Evaluation of Hardening Techniques for Privacy-Preserving Record Linkage

Martin Franke
University of Leipzig
Germany
franke@informatik.uni-leipzig.de

Ziad Sehili
University of Leipzig
Germany
sehili@informatik.uni-leipzig.de

Florens Rohde
University of Leipzig
Germany
rohde@informatik.uni-leipzig.de

Erhard Rahm
University of Leipzig
Germany
rahm@informatik.uni-leipzig.de

ABSTRACT

Privacy-preserving record linkage aims at integrating person-related data from different sources while protecting the privacy of individuals by securely encoding and matching quasi-identifying attributes, like names. For this purpose Bloom-filter-based encodings have been frequently used in both research and practical applications. Simultaneously, however, weaknesses and attack scenarios were identified emphasizing that Bloom filters are in principal susceptible to cryptanalysis. To counteract such attacks, various encoding variants and tweaks, also known as hardening techniques, have been proposed. Usually, these techniques bear a trade-off between privacy (security) and the linkage quality outcome. Currently, a comprehensive evaluation of the suggested hardening methods is not available. In this work, we will therefore review and categorize available Bloom-filter-based encoding schemes and hardening techniques. We also comprehensively evaluate the approaches in terms of privacy (security) and linkage quality to assess their practicability and their effectiveness in counteracting attacks.

1 INTRODUCTION

Linking records from different independent sources is an essential task in research, administration and business to facilitate advanced data analysis [6]. In many applications, these records are about individuals and thus contain sensitive information, e. g., personal, health, criminal or financial information. Due to several laws and regulations, data holders have to protect the privacy of individuals [33]. As a consequence, data holders have to ensure that no sensitive or confidential information is revealed during a linkage process.

Privacy-preserving record linkage (PPRL) addresses this problem by providing techniques for linking records referring to the same real-world entity while protecting the privacy of these entities. In contrast to traditional record linkage [6], PPRL encodes sensitive identifying attributes, also known as quasi-identifiers, for instance, names, date of birth or addresses, and then conduct the linkage on the encoded attribute values.

Over the last years, numerous PPRL approaches have been published [33]. However, many approaches are not suited for real-world applications as they either are not able to sufficiently handle dirty data, i. e., erroneous, outdated or missing values, or do not scale to larger datasets. More recent work mainly focuses on encoding techniques utilizing Bloom filters [2] as error-tolerant and privacy-preserving method to encode records containing sensitive information. While Bloom-filter-based encodings have become the quasi-standard in PPRL approaches, several studies analyzed weaknesses and implemented successful attacks on Bloom filters [7, 8, 18, 20, 21, 24]. In general, it was observed that Bloom filters carry a non-negligible re-identification risk because they are vulnerable to frequency-based cryptanalysis. In order to prevent such attacks, various Bloom filter hardening techniques were proposed [7, 25]. Such techniques aim at reducing patterns and frequency information that can be obtained by analyzing the frequency of individual Bloom filters or (co-occurring) 1-bits.

Previous studies on Bloom filter hardening techniques only consider individual methods and do not analyze the effects of combining different approaches. Moreover, many of the proposed hardening techniques have received only limited evaluation on small synthetic datasets making it hard to assess the possible effects on the linkage quality.

The aim of this work is to review hardening techniques proposed in the literature and to evaluate their effectiveness in terms of achieving high privacy (security) and linkage quality.

In particular, we make the following contributions:

- We survey Bloom filter variants and hardening techniques that have been proposed for use in PPRL scenarios to allow secure encoding and matching of sensitive person-related data.
- We categorize existing hardening techniques to generalize the Bloom filter encoding process and thus highlight the different possibilities for building tailored Bloom filter encodings that meet the privacy requirements of individual application scenarios.
- We explore additional variants of hardening techniques, in particular salting utilizing blocking approaches and attribute-specific salting on groups of attributes.
- We propose and analyze measures that allow us to quantify the privacy properties of different Bloom filter variants.
- We comprehensively evaluate different Bloom filter variants and hardening techniques in terms of privacy (security) and linkage quality using two real-world datasets containing typical errors and inconsistencies.

2 BLOOM FILTER

The use of Bloom filters [2] for PPRL has been proposed by Schnell and colleagues [26] and has become the quasi-standard for recent PPRL approaches in both research and real applications [33].

2.1 Basics

A Bloom filter (BF) is a space-efficient probabilistic data structure for representing a set $E = \{e_1, \ldots, e_n\}$ of n elements or features and testing set membership. Therefore, a bit vector of fixed size m is allocated and initially all bits are set to zero. A set of $\mathbf{k}$ hash functions is selected where each function $H_1, \ldots, H_k$ outputs a value in $[0, m-1]$. To represent the set E in the BF, each element is (hash) mapped to the BF by using each of the k hash functions and setting the bits at the resulting positions to one.

To check the membership of an element, the same hash functions are calculated and the bits at the resulting positions are checked. If all bits are set to one, the element probably is in the set. Due to collision, i. e., two or more elements may set the same bit position for the same or different hash functions, BFs have a false-positive probability that is $\text{fpr} = (1 - e^{\frac{k \cdot n}{m}})^k$ [3]. On the other hand, if at least one bit is zero, the element is definitively not in the set.

Union and intersection of BFs with the same size and set of hash functions can be implemented with bit-wise OR and AND operations respectively. While the union operation is lossless, i. e., the resulting BF will be equal to a BF that was build using the union of the two sets, the intersection operation produces a BF that may have a larger false-positive probability [3].

By using BF union and intersection, set-based similarity measures can be used to calculate the similarity of two BFs. Here the **Jaccard coefficient** is frequently used as a similarity measure. Given two BFs x, y the Jaccard coefficient is defined as

$$J(x, y) = \frac{|x \cap y|}{|x \cup y|} = \frac{|x \text{ AND } y|}{|x \text{ OR } y|}$$

For instance, given the BFs $x = [\mathbf{1}0011001]$ and $y = [\mathbf{0}0011001]$ the Jaccard coefficient is $3/4$. The BF similarity is an approximation of the similarity of the underlying (represented) sets.

2.2 Utilization in PPRL

The main idea for utilizing BFs in PPRL scenarios is to use a BF to represent the records attribute values, i. e., all quasi-identifying attributes of a person that are relevant for linkage, e. g., first name, last name and date of birth. The BFs hash functions need to be cryptographic (one-way) hash functions that are keyed (seeded) with a secret key $\mathcal{S}$, i. e., keyed-hash message authentication codes (HMACs) like MD5 or SHA-1 [23]. For approximate matching, the granularity of the record attributes is increased by segmentation into features. A widely used approach is to split the attribute values into small substrings of length q, called **q-grams**, typically setting $1 \leq q \leq 4$. Consequently, in PPRL a BF represents a set of attribute value segments, that we term record (attribute) features. Thus, the number of common 1-bits of two BFs approximates the number of common (overlapping) features between two records.

2.2.1 Types. There are two ways of encoding records into BFs: either one BF is built for each record attribute, which is known as field- or **attribute-level BF**, or a single BF is built for all relevant attributes, which is known as **record-level BF**. For constructing record-level BFs there are two approaches: The first approach [27] builds a single BF in which all record attributes are hashed. The second approach [9] first constructs field-level BFs and then selects bits from these individual BFs according to the weight of the respective attribute. In this work, we will focus on the first approach since it is heavily used in literature and practice [33]. We illustrate the basic BF building process in Fig. 1. By using

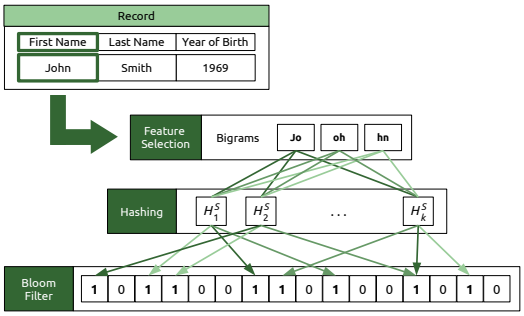


Figure 1: Basic Bloom filter building process.

the first name attribute the figure shows how an attribute value (*'John'*) is segmented into q-gram segments (here $q = 2$), which are then mapped to the bit vector using the k hash functions. Other attributes are mapped in the same way, although a different segmentation strategy can be used. The advantage of using field-level BFs is that individual BFs are produced allowing the use of sophisticated matching techniques known from traditional record linkage, e. g., classification based on attribute weights and attribute error rates, as well as approaches for handling composite fields, for instance, name attributes with compounds (multiple given names). However, as we discuss in the next section, field-level BFs fulfill much weaker privacy properties compared to record-level BFs.

2.2.2 Privacy Properties. The privacy-preserving property of BFs rely on the following aspects:

(1) An adversary has no information on how the record features are obtained, e. g., selected attributes or length of substrings (q-grams).
(2) The selected hash functions, the secret key $\mathcal{S}$ and thus the hash mapping of record features to bit positions is unknown to an adversary. In particular, the use of keyed hash functions is essential to prevent dictionary attacks.
(3) Due to collisions multiple record features will map to a single bit position in general. Keeping the BF size m fixed, the more hash functions are used and the more features are mapped to the BF, the higher will be the number of collisions and thus the confusion.
(4) There is no coherence or positional information: Since a BF encodes a *set* of record features, it is not obvious from where features were obtained, i. e., within an attribute and for record-level BF even from which attribute.

