International Journal of Data Science and Analytics
https://doi.org/10.1007/s41060-022-00377-2

REGULAR PAPER

Privacy-preserving record linkage using autoencoders

Victor Christen[1] · Tim Häntschel[1] · Peter Christen[2] · Erhard Rahm[1]

Received: 11 August 2022 / Accepted: 20 November 2022
© The Author(s) 2022

Abstract
Privacy-preserving record linkage (PPRL) is the process aimed at identifying records that represent the same real-world entity across different data sources while guaranteeing the privacy of sensitive information about these entities. A popular PPRL method is to encode sensitive plain-text data into Bloom filters (BFs), bit vectors that enable the efficient calculation of similarities between records that is required for PPRL. However, BF encoding cannot completely prevent the re-identification of plain-text values because sets of BFs can contain bit patterns that can be mapped to plain-text values using cryptanalysis attacks. Various hardening techniques have therefore been proposed that modify the bit patterns in BFs with the aim to prevent such attacks. However, it has been shown that even hardened BFs can still be vulnerable to attacks. To avoid any such attacks, we propose a novel encoding technique for PPRL based on autoencoders that transforms BFs into vectors of real numbers. To achieve a high comparison quality of the generated numerical vectors, we propose a method that guarantees the comparability of encodings generated by the different data owners. Experiments on real-world data sets show that our technique achieves high linkage quality and prevents known cryptanalysis attacks on BF encoding.

Keywords Data linkage · Bloom filters · Reidentification · Sensitive data · Personal data · Privacy attack

1 Introduction

It is generally recognised that linked individual-level databases facilitate data analysis that is not feasible on a single database [3]. Therefore, in domains ranging from business analytics and national security to health and social science research, increasingly records about individuals need to be linked across databases that are often held by different organisations. *Record linkage* has been an active research area since the 1950s [27].

The lack of common unique entity identifiers (such as social security numbers or patient identifiers) across the databases to be linked means that linking records is commonly based on available quasi-identifiers (QIDs), such as the names, addresses, and dates of birth of the individuals whose records are to be linked [7]. Given these are personally identifiable information [26], concerns about privacy and confidentiality limit or even prevent such personal data from being used for the linkage of records across databases [16,40].

Techniques generally known as *privacy-preserving record linkage* (PPRL) have been developed in the past two decades [16,43] with the aim of tackling the challenge of linking sensitive data without revealing any private or sensitive information about the entities being linked. The general approach of PPRL techniques is to encode or encrypt sensitive identifying information and conduct the linkage using these encoded or encrypted values. At the end of a PPRL process, only the organisations being involved learn which of their records are matches (based on some decision model) with records from the other database(s), but no organisation is able to learn any sensitive information about records in the database(s) held by other organisations. Furthermore, external adversaries must be denied the discovery of any meaningful information about the sensitive data [7].

✉ Victor Christen
christen@informatik.uni-leipzig.de

Tim Häntschel
timhaentschel@yahoo.de

Peter Christen
peter.christen@anu.edu.au

Erhard Rahm
rahm@informatik.uni-leipzig.de

[1] Department of Computer Science, University of Leipzig, Leipzig, Germany

[2] School of Computing, The Australian National University, Canberra, Australia

A diverse range of PPRL techniques has been developed [42], including techniques based on secure multiparty computation (SMC), secure hash encoding, and encoding of values into bit vectors. While SMC techniques are accurate and provably secure, because PPRL generally requires the calculation of similarities between encoded values (due to errors and variations that can occur in QID values [9]) these techniques often have high computational costs [16]. PPRL techniques based on some form of hashing or embedding of sensitive values, known as perturbation-based techniques [43], on the other hand provide adequate privacy, linkage quality, and scalability to link large sensitive databases. However, perturbation-based techniques commonly lack the security proofs provided by SMC techniques [7].

As we discuss further in Sect. 3, one popular perturbation technique used in PPRL is based on BF encoding [4], where elements of a set (such as character q-grams extracted from QID values) are hashed into bit vectors [32]. BF-based PPRL is now being employed in practical linkage applications, mainly in the health domain [5,29,31].

The general PPRL workflow follows a three-party protocol [7], where the (DOs) generate BFs based on encoding the QID values of records, and send these BFs to a third party, called the linkage unit (LU). The LU then compares pairs of BFs to calculate their similarity and classifies pairs into *matches* (two BFs assumed to represent the same entity) and *non-matches* (two BFs assumed to represent two different entities).

While BF encoding facilitates scalable and accurate linkage of large databases, its drawback is the lack of provable security. As a result, various attacks on BF-based PPRL encoding techniques have been developed [47]. These attacks mainly exploit the bit patterns and their frequencies in a set of BFs [6,8,10,20–22,25,28], or the similarities between BFs [11,44].

To overcome such attacks, different *hardening* techniques have been proposed [30,34,35,39]. These techniques modify the bit patterns of BFs with the aim of removing the association of frequent patterns or the positions of 1 bits with encoded values (such as character q-grams) that would allow the re-identification of encoded values [47]. Other methods add fake records or BFs to perturb frequency patterns [18]. However, due to the addition of noise or modification of bit patterns, existing hardening techniques have shown to negatively influence the final linkage quality by increasing the number of false matches (false positives) and/or reducing the number of true matches (false negatives) [14].

In this paper, we propose a novel encoding technique using autoencoder networks [1] to transform bit patterns in BFs that encode sensitive values into numerical vectors. For each DO, our technique independently trains an autoencoder network using the DO's BFs. To guarantee comparability of the encodings generated from the different autoencoders, we

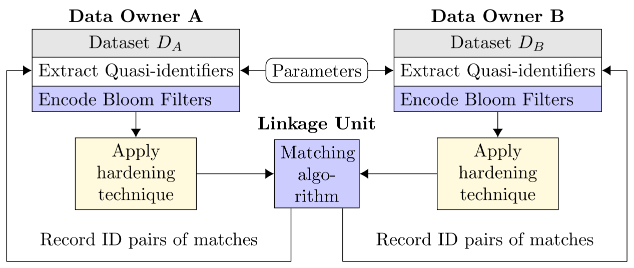

Data Owner A
Dataset D_A
Extract Quasi-identifiers
Encode Bloom Filters
Apply hardening technique
Record ID pairs of matches
Parameters
Linkage Unit
Matching algorithm
Data Owner B
Dataset D_B
Extract Quasi-identifiers
Encode Bloom Filters
Apply hardening technique
Record ID pairs of matches

Fig. 1 Three-party protocol for PPRL based on BF encoding and a hardening technique applied

train a mapping function that transforms the encodings from one DO into the latent space of the second DO. This mapping allows the LU to accurately calculate the similarities between the encodings from the different DOs.

