloom filter based PPRL approaches concentrate on frequency attacks and cryptanalysis.

[12] Dalenius in 1977 cited Hansen as early as 1971: "The U.S. Bureau of the Census accepts the view that '(...) it is not feasible to protect against disclosure by collusion"'.

6.3 Attacks on Bloom filters

So far, three studies by two research groups have been published on attacking PPRL Bloom filters. Furthermore, there is a recent unpublished report by one of the groups. All attacks will be described shortly.

Kuzu et al. (2011) sampled 20,000 records from a voter registration list and encrypted the bigrams of forenames (with $k = 15$ and $l = 500$). The authors formulated their attack as a *constraint satisfaction problem (CSP)* which defines a set of variables, which have to satisfy certain constraints. CSP problems can be solved by the application of constraint programming (Marriott & Stuckey 1999). Usually special programs called CSP solvers are used for this. To limit the problem size given to the CSP solver, Kuzu et al. (2011) used a frequency analysis of the identifiers from the voter registration list and of the Bloom filters encodings. For assigning possible names to the Bloom filters, generated frequency intervals for the forenames in the voting list and for the Bloom filters were applied. To restrict the number of alternative names that could correspond to a Bloom filter, two assumptions were used: The encrypted records are a random sample of a known resource and the attacker has access to the resource from which the sample is drawn. Given the described setting the CSP solver assigned the 400 most frequent names of approximately $3,500$ different forenames correctly.

However, the assumptions used by the authors are rarely found in practical applications. In their second paper, Kuzu et al. (2013) evaluated the practical use of their own attack more critically. Personal identifiers in a given application database such as a medical registry are unlikely to be a random sample of an available population list. Thus, Kuzu et al. (2013) investigated whether voter registration records could be used to identify encrypted personal identifiers from a medical center. In this more realistic scenario, the attacked database is not a random sample of the available population list. After several modifications of the attack and restricting the problem to a set of just 42 names, the CSP solver failed even after a week of runtime. Restricting the problem further to the set of only the 20 most frequent names, the CSP assigned four of those 20 names correctly. Nevertheless, Kuzu et al. (2013) concluded, that the attack remains feasible in practice. This remains to been demonstrated with an independently encrypted database. No replication of a CSP attack on Bloom filters from other research groups have been reported in the literature.

A different and highly successful attack on simple Bloom filters has been published by Niedermeyer et al. (2014). Very little computational effort is needed and only publicly available name frequency lists are used for this form of attack. The attack concentrates on the frequency of bigrams as the building blocks of Bloom filters instead of the frequency of entire names.

In the standard Bloom filters used for PPRL (Schnell et al. 2009) the k hash functions are based on the *double hashing scheme*

$$h_i = (f + i \cdot g) \bmod L \quad \text{for} \quad i = 0, \ldots, k-1$$

where L is the length of the Bloom filter and f and g cryptographic hash functions such as MD-5 or SHA-1. This was originally proposed in Kirsch & Mitzenmacher (2006) as

a fast and simple hashing method for Bloom filters yielding satisfactory performance results. Using this scheme, each bigram will set at most k bits to one, since k hash functions are used. Niedermeyer et al. (2014) denote the Bloom filter generated by one bigram as an *atom*. Then, the Bloom filter of a name is the bitwise OR operation of the atoms corresponding to the bigrams of the name.[13]

For the demonstration of the attack, the authors used 10,000 bloom filters with standard settings ($m = 1000, k = 15$). The Bloom filters were sorted and deduplicated, yielding 7,580 unique Bloom filters. For analysis, they used only those 934 Bloom filters, which occurred at least twice. Because they did not know the hash functions used, they generated every possible atom. The pairwise combination of two vectors with 1,000 bits resulted in 1,000,000 possible atoms. The next step was testing which atoms actually occurred in the 934 most frequent Bloom filters. They matched the bit positions set to one in the possible atoms with the bit positions set to one in the given Bloom filters, which resulted in 3,952 atoms.

By excluding possible atoms that do not encrypt a bigram (these atoms have a high probability to have less than k bits set to one), they only used atoms with k bits for further analyses.

After that, they determined which of the remaining 805 possible atoms are contained in each Bloom filter. Finally they assigned the three most frequent bigrams to the most frequent atoms by hand. For that, 934 Bloom filters as well as the 805 atoms were sorted according to their frequencies. By manually assigning further atoms to bigrams, they deciphered all 934 Bloom filters. In total, they found 338 bigrams (of all possible 729 bigrams) through deciphering the 934 most frequent Bloom filters. With more manual effort, nearly all bigrams could have been deciphered. Given the knowledge of nearly all bigrams, every name and even unique names could be deciphered. Therefore, the attack described in that paper can be used for the deciphering of a whole database instead of only a small subset of the most frequent names, as in reported previous research. The authors clearly state that this kind of filtering atoms is only due to the double hashing scheme used. If other hash functions would have been used, this method is not suitable for attacking Bloom filters.

Finally, in a yet unpublished paper Steinmetzer & Kroll (2014) present a fully automated attack on an encrypted database containing Bloom filters, encrypting forenames, surnames and place of birth. They suppose that the attacker only knows some publicly available lists of the most common forenames, surnames and locations. The attack is based on analyzing the frequencies and the combined occurrence of bigrams from the identifiers of these lists.

The authors generated Bloom filters with quasi-identifiers according to the distribution in the population. The identifiers were truncated after the tenth letter and padded with blanks. The bigrams of each record were hashed with $k = 20$ hash functions into one Bloom filter of length 1,000. It is important to note, that they used the double hashing scheme for the generation of k hash functions. They modified an automatic cryptanalysis of substitution ciphers as described by Jakobsen (1995). In addition to the

[13]The bitwise OR of two bits gives 1 if either or both bits are 1, otherwise it returns 0.

general approach of Niedermeyer et al. (2014) to identify *atoms*, Steinmetzer & Kroll (2014) used additional information about the joint frequency distribution of bigrams in the target language. After this modification, the algorithm was able to reconstruct 59.6% of the forenames, 73.9% of the surnames and 99.7% of the locations correctly. For 44% of the 10,000 records all identifier values were reconstructed successfully. However, the authors note that the attack is specific to the use of double hashing scheme. Furthermore, the attack described avoided encoded numerical data. Although the attack seems to be highly successful, the authors doubt that this way of attack will still be a viable option if some of the methods described in the following section are applied.

7 Hardening Bloom filters

To counteract the attacks described above, the construction of Bloom filters for PPRL can be modified. Most of the currently available options will be described here.