However, BFs are susceptible to frequency attacks as the frequencies of set bit positions correspond to the frequencies of record features. Thus, frequently (co-)occurring record features will lead to frequently set bit positions or even to frequent BFs in the case of field-level BFs. By using publicly available datasets containing person-related data, e. g., telephone books, voter registration databases, social media profiles or databases about persons of interests like authors, actors or politicians, an adversary can estimate the frequencies of record features and then try to align those frequencies to the BFs bit frequencies.

A successful re-identification of attribute values encoded in BF is a real threat as shown by several attacks proposed in the literature. Earlier attacks, namely [18, 20, 21, 24], often exploit the hashing method used in [27], the double-hashing scheme, that combines two hash functions to implement the k BF hash

Table 1: Overview of surveyed Bloom filter hardening techniques.

Subject of modification	Technique	Reference	Description
Bloom filter input	Avoidance of padding	[24, 25]	No use of padded q-grams as BF input due to their higher frequency.
	Standardization of attribute lengths	[24, 25]	The length of attribute values is unified to avoid exceptionally short or long values.
Hashing mechanism	Increasing the number of hash functions (k)	[26, 27]	Using more hash functions (k) while keeping the Bloom filter size (m) fixed will lead to more collisions and thus a higher number of features that are mapped to each position.
	Random hashing	[24]	Replacement for the double-hashing scheme [26] which can be exploited in attacks [24].
	Attribute weighting	[9, 32]	Record features are hashed with a different number of hash functions (k) depending on the weight of the attribute from which they were obtained.
	Salting	[24, 27]	Record features are hashed together with an additional attribute specific and/or record specific value.
Output Bloom filter	Balancing	[28]	Each Bloom filter is concatenated with a negative copy of itself and then the underlying bits are permuted.
	XOR-folding	[29]	Each Bloom filter is split into halves which are then combined using the bit-wise XOR-operation.
	Re-hashing	[25]	Sliding window approach where the Bloom filter bits in each window are used to generate a new set of bits.
	Rule90	[30]	Each Bloom filter bit is replaced by the result of XOR-ing its two neighbouring bits.
	Random noise	[1, 24–26, 28]	Bloom filter bits are changed randomly.
	Fake injections	[16]	Addition of artificial records and thus Bloom filters.

functions. This hashing method can easily be replaced by using independent hash functions or other techniques as discussed in Sec. 3.2.1. Furthermore, these attacks rely on many unrealistic assumptions, for instance, that the encoded records are a random sample of a resource known to the adversary [20, 24] or that all parameters of the BF process, including used secret keys for the hash functions, are known to the adversary [21]. However, recent frequency-based cryptanalysis attacks, namely [7] and in particular [8], are able to correctly re-identify attribute values without relying on such assumptions. These attacks are the more successful, the fewer attributes are encoded in a BF and the larger the number of encoded records. Overall, the attacks show the risk of re-identification when using BFs, especially field-level BFs. In this work, we will focus only on record-level BFs.

3 BLOOM FILTER VARIANTS AND HARDENING METHODS

In the following, we review different variations within the BF encoding process. In general, these variations will affect both the BFs privacy and similarity-preserving (matching) properties. Approaches that try to achieve a more uniform frequency distribution of individual BFs or set bit positions are also known as hardening techniques as they are intended to make BF encodings more robust against cryptanalysis. An overview of these techniques is given in Tab. 1. We divide the approaches into three categories: (A) approaches that alter the way of selecting features from the records attributes values, (B) approaches that modify the BFs hashing process and (C) approaches that modify already existing BFs by changing or aggregating bits. In the following subsections, we will describe the approaches of each category.

3.1 Record Feature Selection

We will first focus on how features are selected from the record's attributes. In the encoding process, at first, all attribute values are pre-processed to bring them into the same format and to reduce data quality issues. After that, all linkage-relevant attributes, i. e., the quasi-identifiers of a person, are transformed into their respective feature set. Such features are pieces of information that are usually obtained by segmenting the attribute values into chunks or tokens. This is necessary because instead of a binary decision for equality (true/false), approximate linkage is desired resulting in similarity scores ranging from zero (completely different) to one (equal).

3.1.1 Standardization of Attribute Lengths. Quasi-identifiers, such as names and addresses, show high variation and skewness leading to significant differences in the length of attribute values [11]. For instance, multiple given names, middle names or compound surnames (e. g., *'Hans-Wilhelm Müller-Wohlfahrt'*) will lead to exceptionally long attribute values and consequently a comparatively large amount of 1-bits in the resulting BF. The same applies for very short names (e. g., *'Ed Lee'*) resulting in very few 1-bits in the BF. By analyzing the number of 1-bits in a set of BFs, an adversary can gain information on the length of encoded attribute values. To address this problem, the length of the quasi-identifiers should be standardized by sampling, deletion or stretching of the attribute values [25]. Stretching can be implemented by concatenating short attribute values with (rarely occurring) character sequences.

3.1.2 Segmentation Strategy. The standard segmentation strategy adopted from traditional record linkage is to transform all quasi-identifiers into their respective **q-gram** set. A q-gram set is a set of all consecutive character sequences of length q that can be built from the attribute's string value by using a sliding window approach. For instance, setting $q = 3$ the value *'Smith'* will produce the q-gram set $\{Smi, mit, ith\}$. The idea behind building these q-gram sets is that they allow approximate string comparisons by calculating the number of q-grams two sets have in common. To directly obtain a similarity value, any set-based similarity measure, e. g., Jaccard coefficient, can be used.

The choice of q is important since it can affect the linkage quality. Usually, q is selected in the range $[1..4]$ while most approaches setting $q = 2$. In general, larger values for q are more sensitive to single character differences, e. g., the values *'Smith'* and *'Smyth'* will have two bigrams ($q = 2$), i. e., *'Sm'* and *'th'*, but zero trigrams ($q = 3$) in common. However, choosing a larger q also increases to number of possible q-grams, e. g., for $q = 2$ at maximum $26^2 = 676$ while for $q = 3$ at maximum $26^3 = 17\,576$ are possible. Overall, larger values for q tend to be less error-tolerant and thus possibly lead to missing matches. On the other hand, larger q's are more distinctive and thus tend to reduce false-positives.

As can be seen from the example above, for $q > 1$ each character will contribute to multiple q-grams except the first and last character. Thus, a common extension is to construct **padded q-grams** by surrounding each attribute value with $q - 1$ special characters at the beginning and the end. For our example the

padded q-gram set will be {*++S*, *+Sm*, *Smi*, *mit*, *ith*, *th-*, *h--*}. By using padded q-grams strings with the same beginning and end but variations in the middle will reach larger similarity values, while strings with different beginning and end will produce lower similarity values compared to standard q-grams [6]. It is important to note, that padded q-grams are among the most frequent q-grams and thus can ease any frequency alignment attacks.

There are several other extensions for generating q-grams, two of which have been used in traditional record linkage, but so far not for PPRL: **positional q-grams** and **skip-grams** [6]. Positional q-grams add the information from which position the q-gram was obtained. For our running example, the positional q-gram set for $q = 3$ is $\{(Smi, 0), (mit, 1), (ith, 2)\}$. When determining the overlap between two positional q-gram sets, only the q-grams at the same position or within a specific range are considered. Positional q-grams will be more distinctive and thus tend to reduce false-positives and even the frequency distribution. The idea of skip-grams is to not only consider consecutive characters but to skip one or multiple characters. Depending on the defined skip length multiple skip-gram sets can be created and used in addition to the regular q-gram set.

So far, only a few alternatives to q-grams have been investigated. In [17] and [31] the authors explore methods for handling numerical attribute values. Besides, arbitrary substrings of individual length or phonetic codes, such as Soundex [6], are possible approaches that can be used for feature extraction.

3.2 Modification of the Hashing Mechanism

After transforming all quasi-identifiers in their respective feature set, the features of each set are hashed into one record-level BF. As discussed in Sec. 2.2.2, we do not further consider field-level BFs due to their vulnerabilities. For PPRL several modifications of the standard hashing process of BFs have been proposed which we will discuss below.

3.2.1 Hash Functions. As described in Sec. 2.2, by default k independent (cryptographic) hash functions are used in conjunction with a private key $\mathcal{S}$ to prevent dictionary attacks. However, the authors of [27] proposed the usage of the so-called double-hashing scheme. This scheme only uses two independent hash functions G_1, G_2 to implement the BFs k hash functions. Each hash function is then defined as $H_i(x) = (G_1(x) + (i-1) \cdot G_2(x)) \mod m, \forall i \in \{1, \ldots, k\}$. The attacks described in [18, 24] showed that this specific scheme can be successfully exploited. As a consequence, an alternative method, called **random hashing**, was proposed [24] that utilizes a pseudo-random number generator to calculate the hash values. Therefore, the random number generator is seeded with the private key $\mathcal{S}$ and the actual input of the hash function, i. e., a certain record feature. No attacks against this method are known at present.

3.2.2 Salting. Salting is a well-known technique in cryptography that is often used to safeguard passwords in databases [22]. The idea is to use an additional input, called salt, for the hash functions to flatten the frequency distribution. Already in [27] it is mentioned that a different cryptographic secret key $\mathcal{S}_a$ can be used for each record *attribute a*. We term such kind of key as **attribute salt**. By using this approach, the same feature will be mapped to different positions if it originates from different attributes. For instance, given the first name *'thomas'* and the last name *'smith'* the bigram *'th'* will produce different positions. This approach will smoothen the overall frequency distribution and also reduce false-negatives since features from different attributes will not produce common 1-bits (except due to collision). However, the BF's ability to match exchanged attributes, e. g., transposed first and middle name, is lost. If such errors occur repeatedly this will lead to missing matches. As a compromise, we propose to define groups of attributes, where transpositions are expectable. Then, the same key is used for each attribute from the same group. For instance, all name-related attributes (first name, middle name, last name) could form a group.