We make the following contributions: (1) We propose a novel PPRL encoding technique which applies autoencoders on BFs to improve their privacy by preventing attacks on frequent bit patterns in BFs. (2) Our technique generates linkage results with high quality by using the calculated encodings in a numerical vector space in combination with a mapping function that allows the LU to accurately compare encodings from multiple DOs. (3) We evaluate our proposed technique using real-world data sets considering different parameter settings, and we compare our method with existing hardening techniques regarding linkage quality.

2 Related work

Different methods have been proposed to attack BFs with the goal of re-identifying the sensitive values encoded in a set of BFs [47]. Kuzu et al. [21] developed a method that maps BFs to first names from the North Carolina Voter Registration database based on their frequencies and further constraints regarding common q-grams and common BFs. Christen et al. [6] proposed a frequent pattern mining-based approach that identifies frequent bit patterns in BFs and aligns them to q-grams considering the frequency distribution derived from external resources such as telephone books. In contrast to other attacks, this attack can also be applied on BFs that encode more than one attribute.

Different from previous attacks is a graph-based attack proposed by Vidanage et al. [44] that uses a similarity graph built from BFs that is matched to a similarity graph built from plain-text values. The idea is to generate for each BF and each plain-text value a set of features that represent their neighbourhood in the corresponding similarity graph, and then perform a bipartite matching between the feature vectors of BFs and the feature vectors of plain-text records. However, for accurate matching of these graphs, a mapping between BF and plain-text value similarities is required [44].

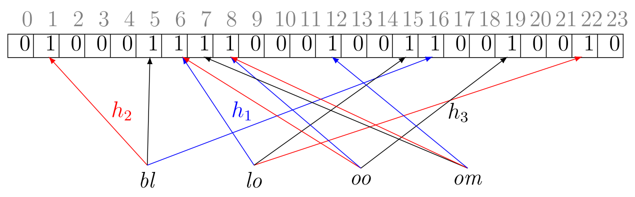


Fig. 2 BF encoding of the word 'bloom' converted into the set of 2-grams (bigrams) {'bl', 'lo', 'oo', 'om'} and mapped into a bit vector of length $l = 24$ using $k = 3$ hash functions which, for example, map 'bl' to bit positions 1, 5, and 16

To avoid the identification of associations between bit patterns and plain-text values, hardening techniques manipulate BFs by adding noise or modifying the encoding process of BFs with respect to the frequency distribution of q-grams. Ranbaduge and Schnell [30] provided a comprehensive overview of different hardening techniques including *XOR* folding, *BLIP*, *Rule 90*, *WXOR*, and *Resampling*. *XOR* folding [34] divides a BF into two halves and applies the bit-wise XOR operation on the resulting shorter half-length BFs. BLIP [34,39] uses a differential privacy mechanism to randomly select a set of bit positions in a BF and flipping them (0 to 1 or vice versa) depending on a flip probability f. *Rule 90* [35] generates a hardened BF by XORing the bit at position p with the bits at position $(p - 1)\ mod\ l$ and $(p + 1)\ mod\ l$, where l is the length of a BF. The modulo function ensures that each position in the hardened BF is based on three bits.

The *WXOR* hardening method [30] uses a window-based XOR approach where two windows, $W1$ and $W2$, of width $w > 1$ slide over a BF of length l. The starting position p of $W1$ slides from 0 to $l - w$, while $W2$ is positioned at $(p + 1)\ mod\ l$. For generating the window at position p, the bit patterns of the two windows are XORed. The *Resample* method [30] determines for each position p of the BF to be hardened the XOR operation of two randomly selected positions, i and j, ranging from 0 to $l - 1$, with replacement.

All these discussed hardening techniques lower the risk of a successful attack at the expense of linkage quality [14]. In contrast, we propose a novel encoding technique based on autoencoders that offers a complete masking of any bit patterns while still providing linkage quality comparable to the quality of unhardened BFs, as we experimentally evaluate in Sect. 6.

3 Background

We now describe the PPRL process, BF encoding, and autoencoders, which form the basis of our approach.

The PPRL process Figure 1 shows the three-party PPRL protocol [7], where a LU receives QID values from two or more DOs that have been encoded, for example, into BFs and (optionally) further hardened. The LU compares these encodings and classifies the corresponding pairs of records as matches or non-matches. The record identifier (ID) pairs of matched encodings are returned to the DOs as result of the linkage.

The DOs encode their own QID values independently according to the agreed encoding method and parameters that define which QIDs are used and how they are to be encoded. For BF encoding [32], this includes the number of hash functions k and the length of BFs l to be used, and so on [7]. The DOs potentially also apply an agreed hardening technique to transform the generated BFs into hardened encodings [30].

Employing a LU avoids the direct exchange of data between the DOs which would increase the risk of revealing sensitive information in the encoded QID values. This is because BFs are easy to decode for DOs that have knowledge about the encoding parameters [25].

Bloom filter encoding BFs are bit vectors of length l with an associated set of k independent hash functions that map the elements of a set to positions in the bit vector [4]. The idea of using a BF is to efficiently determine whether a certain element in a set has been encoded in a BF or not, based on the bit patterns generated by the hash functions.

In the context of PPRL, BFs are generally based on the encoding of textual QID values, such as the names and addresses of people, that are converted into character q-gram sets [32]. Such sets are then mapped to positions in a BF by using k hash functions, h_i (with $0 \leq i < k$), as shown in Fig. 2. Methods to encode numerical values (such as ages or medical data) [19,41] and categorical codes (such as disease or occupation codes) [36] into BFs have also been developed.

Autoencoders Our approach is based on *autoencoders* to further encode BFs to prevent cryptanalysis attacks. Autoencoders [1] are neural networks (NNs) that can generate lower-dimensional representations with a small information loss for high-dimensional input data. In our case, we use the reduction of dimensions and the transformation of BFs from a binary l-dimensional space into a continuous space as an advantage to hide potentially vulnerable bit patterns in BFs.

