Strengthening Privacy-Preserving Record Linkage using Diffusion

Frederik Armknecht
University of Mannheim
Mannheim, Germany
armknecht@uni-mannheim.de

Youzhe Heng
University of Mannheim
Mannheim, Germany
y.heng@uni-mannheim.de

Rainer Schnell
University of Duisburg-Essen
Duisburg, Germany
rainer.schnell@uni-due.de

ABSTRACT

Linking personal records from different databases is an essential step in many data workflows. Privacy-Preserving Record-Linkage (PPRL) techniques have been developed to link persons despite errors in the identifiers without violating their privacy. Designing efficient PPRL schemes with high linkage quality and a strong level of privacy protection is challenging. PPRL based on Bloom filter encoding (BF) is currently one of the most popular methods as they offer high efficiency and linkage quality. However, it turned out that these schemes are vulnerable to several attacks, with pattern mining and graph matching attacks considered to be the most serious by far. While several proposals have been made to strengthen BF-based PPRL schemes against these attacks, all these lack a proper security analysis or do not preserve the high efficiency and linkage quality. This paper shows that both problems can be addressed by extending the scheme with an appropriate linear diffusion layer. As opposed to previous schemes, we provide extensive theoretical and experimental analysis that confirms that the resulting scheme provides high efficiency and linkage quality *and* significantly increases security against the attacks mentioned above.

KEYWORDS

Record Linkage, Bloom Filters, Diffusion, Cryptanalysis.

1 Introduction

Vast amounts of data have been collected due to the digital transformation of commercial and administrative processes. This data is used for many purposes, including research. Notably, new research interests have been created by the ability to link multiple databases. The process linking entities from different databases while still preserving privacy is called privacy-preserving record linkage (PPRL) [8]. The three main challenges of a PPRL technique are (1) scalability or efficiency, (2) linkage quality and (3) level of privacy protection.

To provide privacy, PPRL schemes usually encode the records and execute the linkage process on the encoded records only. Among different encoding techniques used in PPRL, the use of Bloom filter encoding (BF) has been proposed in [19]. Due to their high efficiency and linkage quality level, BF-based PPRL has now been widely used in real applications [2, 5]. However, research [14, 17, 25, 26] has also shown that such schemes can be vulnerable to a variety of attacks. In particular, pattern mining attack [26] and graph matching attack [25] are considered the most practical and advanced threats.

Consequently, various Bloom filter hardening techniques have been proposed to enhance the scheme's security, including salting, balancing, adding random noise, and rule 90 [8, 22]. However, all these techniques reduce the linkage quality or the efficiency [8, 10]. Moreover, as is remarked by [8], no proper security analysis has been provided for any of these hardening techniques. Thus, it was an open question of whether an extension of BF-based PPRL does exist that is (i) able to preserve their advantages and (ii) increases security against the two attacks mentioned above.

In this work, we positively answer this question. The core idea is to adopt an established concept from cipher design and append a linear diffusion layer at the end of the BF-based PPRL scheme. Consequently, we refer to this concept as an BFD scheme (Bloom filter with diffusion). In a nutshell, the idea is that each output bit of the encoding process is a linear combination of secretly chosen BF bits.

Even though this idea seems to be obvious, it hasn't been fully explored so far. For instance, a similar approach was considered in [22]. The authors proposed to apply the so-called 'Rule 90' [28], so that each bit position in Bloom filter is modified by bitwise XORing the predecessor and successor bit.

In [18], randomly sampling bits from Bloom filters and XORing them was suggested. However, for none of these propositions (or any other hardening technique proposed so far), any security analysis has been given.

In contrast, one of the major contributions of this work is the extensive theoretical and experimental analysis of the proposed scheme. Our theoretical analysis shows superior security compared to the basic BF-based PPRL approach while preserving good linkage quality. As a kind of side result, it also shows that the diffusion induced by 'Rule 90' as suggested in [22] is insufficient. Some of the analysis techniques may have value on their own and could be applied to other schemes well.

Besides theoretical analysis, extensive experiments confirm the excellent properties of the BFD. Thanks to the simplicity of the extension, high efficiency is also still given. To the best of our knowledge, this is the first (BF-based) effective PPRL scheme providing good linkage quality where a thorough security analysis has been conducted. Moreover, we discuss how concrete parameter settings can be derived.