Another salting variant is proposed in [24], where *for each record* a specific salt is selected and then used as key for the BFs k hash functions. Therefore, we term such keys as **record salt**, since they depend on a specific record. Record salts can also be combined with the aforementioned attribute salts. Only if the record salt is identical for two records, the same feature (q-gram) will set the same bit positions in the corresponding BFs. However, if the record salts are different, then the probability that the same bit positions are set in the corresponding BFs is very low. Thus, if the attributes (from which the record salts is extracted) contain errors, this will lead to many false-negatives. For this reason only commonly available, stable and small segments of quasi-identifiers, such as year of birth, are suitable as salting key. Consequently, this technique is only an option in PPRL scenarios where the attributes used for salting are guaranteed to be of very high quality which might be rarely the case in practice.

To reduce the aforementioned problem of salting with record-specific keys, we propose to generate the salt by utilizing blocking approaches. Blocking [6] is an essential technique in (privacy-preserving) record linkage to overcome the quadratic complexity of the linkage process since in general each record must be compared to each record of another source. The idea of blocking is to partition records into small blocks and then to compare only records within the same block to reduce the number of record pair comparisons. For this purpose, one or more blocking keys are defined, where each blocking key represents a specific, potentially complex criterion that records must meet to be considered as potential matches. For example, the combination of the first letter of the first and last name and the year of birth might be used as a blocking key. If the attributes used for blocking contain errors, then also the blocking key will be affected leading to many false-negatives, in particular if the blocking key is very restrictive. Hence, often multiple blocking keys are used to increase the probability for records to share at least one blocking key. However, this will lead to duplicate candidates since very similar records will share most blocking keys. The challenge of both, salting and blocking, is to select a key that is as specific as possible (to increase privacy, or to reduce the number of record pair comparisons respectively) and at the same time not prone to errors. For record-dependent salting, only the use of attribute segments was suggested. In contrast, for blocking more sophisticated approaches have been considered, in particular using phonetic codes, e. g., Soundex, or locality-sensitive hashing schemes, e. g., MinHash [4].

3.2.3 Dependency-based Hashing. In traditional record linkage, sophisticated classification models are used to decide whether a record pair represents a match or a non-match. Often these models deploy an attribute-wise or rule-based classification considering the discriminatory power and expected error rate of the attributes [6]. In contrast, PPRL approaches based on record-level BFs only apply classification based on a single similarity threshold since all attributes values are aggregated (encoded) in

a single BF. However, as discussed in Sec. 2.2.1, the record-level BF variant proposed in [9] also considers the weight of attributes by selecting more bits from the field-level BFs of attributes with higher weights. In [32] the authors proposed an extension to the approach of [27] to allow attribute weighting. While also only a single BF is constructed, a different number of hash functions is selected for different attributes according to their weights. Consequently, the higher the weight of an attribute, the more hash functions will be used and thus the more bits the attribute will set in the BF. The idea of varying the number of hash functions can be generalized to dependency-based hashing. For instance, not only the weights of attributes can be considered but instead also the frequency of input features or their position within the attribute value (positional q-grams).

3.3 Bloom Filter Modifications

While the methods described so far modify the way BFs are created, the following approaches are applied directly on the obtained BFs (bit vectors).

3.3.1 Balanced Bloom Filters. Balanced BFs were proposed in [28] for achieving a constant Hamming weight over all BFs. A constant Hamming weight should make the elimination of infrequent patterns more difficult. Balanced BFs are constructed by concatenating a BF with a negative copy of itself and then permuting the underlying bits. For instance, the BF [10011001] will give [10011001] · [01100110] = [1001100101100110] before applying the permutation. Since the size of the BFs is doubled, balanced BFs will increase computing time and required memory for BF comparisons.

3.3.2 XOR-Folding. XOR-folding of bit vectors is a method originating from chemo-informatics to speed up databases queries. In [29] the authors adopted this idea for Bloom-filter-based PPRL for preventing bit pattern attacks. To apply the XOR-folding a BF is split into halves and then the two halves are combined by the XOR-operation. For instance, the BF [11000101] will give [1100] ⊕ [0101] = [1001]. The folding process may be repeated several times. Since the size of the BFs is halved, XOR-folding will decrease computing time and required memory for BF comparisons. The initial evaluation in [29] using unrealistic datasets with full overlap and low error-rates shows that one-time folding does not significantly affect linkage quality. However, n-time folding drastically increases the number of false-positives.

3.3.3 Rule90. In [30] the use of the so-called *Rule90* was suggested to increase the resistance of BFs against bit-pattern-based attacks. The Rule90 is also based on the XOR-operation which is applied on the two neighboring values of each BF bit. Consequently, there are 8 possible combinations (patterns), which are listed in Tab. 2. So each bit b_i ($0 \leq i \leq m-1$) is replaced by the result of XOR-ing the two adjacent bits at positions $(i-1) \bmod m$ and $(i+1) \bmod m$. By using the modulo function the first and the last bit are treated as if they were adjacent. For example, applying Rule90 to the BF [11000101] will lead to the following patterns 1**1**1, 1**1**0, 1**0**0, 0**0**0, 0**0**1, 0**1**0, 1**0**1, 0**1**1 where the middle bit corresponds to the bit at position $i \in \{0, m-1\}$ of the BF. After applying the transformation rules (Tab. 2) we obtain [01101001].

Table 2: Transformation rules for Rule90.

Pattern	111	110	101	100	011	010	001	000
New Bit Value	0	1	0	1	1	0	1	0

3.3.4 Re-hashing. The idea of re-hashing [25] is to use consecutive bits of a BF to generate a new bit vector. Therefore, a window of width w bits is moved over the BF where in each step the window slides forward s positions (step size). At first, a new bit vector v of size m' is allocated. Then, the w bits, which are currently covered by the window, are represented as an integer value. The integer value is then used in combination with a secret key as input for a random number generator (RNG). With that, r new integer values are generated with replacement, each in the range $[0, m'-1]$. Finally, the bits at these r positions are set to one in the bit vector v. For example, given the BF [11000101] and setting $w = 4, s = 2$ will lead to three windows, namely $w_1 = [1100], w_2 = [0001], w_3 = [0101]$. By transforming the bits in each window into an integer value we obtain the seeds 12, 1 and 5. Setting $r = 2$ the RNG might generate the positions $(4, 2), (2, 5), (8, 6)$ for the respective seeds which finally results in the bit vector [001011101]. The evaluation in [28] uses very unrealistic datasets (full overlap, no errors) and shows no clear trend. However, this technique is highly dependent on the choice of the parameters m', w, s and r as well as on the original BFs, in particular the average fill factor (amount of 1-bits).

3.3.5 Random Noise. In order to make the frequency distribution of BFs more uniform, random noise can be added to the BFs [25, 28]. Trivial options are to randomly set bits to one/zero or to flip bits (complement). Additionally, the amount of random noise can depend on the frequency of mapped record features. For instance, for BFs containing frequent q-grams more noise can be added. In [1] a ϵ-differential private BF variant, called BLoom-and-flIP (BLIP), based on permanent randomized response is proposed. Each bit position $b_i, \forall i \in \{0, \ldots, m-1\}$ is assigned a new value b'_i based on the probability f such that

$$b'_i = \begin{cases} 1 & \text{with probability} \quad \frac{1}{2}f \\ 0 & \text{with probability} \quad \frac{1}{2}f \\ b_i & \text{with probability} \quad 1-f. \end{cases}$$

3.3.6 Fake Injections. Another option to modify the frequency distribution of BFs is to add artificial records or attribute values [16]. By inserting random strings containing rarely occurring q-grams the overall frequency distribution will become more uniform making any frequency alignment less accurate. The drawback of fake records is that they produce computational overhead in the matching process. Moreover, it is possible that a fake record will match with another record by chance. Thus, after the linkage, fake records need to be winnowed.

4 BLOOM FILTER PRIVACY MEASURES

Several attacks on BFs have been described in the literature (see Sec. 2.2.2), which show that BFs carry the risk of re-identification of attribute values and even complete records. Currently, the privacy of BF-based encoding schemes is mainly evaluated by simulating attacks and inspecting their results, i. e., the more attribute values and records can be correctly re-identified by an attack, the lower is the assumed degree of privacy of the encoding scheme. However, this way of measuring privacy strongly depends on the used attacks, their assumptions and the used reference dataset. Besides, only a few studies investigated evaluation measures for privacy [33]. These measures are either calculating the probability of suspicion [32] or are based on entropy and information gain between masked and unmasked data [28]. The disadvantage of these measures is that they strongly depend on

the reference data set used. In the following, we therefore propose privacy measures that solely depend on a BF dataset.

To evaluate the disclosure risk of BF-based encoding schemes we propose to analyze the frequency distribution of the BF 1-bits. As described in Sec. 2.2.2, attacks on BFs mostly try to align the frequency of frequent (co-occuring) bit patterns to frequent (co-occuring) record features (q-grams). Thus, the more uniform the frequency distribution of 1-bits is, the less likely an attack will be successful. To measure the uniformity of the bit frequency distribution of a BF dataset $\mathfrak{B}$, we calculate for each BF bit position (column) $0 \leq i < m-1$ the number of 1-bits, given as $\mathbf{c}_i = \sum_{\text{Bf} \in \mathfrak{B}} \text{Bf}(i)$, where $\text{Bf}(i)$ returns the BFs bit value at position i. The total number of 1-bits is then $\mathbf{b} = \sum_{i=0}^{m-1} c_i = \sum_{\text{Bf} \in \mathfrak{B}} |\text{Bf}|$ where $|\text{Bf}|$ denotes the cardinality of a BF (number of 1-bits). We can then calculate for each column its share of the total number of 1-bits, i. e., $\mathbf{p}_i = c_i / b$. Ideally, for a perfect uniform bit distribution, p_i will be close to b/m for all $i \in \{0, m-1\}$.