Autoencoders are generally composed of two connected NNs: an encoder f that maps data into a low-dimensional space (of dimension d, with $d < l$), and a decoder g, that maps values from the low-dimensional space back into the original space. The two NNs are trained in combination, and aim to fit the identity function on the data. Formally, an encoder f and a decoder g are defined by the following functions, where w_1 and w_2 are trainable weights [1]:

$$f_{w_1} : [0, 1]^l \longrightarrow \mathbb{R}^d \qquad g_{w_2} : \mathbb{R}^d \longrightarrow [0, 1]^l$$

An autoencoder A is then represented by the concatenation of both functions utilising the trained NNs: $A = g \circ f$. In contrast to other dimensionality reduction methods, such as principal component analysis or singular value decomposi-

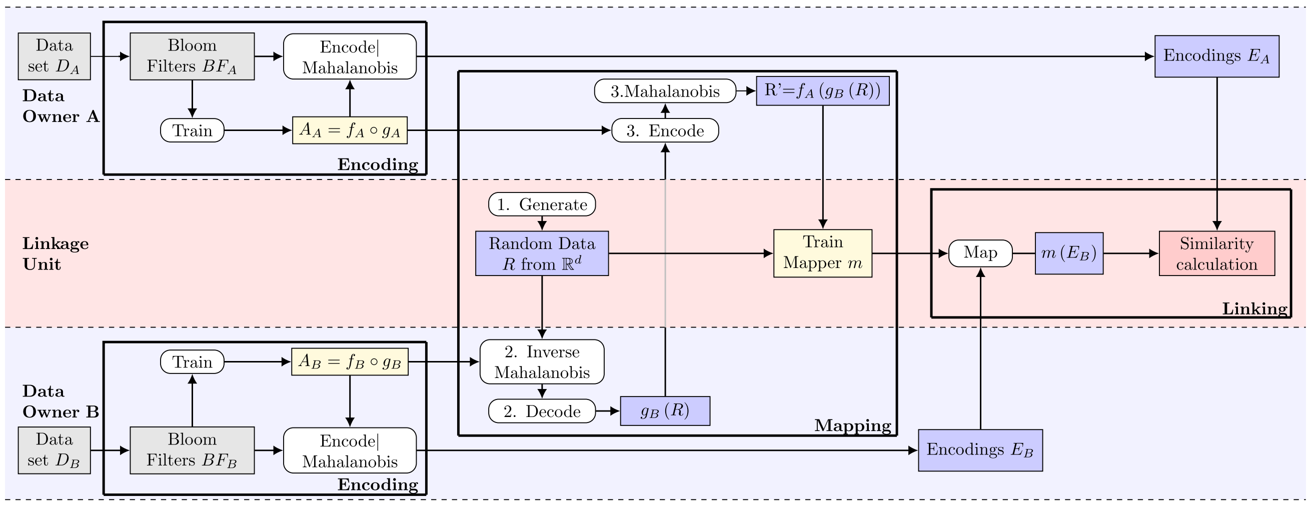


Fig. 3 Extended three-party PPRL protocol using autoencoders in chronological order (left to right) and separated by the different parties (the horizontal layers). The black-lined boxes represent the main steps of the process, and the white boxes show the specific tasks for generating the models (yellow) and the data (blue) required

tion [13], autoencoders can provide nonlinear transformation functions for generating low-dimensional encodings. The attained encoding function is sensitive towards changes of the initial weights which are randomly drawn, and therefore is a non-deterministic function. Both of these properties suit our approach, because (1) the space of possible BFs of length l, $\mathcal{B} = \{0, 1\}^l$, is not isomorphic to a low-dimensional linear space and therefore requires a nonlinear mapping for good low-dimensional representation; and (2) a deterministic mapping would be easier to attack compared to non-deterministic mapping because the encoding dimensions might carry some specific semantics.

4 PPRL using autoencoders

To decrease the risk of attacks on BFs [47], we develop a novel PPRL technique based on autoencoders [1]. One main requirement for any encoding to be used for PPRL is to preserve similarities [7]. To achieve this goal, we have to select the autoencoder layout such that information loss is minimal, and apply data transformation steps to normalise the output of the encoder. The first requirement ensures that most of the information being encoded in BFs is preserved, while the second requirement homogenises the similarities across the different dimensions of the encodings. We now present an extended PPRL protocol and describe its essential parts in detail, as also outlined in Fig. 3.

Bloom filter hardening with autoencoders In addition to the BF encoding step being the same as in the basic protocol from Fig. 1, the extended protocol consists of an *encoding*, a *mapping*, and a *linking* step.

In the *encoding* step, each DO trains their own autoencoder model as shown in Fig. 3. Each layer is fully connected with the next layer since we cannot make any assumptions about the order of bits in BFs (as shown in Fig. 2). As activation function, we use the below function (which we call *leaky-capped ReLU*), where x is the sum of the input values of a neuron multiplied by the trainable weights, and α is a leakage parameter:

$$SR_\alpha(x) = \begin{cases} \alpha \cdot x & x < 0 \\ x & 0 \le x < 1 \\ 1 + \alpha(x-1) & 1 \le x \end{cases} \tag{1}$$

The use of this activation function is motivated by the fact that the correct output values of the autoencoder can only be 0 or 1, so any values outside the interval [0, 1] are handled by mapping them to the boundary of the interval, which is attained by choosing $\alpha = 0$. This would, however, result in a partially constant activation function for values below 0 or larger than 1, and thus, the gradient would be zero for those values, which is undesirable for training.

For each BF of DOs A and B, the resulting encoders f_A and f_B compute the corresponding encoding. To ensure that each dimension of the encodings has the same scale, we normalise the generated encodings using the *Mahalanobis* transformation [24], as illustrated in Algorithm 1. To transform the encodings E_i of a DO A or B by Mahalanobis, the covariance matrix C of E_i and the inverse square root matrix T of C are computed. The encodings E_i are then transformed by computing the dot product between E_i and T.