The rest of the paper is organized as follows. In Section 2, we describe the concept of privacy-preserving record linkage (PPRL), emphasizing the variant based on Bloom filter encoding. In Section 3, we explain the commonly considered attacker model and the two most relevant attacks, namely pattern mining and graph matching attacks. Section 4 gives the rationale of the design and a

Proceedings on Privacy Enhancing Technologies 2023(2), 298–311

https://doi.org/10.56553/popets-2023-0054

formal description of our BFD scheme. The security analysis of the scheme is given in Section 5. In Section 6, we evaluate the scheme for practical applications by investigating efficiency and linkage quality. Parameter choices are discussed in Section 7. Section 8 summarizes the contribution and gives an outlook on potential future work.

2 Privacy-Preserving Record Linkage

2.1 General Description

Privacy-preserving record linkage (PPRL) refers to identifying records in different databases that refer to the same person without revealing the identity of any individual in one of these databases.

Without loss of generality, we restrict our description to the case of two databases. We consider two database holders, who own databases D and D', respectively. One assumes that each record in any database can be written as $r = (\mathsf{ID}, \lambda, \mu)$ where ID is some randomly generated, unique identifier, λ is so-called linkage data that will be used for linking records, e.g., name, and μ is the microdata that is relevant for the analysis. Essentially, the task of record linkage is to identify records $r = (\mathsf{ID}, \lambda, \mu) \in D$ and $r' = (\mathsf{ID}', \lambda', \mu') \in D'$ that according to λ and λ' refer to the same person and hence shall be linked.

In PPRL, this process is executed by a third party called 'linkage unit' $\mathcal{L}$. To prevent any re-identification of individuals from λ, each database holder applies an encoding algorithm to transform the plain linkage data λ into encoded linkage data, written as $[\lambda]$. The linkage unit $\mathcal{L}$ receives encoded linkage data $[\lambda]$ along with their identifiers ID from holders and conducts the linkage using a linkage algorithm. That is, for any pair of tuples $(\mathsf{ID}, [\lambda])$ and $(\mathsf{ID}', [\lambda'])$, the linkage unit $\mathcal{L}$ computes a similarity function sim on the encoded linkage data. If $sim([\lambda], [\lambda'])$ is above a predefined threshold, the records are classified as a potential match and $(\mathsf{ID}, \mathsf{ID}')$ will be recorded and be sent to another party for actually joining the databases D and D'.

2.2 PPRL based on Bloom Filters

As the linkage process in a PPRL scheme is applied on the coded linkage data, the choice of the encoding procedure has a significant impact on the quality of the whole scheme. One encoding procedure that became very popular in this context are Bloom filters. Bloom Filters (BF) [3] were developed to allow to test the membership of set elements without the need to access the set directly. A BF is parameterized by a bit string of length ℓ_{BF} (also called the length of the BF), a universe $\mathcal{U}$ of elements, and k hash functions $h_1, \ldots, h_k$ where $h_i : \mathcal{U} \to \{1, \ldots, \ell_{\mathrm{BF}}\}$ for each i.

Given a set $S \subseteq \mathcal{U}$, first a bit string B of length ℓ_{BF} is initialized where each entry is set to zero, i.e., $B[i] = 0$ for $i = 1, \ldots, \ell_{\mathrm{BF}}$. Then, for each $s \in S$, (up to) k indexes $h_1(s), \ldots, h_k(s) \in \{1, \ldots, \ell_{\mathrm{BF}}\}$ are computed and the bits in B at these positions are set to one. That is, $B[h_i(s)] := 1$ for each $s \in S$ and each hash function h_i.

In the case of BF-based PPRL schemes, Bloom filters are used to encode the linkage data λ as follows. Attributes values contained in λ are represented over some common alphabet Σ. A q-gram (for some integer $q \geq 1$) is a string of length q over characters from Σ. To encode λ into a Bloom filter, it is interpreted as a sequence over Σ, and then, using a sliding window approach, the set of q-grams contained in λ is extracted. For instance, when using $q = 2$ (also known as bigrams), the string "filter" is converted into the set of bigrams $S = \{fi, il, lt, te, er\}$. Then, S is encoded into a ℓ_{BF}-bit string as described above, using k hash functions $h_i : \mathcal{U} = \Sigma^q \to \{1, \ldots, \ell_{\mathrm{BF}}\}$. An example is given in Fig. 1.