In mathematics and economics there are several measures that allow to assess the (non-) uniformity of a certain distribution. Consequently, we are adapting the most promising of these measures to our problem. At first, we consider the **Shannon entropy** $\mathcal{H}(\mathfrak{B}) = -\sum_{i=0}^{m-1} p_i \cdot \log_2(p_i)$ since uniform probability will yield maximum entropy. The maximum entropy is given as $\mathcal{H}_{max}(\mathfrak{B}) = \log_2(m)$. We define the normalized Shannon entropy ranging from 0 (high entropy - close to uniform) to 1 (low entropy) as

$$\widetilde{\mathcal{H}}(\mathfrak{B}) = 1 - \frac{H(\mathfrak{B})}{H_{max}(\mathfrak{B})} \tag{1}$$

Next, we consider the **Gini coefficient** [5, 15], which is well-known in economics as a measure of income inequality. The Gini coefficient can range from 0 (perfect equality – all values are the same) to 1 (maximal inequality – one column has all 1-bits and all others have only 0-bits) and is defined as

$$\mathbf{G}(\mathfrak{B}) = \frac{\sum_{i=0}^{m-1} \sum_{j=0}^{m-1} |c_i - c_j|}{2m \cdot b} \tag{2}$$

Moreover, we calculate the **Jensen-Shannon divergence** (JSD) [14] which is a measure of similarity between two probability distributions. The JSD is based on the Kullback-Leibler divergence (KLD) [19], but has better properties for our application: In contrast to the KLD, the JSD is a symmetric measure and the square root of the JSD is a metric known as **Jensen-Shannon distance** (D_{JS}) [10]. For discrete probability distributions P and Q defined on the same probability space, the JSD is defined as

$$\text{JSD}(P \,||\, Q) = \frac{1}{2} \text{KLD}(P \,||\, M) + \frac{1}{2} \text{KLD}(Q \,||\, M) \quad \text{where}$$

$$\text{KLD}(P \,||\, Q) = \sum_{s \in \mathcal{S}} P(s) \cdot \log_2 \left(\frac{P(s)}{Q(s)} \right) \quad \text{and} \quad M = \frac{1}{2}(P + Q).$$

The JSD also provides scores between 0 (identical) to 1 (maximal different). Since we want to measure the uniformity of the bit frequency distribution of a BF dataset $\mathfrak{B}$, we calculate the Jensen-Shannon distance given as

$$\mathbf{D}_{\text{JS}}(\mathfrak{B}) = \sqrt{\text{JSD}(\mathfrak{B})} \tag{3}$$

where

$$\text{JSD}(\mathfrak{B}) = \frac{1}{2} \left(\sum_{i=0}^{m-1} \frac{1}{m} \cdot \log_2 \left(\frac{\frac{1}{m}}{\frac{1}{2} \cdot (p_i + \frac{1}{m})} \right) \right) + \frac{1}{2} \left(\sum_{i=0}^{m-1} p_i \cdot \log_2 \left(\frac{p_i}{\frac{1}{2} \cdot (p_i + \frac{1}{m})} \right) \right)$$

Finally, we measure how many different record features (q-grams) are mapped to each bit position, which we denote as **feature ratio** (fr). The more features are mapped to each position, the harder becomes a one-to-one assignment between bit positions and record features which will limit the accuracy of an attack.

5 EVALUATION SETUP

Before presenting the evaluation results we describe our experimental setup as well as the datasets and metrics we use.

5.1 PPRL Setup

We implement the PPRL process as a three-party protocol assuming a trusted linkage unit [31]. Furthermore, we set the BF length $m = 1024$. To overcome the quadratic complexity of linkage, we use LSH-based blocking based on the Hamming distance [13]. We empirically determined the necessary parameters leading to high efficiency and effectiveness. As a result, we set $\Psi = 16$ (LSH key length) and $\Lambda = 30$ (number of LSH keys) as default. Finally, we calculate the Jaccard coefficient to determine the similarity of candidate record pairs. We classify every record pair with a similarity equal or greater than t as a match. Finally, we apply a one-to-one matching constraint, i. e., a record of one source can match to at maximum one record of another source, utilizing a symmetric best match approach [12].

5.2 Datasets

For evaluation, we use two real datasets that are obtained from the North Carolina voter registration database (NCVR) (https://www.ncsbe.gov/) and the Ohio voter files (OHVF) (https://www.ohiosos.gov/). For both datasets, we select subsets of two snapshots at different points in time. Due to the time difference records contain errors and inconsistencies, e. g., due to marriages/divorces or moves. Please note that we do not insert artificial errors or otherwise modify the records. We only determine how many attributes of a record have changed and use this information to construct subsets with a specific amount of records containing errors. An overview of all relevant dataset characteristics is given in Tab. 3. Each dataset consists of two subsets, S_A and S_B, to be linked with each other. The two subsets are associated with two data owners (or sources) A and B respectively.

Table 3: Dataset characteristics

Characteristic	Dataset	
	N	O
Type	Real (NCVR)	Real (OHVF)
$\lvert S_A \rvert$	50 000	120 000
$\lvert S_B \rvert$	50 000	80 000
$\lvert S_A \cap S_B \rvert$	10 000	40 000
Attributes	{First, middle, last} name, year of birth (YOB), city	{First, middle, last} name, date of birth (BD), city
|Errors|/record	0 (40 %), 1 (30 %), 2 (20 %), 3 (10 %)	0 (37.5 %), 1 (55 %), 2 (6.875 %), 3 (0.625 %)

5.3 Metrics

To assess the linkage quality we determine **recall**, **precision** and **F-measure** (F1-score). Recall measures the proportion of true-matches that have been correctly classified as matches after the linkage process. Precision is defined as the fraction of classified matches that are true-matches. F-measure is the harmonic mean of recall and precision. To assess the privacy (security) of the

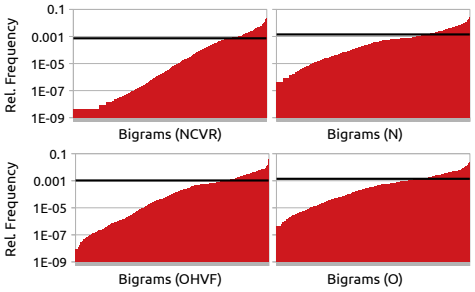
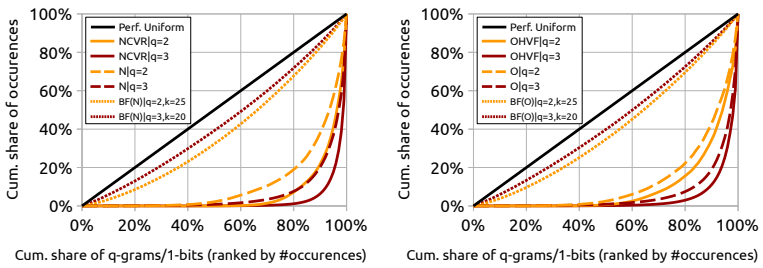


Figure 2: Relative bigram frequencies for used datasets.



(a) NCVR **(b) OHVF**

Figure 3: Comparison of Lorenz curves for plaintext and Bloom filters.

different Bloom-filter-based encoding schemes, we analyze the frequency distribution of the BF's 1-bits in order to determine the **normalized Shannon entropy**, the **Gini coefficient** and the **Jensen-Shannon distance** (see Sec. 4). Furthermore, we calculate the **feature ratio** (fr) that determines how many record features are mapped on average to each bit position.

5.4 Q-Gram Frequencies

Before we begin our evaluation on BFs, we analyze the plaintext frequencies of our datasets N and O as well as the complete NCVR and OHVF datasets. At first, we measure the relative bigram frequencies as shown in Fig. 2. What can be seen in this figure is the high dispersion of bigrams. For the complete NCVR and OHVF the non-uniformity is a bit higher than in our datasets which is mainly due to the larger number of infrequent bigrams. Since our datasets are only subsets from the respective voter registrations (NCVR/OHVF), some of these rare bigrams do simply not occur in our dataset subsets. In Fig. 3 we plot the Lorenz curves [15] for the plaintext datasets as well as BFs (see Sec. 6). These diagrams again illustrate the high dispersion for the plaintext values. Comparing bigrams and trigrams, it can be seen that the non-uniformity for trigrams is even higher than for bigrams. Our observations are confirmed by our uniformity (privacy) measures (see Sec. 4) which we calculate for the datasets as listed in Tab. 4. We use these values as a baseline for the BF privacy analysis. The closer the values for a set of BFs are to these values, the more likely a frequency alignment will be successful. On the other hand, the larger the difference between the values for plaintext and BFs, the better the BFs can hide the plaintext frequencies and thus the less likely a successful frequency alignment becomes. Comparing our three measures, it can be seen that the values for the normalized Shannon entropy ($\widetilde{\mathcal{H}}$) are much lower than the values for the Gini coefficient (G) and the Jensen-Shannon distance (D_{JS}). However, all measures clearly indicate the differences in the frequency distribution of bigrams and trigrams. Comparing both datasets, it can be seen that the non-uniformity of bi- and trigrams is slightly higher for the NCVR than for the OHVF dataset.

Table 4: Analysis of q-gram frequency distribution.