Due to different autoencoders resulting from different training data held by the DOs, the generated encodings of the

Algorithm 1 Mahalanobis transformation applied on the encoded records E_i from data owner $i \in \{A, B\}$.

```
1: function MAHALANOBIS($E_i$)
2:     $E_i \leftarrow E_i$ - COLUMN_MEAN($E_i$)          ▷ Row-wise subtraction of column means
3:     $C \leftarrow 1/(n-1) \cdot E_i^T \cdot E_i$          ▷ Calculate covariance matrix, where $n$ is the number of encodings in $E_i$
4:     $T \leftarrow$ INVERSE_MATRIX_ROOT($C$)          ▷ Calculate the square root of the inverse of the covariance matrix
5:     return $E_i \cdot T$
6: end function
7: function INVERSE_MATRIX_ROOT($M$)
8:     $\Lambda, B \leftarrow$ EIGEN($M$)          ▷ Compute the list of eigenvalues $\Lambda$ and the corresponding list of eigenvectors $B$ of $M$
9:     $S \leftarrow$ DIAG($1/\sqrt{\lambda}\ for \lambda\ in\ \Lambda$)          ▷ Compute the diagonal matrix using the eigenvalues
10:    $T \leftarrow B \cdot S \cdot B^T$          ▷ Determine the inverse square root matrix
11:    return $T$
12: end function
```

same or similar BFs are potentially quite dissimilar, since the internal representation of the learned function is highly sensitive to the training data. Therefore, a direct comparison of encodings does not lead to meaningful results. To guarantee comparability, in the *mapping* step the LU trains a function m enabling the transformation of encodings E_B from DO B to the vector space of DO A, as we describe in detail below.

The normalised encoded BFs, E_A and E_B, from both DOs A and B are sent to the LU where the encodings E_B are transformed using the trained function m. In the *linking* step, the LU calculates the similarities between the transformed encodings E_B and E_A being used to classify record pairs into matches and non-matches according to a similarity threshold δ. Due to the high computational effort for evaluating the full Cartesian product between the encodings E_A and E_B by the LU, we use an approximate nearest neighbour method [17] to reduce the number of encoded record pairs to be compared. As similarity measure, we use the Cosine similarity and employ a threshold for classifying record pairs as matches and non-matches. At the end, the classified matches are sent as the result of the linkage process from the LU back to the DOs, where each match consists of a pair of record identifiers [7].

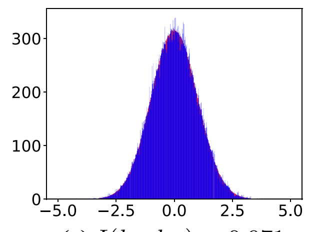
Comparing separately generated encodings The crucial issue of separated encoder models is that the resulting encodings are not directly comparable. Therefore, the LU trains a mapping function m to map an encoding $e \in E_B$ to the space of E_A. Training such a mapping function requires knowledge of a large number of pairs of encodings, generated by the two encoder networks, for the same BF. Due to privacy issues, it is, however, impracticable to generate such a set of BFs that can be shared between the DOs and the LU.

Therefore, we propose a different method for generating training data, exploiting the fact that decoders can generate records that resemble actual data, when fed with random noise of the same distribution as the actual encodings. Having two autoencoders $A_A = g_A \circ f_A$ and $A_B = g_B \circ f_B$ (for two DOs A and B), the relevant mapping is given by $m : f_B(b) \mapsto f_A(b)$ for any BF b. This mapping can formally be approximated by $m = f_A \circ g_B$, where f_A and g_B are known. Due to the sensitivity regarding the privacy aspects we will discuss in Sect. 5, both the decoder function g_B of DO B and the encoder function f_A of DO A are learnt by the DOs independently and not shared with any other party. To determine a model for the function m, we generate pairs $(x, m(x))$, where $x \in \mathbb{R}^d$ is a random point from the encoding space. To prevent having to send g_B and f_A, as well as the original BFs to the LU, we use several steps as we describe next. This generation of training data is shown in the *Mapping* box in Fig. 3:

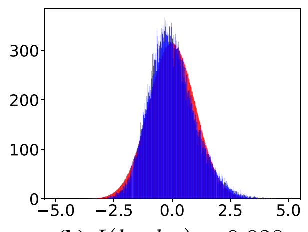
1. The LU generates a list of random vectors R from the encoding space $\mathbb{R}^d$. As the Mahalanobis normalisation from Algorithm 1 is applied to encodings, and those are approximately normally distributed based on the *Central Limit Theorem* [23], these vectors can be drawn from a d-dimensional standard normal distribution, where d is the dimension of the encoded vectors generated by the encoders. The generated random vectors R are sent from the LU to DO B.
2. DO B applies the inverse Mahalanobis transformation with the parameters of its normalisation on the received random data from the LU and decodes the normalised random vectors R with its decoder g_B.
3. The resulting BF-like bit vectors are sent to DO A that encodes them with its encoder f_A and normalises the resulting vectors by applying the Mahalanobis transformation (using the same parameters as for normalising its real encodings). DO A then sends these vectors back to the LU. The resulting set of encodings R' represents the output of the concatenation of the decoder g_B and the encoder f_A.

The randomly generated encodings R and the computed encodings R' are utilised by the LU to train a NN representing the mapping function m to map DO B's encoded data to DO A's encoding space.

While in general the LU should have minimal information about the encoding, having some knowledge about the complexity of the autoencoders used by the DOs does not



(a) $I(he, hn) = 0.971$



(b) $I(he, hn) = 0.928$

Fig. 4 Example of histograms of two dimensions resulting in different histogram intersections for a set of 50,000 encodings he (blue) and the corresponding normal distribution hn (red) using $b = 634$ bins

enable it to decode the actual encodings E_A and E_B. It will, however, be beneficial for choosing a suitable network layout, as this network should replicate the concatenation of B's decoder and A's encoder. Choosing the layout too small would result in a low accuracy of the mapping whereas an overly complex layout could lead to overfitting [2]. Therefore, we choose a configuration to simulate the concatenation of a decoder and an encoder, such that the first half replicates the layer dimensions of the decoder, while the second half replicates the layer dimensions of the encoder. We thereby obtain a NN with input and output dimensions being equal to the encoding dimension d.

5 Discussion of privacy aspects

The goal of applying autoencoders on sensitive data encoded into BFs is to increase the robustness of these encodings against attacks. We claim that this encoding method can decrease the vulnerability of BF encoding by improving the following privacy aspects:

1. The majority of existing attacks on BFs exploit information about the frequencies of bit patterns in sets of BFs and the corresponding q-gram frequencies [47].
   Due to the transformation of BFs from $\{0, 1\}^l$ into encodings E in $\mathbb{R}^d$, existing attacks on bit patterns [8,46] are not applicable. This is because pattern analysis methods such as frequent pattern mining cannot be used to determine frequent 1-bit patterns from numerical vectors in $\mathbb{R}^d$.
   No existing attacks can determine a correlation between the frequency of q-grams and the frequency of a certain pattern of numerical values.
2. The dimensionality reduction from l to d (with $d < l$) results in information loss which potentially is relevant for an attack. This loss therefore decreases the accuracy of a possible attack on the encodings generated by the autoencoders.