7.1 Randomly selected hash values

The attack described by Niedermeyer et al. (2014) was conceived for the construction scheme of the hash functions as used by Schnell et al. (2009). Therefore, this vector of attack can be eliminated by modifying the hash function. This could be done by randomly selecting hash values. If the k hash functions $h_0, \ldots, h_{k-1}$ are randomly sampled from the set of all functions that map the bigram set Σ^2 to $\{0, \ldots, 999\}$, the number of possible atoms increases. For example, given the parameters $l = 1000$ and $k = 15$, the number of candidate atoms increases from 493,975 (when using the double hashing scheme) to approximately $6.9876 \cdot 10^{32}$ when permitting arbitrary hash functions. Therefore, Niedermeyer et al. (2014) strongly suggested the use of random independent hash functions for applications. By the use of randomly selected hash values, filtering phantom atoms by selecting atoms with maximal Hamming weight would result in a smaller reduction of remaining atoms, making this way of attack unusable in practice.

7.2 Random bits

An attack on a Bloom filter encrypted quasi-identifier might use the fact that the same q-gram will be mapped to identical positions in all Bloom filters. The frequency distribution of such bit patterns might be used in different ways for an attack. For example, some patterns frequently occur together. By using information on the joint distribution of q-grams in unencrypted quasi-identifiers in combination with the joint frequency distribution of bit patterns, assignments of q-grams to bit patterns can be checked with an automated procedure. Many different ways of setting up such a system are possible, obvious candidates are generic algorithms (Haupt & Haipt 2004, for a cryptographic application, see Morelli & Walde 2003) or constrained satisfaction problems solvers (Marriott & Stuckey 1999, for an application to Bloom filters see Kuzu et al. 2011). If the constraints or optimization criteria of such automatic assignment systems depend on deterministic rules, such attacks can be made much harder, if random noise is added to

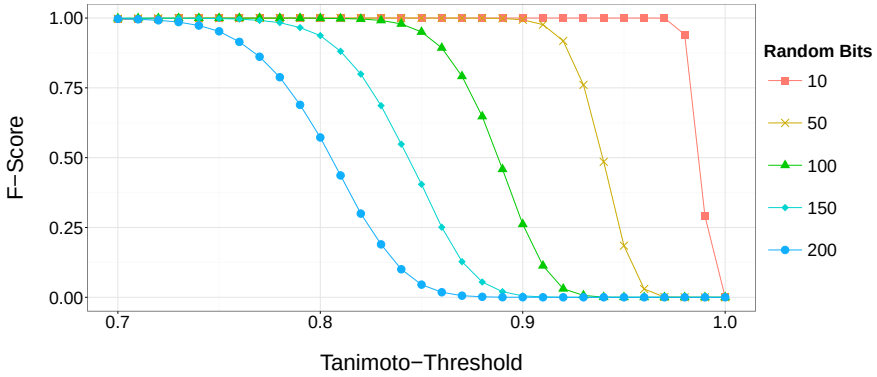

Figure 6: The effect of adding random bits to files with 10,000 Bloom filters each on the F-Score depending on the similarity threshold. Initial identifiers have no errors.

the bit-patterns. Adding random noise to encrypted information is common in computer science, see for example Rivest (1998). In PPRL, the idea of adding random bits has been mentioned by Schnell et al. (2009). The simplest option is adding noise randomly to a Bloom filter. Here bit positions are sampled randomly and set to one (*srs-1*). Alternatively, the complement of the bit is used. The random noise can be made dependent on the q-gram, so that for different q-grams different noise is added. For example, frequent q-grams can be encoded with more bits set randomly. To our knowledge, the effects of this variations on linkage quality and privacy measures have not been studied systematically.[14] As an illustration, the effect of *srs-1* on F- Scores will be described in the following.

Adding random bits will affect precision and recall of PPRL procedures. Since PPRL should be resistant to errors, adding small amounts of random bits will affect performance only slightly. However, very high rates of random error will reduce recall and precision. The amount of loss will depend on error rates in the identifiers themselves. For example, given perfect identifiers, adding even 100 random bits to a Bloom filter ($l = 1000$) shows small effects on the performance for a CLK using first name, second name, sex, date of birth and city with $k = 20$ (see figure 6, n=10,000 records). Choosing a similarity threshold of .85 would have reduced the F-Score to .95 with 100 random bits.

However, if identifiers with errors have to be used, the amount of random bits which can be added without loss in precision and recall will be lower. Repeating the experiment described above with 20% errors per identifier gives the result shown in figure 7. 100 random bits would have reduced the F-Score to about .25 given a similarity threshold of .85. Therefore, at most 50 random bits should have been used, giving a F-Score of about .93.

[14] Note that this is different from differential privacy (for an introduction, see Dwork & Pottenger 2013) for the items encoded in a Bloom filter. This problem has been studied by Alaggan et al. (2012). They show that differential privacy can be satisfied if each bit of a Bloom filter is flipped with a probability $p = 1/(1 + e^{\epsilon/k})$, where k is the number of hash functions and ϵ is the privacy parameter.

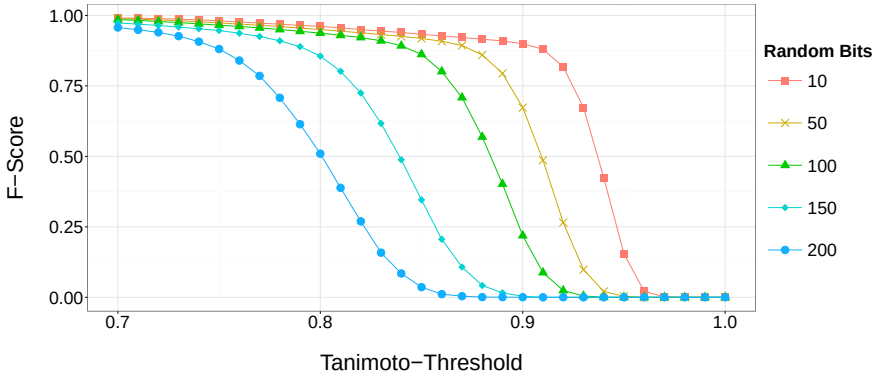

Figure 7: The effect of adding random bits to files with 10,000 Bloom filters each on the F-Score depending on the similarity threshold. 20% errors in the identifiers.

In contrast to our expectation, the amount of overlap between the two files to be linked does not show an effect on the amount of random noise which could be added (see figure 8). Even files with little overlap of records (50%) and high error rates in identifiers (20% per identifier), a similarity threshold of .85 yields a precision of 1.0 and recall of .8706.