Measure	Datasets							
	Bigrams				Trigrams			
	N	NCVR	O	OHVF	N	NCVR	O	OHVF
$\widetilde{\mathcal{H}}$	0.1848	0.2497	0.1670	0.2027	0.2151	0.2781	0.2142	0.2491
G	0.7709	0.8728	0.7466	0.8047	0.8705	0.9425	0.8729	0.9189
D_{JS}	0.6315	0.7516	0.6107	0.6724	0.7340	0.8392	0.7362	0.8007

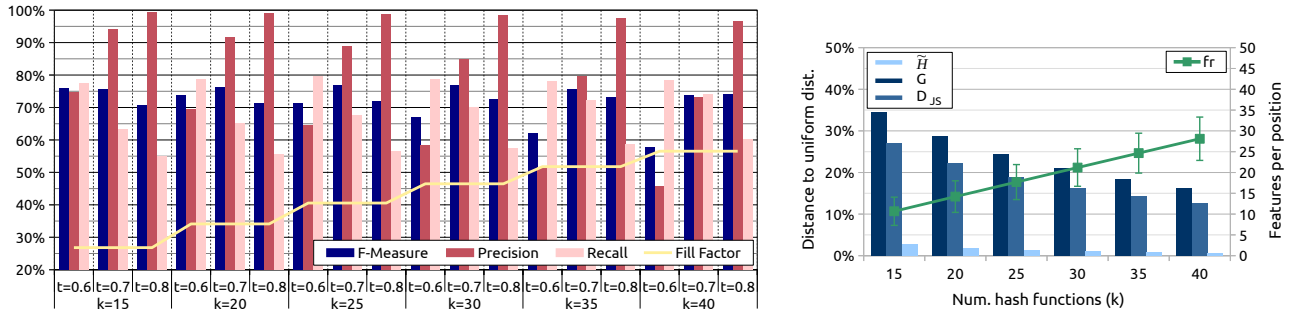
6 RESULTS AND DISCUSSION

In this section, we evaluate various BF variants and hardening techniques in terms of linkage quality and privacy (security).

6.1 Hash Functions and Fill Factor

In the following, we evaluate the linkage quality outcome and the privacy properties of basic BFs by inspecting the frequency distribution of the BF 1-bits compared to the q-gram frequencies. At first, we vary the number of hash functions (k), selecting $k \in \{15, 20, \ldots, 40\}$ and bigrams ($q = 2$), to adjust the fill factor (amount of 1-bits) of the BFs. The results for dataset N are depicted in Fig. 4. The results show, that for the high similarity thresholds of $t = 0.8$, all configurations achieve high precision $\geq 96.58\,\%$, but low recall $\leq 60.19\,\%$, leading to a max. F-measure of 74.16 %. For lower similarity thresholds ($t = \{0.7, 0.6\}$), precision is reduced drastically the more hash functions are used. For instance, setting $t = 0.6$ and $k = 15$, the highest precision of 75.93 % is achieved, while for $k = 40$ the precision is only 45.74 %. In contrast, the higher the number of hash functions, the higher the recall. For instance, setting $t = 0.6$ and $k = 15$, the recall is 73.28 %, while for $k = 40$ it increases to 78.51 %. However, the impact on precision is much higher (difference of around 34 %) than on recall (difference of around 5 %). Overall, the configuration with $t = 0.7$ and $k = 25$ achieves the best F-measure of 76.89 %. However, the other configuration except those with a fill factor over 50 % ($k \in \{35, 40\}$) achieve only slightly less F-measure. When averaging precision and recall for each k over all thresholds, the configurations with $k \leq 25$ achieve a mean F-measure of over 75 %, while for larger k it declines from around 74 % for $k = 30$ to around 71 % for $k = 40$.

Next, we analyze our privacy measures, which are depicted in Fig. 4(b). The figure shows that the more hash functions are used (and thus the higher the fill factor of the BFs) the higher is the avg. number of features that are mapped to each bit position. Even for the lowest number of hash functions, on average around 10 different bigrams are mapped to each individual bit position. Compared to the plaintext frequencies (see Tab. 4), we see that basic BFs have a significantly more uniform frequency distribution than the original plaintext dataset. For instance, using $k = 25$ hash functions, we obtain a Gini coefficient of 0.2443 and a Jensen-Shannon distance of 0.1891 compared to 0.7709 and 0.6315 for the unencoded dataset. Although for the Shannon entropy also a difference is visible, i. e., from 0.1848 for plaintext to 0.0137 for BFs setting $k = 25$, the values are in general much



(a) Quality **(b) Privacy**

Figure 4: Evaluation of standard Bloom filters using bigrams without padding for varying number of hash functions (k) on dataset N.

closer to zero and thus less intuitive to compare. As a consequence, in the following, we will focus on the other two privacy measures. Finally, the privacy measures indicate, that the more hash functions are used, the more closer the 1-bit distribution will get to uniform. However, the effect is not linear, such that the privacy gain is continuously getting lower, in particular for $k \geq 30$.

Table 5: Comparison of Bloom filter encodings using bi- and trigrams with and without padding for dataset N.

q	Pad.	k	t	Recall [%]	Prec. [%]	F-Meas. [%]	Mean Recall [%]	Mean Prec. [%]	Mean F-Meas. [%]
2	No	15	0.6	77.34	74.58	75.93	85.21	67.09	75.07
			0.7	63.27	94.12	75.67			
			0.8	54.96	**99.36**	70.77			
		20	0.6	78.71	69.54	73.84			
			0.7	65.21	91.75	76.24			
			0.8	55.64	99.19	71.29			
		25	0.6	**79.51**	64.58	71.27			
			0.7	67.71	88.95	**76.89**			
			0.8	56.50	98.81	71.89			
		30	0.6	78.68	58.51	67.11			
			0.7	70.14	84.80	76.78			
			0.8	57.43	98.37	72.52			
	Yes	10	0.6	81.48	84.06	**82.75**	90.21	**69.20**	78.32
			0.7	64.56	97.71	77.75			
			0.8	55.08	**99.67**	70.95			
		15	0.6	**83.77**	76.18	79.79			
			0.7	68.46	96.17	79.98			
			0.8	56.02	99.53	71.69			
		20	0.6	83.66	66.15	73.88			
			0.7	72.33	93.07	81.40			
			0.8	57.42	99.37	72.78			
3	No	15	0.6	69.83	86.85	77.42	91.12	63.28	74.69
			0.7	59.49	96.99	73.75			
			0.8	53.71	**99.57**	69.78			
		20	0.6	72.30	83.38	**77.45**			
			0.7	60.71	96.19	74.44			
			0.8	54.30	99.54	70.27			
		25	0.6	73.53	79.83	76.55			
			0.7	62.08	94.93	75.07			
			0.8	54.76	99.41	70.62			
		30	0.6	74.18	75.71	74.94			
			0.7	63.64	93.34	75.68			
			0.8	55.30	99.15	71.00			
		35	0.6	**74.25**	71.66	72.93			
			0.7	65.22	91.43	76.13			
			0.8	55.96	98.86	71.47			
	Yes	10	0.6	79.17	91.99	**85.10**	**93.93**	67.30	**78.42**
			0.7	61.77	98.91	76.05			
			0.8	53.87	**99.70**	69.95			
		15	0.6	**80.87**	86.61	83.64			
			0.7	66.74	98.00	79.40			
			0.8	54.82	99.63	70.72			
		20	0.6	80.31	75.42	77.79			
			0.7	71.96	95.60	82.11			
			0.8	56.20	99.48	71.82			

6.2 Choice of q and the Impact of Padding

In Tab. 5 we compare the linkage quality of BFs using different configurations for $q \in \{2, 3\}$ and padding for dataset N. Without the use of padding the best configuration for bigrams, i. e., $k = 25$ and $t = 0.7$, achieves a slightly less F-measure of 76.89 % than the best configuration for trigrams, i. e., $k = 20$ and $t = 0.6$, of 77.45 %. However, considering the mean over all configurations, using bigrams achieves a slightly higher F-measure of 75.07 % compared to 74.69 % for trigrams. Surprisingly, using trigrams results in an overall higher recall but lower precision if we average the results over all configurations. Moreover, the use of padding leads to a higher linkage quality, i. e., the best configurations for bigrams achieves a F-measure of 82.75 % while for trigrams even 85.10 % is attained. Averaged over all configurations, by using padding recall is increased about 5 % for bigrams and around 2.8 % for trigrams. Interestingly, also precision is increased by around 2.11 % for bigrams and around 4.02 % for trigrams. Thus, for both bigrams and trigrams, the mean F-measure can be increased by padding by more than 3 %. We repeat the experiments on dataset O and report the best configurations in Tab. 6. The results

Table 6: Comparison of Bloom filter encodings using bi- and trigrams with and without padding for dataset O.

q	Padding	k	t	Recall [%]	Precision [%]	F-Meas. [%]
2	No	25	0.7	68.17	88.32	76.95
	Yes	10	0.6	93.99	83.17	88.25
3	No	15	0.6	72.68	82.76	77.39
	Yes	10	0.6	95.49	90.14	92.74

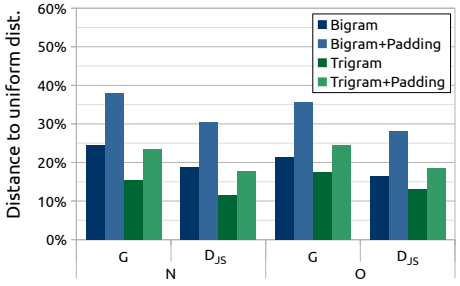
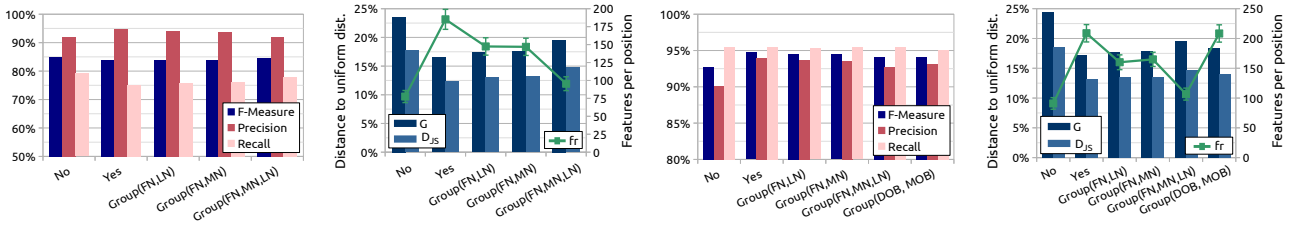


Figure 5: Comparison of Bloom filter privacy for bigrams and trigrams with and without using padding for datasets N and O.