The transformation of BFs into numerical vectors might, however, result in new patterns in the data that were not available previously. We therefore need to assess the distribution of the encodings in order to establish their resilience against privacy attacks.

Bloom filter reconstruction We first consider strategies for reconstructing BFs based on the generated encodings and information about the trained autoencoder models.

The task of the decoders, g_A and g_B, is to reconstruct BFs. Therefore, the DOs have to guarantee that these models are secure and they are not shared. Moreover, the encoders, f_A and f_B, allow the generation of training data that can be used to train a NN that determines the inverse mapping of the encoder and therefore replicates the decoder. Therefore, the encoders must also be kept private by the DOs.

Similar to the autoencoders, publication of the mapping function m by the LU also imposes a security risk, because it would allow DO A to transform the encoding from DO B into its vector space and decode the results, while DO B could run a similar attack by training an inverse mapping. After that, DO A could use its decoder to approximately reconstruct the BFs of DO B, from which it can potentially identify q-grams in the QIDs held by DO B.

Considering this potential attack, the LU would have to collude with one DO by releasing the private information about the relation between the different encodings in the form of the mapping function m. Furthermore, the adversary would have to gain access to the DO's encoded data set E_i, either directly or via the LU.

In addition to the publication of the function m, a further risk is a decomposition of m so that the LU can generate a function g_{LU} that can decode encodings E_B from DO B. Due to the loss in the training process of m, we assume that the component of m approximating g_B will results in inaccurately decoded BFs. We plan to investigate possible attacks based on the decomposition of m by the LU as future work.

Distribution analysis Similarly to the pattern-based attacks on BFs [6,46], we assume that our autoencoder-based encoding might be vulnerable to attacks if it is possible for an adversary to extract specific characteristics from similarities in the encodings, for example by clustering groups of vectors of encodings. The resulting clusters might contain information about encodings, such as common q-gram combinations, which could be assigned to characteristics of plain-text values (like q-grams) using frequency information extracted from plain-text data such as telephone directories or voter databases [47].

To analyse the possible effectiveness of such cluster-based attacks, we consider the distribution of encodings in the generated numerical vector space. We assume that an arbitrary clustering approach does not result in accurate and well separated clusters if the encodings generated by the autoencoders have a distribution that is close to a multidimensional normal distribution (a normal distribution in all d dimensions). We therefore evaluate how closely the generated encodings approximate such a normal distribution as an indicator for their vulnerability with regard to such a clustering attack.

To quantify the similarity of the distribution of a set of autoencoder encodings compared to a normal distribution $\mathcal{N}(0, 1)$, we use the histogram intersection measure [38,45][1]. For each of the d dimensions, we generate a histogram he of the encodings using b bins of equal width and similarly generate a histogram hn for a normal distribution with the same width and the same number of bins and data points as for the number of encodings. We automatically determine the width and the number of bins using the approach by Freedman and Diaconis [15]. We show examples of these normal distributions for two selected dimensions in Fig. 4.

We calculate the histogram intersection $I(he, hn)$ as follows [38], where $he[i]$ and $hn[i]$ represent the number of data points in bucket i (with $1 \leq i \leq b$):

$$I(he, hn) = \sum_{i=1}^{b} \min(\text{he[i]}, \text{hn[i]}). \tag{2}$$

To obtain a value between 0 and 1, we normalise $I(he, hn)$ by the number of records, and to obtain a single privacy evaluation measure, we calculate the average of the $I(he, hn)$ over all d dimensions. The closer the resulting value is to 1.0, the more similar the distribution of encodings is to a normal distribution.

Vulnerability to similarity attacks Attacks on PPRL based on similarity graphs [11,44] compare a graph generated when comparing plain-text values with a graph generated comparing encoded values, where the aim is to determine correspondences between plain-text values and encoded values based on node features. The success of a similarity graph attack depends on the comparability of both similarity graphs, and therefore, any PPRL method that calculates accurate similarities between encodings can be vulnerable to a similarity attack [47]. Our autoencoder-based PPRL approach also calculates similarities; therefore, we cannot prevent similarity attacks completely. Nevertheless, due to the use of encodings in $\mathbb{R}^d$, a mapping between the different similarity spaces is not trivially derivable. We plan to investigate how to prevent similarity attacks on our approach in the future.

[1] Because we do not have probability distributions we cannot use Kullback–Leibler divergence or similar measures, while tests for normal distributions such as the Shapiro–Wilk test [37] are known not to work well on large data sets.

Table 1 Used QID attributes and the average number of q-grams per record from the NCVR (N) and Ohio (O) voter data sets used to generate BFs

Data Set	Attributes	Avr. q-gram
N-A4	First name, Last name, Middle name, Year of birth	14.8
N-A5	N-A4 ∪ {City}	21.4
O-A4	First name, Last name, Middle name, Birth date	14.3
O-A5	O-A4 ∪ {City}	21.4

6 Experimental evaluation

In this section, we evaluate our proposed autoencoder-based technique using real-word data sets. We first compare the linkage results of our technique considering a range of autoencoder layouts. We then compare our technique with a standard BF-based PPRL method as baseline, as well as existing hardening techniques [30].

Data sets To evaluate our proposed encoding technique, we use voter registration databases from the US states of North Carolina (N)[2] and the Ohio (O)[3] as used by Franke et al. [14]. We use the same subsets obtained by selecting records from two different snapshots with a certain overlap in matching records and different ratios of variations and errors per record. The North Carolina subset consists of two data sources of 50,000 sampled records each where 10,000 record pairs are matching, while the Ohio voter files consist of two data sources containing 120,000 records and 80,000 records, respectively, with 40,000 matching record pairs.

We consider two sets, A4 and A5, of different QID attribute combinations from which we generate the q-gram sets to be encoded, as shown in Table 1. We use $q = 2$ (bigrams) and employ $k = 30$ hash functions and BFs of length $l = 1024$ using Random Hashing [33] for both data sets, where we generate one BF per record.

Experimental setup To evaluate the BF baseline using the original BFs and existing hardening techniques, we use the Dice coefficient to calculate similarities between BFs [7] and the cosine similarity for the autoencoder encodings. To compare the linkage quality of the different methods, we calculate the area under the precision–recall curve (AUC-PR) [12] with respect to different similarity thresholds δ to classify matches, ranging from 0.4 to 1 in intervals of 0.02. To efficiently compare encodings and BF, we use the Annoy library[4] for nearest neighbour search. To facilitate repeatability, we make our code and data sets available at https://github.com/vicolinho/pprl_autoencoder.