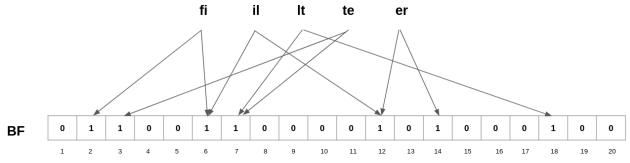

Figure 1: Example of generation of Bloom filters. $\ell_{\mathrm{BF}} = 20, k = 2$

Intuitively, two linkage data λ and λ' are similar if the corresponding sets of q-gram strongly overlap. However, for privacy reasons, the linkage unit $\mathcal{L}$ does not have access to these sets directly but only sees the encoded linkage data $B = [\lambda]$ and $B' = [\lambda']$. In BF-based PPRL schemes the similarity of these binary strings is captured by the Dice coefficient Dice. Formally, the Dice coefficient of two binary strings is defined as

$$dice(B, B') = \frac{2\,|supp(B) \cap supp(B')|}{|supp(B)| + |supp(B')|}, \tag{1}$$

where $supp(B) := \{i | B[i] \neq 0\}$ refers to the set of indices of the bits that are equal to 1. If the Dice coefficient is sufficiently large, i.e. greater or equal to a pre-defined threshold τ, two records $(\mathsf{ID}, \lambda, \mu)$ and $(\mathsf{ID}', \lambda', \mu')$ should be linked.

3 Security

3.1 Model

The workflow of PPRL is that each database holder encodes its database and sends the result to the linkage unit $\mathcal{L}$. No database holder sees the data of any other database holder. However, database holders may agree on shared secrets, e.g., the deployed hash functions in the case of BF-based PPRL, before the encoding step. The linkage unit $\mathcal{L}$ has access to all encoded databases and possibly knows certain meta-data about the data sets, e.g., that they belong to a specific hospital, the individuals are residents of a particular area, etc. The goal of $\mathcal{L}$ as an attacker is to correctly re-identify individuals represented in the databases from the encoded linkage data better than pure guessing [8].

The overall goal of this work is to propose an extension for BF-based PPRL schemes and to show that this modification makes it more secure while preserving its practicability. Ideally, one would rely here on a concise security definition and provide a proof of security based on this definition. Unfortunately, we would face two problems then.

First, the only formal security definition we are aware of is the one provided in [12]. However, it models an interactive scenario while we consider a one-shot-scenario where each database holder obfuscates his database once and sends the result to the linkage unit. The reason for this choice is that this is the predominant use case in practice. For this scenario, there is no formal security definition

(yet). For the same reason, we also rule out interactive PPRL based on multiparty computation, e.g., see [24].

Second, proofs of security reduce the security of one cryptographic scheme to the (alleged) difficulty of another problem. Experience shows that provably secure schemes either rely on problems that are not thoroughly analyzed or the resulting schemes are less efficient than other, non-provably secure schemes. Actually, no symmetric cryptographic scheme in practical use, e.g., the encryption standard AES or the hash functions SHA2/SHA3, is provably secure.

Instead, the conventional approach in the symmetric cryptography community is to analyze the resistance of schemes against the best-known attacks. In this work, we follow this approach. This is also motivated by the fact that BF-PPRL exhibits the characteristics of a symmetric-key cryptographic scheme. On the one hand, secrets are shared between the database holders. On the other hand, the efficiency of PPRL is of high importance.

While several attacks have been found against BF-based PPRL (for an overview, see [13, 27]), the pattern mining [26] and graph matching attacks [25] are considered to be the most powerful attacks against Bloom filter based PPRL techniques among all other attacks. Consequently, we aim to adapt BF-based PPRL to make these attacks less effective without sacrificing the efficiency and linkage quality.

In the following, we describe pattern mining attacks 3.2 and graph matching attacks 3.3. In particular, we identify for each attack a necessary property of BF-based PPRL. These properties will also serve as the basis of the security analysis 3.

3.2 Pattern Mining Attacks

A pattern mining attack [26] is one of the most practical and advanced attack methods as it is quite effective while assuming a weak attacker model. In this attack, the attacker has access to an encoded database and has some knowledge about the distribution of q-grams in plaintext records, e.g., statistics from a population. In particular, the attacker knows the frequency of the most common q-grams.