(a) Quality - N dataset **(b) Privacy - N dataset** **(c) Quality - O dataset** **(d) Privacy - O dataset**

Figure 6: Impact of attribute salting.

confirm our previous observation that trigrams with padding lead to the highest linkage quality. Here, the best configuration using trigrams and padding outperforms that with bigrams and padding even slightly more than for dataset N, i. e., F-measure increases 2.35 % for N and 4.49 % for O.

Fig. 5 shows our privacy measures for the best configuration in each group. In general, the use of bigrams leads to a less uniform distribution of 1-bits and thus lower privacy. Also, the use of padding leads to a higher dispersion of the BFs 1-bits. However, even the worst configuration, namely bigrams using padding, leads to a significantly less Gini coefficient as for the plaintext datasets. For N, for instance, the Gini coefficient is reduced from 0.7709 to 0.3801 (see Tab. 4 and Fig. 3). Also the Jensen-Shannon distance reduces drastically, e. g., for dataset N from 0.6315 for the plaintext dataset to 0.3045 for the BF dataset using bigrams with padding. In contrast, the use of trigrams leads to a more even distribution of 1-bits, so that despite using padding, a slightly more uniform frequency distribution is achieved than with bigrams and without using padding.

To summarize, the highest linkage quality is achieved by using padding which indeed leads to less uniform 1-bit distribution making frequency-based cryptanalysis more likely to be successful. However, this can be compensated by using trigrams leading even to a slightly better linkage quality than for bigrams. Consequently, for our following evaluation, we select the best configuration using trigrams and padding with $k = 10$ as a baseline for our experiments.

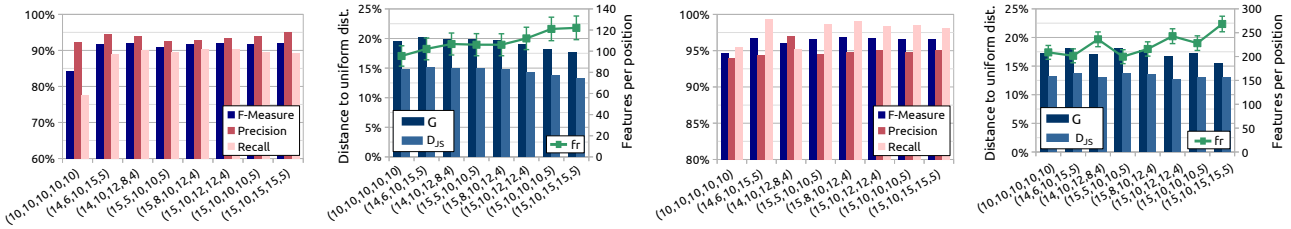
6.3 Salting and Weighting

In this section, we evaluate the impact of methods that alter the BFs hashing process by varying the number of hash functions and using salting keys to modify the hash mapping.

6.3.1 Attribute Salts. Fig. 6 depicts the results for BFs where the used hash functions are keyed (seeded) with a salt depending on the attribute a feature belongs to. For dataset N we observe that using an individual salt for each attribute increases precision from 91.99 % to 94.69 % but also decreases recall from 79.17 % to 75.26 % leading to a F-measure loss of around 1.2 %. Surprisingly, for dataset O precision increases from 90.14 % to 93.93 % while recall remains stable. Simultaneously, the average number of features that are mapped to each bit position increases by more than a factor of two for both datasets (Fig. 6 (b)/(d)). Furthermore, also the Gini coefficient and the Jensen-Shannon distance are significantly decreased and thus indicating an additional smoothing of the 1-bit distribution.

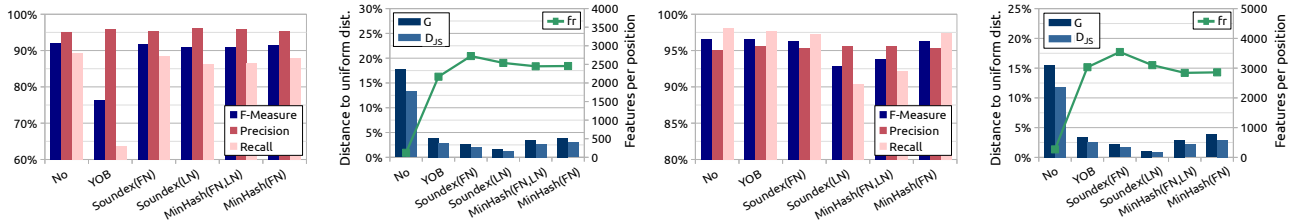
To be tolerant of swapped attributes, we build groups containing name-related attributes, i. e., one group for first name (FN) and last name (LN), one for first name and middle name (MN) and one for all three name components. Additionally, for dataset O, we build a group containing day and month of birth (DOB, MOB). For all attributes within one group, the same attribute salt is used. For dataset N we observe that all groups can slightly increase F-measure, while the group `(FN,MN,LN)` performs best and can increase F-measure to 84.48 %. Compared to the variant without using attribute salts, F-measure is therefore only decreased by 0.6 %. On dataset O, all groups achieve similar results, whereby precision and thus F-measure is always slightly lower than without using groups. Accordingly, swapped attributes seem to occur only rarely in dataset O. Using attribute salt groups also reduce the feature ratio and are also less effective in flattening the 1-bit distribution. Overall, however, the use of attribute salts can significantly reduce the dispersion of 1-bits while maintaining a high linkage quality. Building attribute salt groups can be beneficial for linkage quality, namely for applications where attribute transpositions are likely to occur. In the following, we include attribute salting as a baseline for our experiments, where for dataset N the group `(FN,MN,LN)` is used.

6.3.2 Impact of Attribute Weighting. In the following, we evaluate the impact of attribute weighting. Therefore, the number of hash functions is varied for each attribute depending on attribute weight. We tested several configurations and report the results in Fig. 7. The number of hash functions for each attribute is denoted



(a) Quality - N dataset **(b) Privacy - N dataset** **(c) Quality - O dataset** **(d) Privacy - O dataset**

Figure 7: Evaluation of varying number of hash functions based on attribute weights.



(a) Quality - N dataset **(b) Privacy - N dataset** **(c) Quality - O dataset** **(d) Privacy - O dataset**

Figure 8: Impact of record salting.

in the order (FN,LN,MN,YOB/BD,City). We observe that using attribute weighting strongly affects the linkage quality. All configurations that use a lower number of hash functions to map the attribute *city* can significantly increase both recall and precision. As a consequence, F-measure is improved by more than 6 % to over 91 % for dataset N and by around 2 % to over 96 % for dataset O. Analyzing the privacy results depicted in Fig. 7 (b)/(d), we observe that most weighting configurations can slightly increase the feature ratio and also slightly decrease the non-uniformity of 1-bits. By comparatively analyzing linkage quality and privacy, we select the configuration (15,10,15,15,5) as new baseline since it achieves the highest privacy while F-measure is only minimal less than for (14,10,12,8,4) (dataset N) and (15,10,12,12,4) (dataset O).

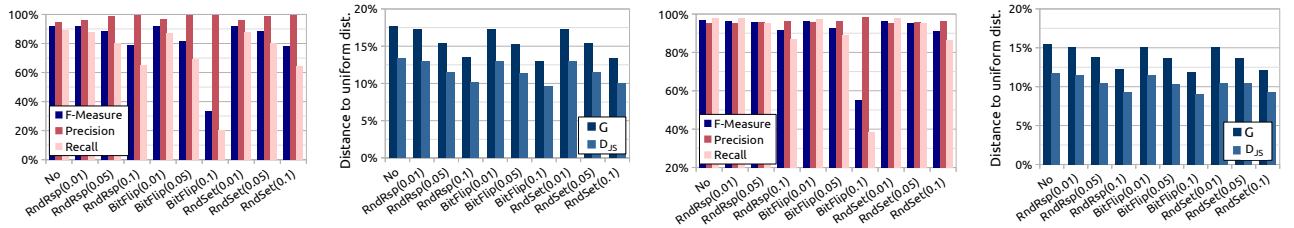
6.3.3 Record Salts. We now evaluate the approach of using a hash function salt that is individually selected for *each record*. The record salt is used in addition to the attribute salt we selected in the previous experiment. We tested several configurations using different attributes (year of birth, first name, last name). As Fig. 8(a)/(c) illustrate, record salts highly affect the linkage quality outcome. If we use the person's year of birth (YOB) as record-specific salt for the BFs hash function, recall drops drastically from 89.20 % (baseline) to only 63.58 % for dataset N. Apparently, in this dataset, this attribute is often erroneous and thus not suitable as record salt. In contrast, applying this configuration on dataset O, recall is only slightly reduced while precision is slightly increased, resulting in nearly the same F-measure. In order to compensate erroneous attributes, we test two techniques that are often utilized as blocking approaches, namely Soundex and MinHashing that we apply on the first and/or last name attribute. All tested approaches can slightly increase precision as they make the hash-mapping of the record features more unique. However, the Soundex and MinHash-based approaches also decrease recall, depending on the attribute(s) used. For instance, using Soundex on last name leads to relatively low recall in both datasets indicating many errors, e. g., due to marriages or divorces. Nevertheless, with the approaches using the first name, a similar high F-measure (loss ≤ 1 %) can be achieved as with the baseline.