Adding 5% random bits to bloom filters seem to affect PPRL performance only slightly. Given the fact that deterministic attacks are likely to fail with noisily Bloom filters, adding up to 5% random bits will enhance security of Bloom filter PPRL without serious losses in precision and recall.

7.3 Avoiding padding

In record linkage applications based on q-grams, adding blanks at the beginning and end of an identifier is common practice. This *padding* allows the identification of q-grams at the beginning or end of an identifier in a q-gram set. Usually, padding at least increases precision in a record-linkage process. However, padded q-grams are usually among the most frequent q-grams in identifiers. For example, among the 50 most frequent bigrams in German surnames, 18 are padded bigrams. Since frequent padded q-grams can be identified easily even after encryption, not padding identifiers is recommended for PPRL.

7.4 Standardizing the length of identifiers

The distributions of quasi-identifiers such as names are usually not uniform, but show high variations and skewness. For example, in a large German administrative database covering about 40% of the population, the length of first names varies between 2 and 24 with a mean of 9.7 (and a skewness of 0.52); the length of last names varies between 2 and 46 with a mean of 7.2 (skewness 1.9). Very long or very short names can be easily identified in a Bloom filter, since the number of bits set to one are exceptionally large or small. Therefore, to protect against this way of re-identification, the length of identifiers should be standardized. Long names could be shortened by deletion or

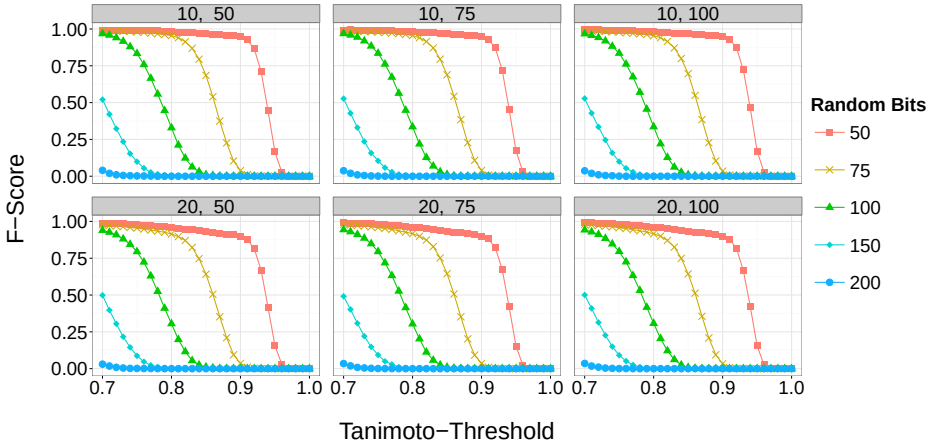

Figure 8: The effect of adding random bits to files with 10,000 Bloom filters each on the F- Score depending on the similarity threshold, errors in identifiers and different overlap. First row: 10% errors in the identifiers, second row: 20% errors in the identifiers. First column: 50% overlap, second column 75% overlap, third column 100% overlap.

sampling of q-grams. Shortening the identifiers will prevent the identification of longer names. Furthermore, the distribution of bigram frequencies will change. Short identifiers may be extended by *key stretching* (Kelsey et al. 1998). The basic idea of key stretching is the extension of a g-bit key to $g + t$ bits. Key stretching is a standard technique in cryptography, but so far has not been used in PPRL. Key stretching for PPRL can for example be implemented by random sampling $q-$grams of an extended alphabet and concatenating these with the short name. Of course, key stretching will also modify the frequency distribution of $q-$grams. Standardizing the length of identifiers will make re-identification of rare short or long names more difficult, furthermore attacks based on $q-$gram frequencies will need additional information. However, no empirical studies on the cryptographic and PPRL properties of these techniques seem to be currently available.

7.5 Sampling bits for composite Bloom filters

Sampling from identifiers can be seen as a special case of deleting characters. Therefore, instead of deleting characters, sampling can be used. Durham (2012) published a variation of CLKs, denoted by Durham et al. (2013) as *composite Bloom filters*. Bit positions from separate Bloom filters for each identifier are sampled. The initial Bloom filters are built either statically (constant values for the length of the Bloom filters and the number of hash functions) or dynamically (such that an approximately equal number of bit positions is set to one in each Bloom filter). Next, bits are sampled from these separate Bloom filters, either the same number of bits from each or proportional to the discriminatory power of each Bloom filter. Finally the sampled bits are concatenated and

permuted. Applying the attack described above on these kind of Bloom filters would not be promising, because composite Bloom filters consist of multiple identifiers, like CLKs, and their bit positions are sampled from the initial Bloom filters.

7.6 Re-hashing

To impede attacks on q-gram-encodings, the resulting bit arrays can be transformed. Obvious ideas for the transformation of bit arrays such as rotating (Schnell et al. 2011) or permutation (Durham 2012) offer no protection against attacks as described above. Therefore, other techniques are needed. As an example, a new technique will be described here. The method is based on a second hashing of the already hashed q-grams.

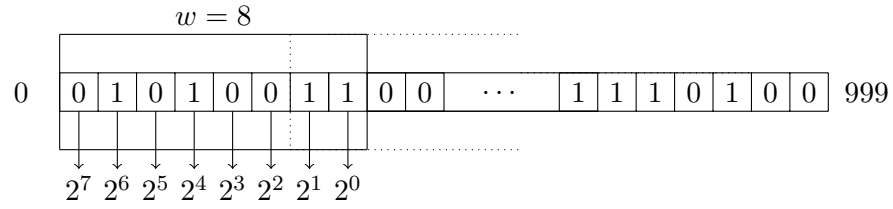

Figure 9: Re-hashing a CLK ($l = 1000$) with a window of size $w = 8$. The bits covered by the window are used as integer seed for the generation of a set of k_{re} random numbers

Consider a database of CLKs or other Bloom filters of length m equal to 1,000 bits, built from multiple identifiers. For re- hashing a CLK, we define a window of w bit positions, which slides along the CLK. The number of bit positions the window slides forward is denoted with s. The bits covered by the window are used as the binary representation of an integer. This integer serves as a seed for a random sample of k_{re} integers, which are drawn with replacement from the set $\{0, \ldots, m-1\}$. Next, a new second stage bit array is initialized with zeros. The bit positions that correspond to the random numbers are set to one. Finally, the window will shift s positions to the right and the re-hashing procedure is repeated. If the combination of window size and step size does not divide the length of the CLK, the window covers the missing number of bits from the beginning of the CLK.