[2] https://www.ncsbe.gov/.

[3] https://www.ohiosos.gov/.

[4] https://github.com/spotify/annoy.

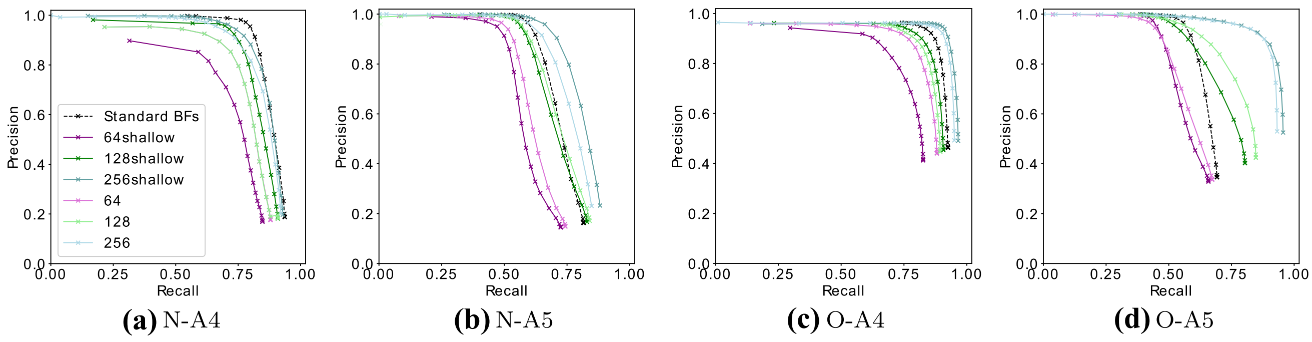


Fig. 5 Precision–recall curves showing results for different autoencoder models with different numbers of dimensions (64, 128, and 256) and their complexity (normal versus shallow) for the four data sets

For our proposed method, we evaluate different autoencoder layouts. In order to generate the training data for the mapping function m, we sampled 200,000 d-dimensional standard normal distributed vectors (where d is the encoding dimension). The required training pairs $(x, m(x))$, where $x, m(x) \in \mathbb{R}^d$, are then generated in the *mapping* step described in Sect. 4.

Autoencoder layouts and parameters For the autoencoders used in our encoding technique, we evaluated different layouts to investigate the resulting linkage quality. Specifically, we considered different dimensions of the encoding layer ($d = 64$, 128, and 256) and two different depths for the network where the shallow network consists of three layers (input, encoding, and output) while the other network includes an additional hidden layer of 512 neurons, both in the encoder and in decoder networks. We set the value of the customised activation function in Eq. (1) to $\alpha = 0.2$ for all layouts as this provided good results in setup experiments.

Figure 5 shows the linkage quality for the four data sets. We observe that an increase in the dimension leads to an improvement of quality. For instance, the AUC-PR values increase by up to 0.1 for N-A5 and by around 0.2 for O-A5 when using 256 rather than 128 dimensions for the shallow layout autoencoder model.

The network complexity influences the quality depending on the number of dimensions. The models with an additional layer perform better up to a certain dimension, namely 128 for N-A5 and O-A5. With more dimensions, the shallow models performed better than the normal ones, as can be seen from the PR curves of the models with 256 dimensions (where the shallow model performs better). We hypothesise that autoencoders with an extra layer can represent complex patterns more effectively than shallow models up to a certain dimensionality. However, the additional layer seems to lead to overfitting for higher-dimensional models because linkage quality decreases.

Comparison with hardening methods We now compare our proposed method with the hardening techniques [30] XOR, WXOR, BLIP, Rule-90 (R90), and Resample (RES), as we described in Sect. 2. In Table 2, we show the AUC-PR results for the different data sets and hardening techniques.

To investigate the impact of different BF encodings, we use different numbers of hash functions, k. The results we obtain indicate that our autoencoder-based technique achieves results comparable to existing hardening techniques. Our approach outperforms the BF-based approaches for the larger data sets O-A4 and O-A5 when using the shallow networks with 256 dimensions.

We explain these improvements in linkage quality in that the autoencoder learns to distinguish differences in BFs resulting from rare q-gram variations compared to common variations. Rare variations are usually a non-frequently occurring character sequence, and therefore, the corresponding 1-bit patterns do occur rare in BFs. Due to their rareness, their impact on the loss function is negligible if these 1-bit patterns are ignored during the training of autoencoders.

Moreover, the results show the robustness of the results regarding linkage quality when we consider different fill ratios, collision ratios, and number of hash functions. This robustness can be seen by similar results obtained with a difference below 0.07 AUC-PR for 256 dimensions considering different attribute combinations being encoded. In contrast to the autoencoder method, the results of using BF-based methods show drops of up to 0.25 AUC-PR between the O-A4 and O-A5 data sets for $k = 20$.

We conclude that BF-based methods are more sensitive with regard to the number of hash functions and the ratio of collisions in BFs. In general, a smaller number of hash functions leads to a decreasing linkage quality, while a higher average collision ratio per record results in a lower AUC-PR, as given in Table 2. The sensitivity of the encoding parameters is also shown for the data set N4, as the BF-based methods lead to a higher AUC-PR by 0.01 compared to our method only for the configuration k=30 and a certain fill rate.

Privacy analysis We now discuss potential privacy risks based on the analysis of the two data sets. Similarly to pattern mining attacks [6,46] the encodings are vulnerable if they are clearly separable and the separated encodings can be mapped

Table 2 AUC of precision-recall curves for different hardening techniques and autoencoder layouts considering 128 and 256 dimensions, as well as the shallow (s) and normal models for the NCVR and Ohio data set with different attribute combinations and different numbers of hash functions k. The best results are highlighted in bold font