Inspecting the privacy results depicted in Fig. 8 (b)/(d), we observe that the number of features that are mapped to each individual bit position is greatly increased by at least a factor of 10. At the same time, using record salts leads to a much more uniform 1-bit distribution. For instance, the Gini coefficient can be reduced from 0.1772 (baseline dataset N) and 0.1549 (baseline dataset O) to less than 0.04 for all tested approaches. The most uniform 1-bit distribution is achieved by using Soundex applied on last name, which leads to a Gini coefficient of less than 0.02. This implies that the 1-bit distribution is almost perfectly uniform which will make any frequency-based attack very unlikely to be successful. By analyzing privacy in relation to quality, we conclude that for both datasets Soundex applied to the first name performs the best and is able to achieve high linkage quality while effectively flattening the 1-bit distribution.

6.4 Modifications

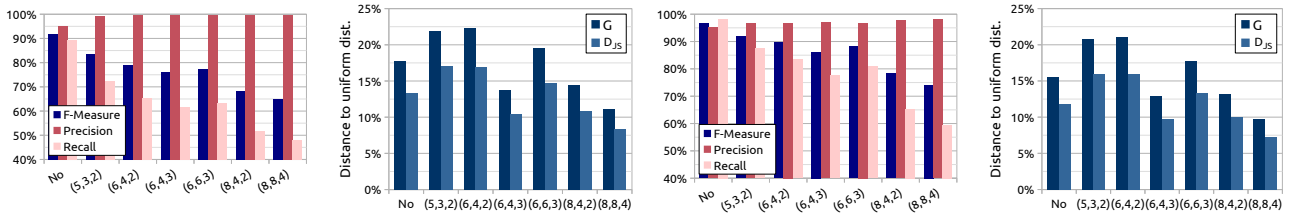
In the following, we evaluate hardening techniques that are applied directly on BFs (bit vectors).

6.4.1 Adding Random Noise. There are several ways of adding random noise to a BF (see Sec. 3.3.5). We compare the randomized response technique (RndRsp), random bit flipping (BitFlip) and randomly setting bits to one (RndSet) with each other. We vary the probability for changing an individual bit by setting $\rho = \{0.01, 0.05, 0.1\}$. The results are depicted in Fig. 9. As expected, recall and F-measure decrease with increasing ρ. While for $\rho = 0.01$ the loss is relatively small, it becomes significantly large for $\rho = 0.1$, in particular for the bit flip approach where recall drastically drops below 20 % for N and below 40 % for O. Interestingly, precision can be raised for all approaches and configurations up to 4.7 % (for $\rho = 0.1$). Overall, the bit flipping approach leads to the highest loss in linkage quality.



(a) Quality - N dataset **(b) Privacy - N dataset** **(c) Quality - O dataset** **(d) Privacy - O dataset**

Figure 9: Evaluation of random noise approaches.



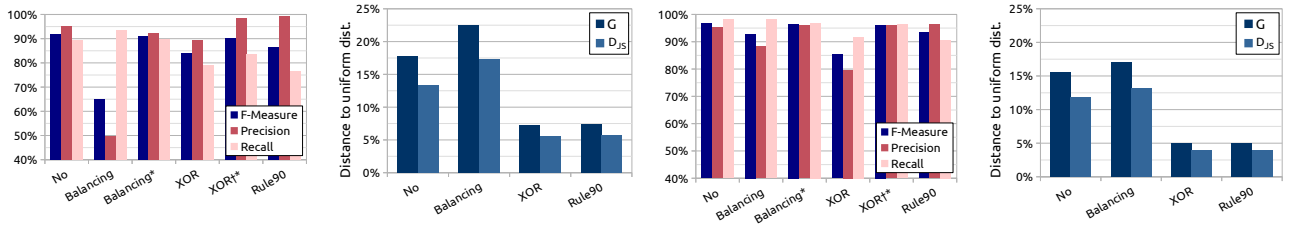
(a) Quality - N dataset **(b) Privacy - N dataset** **(c) Quality - O dataset** **(d) Privacy - O dataset**

Figure 10: Evaluation of re-hashing.

By analyzing the privacy results shown in Fig. 9 (b)/(d), it is evident that all random noise approaches can reduce the frequency information only a little. Only at a high ρ-value of 0.1 the Gini coefficient can be reduced by up to 4.81 % and the Jensen-Shannon distance up to 3.65 %. Analyzing the trade-off between linkage quality and privacy, we observe that a high value for ρ leads to an unacceptable loss in linkage quality. For lower ρ-values both techniques, randomized response and randomly setting bits to one, lead to relatively small losses in linkage quality. However, they are not able to significantly flatten the 1-bit distribution. Nevertheless, hardening techniques based on random noise may impede deterministic attacks by increasing the number of unique bit patterns.

6.4.2 Re-Hashing. To evaluate re-hashing, we tested several configurations regarding window size w, step size s and the number of re-hashed values r. In Fig. 10 we report the best configurations which are denoted in the order `(w,s,r)` setting $m' = m$. The results regarding linkage quality show that re-hashing increases precision but in contrast drastically decreases recall. In general, the larger the window size w the lower the recall that can be achieved. This effect is due to the fact that with larger windows there is a higher probability that a bit in the window is different for two similar BFs. This will result in another integer value (seed) on which the re-hashed 1-bit positions are selected. Even for the configuration with the smallest window size $w = 5$, recall decreases by more than 16 % for both datasets. We could not further decrease the window size, as with $w = 4$ only 16 different bit patterns are possible, so the re-hashed values will be often the same. This observation is confirmed by inspecting the privacy measures illustrated in Fig. 10 (b)/(d). Surprisingly, several configurations, namely `(5,3,2)`, `(6,4,2)` and `(6,6,3)`, will increase the non-uniformity of 1-bits. This is because the re-hashed values will be mapped only to a small range and thus increase the frequencies of these bits. In contrast, the configuration `(6,4,3)` and those with $w = 8$ can flatten the similarity distribution moderately. At the same time, however, these configurations will lead to an unacceptable low recall, e. g., for dataset N to only 61.66 % for `(6,4,3)` or even less then 50 % for `(8,8,4)`. As illustrated by the two configurations `(8,8,4)` and `(8,4,4)`, a reduction of the step size can increase recall since configurations with $s < w$ will lead to overlapping windows and thus a higher chance of finding overlapping bit patterns between two BFs. However, this again increases the unequal distribution of 1-bits. To summarize, we observe that re-hashing will decrease linkage quality while being not effective in increasing the uniformity of 1-bits. Therefore, we can not recommend this method for practical applications.

6.4.3 Balanced Bloom Filter, XOR-folding and Rule90. Finally, we evaluate balanced BFs, XOR-folding and applying Rule90 in terms of linkage quality and privacy. The results are depicted in Fig. 11. The results indicate that balancing reduces precision. While for dataset O precision decreases moderately by 6.93 %, for dataset N it drops drastically by 45.19 %. In contrast, recall remains stable for dataset O whereas it is slightly increased for dataset O. We found that changing our basic similarity threshold from 0.6 to 0.7 can significantly improve linkage quality for balancing. This might be due to the fact that balancing doubles the size of the BFs. Thus, we included the starred version of balancing indicating that a different threshold was used. With this configuration, balancing reduces F-measure only slightly for both datasets. For dataset N this is due to a little less precision and a little higher recall than for the baseline. For dataset O, however, it is the other way around, i. e., less recall and higher precision. XOR-folding also causes a reduction in linkage quality for both datasets. Since XOR-folding halves the size of the BFs, LSH-based blocking is affected in such a way that the amount of bits selected for the LSH keys is comparatively large. We therefore reduced the LSH key length from $\Psi = 16$ to $\Psi = 10$ and indicate this configuration with a dagger (†). By using this configuration and setting $t = 0.7$, for both datasets XOR-folding results in a minor loss of F-measure of less than 1 % compared to the baseline. Primarily accountable for the high F-measure is the high precision, which is slightly increased. Furthermore, we observe that



(a) Quality - N dataset **(b) Privacy - N dataset** **(c) Quality - O dataset** **(d) Privacy - O dataset**

Figure 11: Evaluation of balanced Bloom filters, XOR-folding and Rule90.

applying Rule90 also leads to a relatively high loss of recall of around 12.5 % for datast N and 7.6 % for dataset O. Again, Rule90 increases precision slightly thus leading to a moderate loss of F-measure of around 5.5 % for dataset N and 3.2 % for dataset O.

Examining Fig. 11 (b)/(d), we can see that balancing interestingly increases the dispersion of 1-bits. For dataset N, for instance, the Gini coefficient is increased by around 4.8 % and the Jensen-Shannon distance by around 4 %. In contrast, XOR-folding and Rule90 lead to a more uniform distribution of 1-bits. Both approaches reduce the Gini coefficient by about 10 % and the Jensen-Shannon distance by more than 7.5 %. Considering both, linkage quality and privacy, we conclude that XOR-folding performs the best by maintaining high linkage quality while effectively flattening the 1-bit distribution.