The procedure will be explained using an example: Assume a CLK of length $m =$ 1,000 and a window of size $w = 8$ with step size $s = 6$ (see figure 9).Then the binary vector $v = (0, 1, 0, 1, 0, 0, 1, 1)$ of length 8, which is defined by the window, serves as pattern for the binary representation of an integer:

$$\begin{aligned} 1 \cdot 2^0 + 1 \cdot 2^1 + 0 \cdot 2^2 + 0 \cdot 2^3 + 1 \cdot 2^4 + 0 \cdot 2^5 + 1 \cdot 2^6 + 0 \cdot 2^7 \\ = 1 + 2 + 16 + 64 \\ = 83. \end{aligned}$$

Next, using 83 as the seed for a random generator, a random sample of $k_{re} = 3$ numbers from the set $\{0, \ldots 999\}$ is drawn and assigned to 83. Thus, for each binary representation

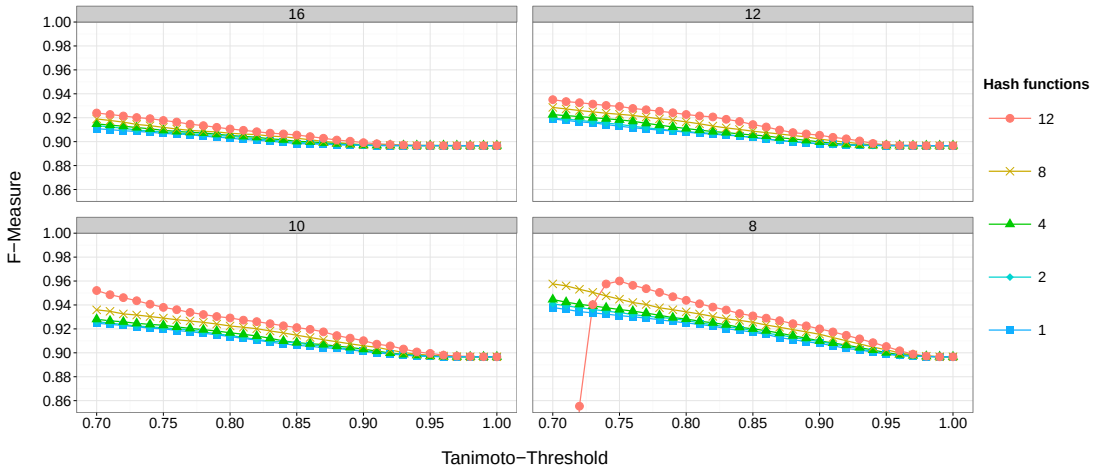

Figure 10: F-Scores for PPRL of two files with 10,000 records after re-hashing of CLKs with $k = 1, 2, 4, 8, 12$ hash functions, window size $w = 8$ and step size $s = 8, 10, 12, 16$

which produces the integer 83 the same positions in the new bit array are set to one. Finally, the window is shifted 6 bits to the right. The bits that are now covered by the window are used as binary representation of another integer. Finally, a new random sample of three numbers is assigned to this integer and the bit positions corresponding to the new sample are also set to one in the new bit array. This new second stage CLK is used like a traditional Bloom filter for PPRL.

The results of a small simulation of PPRL using two files with 10,000 records is shown in figure 10. Larger step sizes decrease the performance (at least for the given window size here). Larger number of hash functions increase performance for re-hashing. For the parameter settings used here (no additional errors, overlap of 100%), lower similarity thresholds show better results for all but one experiment. The lower right figure ($s = 8$) demonstrates a decrease in linkage quality below a similarity threshold of .75. Below this threshold, the number of false positives increases sharply; therefore the precision and accordingly the F-Score becomes unacceptable. However, high values of precision and recall can be achieved with small ($5 < s < 10$) step sizes, high numbers of hash functions ($k \geq 12$) and moderate similarity thresholds ($0.75 \leq t \leq 0.8$). Therefore, the idea of re-hashing CLKs or other Bloom filters seems to deserve further study. Since the choice of these parameters depend on m and k and the number of q-grams per identifier, the choice of optimal parameters for re-hashing is subject to ongoing research.

7.7 Salting keys with record specific data

In computer science, salt is an additional random string that is concatenated with passwords before a one-way function for hashing is used (Schneier 1996; Morris & Thompson 1979). In this application, salts are a defence against dictionary attacks. The idea can

be used for Bloom filter based PPRL as suggested by the author in Niedermeyer et al. (2014). *Salting* describes the process of generating hash values that depend on record-specific keys. Short identifiers such as date or year of birth are obvious candidates for such a key. Then, for a single bigram b appearing in two names, the same bit positions will be set to one in the corresponding Bloom filters only if the keys coincide. If the keys are not equal, it is very unlikely that all hash values for the bigram b are the same. If the keys are erroneous in one file, the records will not match, since the bit pattern is different for the same records. Therefore, the key values for salting should contain no errors at all. In practice, very few identifiers beyond date of birth are suitable candidates. If such keys can be used, salting in PPRL increases precision and does not reduce recall. The effect of salting is similar to blocking. However, the use of salting increases the entropy of encrypted identifiers. For example, the bigrams resulting from the name FELLEGI would be hashed completely different if the year of birth is used as a key and two persons with this name were born in 1959 and 1972. Therefore, filtering phantom atoms as described above would be much harder with salting. Hence, if suitable blocking variables with very low error rates are available, the values of the blocking variables should be used within each block as a salt for hashing.

7.8 Fake injections

All proposed attacks on Bloom filters strongly depend on comparing observed frequencies of bit patterns and expected frequencies of encoded entities. Names and q-grams of names show highly skewed distributions.[15] Hence, the success of these attacks can be reduced if the frequency distribution of the input is modified artificially. This could be achieved, for example, by inserting random strings which contain rare q- grams. Thus, the overall frequency distribution of hashed q- grams will be closer to a uniform distribution, which makes re- identification more difficult. Karakasidis et al. (2012) described three such fake injection methods in the context of phonetic codes. These approaches could be adopted to Bloom filters. However, all these methods have serious drawbacks.