To explain the attack, we introduce some notation that we will re-use later in Section 5.2 for the security analysis. Let γ denote an arbitrary q-gram, and let

$$\mathcal{I}(\gamma) := \{h_1(\gamma), \ldots, h_k(\gamma)\} \subseteq \{1, \ldots, \ell_{\mathrm{BF}}\}, \tag{2}$$

where h_i denotes the involved hash functions. By definition of Bloom filters, it holds that if of a record contains γ, then we have for the corresponding Bloom filter B that $B[i] = 1$ for all $i \in \mathcal{I}(\gamma)$. For a given set $\mathcal{I} \subseteq \{1, \ldots, \ell_{\mathrm{BF}}\}$, we denote by $\mathbb{E}_{\mathcal{I}}$ the event that $B[i] = 1$ for all $i \in \mathcal{I}$. Thus, we can rephrase the statement above as follows:

$$\gamma \text{ occurs} \quad \Rightarrow \quad \Pr(\mathbb{E}_{\mathcal{I}(\gamma)}) = 1. \tag{3}$$

Note that $\mathcal{I}(\gamma)$ usually is unknown to an attacker, due to the use of secret hash functions. However, the attack exploits the fact that frequent q-grams incur frequent '1'-patterns in the Bloom filters.

Given this, let γ^* denote the most frequent q-gram according to the known distribution of q-grams in plaintext records. The attacker searches the Bloom filters for the most frequent patterns, i.e., a set of indices $\mathcal{I}^*$ that are jointly equal to 1 with high frequency. The assumption is that the most frequent pattern results from the most frequent q-gram, i.e., $\mathcal{I}(\gamma) \subseteq \mathcal{I}^*$. In particular, if $\mathbb{E}_{\mathcal{I}^*}$ holds for a given Bloom filter, one assumes that the underlying record includes γ. That is, one q-gram is identified from the underlying records. Analogously, the second-most-frequent pattern is identified and so on. Thus, step by step, certain q-grams are identified in the encoded linkage data and eventually may result in the re-identification of individuals.

In [27], the authors made a taxonomy of all existing attacks and their results. Using the publicly available database called 'NCVR' (North Carolina Voter Registration), in [27] it's stated that at most 27,665 out of 222, 251 records are correctly re-identified by pattern mining attacks.

3.3 Graph Matching Attacks

A graph matching attack [25] is considered to be the most powerful attack against PPRL schemes, but assumes a stronger attacker model. The attacker has access to a plaintext database D and an encoded database $[D']$, being the encoding of some database D'. Such a scenario might be given if for example an attacker has some knowledge about a super-set of the records contained in D', e.g., D' being a database of patient records from a hospital while D is publicly available phone book.

For a graph matching attack, the databases D and D' need to have a significant overlap (in the sense of linkability). This is expressed by the overlap rate $\frac{2 \cdot |\mathrm{Matches}(D, D')|}{|D| + |D'|}$ where $\mathrm{Matches}(D, D')$ is the set of all pairs in $D \times D'$ that should/can be linked. The overlap rate ranges from 0% to 100%, the latter being the case if D equals $[D]$. In [25], it was demonstrated that graph matching attacks are very powerful if the overlap rate is *high*, i.e., very close to 100%. Given this, the attacker aims to identify elements in D' that can be linked to records in D. As D is given in plaintext, this results in a re-identification of the individual encoded in $[D']$.

A second requirement for graph matching attack is that the encoding scheme preserves the level of similarity. In the case of BF-based PPRL, consider any pair of plaintext records with linkage data λ and λ' and let B and B' be the corresponding Bloom filter encodings. Let sim be the function that expresses the similarity between plaintext records while the Dice coefficient expresses the similarity between the Bloom filters (cf. Eq. 1). Then, a graph matching attack exploits that these notions of similarity are correlated. We express this by

$$sim(\lambda, \lambda') \sim \mathrm{Dice}([\lambda], [\lambda']) = \mathrm{Dice}(B, B') \tag{4}$$

For example, in the case of BF-based PPRL, it holds that $\mathrm{Dice}(B, B')$ grows *linearly* with $sim(\lambda, \lambda')$ (cf. Fig. 2).