	Data			Autoencoder based				Existing hardening techniques					
k	set	Avr. 1-bit	Avr. coll.%	128s	128	256s	256	None	XOR	WXOR	R90	BLIP	RES
30	N-A4	358.5	23.9	0.82	0.78	0.87	0.85	**0.88**	**0.88**	**0.88**	**0.88**	**0.88**	0.86
	N-A5	475.4	35.3	0.71	0.72	**0.81**	0.77	0.73	0.72	0.72	0.73	0.72	0.71
	O-A4	347.0	23.4	0.85	0.84	**0.92**	0.91	0.88	0.88	0.88	0.88	0.87	0.84
	O-A5	490.4	31.2	0.73	0.79	**0.93**	0.91	0.66	0.67	0.66	0.66	0.65	0.64
20	N-A4	256.6	15.3	0.80	0.82	**0.85**	**0.85**	0.73	0.72	0.73	0.72	0.72	0.70
	N-A5	349.7	22.6	0.73	0.74	**0.81**	0.77	0.75	0.75	0.75	0.75	0.75	0.72
	O-A4	248.0	15.2	0.73	0.76	**0.89**	**0.89**	0.88	0.88	0.88	0.88	0.87	0.84
	O-A5	347.1	23.6	0.66	0.68	**0.89**	0.88	0.63	0.64	0.63	0.63	0.63	0.61

Table 3 Average histogram intersection of encodings for the four data sets compared to a multidimensional normal distribution, as discussed in Sect. 5

Data Set	Layout					
	64s	64	128s	128	256s	256
N-A4	0.959	0.959	0.963	0.949	**0.964**	0.955
N-A5	0.961	**0.969**	0.964	0.958	0.966	0.961
O-A4	0.950	0.924	0.963	0.946	**0.968**	0.956
O-A5	0.960	**0.977**	0.968	0.966	0.973	0.964

to a corresponding clear text value or q-gram cluster [47]. As an indicator how well a data set is separable, we proposed a method for measuring how similar our encodings are to a multidimensional normal distribution.

Therefore, we quantify the similarity by analysing the distribution of the data set using the distribution analysis described in Sect. 5. The average histogram intersection regarding all dimensions is shown in Table 3.

We observe that the average histogram intersection is higher for shallow models compared to models with an extra hidden layer except for $d = 64$ using 5 attributes. Moreover, an increasing number of dimensions leads to higher histogram intersection results considering the shallow networks.

7 Conclusion

Privacy-preserving record linkage is an essential process for integrating sensitive data [7], where Bloom filter (BF) encoding is a popular technique used to efficiently mask plain-text values and facilitate similarity calculations between encoded values. However, research has shown the vulnerability of BF encoding with regard to various attacks [47]. This has led to the development of multiple hardening techniques which manipulate BFs such that the likelihood of associating a given BFs or its bit pattern to a plain-text value, and therefore any possible re-identification, decreases [14,30].

In this paper, we have proposed a novel encoding technique based on autoencoders [1] which transforms BFs into numerical vectors. These vectors prevent existing attacks that have shown to be successful on BFs. Moreover, compared to other hardening techniques for BFs, our technique generates high-quality linkage results by training a mapping function which transforms the encodings of one DO into the vector space of the other DO. This transformation guarantees the comparability of numerical vectors. We showed that our technique can outperform existing hardening techniques for BF both in terms of privacy and linkage quality.

In future work, we plan to analyse different autoencoder architectures for the encoding process and investigate the vulnerability of our method to clustering and graph-based attacks in more detail [11]. Clustering attacks are similar to pattern mining attacks [46] on BFs in that they exploit the similarities between the frequencies of patterns in plain-text and encoded values. Graph-based attacks aim at aligning the nodes in two similarity graphs generated from a plain-text and an encoded data set, respectively, based on attribute and neighbourhood similarities [44].

Author Contributions VC and TH developed the methods proposed in this work, TH implemented the methods, VC run the experiments, and all authors worked on the manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL. No support was received for the submitted work.

Declarations

Conflict of interest The authors declare that they have no conflicts of interest.