7.9 Evaluation of Bloom filter hardening procedures

Most of the methods described here have been proposed quite recently. This is also true for the criteria used for evaluating these methods. The most elaborated contribution here by Vatsalan et al. (2014) suggested a series of standardized measures of disclosure risk DR based on $P_s(a^M)$ as *probability of suspicion*. Given n_g as the number of values in the dataset $\mathbf{G^M}$ matching a value a^M in the anonymized dataset $\mathbf{D^M}$, the probability of suspicion is $1/n_g$. Normalizing this to $0 \leq P_s(a^M) \leq 1$ and denoting with N the

[15] Fox & Lasker (1983) showed that the distribution of English surnames is highly skewed and obeys Zipf's law by a discrete Pareto distribution. The same phenomenon is observed for the q-gram frequencies.

number of records in $\mathbf{G^M}$ yields

$$P_s(a^M) = \frac{1/n_g - 1/N}{1 - 1/N}$$

This probability is primarily used for the definition of

maximum risk, DR_{max} as the maximum $P_s(a^M)$ over all values in $\mathbf{D^M}$,

marketer risk, DR_{Mark} as the number of records in $\mathbf{D^M}$, which can be re-identified with $P_s(a^M) = 1.0$,

mean risk, DR_{Mean} as the mean of all $P_s(a^M)$.

Although Vatsalan et al. (2014) presented first results for some PPRL techniques, their proposal must still be proven to be useful in practice. However, systematically testing the privacy properties of the procedures discussed above is currently the most urgent research problem within Bloom filter based PPRL. It must kept in mind that a cryptanalytic attack on a building block of a PPRL (such as as a cryptographic hash functions) can invalidate the results of computing privacy metrics. Since it is impossible to prove that information protection designs are unbreakable (Pieprzyk et al. 2003), privacy metrics are just a necessary first step for the evaluation of PPRL procedures.

8 Future Research

Progress in PPRL has been remarkable during the last 5 years. PPRL is now a very active research field in statistics, computer science and application areas such as medicine and official statistics. However, for practical applications using real world population size databases, some problems deserve further study.

Most PPRL approaches are currently based on encrypted string data. However, in some situations either geographical proximities or temporal distances are among the set of available quasi-identifiers. Despite some recent progress (Farrow 2014), using the ordinal or metric information as quasi-identifiers in PPRL is an unresolved topic up to now.

One of the most urgent problems is due to missing identifiers or quasi-identifiers. This problem has received very little attention (Ong et al. 2014). Regarding encrypted identifiers in PPRL, currently very little is known. Therefore, for this important practical problem, systematic research is lacking.

Privacy preserving blocking for datasets with about 100 million records on current hardware needs about a day of computing time. Using the most promising approaches from PPRL blocking, massive parallel or GPU implementations should be easy by comparison, since the results of blocks are in many applications independent. Although, parallel implementations of sorted neighborhood (Kolb et al. 2012) and canopy clustering (https://mahout.apache.org) have been published, practical parallel implementations of PPRL blocking for secure environments are currently not readily available.

Instead of using functions of identifiers to compute similarities, Wong & Kim (2013) suggested a protocol to compute similarity coefficients of binary vectors by exchanging only some temporary summation variables with the help of a homomorphic encryption. The authors did not intend this approach to be used in PPRL, but these building blocks can be used as part of a PPRL protocol. Although Zhang & Zhang (2012) showed that the approach of Wong & Kim (2013) has security problems, the idea of using summation variables for comparing Bloom filters is new to PPRL. A different approach for cryptographic secure Bloom filter similarity computations was introduced by Beck & Kerschbaum (2013). They encrypt every bit of a Bloom filter in a two party protocol to compute an approximation to the similarity of two Bloom filters. However, although none of these protocols meets the requirements for applications in real-world settings of medical research, they open new areas of research in PPRL.

9 PPRL research and implementation with national databases

Finally, it must be emphasized that the most important obstacles against the widespread use of PPRL techniques are not technical problems or the availability of easy to use software. More important than precision, recall and run time performance is the acceptance of the protocols by the data protection agencies supervising the data custodians according to the relevant jurisdiction. Even if trusted and tested standard techniques such as ALCs are used, getting the legal permission for a national linkage project seems to require about two years in most jurisdictions. If new techniques have to be approved, longer delays should be expected. A recent example is given by a report for the British Office for National Statistics. They considered several methods for linking administrative data within the *Beyond 2011 Programme*, but refrained from all "(...) recent innovations, such as bloom filter encryption (...)" since they "(...) have not been fully explored from an accreditation perspective" (Office for National Statistics 2013). Therefore, if new PPRL techniques are to be widely used, systematic research on cryptographic properties according to European and other international legal standards has to be done. This will require the joint effort of lawyers, computer scientists, statisticians and cryptographic experts. Coordinated efforts across disciplines and across national borders within existing research structures usually require many years. Therefore, the actual implementation of new PPRL procedures may require at least as much political action as additional research efforts.

Acknowledgement

For computational assistance, I am indebted to Christian Borgs. For the clarification of some mathematical details I have to thank Simone Steinmetzer. This research has been partly supported by the grant SCHN 586/17-1 of the German Research Foundation (DFG) awarded to author.

References

Adly, Noha, 2009: Efficient Record Linkage Using a Double Embedding Scheme. in: *Robert Stahlbock*, *Sven F. Crone* & *Stefan Lessmann* (Hg.), Proceedings of the International Conference on Data Mining: 13–16 July 2009; Las Vegas, S. 274–281. Athens, GA: CSREA Press.

Alaggan, Mohammad, *Sebastien Gambs* & *Anne-Marie Kermarrec*, 2012: BLIP: Non-interactive Differentially-Private Similarity Computation on Bloom Filters. in: *Andrea W. Richa* & *Christian Scheideler* (Hg.), Stabilization, Safety, and Security of Distributed Systems: 14th International Symposium, 2012, Toronto, Canada, October 1–4, 2012. Proceedings, S. 202–216. Berlin: Springer.

Atallah, Mikhail J., *Florian Kerschbaum* & *Wenliang Du*, 2003: Secure and Private Sequence Comparisons. in: *P. Samarati* & *P. Syverson* (Hg.), Proceedings of the ACM Workshop on Privacy in the Electronic Society: 30 October 2003; Washington, DC, S. 39–44. Washington, DC.

Bachteler, T., *J. Reiher* & *R. Schnell*, 2013: Similarity Filtering with Multibit Trees for Record Linkage. Working Paper WP-GRLC-2013-02, German Record Linkage Center, Nuremberg.

Bachteler, Tobias, *Rainer Schnell* & *Jrg Reiher*, 2010: An Empirical Comparison of Approaches to Approximate String Matching in Private Record Linkage. in: Proceedings of Statistics Canada Symposium 2010: Social Statistics: The Interplay among Censuses, Surveys and Administrative Data, S. 290–295. Ottawa: Statistics Canada.

Bachteler, Tobias, *Jörg Reiher* & *Rainer Schnell*, 2011: A Simplified Variant of a Protocol for Privacy-preserving String Comparison. Technischer Bericht 2011-01, German Record Linkage Center.