Thus, if a set of records in D is also contained[1] in $[D]$ (in encoded form), the relation between them in terms of similarity should have a similar structure in both databases. The attacker constructs "similarity graphs" for each database as follows. The nodes of the graphs are the (encoded) entries, and the edges between nodes are weighted with the level of similarity of their endpoints (the Dice coefficient in the case of BF-based PPRL). The idea is now to look for sub-graphs in both graphs structurally similar, i.e., match. The

[1] In the sense of being linked to.

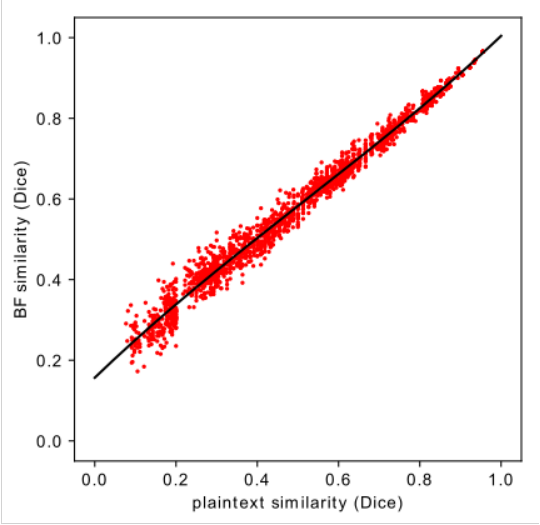

Figure 2: Relation between the similarity of plaintext records and the similarity of the corresponding Bloom filters (using the Dice coefficient in both cases).

assumption is that this indicates that the nodes of the sub-graph constructed from $[D']$ match the nodes in the sub-graph derived from D. As the latter is given in plaintext, these records from $[D']$ have been re-identified.

Using the same database called 'NCVR' as in [26], the authors successfully re-identified 55.1% of the records using graph matching attacks according to [27]. In [15], the authors pointed out that a successful attack should result in more correct re-identifications than random guessing. An equation given by [15] suggests one re-identification as the expected value for our setting. Thus, any privacy attack that correctly identifies significantly more than one record should be considered a success, at least to some extent.

4 The BFD Scheme

4.1 Design Rationale

As motivated in Section 1, the challenge we address in this paper is to extend the BF-based PPRL scheme explained in Section 2.2 such that (i) the extension does not change the original scheme too much to preserve its benefits but (ii) one can show the security is given against pattern mining attacks (Sec. 3.2) and graph matching attacks (Sec. 3.3).

As a pattern mining attack is similar to a frequency attack against the substitution cipher, it is reasonable to consider as a countermeasure the same protection mechanism that also helps to thwart frequency attacks – diffusion. In cryptography, the purpose of diffusion [23] is to hide the statistical relationship between the ciphertext and the plaintext. Commonly, this is achieved by requiring that small changes in the input have a potentially large impact on the output. In the case of BF-based PPRL, we use this concept to break the strong correlation between q-grams (input) and bits of the encoded linkage data (output). More precisely, a linear diffusion layer is appended to the computation of the Bloom filters. Given some Bloom filter B of length ℓ_{BF}, a bitstring E, also called an encoded linkage data, of length ℓ_{ELD} is constructed where each bit in E is the XOR-sum of selected bits from B. An example is shown in Fig. 3. There, BF refers to the Bloom filter alone while BFD stands for the encoding of BF by applying a diffusion layer.

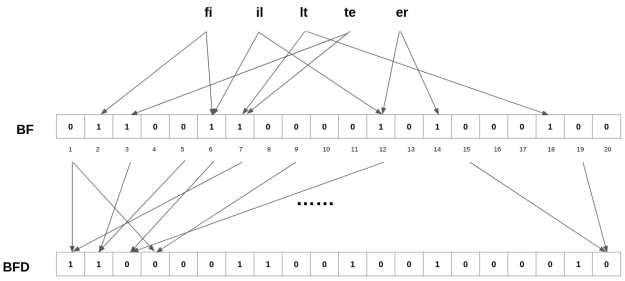

Figure 3: Example of the proposed extension. In the example, each output bit is the sum of two Bloom filter bits.