References

1. Bank, D., Koenigstein, N., Giryes, R.: Autoencoders. CoRR arXiv:2003.05991 (2020)
2. Bejani, M.M., Ghatee, M.: A systematic review on overfitting control in shallow and deep neural networks. Artif. Intell. Rev. **54**(8), 6391–6438 (2021)
3. Binette, O., Steorts, R.C.: (Almost) all of entity resolution. Sci. Adv. **8**(12), eabi8021 (2022)
4. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Commun. ACM **13**(7), 422–426 (1970)
5. Boyd, J.H., Randall, S.M., Ferrante, A.M.: Application of privacy-preserving techniques in operational record linkage centres. In: Gkoulalas-Divanis, A., Loukides, G. (eds.) Medical Data Privacy Handbook. Springer, New York (2015)
6. Christen, P., Vidanage, A., Ranbaduge, T., Schnell, R.: Pattern-mining based cryptanalysis of Bloom filters for privacy-preserving record linkage. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. pp. 628–640. Springer, Melbourne (2018)
7. Christen, P., Ranbaduge, T., Schnell, R.: Linking Sensitive Data. Springer, Heidelberg (2020)
8. Christen, P., Ranbaduge, T., Vatsalan, D., Schnell, R.: Precise and fast cryptanalysis for Bloom filter based privacy-preserving record linkage. Transactions Knowl. Data Eng. **18**(11), 2164–2177(2018)
9. Christen, P., Schnell, R.: Common misconceptions about population data. arXiv preprint arXiv:2112.10912 (2021)
10. Christen, P., Schnell, R., Vatsalan, D., Ranbaduge, T.: Efficient cryptanalysis of Bloom filters for privacy-preserving record linkage. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. , vol. LNAI 10234, pp. 628–640. Springer, Jeju, Korea (2017)
11. Culnane, C., Rubinstein, B.I., Teague, V.: Vulnerabilities in the use of similarity tables in combination with pseudonymisation to preserve data privacy in the UK Office for National Statistics' privacy-preserving record linkage. arXiv Preprint (2017)
12. Davis, J., Goadrich, M.: The relationship between precision-recall and ROC curves. In: International Conference on Machine Learning. pp. 233–240. ACM, Pittsburgh (2006)
13. Dwork, C., Talwar, K., Thakurta, A., Zhang, L.: Analyze Gauss: optimal bounds for privacy-preserving principal component analysis. In: Symposium on Theory of Computing. pp. 11–20. ACM, New York (2014)
14. Franke, M., Sehili, Z., Rohde, F., Rahm, E.: Evaluation of hardening techniques for privacy-preserving record linkage. In: Proceedings of the 24th International Conference on Extending Database Technology, EDBT 2021, Nicosia, Cyprus, 23-26 March 2021, pp. 289–300 (2021)
15. Freedman, D., Diaconis, P.: On the histogram as a density estimator:l2 theory. Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete **57**, 453–476 (1981)
16. Gkoulalas-Divanis, A., Vatsalan, D., Karapiperis, D., Kantarcioglu, M.: Modern privacy-preserving record linkage techniques: an overview. Transactions Informations Forensics Secur. (2021)
17. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: Symposium on the Theory of Computing. pp. 604–613. ACM, Dallas (1998)
18. Karakasidis, A., Verykios, V.S., Christen, P.: Fake injection strategies for private phonetic matching. In: International Workshop on Data Privacy Management. Leuven, Belgium (2011)
19. Karapiperis, D., Gkoulalas-Divanis, A., Verykios, V.S.: Distance-aware encoding of numerical values for privacy-preserving record linkage. In: IEEE International Conference on Data Engineering. pp. 135–138. San Diego (2017)
20. Kroll, M., Steinmetzer, S.: Who is 1011011111...1110110010? Automated cryptanalysis of Bloom filter encryptions of databases with several personal identifiers. In: International Joint Conference on Biomedical Engineering Systems and Technologies. pp. 341–356. Lisbon (2015)
21. Kuzu, M., Kantarcioglu, M., Durham, E., Malin, B.: A constraint satisfaction cryptanalysis of Bloom filters in private record linkage. In: International Symposium on Privacy Enhancing Technologies Symposium. pp. 226–245. Springer (2011)
22. Kuzu, M., Kantarcioglu, M., Durham, E.A., Toth, C., Malin, B.: A practical approach to achieve private medical record linkage in light of public resources. J. Am. Med. Inform. Assoc. **20**(2), 285–292 (2013)
23. Le Cam, L.: The central limit theorem around 1935. Statistical Sci. **1**(1), 78–91 (1986)
24. Mahalanobis, P.C.: On the generalized distance in statistics. Proc. Nat. Inst. Sci. (Calcutta) **2**, 49–55 (1936)
25. Mitchell, W., Dewri, R., Thurimella, R., Roschke, M.: A graph traversal attack on Bloom filter-based medical data aggregation. Int. J. Big Data Intell. **4**(4), 217–226 (2017)
26. Narayanan, A., Shmatikov, V.: Myths and fallacies of personally identifiable information. Commun. ACM **53**(6), 24–26 (2010)
27. Newcombe, H., Kennedy, J., Axford, S., James, A.: Automatic linkage of vital records. Science **130**(3381), 954–959 (1959)
28. Niedermeyer, F., Steinmetzer, S., Kroll, M., Schnell, R.: Cryptanalysis of basic Bloom filters used for privacy preserving record linkage. German Record Linkage Center, Working Paper Series, No. WP-GRLC-2014-04 (2014)
29. Pita, R., Pinto, C., Sena, S., Fiaccone, R., Amorim, L., Reis, S., Barreto, M., Denaxas, S., Barreto, M.: On the accuracy and scalability of probabilistic data linkage over the Brazilian 114 million cohort. J. Biomed. Health Inform. **22**(2), 346–353 (2018)
30. Ranbaduge, T., Schnell, R.: Securing Bloom filters for privacy-preserving record linkage. In: International Conference on Information and Knowledge Management. pp. 2185–2188. ACM, Galway (2020)
31. Randall, S.M., Ferrante, A.M., Boyd, J.H., Bauer, J.K., Semmens, J.B.: Privacy-preserving record linkage on large real world datasets. J. Biomed. Inform. **50**, 205–212 (2014)
32. Schnell, R., Bachteler, T., Reiher, J.: Privacy-preserving record linkage using Bloom filters. BMC Med. Inform. Decisi. Mak. **9**(1), 1–11 (2009)
33. Schnell, R., Borgs, C.: Randomized response and balanced Bloom filters for privacy preserving record linkage. In: International Conference on Data Mining Workshops. pp. 218–224. IEEE, Barcelona (2016)
34. Schnell, R., Borgs, C.: XOR-folding for Bloom filter-based encryptions for privacy-preserving record linkage. German Record Linkage Center 22 (2016)
35. Schnell, R., Borgs, C.: Hardening encrypted patient names against cryptographic attacks using cellular automata. In: International Conference on Data Mining Workshops. pp. 518–522. IEEE, Singapore (2018)

36. Schnell, R., Borgs, C.: Encoding hierarchical classification codes for privacy-preserving record linkage using Bloom filters. In: Workshop on Data Integration and Applications. held at ECML/PKDD, pp. 142–156. Springer, Würzburg (2019)
37. Shapiro, S.S., Wilk, M.B.: An analysis of variance test for normality. Biometrika **52**(3/4), 591–611 (1965)
38. Swain, M.J., Ballard, D.H.: Color indexing. Int. J. Comput. Vis. **7**(1), 11–32 (1991)
39. Vaiwsri, S., Ranbaduge, T., Christen, P.: Reference values based hardening for Bloom filters based privacy-preserving record linkage. In: Australasian Conference on Data Mining. pp. 189–202. Springer, Bathurst (2018)
40. Vaiwsri, S., Ranbaduge, T., Christen, P.: Accurate and efficient privacy-preserving string matching. Int. J. Data Sci. Anal. **14**, 191–125(2022)
41. Vatsalan, D., Christen, P.: Privacy-preserving matching of similar patients. J. Biomed. Inform. **59**, 285–298 (2016)
42. Vatsalan, D., Christen, P., Verykios, V.S.: A taxonomy of privacy-preserving record linkage techniques. Information Syst. **38**(6), 946–969 (2013)
43. Vatsalan, D., Sehili, Z., Christen, P., Rahm, E.: Privacy-preserving record linkage for Big Data: current approaches and research challenges. In: Zomaya, A.Y., Sakr, S. (eds.) Handbook of Big Data Technologies. Springer, New York (2017)
44. Vidanage, A., Christen, P., Ranbaduge, T., Schnell, R.: A graph matching attack on privacy-preserving record linkage. In: International Conference on Information and Knowledge Management. pp. 1485–1494. ACM (2020)
45. Vidanage, A., Ranbaduge, T., Christen, P., Randall, S.: A privacy attack on multiple dynamic match-key based privacy-preserving record linkage. Int. J. Popul. Data Sci. **5**(1),13 (2020)
46. Vidanage, A., Ranbaduge, T., Christen, P., Schnell, R.: Efficient pattern mining based cryptanalysis for privacy-preserving record linkage. In: International Conference on Data Engineering. IEEE, Macau (2019)
47. Vidanage, A., Ranbaduge, T., Christen, P., Schnell, R.: A taxonomy of attacks on privacy-preserving record linkage. J. Priv. Confid. **12**(1), 35 (2022)