BBC-News, 2014: Community Health Systems data hack hits 4.5 million. http://www.bbc.com/news/technology-28838661.

Beck, Martin & *Florian Kerschbaum*, 2013: Approximate Two-Party Privacy-Preserving String Matching with Linear Complexity. in: IEEE International Congress on Big Data, BigData Congress 2013, June 27 2013–July 2, 2013, S. 31–37. Washington.

Bloom, Burton H., 1970: Space/Time Trade-Offs in Hash Coding with Allowable Errors. Communications of the ACM 13 (7): 422–426.

Borst, Franois, *Franois-Andr Allaert* & *Catherine Quantin*, 2001: The Swiss solution for anonymous chaining patient files. in: *V. Patel*, *R. Rogers* & *R. Haux* (Hg.), Proceedings of the 10th World Congress on Medical Informatics, S. 1239–1241. Amsterdam: IOS Press.

Broder, Andrei Z., 1997: On the Resemblance and Containment of Documents. in: Proceedings of Compression and Complexity of Sequences, Positano, Italy., S. 21–29. Washington: Institute of Electrical and Electronics Engineers.

Chen, Bee-Chung, *Daniel Kifer*, *Kristen LeFevre* & *Ashwin Machanavajjhala*, 2009: Privacy-Preserving Data Publishing. Foundations and Trends in Databases, Vol 2. Hanover: now Publishers.

Christen, Peter, 2008: Febrl: an open source data cleaning, deduplication and record linkage system with a graphical user interface. in: *Ying Li*, *Bing Liu* & *Sunita Sarawagi* (Hg.), Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, S. 1065–1068. o. O.: ACM.

Christen, Peter, 2012a: Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection. Data-Centric Systems and Applications. Berlin: Springer.

Christen, Peter, 2012b: A survey of indexing techniques for scalable record linkage and deduplication. IEEE Transactions on Knowledge and Data Engineering 24 (9): 1537–1555.

Christen, Peter, *Vassilios Verykios* & *Dinusha Vatsalan*, 2013: A Tutorial on Techniques for Scalable Privacy-preserving Record Linkage. http://cs.anu.edu.au/ Peter.Christen/cikm2013pprl-tutorial/cikm-2013-pprl-tutorial-slides.pdf.

Dalenius, T., 1977: Towards a methodology for statistical disclosure control. Statistik Tidskrift 15 (429-444): 2–1.

Durham, Elizabeth Ashley, 2012: A Framework for Accurate, Efficient Private Record Linkage. Dissertation, Nashville: Vanderbilt University.

Durham, Elizabeth Ashley, *Murat Kantarcioglu*, *Yuan Xue*, *Csaba Toth*, *Mehmet Kuzu* & *Bradley Malin*, 2013: Composite Bloom Filters for Secure Record Linkage. IEEE Transactions on Knowledge and Data Engineering 99 (preprints): 1.

Dwork, Cynthia & *Rebecca Pottenger*, 2013: Toward practicing privacy. Journal of the American Medical Informatics Association 20 (1): 102–108.

Emam, Khaled El, *Elizabeth Jonker*, *Luk Arbuckle* & *Bradley Malin*, 2011: A Systematic Review of Re-identification Attacks on Health Data. PLoS One 6 (12): e28071.

Farrow, James, 2014: Privacy Preserving Distance-Comparable Geohashing. Presentation at International Health Data Linkage Conference, IHDL.

Ford, Jane B, *Christine L Roberts* & *Lee K Taylor*, 2006: Characteristics of unmatched maternal and baby records in linked birth records and hospital discharge data. Paediatric and perinatal epidemiology 20 (4): 329–337.

Fox, W. R. & G. W. Lasker, 1983: The Distribution of Surname Frequencies. International Statistical Review 51 (1): 81–87.

Ganta, Srivatsava Ranjit, *Shiva Prasad Kasiviswanathan* & *Adam Smith*, 2008: Composition attacks and auxiliary information in data privacy. in: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, S. 265–273. o. O.: ACM.

Grannis, Shaun J, *J Marc Overhage* & *Clement J McDonald*, 2002: Analysis of identifier performance using a deterministic linkage algorithm. in: Proceedings of the AMIA Symposium, S. 305–309. o. O.: American Medical Informatics Association.

Hall, Rob & *Stephen E. Fienberg*, 2010: Privacy-Preserving Record Linkage. S. 269–283 in: *Josep Domingo-Ferrer* & *Emmanouil Magkos* (Hg.), Privacy in Statistical Databases. Lecture Notes in Computer Science, Nr. 6344. o. O.: Springer Berlin Heidelberg.

Haupt, Randy L. & *Sue Ellen Haupt*, 2004: Practical genetic algorithms. Hoboken: Wiley.

Hentschel, Stefan & *Alexander Katalinic* (Hg.), 2008: Das Krebsregister-Manual der Gesellschaft der epidemiologischen Krebsregister in Deutschland e.V. München: Zuckerschwerdt Verlag.

Hernández, Mauricio A. & *Salvatore S. Stolfo*, 1998: Real-world data is dirty: data cleansing and the merge/purge problem. Data Mining and Knowledge Discovery 2 (1): 9–37.

Herzog, Thomas N., *Fritz J. Scheuren* & *William E. Winkler*, 2007: Data Quality and Record Linkage Techniques. New York: Springer.

Hjaltason, Gisli R. & *Hanan Samet*, 2003: Properties of Embedding Methods for Similarity Searching in Metric Spaces. IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (5): 530–549.

Hundepool, Anco, *Josep Domingo-Ferrer*, *Luisa Franconi*, *Sarah Giessing*, *Eric Schulte Nordholt*, *Keith Spicer* & *Peter-Paul de Wolf*, 2012: Statistical Disclosure Control. Chichester: Wiley.

Indyk, Piotr & *Rajeev Motwani*, 1998: Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. in: *Jeffrey Scott Vitter* (Hg.), Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998, S. 604–613. o. O.: ACM.

ITRC, 2014: 2013 Data Breach Category Summary. Technischer Bericht, Identity Theft Resource Center, San Diego. www.idtheftcenter.org.

Jakobsen, T., 1995: A fast method for the cryptanalysis of substitution ciphers. Cryptologia 19 (3): 265–274.

Karakasidis, A. & V. S. Verykios, 2009: Privacy Preserving Record Linkage Using Phonetic Codes. in: Informatics, 2009. BCI '09. Fourth Balkan Conference in, S. 101–106.