7 CONCLUSION

Bloom filters are frequently used in both research and practice for PPRL applications. In this paper, we reviewed and classified various BF variants and hardening techniques that aim at making Bloom filters more robust against cryptanalysis. Currently, no privacy measure exists that allows comparison of different encoding schemes in terms of privacy (security) and is independent of any reference dataset. We therefore proposed three privacy measures that allow assessing the privacy properties of Bloom filter encodings. These measures are based solely on a set of Bloom filters and do not need any reference dataset or other information. Moreover, we comprehensively evaluated the Bloom filter variants and hardening techniques in terms of both linkage quality and privacy. The evaluation showed that multiple hardening techniques drastically reduce linkage quality and are thus not applicable in real-world scenarios. However, in particular two techniques, namely salting and XOR-folding, drastically reduce any frequency information while maintaining high linkage quality. Carefully selected Bloom filter parameters in combination with these techniques will make any frequency-based cryptanalysis very unlikely to be successful.

For future work, we aim to evaluate these approaches against modern Bloom filter attacks described in the literature to further verify our findings.

REFERENCES

[1] Mohammad Alaggan, Sébastien Gambs, and Anne-Marie Kermarrec. 2012. BLIP: Non-interactive Differentially-Private Similarity Computation on Bloom filters. In *Stabilization, Safety and Security of Distributed Systems*. 202–216. https://doi.org/10.1007/978-3-642-33536-5_20

[2] Burton H. Bloom. 1970. Space/Time Trade-Offs in Hash Coding with Allowable Errors. *CACM* 13, 7 (1970), 422–426. https://doi.org/10.1145/362686.362692

[3] Andrei Broder and Michael Mitzenmacher. 2004. Network Applications of Bloom Filters: A Survey. *Internet Mathematics* 1, 4 (2004), 485–509. https://doi.org/10.1080/15427951.2004.10129096

[4] A. Z. Broder. 1998. On the Resemblance and Containment of Documents. In *Compression and Complexity of Sequences*. 21–29. https://doi.org/10.1109/SEQUEN.1997.666900

[5] Lidia Ceriani and Paolo Verme. 2012. The Origins of the Gini Index: Extracts from Variabilità e Mutabilità (1912) by Corrado Gini. *The Journal of Economic Inequality* 10, 3 (2012), 421–443. https://doi.org/10.1007/s10888-011-9188-x

[6] Peter Christen. 2012. *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. https://doi.org/10.1007/978-3-642-31164-2

[7] Peter Christen, Thilina Ranbaduge, Dinusha Vatsalan, and Rainer Schnell. 2018. Precise and Fast Cryptanalysis for Bloom Filter Based Privacy-Preserving Record Linkage. *IEEE TKDE* (2018). https://doi.org/10.1109/TKDE.2018.2874004

[8] Peter Christen, Anushka Vidanage, Thilina Ranbaduge, and Rainer Schnell. 2018. Pattern-Mining Based Cryptanalysis of Bloom Filters for Privacy-Preserving Record Linkage. In *PAKDD*. Vol. 10939. Springer, 530–542. https://doi.org/10.1007/978-3-319-93040-4_42

[9] Elizabeth A. Durham, Murat Kantarcioglu, Yuan Xue, Csaba Toth, Mehmet Kuzu, and Bradley Malin. 2014. Composite Bloom Filters for Secure Record Linkage. *IEEE TKDE* 26, 12 (2014), 2956–2968. https://doi.org/10.1109/TKDE.2013.91

[10] D.M. Endres and J.E. Schindelin. 2003. A New Metric for Probability Distributions. *IEEE Transactions on Information Theory* 49, 7 (2003), 1858–1860. https://doi.org/10.1109/TIT.2003.813506

[11] Wendy R. Fox and Gabriel W. Lasker. 1983. The Distribution of Surname Frequencies. *Int. Statistical Review* 51, 1 (1983), 81–87. https://doi.org/10.2307/1402733

[12] Martin Franke, Ziad Sehili, Marcel Gladbach, and Erhard Rahm. 2018. Post-Processing Methods for High Quality Privacy-Preserving Record Linkage. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Springer, 263–278. https://doi.org/10.1007/978-3-030-00305-0_19

[13] Martin Franke, Ziad Sehili, and Erhard Rahm. 2018. Parallel Privacy-Preserving Record Linkage Using LSH-Based Blocking. In *IoTBDS*. 195–203. https://doi.org/10.5220/0006682701950203

[14] B. Fuglede and F. Topsoe. 2004. Jensen-Shannon Divergence and Hilbert Space Embedding. In *Int. Symposium on Information Theory (ISIT)*. 31–. https://doi.org/10.1109/ISIT.2004.1365067

[15] Joseph L. Gastwirth. 1972. The Estimation of the Lorenz Curve and Gini Index. *The Review of Economics and Statistics* 54, 3 (1972), 306. https://doi.org/10.2307/1937992

[16] Alexandros Karakasidis, Vassilios S. Verykios, and Peter Christen. 2012. Fake Injection Strategies for Private Phonetic Matching. In *Data Privacy Management and Autonomous Spontaneus Security*. 9–24. https://doi.org/10.1007/978-3-642-28879-1_2

[17] Dimitrios Karapiperis, Aris Gkoulalas-Divanis, and Vassilios S. Verykios. 2018. FEDERAL: A Framework for Distance-Aware Privacy-Preserving Record Linkage. *IEEE TKDE* 30, 2 (2018), 292–304. https://doi.org/10.1109/TKDE.2017.2761759

[18] Martin Kroll and Simone Steinmetzer. 2014. Automated Cryptanalysis of Bloom Filter Encryptions of Health Records. *German Record Linkage Center, Working Paper Series* (2014). https://doi.org/10.2139/ssrn.3530864

[19] S. Kullback and R. A. Leibler. 1951. On Information and Sufficiency. *The Annals of Mathematical Statistics* 22, 1 (1951), 79–86. https://doi.org/10.1214/aoms/1177729694

[20] Mehmet Kuzu, Murat Kantarcioglu, Elizabeth Durham, and Bradley Malin. 2011. A Constraint Satisfaction Cryptanalysis of Bloom Filters in Private Record Linkage. In *Privacy Enhancing Technologies*. 226–245. https://doi.org/10.1007/978-3-642-22263-4_13

[21] William Mitchell, Rinku Dewri, Ramakrishna Thurimella, and Max Roschke. 2016. A Graph Traversal Attack on Bloom Filter Based Medical Data Aggregation. *Int. Journal of Big Data Intelligence* (2016). https://doi.org/10.1504/IJBDI.2017.086956

[22] Robert Morris and Ken Thompson. 1979. Password Security: A Case History. *Commun. ACM* 22, 11 (1979), 594–597. https://doi.org/10.1145/359168.359172

[23] National Institute of Standards and Technology. 2008. *The Keyed-Hash Message Authentication Code (HMAC)*. Technical Report NIST FIPS 198-1. https://doi.org/10.6028/NIST.FIPS.198-1

[24] Frank Niedermeyer, Simone Steinmetzer, Martin Kroll, and Rainer Schnell. 2014. Cryptanalysis of Basic Bloom Filters Used for Privacy Preserving Record Linkage. *Journal of Privacy and Confidentiality* 6, 2 (2014). https://doi.org/10.29012/jpc.v6i2.640

[25] Rainer Schnell. 2014. Privacy Preserving Record Linkage. In *Methodological Developments in Data Linkage*. 201–225.

[26] Rainer Schnell, Tobias Bachteler, and Jörg Reiher. 2009. Privacy-Preserving Record Linkage Using Bloom Filters. *BMC Medical Informatics and Decision Making* 9, 1 (2009). https://doi.org/10.1186/1472-6947-9-41

[27] Rainer Schnell, Tobias Bachteler, and Jörg Reiher. 2011. A Novel Error-Tolerant Anonymous Linking Code. (2011). https://doi.org/10.2139/ssrn.3549247

[28] Rainer Schnell and Christian Borgs. 2016. Randomized Response and Balanced Bloom Filters for Privacy Preserving Record Linkage. In *IEEE ICDMW*. 218–224. https://doi.org/10.1109/ICDMW.2016.0038

[29] Rainer Schnell and Christian Borgs. 2016. XOR-Folding for Hardening Bloom Filter-Based Encryptions for Privacy-Preserving Record Linkage. *German Record Linkage Center, Working Paper Series* (2016). https://doi.org/10.2139/ssrn.3527984

[30] Rainer Schnell and Christian Borgs. 2018. Hardening Encrypted Patient Names Against Cryptographic Attacks Using Cellular Automata. In *IEEE ICDMW*. 518–522. https://doi.org/10.1109/ICDMW.2018.00082

[31] Dinusha Vatsalan and Peter Christen. 2016. Privacy-Preserving Matching of Similar Patients. *Journal of Biomedical Informatics* 59 (2016), 285–298. https://doi.org/10.1016/j.jbi.2015.12.004

[32] Dinusha Vatsalan, Peter Christen, Christine M. O'Keefe, and Vassilios S. Verykios. 2014. An Evaluation Framework for Privacy-Preserving Record Linkage. *Journal of Privacy and Confidentiality* 6, 1 (2014).

[33] Dinusha Vatsalan, Ziad Sehili, Peter Christen, and Erhard Rahm. 2017. Privacy-Preserving Record Linkage for Big Data: Current Approaches and Research Challenges. In *Handbook of Big Data Technologies*. Springer, 851–895. https://doi.org/10.1007/978-3-319-49340-4_25