The diffusion layer may also protect against graph matching attacks. Consider two Bloom filters B and B' and their respective encoded linkage data E and E'. To preserve linkage quality, it needs to hold that if $\text{Dice}(B, B')$ is high, then this implies a high value for $\text{Dice}(E, E')$ as well. This requirement is because we use the same criteria for deciding whether or not encoded linkage data shall be linked. However, the lower $\text{Dice}(B, B')$, the less shall it be correlated to the value of $\text{Dice}(E, E')$. If this is given, the sub-graphs in $[D]$ do not reflect the structure anymore in D, and hence the attack would fail.

4.2 Description

In the following, we provide a complete description of BFD, being the BF-based PPRL scheme extended by a diffusion layer. The BFD scheme comprises three algorithms: (i) an algorithm for choosing the diffusion layer (Sec. 4.2.1), (ii) the encoding of the linkage data (Sec. 4.2.2), and (iii) the linkage algorithm (Sec. 4.2.3).

4.2.1 Choosing the Diffusion Layer. During the setup phase of a BF-based PPRL, i.e., when the database holders agree on shared secrets such as the deployed hash functions, also the diffusion layer needs to be selected. We assume this to be done by one database holder, and the result is shared with the others.

Note that the diffusion layer transforms a Bloom filter $B \in \{0,1\}^{\ell_{\text{BF}}}$ into an encoded linkage data $E \in \{0,1\}^{\ell_{\text{ELD}}}$ where each bit of E is the XOR-sum of some bits from B. Thus, the diffusion layer is specified by a list $\mathcal{I}$ of ℓ_{ELD} sets $\mathcal{I}_j \subset \{1, \ldots, \ell_{\text{BF}}\}$.

Alg. 1 displays the algorithm for selecting the diffusion layer. To maximize the diffusion effect, we aim for two effects. First, the change of any Bloom filter bit should impact as many bits in the encoded linkage data as possible. Second, each bit should depend on as many Bloom filter bits as possible.

To accomplish the first effect, Alg. 1 ensures in lines 4 and 9 that each $\mathcal{I}_j$ is exactly of size t. As we are going to discuss in Sec. 3 and Sec. 6, the parameter t will be one of the main parameters to adjust the level of security and linkage quality.

To achieve the second effect, one would require to ensure that for any indices j_1 and j_2, the intersection $\mathcal{I}_{j_1} \cap \mathcal{I}_{j_2}$ is as small as possible. However, this would require to solve an involved combinatoric problem. To keep this procedure as efficient as possible, we opt for a greedy approach instead. That is, we aim for that each index bit in $\{1, \ldots, \ell_{\text{BF}}\}$ should appear on average in the same number of index sets $\mathcal{I}_j$. This is accomplished by using a set *Indices* that keeps tracks of indices chosen so far. After being initialized to $\{1, \ldots, \ell_{\text{BF}}\}$ (line 1), pairwise disjoint index sets $\mathcal{I}_j$ of size t are removed from *Indices* as long as possible (lines 3–5). Once the number of remaining indices is not sufficient (lines 6–11), the remaining indices are used for the current index set (line 7), *Indices* is re-initialized to $\{1, \ldots, \ell_{\text{BF}}\}$ (line 8), and the procedure continues as before.

Algorithm 1 Choosing diffusion layer

Input: t: The number of selected Bloom filter bits ($t \leq \ell_{\text{BF}}$)
ℓ_{BF}: The length of Bloom filters
ℓ_{ELD}: The length of the encoded linkage data
Output: $\mathcal{I}$: The index list, specifying the diffusion layer

1: $Indices := \{1, ..., \ell_{\text{BF}}\}$ //Initialize set of indices
2: **for** $j \in \{1, ..., \ell_{\text{ELD}}\}$ **do**
3: **if** $|Indices| \geq t$ **then**
4: Choose random $\mathcal{I}_j \subseteq Indices$ with $|\mathcal{I}_j| = t$
5: $Indices := Indices \backslash \mathcal{I}_j$
6: **else** # $|Indices| < t$
7: $\mathcal{I}_j := Indices$
8: $Indices := \{1, ..., \ell_{\text{BF}}\}$ //Reset *Indices*
9: Choose random $S \subseteq (Indices \backslash \mathcal{I}_j)$ with $|S| = t - |\mathcal{I}_j|$
10: $\mathcal{I}_j = \mathcal{I}_j \bigcup S$
11: $Indices := Indices \backslash S$
12: Return $\mathcal{I} := (\mathcal{I}_1, ..., \mathcal{I}_{\ell_{\text{ELD}}})$

4.2.2 Encoding. The encoding algorithm (Alg. 2) first computes the standard BF encoding of linkage data (line 3) and then applies the diffusion layer $\mathcal{I}$ produced by Alg. 1 (lines 4–6). We assume that the parameters of the Bloom filters have been already negotiated between the database holders (cf. Sec. 2.2).