Karakasidis, A & V.S. Verykios, 2012a: A Sorted Neighborhood Approach to Multidimensional Privacy Preserving Blocking. in: 2012 IEEE 12th International Conference on Data Mining Workshops (ICDMW), S. 937–944.

Karakasidis, A., V. S. Verykios & P. Christen, 2012: Fake Injection Strategies for Private Phonetic Matching. S. 9–24 in: *J. Garcia-Alfaro*, *G. Navarro-Arribas*, *N. Cuppens-Boulahia & S. de Capitani di Vimercati* (Hg.), Data Privacy Management and Autonomous Spontaneus Security. Lecture Notes in Computer Science. Bd. 7122. Berlin: Springer.

Karakasidis, Alexandros & Vassilios S Verykios, 2011: Secure Blocking + Secure Matching= Secure Record Linkage. Journal of Computing Science and Engineering 5 (3): 223–235.

Karakasidis, Alexandros & Vassilios S. Verykios, 2012b: Reference Table Based k-anonymous Private Blocking. in: Proceedings of the 27th Annual ACM Symposium on Applied Computing, S. 859–864. o. O.: ACM.

Karapiperis, Dimitrios & Vassilios S. Verykios, 2014: A distributed near-optimal LSH-based framework for privacy-preserving record linkage. Computer Science and Information Systems 11 (2): 745–763.

Kelsey, John, *Bruce Schneier*, *Chris Hall & David Wagner*, 1998: Secure applications of low-entropy keys. S. 121–134 in: *Eiji Okamoto*, *George Davida & Masahiro Mambo* (Hg.), Information Security. o. O.: Springer.

Kirsch, Adam & Michael Mitzenmacher, 2006: Less Hashing, Same Performance: Building a Better Bloom Filter. in: *Yossi Azar & Thomas Erlebach* (Hg.), Proceedings of the 14th Annual European Symposium on Algorithms, Zurich, Switzerland, September 11–13, S. 456–467. Springer: Heidelberg.

Kolb, Lars, *Andreas Thor & Erhard Rahm*, 2012: Multi-pass Sorted Neighborhood Blocking with MapReduce. Computer Science - Research and Development 27 (1): 45–63. http://dx.doi.org/10.1007/s00450-011-0177-x.

Kristensen, Thomas G., *Jesper Nielsen & Christian N. S. Pedersen*, 2010: A tree-based method for the rapid screening of chemical fingerprints. Algorithms for Molecular Biology 5 (9).

Kuehni, C. E., *C. S. Rueegg*, *G. Michel*, *C. E. Rebholz*, *M.-P. F. Strippoli*, *F. K. Niggli*, *M. Egger & N. X. von der Weid*, 2011: Cohort Profile: The Swiss Childhood Cancer Survivor Study. International Journal of Epidemiology S. 1–12.

Kuzu, Mehmet, *Murat Kantarcioglu*, *Elizabeth Durham & Bradley Malin*, 2011: A Constraint Satisfaction Cryptanalysis of Bloom Filters in Private Record Linkage. in: *S. Fischer-Hübner & N. Hopper* (Hg.), The 11th Privacy Enhancing Technologies Symposium, S. 226–245. Berlin: Springer.

Kuzu, Mehmet, *Murat Kantarcioglu*, *Elizabeth Ashley Durham*, *Csaba Toth & Bradley Malin*, 2013: A Practical Approach to Achieve Private Medical Record Linkage in Light of Public Resources. Journal of the American Medical Informatics Association 20 (2): 285–292.

Leskovec, Jure, *Anand Rajaraman & Jeffrey D. Ullman*, 2014: Mining of Massive Datasets. www.mmds.org.

Liginlal, Divakaran, *Inkook Sim & Lara Khansa*, 2009: How significant is human error as a cause of privacy breaches? An empirical study and a framework for error management. computers & security 28 (3): 215–228.

Lindell, Yehuda & Benny Pinkas, 2009: Secure Multiparty Computation for Privacy-Preserving Data Mining. Journal of Privacy and Confidentiality 1 (1). http://repository.cmu.edu/jpc/vol1/iss1/5.

Marriott, Kim & Peter J. Stuckey, 1999: Programming with Constraints: an Introduction. Cambridge: The MIT Press.

Martin, Keith M., 2012: Everyday Cryptography. Fundamental Principles and Applications. Oxford: Oxford University Press.

McCallum, Andrew, *Kamal Nigam & Lyle H. Ungar*, 2000: Efficient clustering of high-dimensional data sets with application to reference matching. in: *Raghu Ramakrishnan*, *Sal Stolfo*, *Roberto Bayardo & Ismail Parsa* (Hg.), Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining: 20–23 August 2000; Boston, S. 169–178. New York: ACM.

Morelli, Ralph & Ralph Walde, 2003: A Word-Based Genetic Algorithm for Cryptanalysis of Short Cryptograms. in: *Ingrid Russell & Susan M. Haller* (Hg.), FLAIRS Conference, S. 229–233. o. O.: AAAI Press.

Morris, Robert & Ken Thompson, 1979: Password security: A case history. Communications of the ACM 22 (11): 594–597.

Napoleão Rocha, M. C., 2013: Vigilância dos óbitos Registrados com Causa Básica Hanseníase. Master Thesis, Brasilia: Universidade de Brasília.

Navarro-Arribas, Guillermo & Vicenc Torra, 2012: Information Fusion in Data Privacy: a Survey. Information Fusion 13 (4): 235–244.

Newman, Thomas B & Andrew N Brown, 1997: Use of commercial record linkage software and vital statistics to identify patient deaths. Journal of the American Medical Informatics Association 4 (3): 233–237.

Niedermeyer, Frank, *Simone Steinmetzer*, *Martin Kroll* & *Rainer Schnell*, 2014: Cryptanalysis of basic Bloom filters used for Privacy Preserving Record Linkage. Working Paper 2014-4, German Record Linkage Center, Nuremberg.

Office for National Statistics, 2013: Beyond 2011: Matching Anonymous Data. Methods & Policies M9, ONS, London.

Ohm, Paul, 2010: Broken Promises of Privacy: Responding to the Surprising Failure of Anonymization. UCLA Law Review 57: 1701–1777.

Ong, Toan C, *Michael V Mannino*, *Lisa M Schilling* & *Michael G Kahn*, 2014: Improving Record Linkage Performance in the Presence of Missing Linkage Data. Journal of biomedical informatics S. preprint.

Pang, Chaoyi & *David Hansen*, 2006: Improved Record Linkage for Encrypted Identifying Data. in: *Johanna Westbrook*, *J. Callen*, *G. Margelis* & *J. Warren* (Hg.), HIC 2006 and HINZ 2006: Proceedings, S. 164–168. Brunswick East, Vic.: Health Informatics Society of Australia.