Algorithm 2 Encoding

Input: D: Database of records $r = (\text{ID}, \lambda, \mu)$
$\mathcal{I}$: The index list
Output: $[D]$: Encoded database

1: Initialize $[D] := \varnothing$
2: **for** $r = (\text{ID}, \lambda, \mu) \in D$ **do**
3: Compute Bloom filter $B = B[1, ..., \ell_{\text{BF}}]$ from λ
4: Let $E = E[1, ..., \ell_{\text{ELD}}] \in \{0, 1\}^{\ell_{\text{ELD}}}$
5: **for** $j \in \{1, ..., \ell_{\text{ELD}}\}$ **do**
6: $E[j] = \bigoplus_{a \in \mathcal{I}_j} B[a]$
7: Insert (ID, E) into $[D]$
8: Return $[D]$

4.2.3 Linking. The linking procedure (Alg. 3) is essentially the same as for BF-based PPRL without diffusion. For a given pair of encoded linkage data E and E', the Dice coefficient $\text{Dice}(E, E')$ is computed. Suppose this value is above a certain threshold τ. In that case, this indicates a high level of similarity of the underlying linkage data, and hence the corresponding pairs of identifiers (ID, ID') are added to the linkage result. We stress that the threshold τ deployed here is not necessarily the threshold used for Bloom filters and needs to be adapted.

Algorithm 3 Linking

Input: $[D]$, $[D']$: Encoded databases
τ: Threshold for linkage
Output: Pairs of identifiers, indicating that these records shall be linked.

1: Initialize $L := \varnothing$
2: **for** $(\text{ID}, E) \in [D]$ **do**
3: **for** $(\text{ID}', E') \in [D']$ **do**
4: **if** $\text{Dice}(E, E) \geq \tau$ **then** Insert (ID, ID') into L
5: Return L

5 Security Analysis

5.1 Preliminaries

The following section provides a theoretical and experimental security analysis of the proposed diffusion extension against pattern mining attacks and security against graph matching attacks.

Piling-Up Lemma. In the theoretical analysis, we make use of the so-called piling-up lemma:

Lemma 1 (Piling-up Lemma [16]). *Given independent binary variables $X_1, ..., X_n$ with $\Pr(X_i = 0) = \frac{1}{2} + \varepsilon_i$ for $i = 1, \ldots, n$, it holds that*

$$\Pr(X_1 \oplus X_2 \oplus \cdots \oplus X_n = 0) = \frac{1}{2} + 2^{n-1} \prod_{i=1}^{n} \varepsilon_i. \tag{5}$$

The value $-0.5 \leq \varepsilon_i \leq 0.5$ is called 'bias' of the variable. The smaller $|\varepsilon_i|$, the less biased the random variable X_i is and the closer to the uniform distribution.

For the analysis, we are going to refer several times to the following notation and (trivial) consequence of the piling-up lemma:

Lemma 2. *Let $X_1, ..., X_n$ be independent binary variables with $\Pr(X_i = 0) = \frac{1}{2} + \varepsilon_i$ for $i = 1, \ldots, n$. Then, it holds for any set $I \subseteq \{1, \ldots, n\}$ that*

$$\Pr(\bigoplus_{i \in I} X_i = 0) = \frac{1}{2} + \varepsilon_I. \tag{6}$$

where

$$\varepsilon_I := 2^{|I|-1} \prod_{i \in I} \varepsilon_i. \tag{7}$$

Moreover, if $J \subseteq I \subset \{1, \ldots, n\}$, it holds that $|\varepsilon_I| \leq |\varepsilon_J|$.