Pang, Chaoyi, *Lifang Gu*, *David Hansen* & *Anthony Maeder*, 2009: Privacy-Preserving Fuzzy Matching Using a Public Reference Table. S. 71–89 in: *Sally McClean*, *Peter Millard*, *Elia El-Darzi* & *Chris Nugent* (Hg.), Intelligent Patient Management. Berlin: Springer.

Pieprzyk, Josef, *Thomas Hardjono* & *Jennifer Seberry*, 2003: Fundamentals of Computer Security. Berlin: Springer.

Randall, Sean M, *Anna M Ferrante*, *James H Boyd* & *James B Semmens*, 2013: The effect of data cleaning on record linkage quality. BMC medical informatics and decision making 13: 64.

Randall, Sean M, *Anna M Ferrante*, *James H Boyd*, *Jacqueline K Bauer* & *James B Semmens*, 2014: Privacy-preserving record linkage on large real world datasets. Journal of Biomedical Informatics 50: 205–212.

Ravikumar, Pradeep, *William W Cohen* & *Stephen E Fienberg*, 2004: A secure protocol for computing string distance metrics. in: Proceedings the Workshop on Privacy and Security Aspects of Data Mining at the International Conference on Data Mining, S. 40–46.

Richter, Anke, 2013: Ergebnis des Abgleichs Nr. 2. Memo, Cancer Registry Lübeck.

Rivest, Ronald R., 1998: Chaffing and Winnowing: Confidentiality without Encryption. http://theory.lcs.mit.edu/ rivest/chaffing.txt.

Scannapieco, Monica, *Ilya Figoti*, *Elisa Bertino* & *Ahmed Elmagarmid*, 2007: Privacy Preserving Schema and Data Matching. in: Proceedings of the SIGMOD Conference, S. 653–664. New York: ACM.

Schneier, Bruce, 1996: Applied Cryptography: Protocols, Algorithms, and Source Code in C. New York: Wiley.

Schnell, R., 2013: Privacy-Preserving Record Linkage and Privacy-Preserving Blocking for Large Files with Cryptographic Keys using Multibit Trees. in: Proceedings of the Joint Statistical Meetings, American Statistical Association, S. 187–194. o. O.: American Statistical Association.

Schnell, R., 2014: An efficient privacy-preserving record linkage technique for administrative data and censuses. Statistical Journal of the IAOS S. in print.

Schnell, R., *Paul B. Hill* & *Elke Esser*, 2013: Methoden der empirischen Sozialforschung. Munich: Oldenbourg.

Schnell, R., *A. Richter* & *C. Borgs*, 2014: Performance of different methods for privacy preserving record linkage with large scale medical data sets. Presentation at International Health Data Linkage Conference, IHDL.

Schnell, Rainer, *Tobias Bachteler* & *Jörg Reiher*, 2009: Privacy-preserving Record Linkage Using Bloom Filters. BMC Medical Informatics and Decision Making 9 (41): 1–11.

Schnell, Rainer, *Tobias Bachteler* & *Jörg Reiher*, 2011: A Novel Error-Tolerant Anonymous Linking Code. Working Paper WP-GRLC-2011-02, German Record Linkage Center, Nuremberg.

Stallings, William, 2011: Cryptography and Network Security: Principles and Practice. Boston: Prentice Hall.

Steinmetzer, Simone & *Martin Kroll*, 2014: Automated Cryptanalysis of Bloom Filter Encryptions of Health Data. Duisburg.

Steorts, Rebecca C., *Samuel L. Ventura*, *Mauricio Sadinle* & *Stephen E. Fienberg*, 2014: A Comparison of Blocking Methods for Record Linkage.

Swamidass, S. Joshua & *Pierre Baldi*, 2007: Bounds and Algorithms for Fast Exact Searches of Chemical Fingerprints in Linear and Sublinear Time. Journal of Chemical Information and Modelling 47: 302–317.

Trinckes, John J., 2013: The Definitive Guide to Complying with the HIPAA/HITECH Privacy and Security Rules. Boca Raton: CRC Press.

Vaidya, Jaideep & *Chris Clifton*, 2003: Privacy-preserving k-means clustering over vertically partitioned data. in: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, S. 206–215. o. O.: ACM.

Vatsalan, Dinusha, *Peter Christen* & *Vassilios S. Verykios*, 2011: An Efficient Two-Party Protocol for Approximate Matching in Private Record Linkage. in: Proceedings

of the Ninth Australasian Data Mining Conference-Volume 121, S. 125–136. o. O.: Australian Computer Society.

Vatsalan, Dinusha, *Peter Christen* & *Vassilios S Verykios*, 2013: Efficient two-party private blocking based on sorted nearest neighborhood clustering. in: Proceedings of the 22nd ACM international conference on Conference on information & knowledge management, S. 1949–1958. o. O.: ACM.

Vatsalan, Dinusha, *Peter Christen*, *Christine O'Keefe* & *Vassilios S. Verykios*, 2014: An Evaluation Framework for Privacy-preserving Record Linkage. Journal of Privacy and Confidentiality 6 (1): 35–75.

Verykios, Vassilios S. & *Peter Christen*, 2013: Privacy-preserving Record Linkage. WIREs Data Mining and Knowledge Discovery, (3): 321–332.

Winkler, William E., 2009: Record Linkage. S. 351–380 in: *Danny Pfeffermann* & *C.R. Rao* (Hg.), Handbook of Statistics Vol. 29A, Sample Surveys: Design, Methods and Applications. Amsterdam: Elsevier, North-Holland.

Wong, Kok-Seng & *Myung Ho Kim*, 2013: Privacy-preserving similarity coefficients for binary data. Computers & Mathematics with Applications 65 (9): 1280–1290.

Yakout, Mohamed, *Mikhail J. Atallah* & *Ahmed Elmagarmid*, 2009: Efficient Private Record Linkage. in: Proceedings of the 25th International Conference on Data Engineering, S. 1283–1286. o. O.: IEEE.

Zhang, Bo & *Fangguo Zhang*, 2012: Secure similarity coefficients computation with malicious adversaries. in: Intelligent Networking and Collaborative Systems (INCoS), 2012 4th International Conference on, S. 303–310. o. O.: IEEE.

Zingmond, David S, *Zhishen Ye*, *Susan L Ettner* & *Honghu Liu*, 2004: Linking hospital discharge and death records – accuracy and sources of bias. Journal of Clinical Epidemiology 57 (1): 21–29.