Proof. The claim in (6) is a straightforward consequence of the piling-up lemma. Now, let $J \subseteq I \subset \{1, \ldots, n\}$. As $|\varepsilon_i| \leq 1/2$, it holds that $|2 \cdot \varepsilon_i| \leq 1$ for any i. We have

$$|\varepsilon_I| = \left| 2^{|I|-1} \prod_{i \in I} \varepsilon_i \right| = \frac{1}{2} \cdot \prod_{i \in I} |2 \cdot \varepsilon_i| \quad (8)$$

$$= \frac{1}{2} \cdot \prod_{i \in J} |2 \cdot \varepsilon_i| \cdot \prod_{i \in I \setminus J} |2 \cdot \varepsilon_i| \quad (9)$$

$$\leq \frac{1}{2} \cdot \prod_{i \in J} |2 \cdot \varepsilon_i| \cdot \prod_{i \in I \setminus J} 1 = \frac{1}{2} \cdot \prod_{i \in J} |2 \cdot \varepsilon_i| = |\varepsilon_J|. \quad (10)$$

□

Experiments Setup. In the experiments, the records have been sampled from the publicly available North Carolina Voter Registration database of around 220,000 records [2]. This database is not only popular in textbooks [8] but has been used already for parameter choice recommendations [4] of BF-based schemes and to investigate pattern mining attacks [9, 26] and graph matching attacks [25].

This database has been preprocessed as explained in [7]. The records we used contained several attributes, including names, addresses, date of birth, state, etc. Following the suggestion of [4], attributes to be used as linkage data are first name, last name, street address, and city. We also introduced some errors to check for the linkage quality for similar yet different records. As the ground truth was given, the linkage quality could be computed.

Following earlier work [9, 19, 25, 26], for Bloom filters we use the setting: $\ell_{\text{BF}} = 500, 1000$ (size of the Bloom filter), $k = 5, 10$ (number of hash functions). We used databases of size 2000. The database size is sufficient to conduct the attacks since the expected difference in proportions between different samples from the same population of names would be small.[3] BFs were encoded using random hashing [21] and the CLK approach [20]. The length ℓ_{ELD} of the encoded linkage data has been set to the same as the length of the Bloom filters (see also the parameter discussion in Sec. 7).

For the implementation, we used Python 3.8.6. The experiments were run on a computer with a 64-bit Intel Core i7-10750H 2.60 GHz CPU, 32 GBytes of memory, and Ubuntu 20.04.

5.2 Pattern Mining Attack

Pattern mining attacks exploit that the encoding of a q-gram results in a (possibly characteristic) bit pattern, i.e., a selection of positions where the bits are all equal to 1 (see also (3) on page 300). Thus, to thwart such attacks, one needs to ensure that in the encoded database, no characteristic patterns occur anymore. Let γ denote any q-gram, r some plaintext record, B its Bloom filter encoding, and E the resulting encoded linkage data. We can reformulate the necessary condition (3) for pattern mining attacks against the Bloom filter by

$$\gamma \text{ occurs} \in r \quad \Rightarrow \quad \Pr(B[i] = 0) = 0 \quad \forall i \in \mathbb{E}_{\mathcal{I}(\gamma)}. \quad (11)$$

To strengthen our scheme against this type of attack, we aim for

$$\gamma \text{ occurs} \in r \quad \Rightarrow \quad \Pr(E[i] = 0) \approx \frac{1}{2} \quad \forall i. \quad (12)$$

[2] See https://dl.ncsbe.gov/

[3] Similar Bloom filter settings are widely used in practice also for large real-world databases up to 20 million records, see for example [6].

In other words, no q-gram induces an observable pattern into the encoded linkage data E.

In the following, we first show in Sec. 5.2.1 that for any bit $E[i]$ in the encoded linkage data, its value tends to be uniformly distributed with increasing parameter t (see also Fig. 4). Next, we focus on the case that the underlying Bloom filter exhibits a specific pattern. We show that even under this condition, the bit values $E[i]$ still tend towards uniform distribution (Sec. 5.2.2) and that for any two different bits in the encoded linkage data, the probability of these being the same is also getting close to 1/2 (Sec. 5.2.3).

5.2.1 Distribution. First, we estimate the distribution of the bit values in the encoded linkage data in dependence on the distribution of the Bloom filter bits. Note that having a close-to-uniform distribution of bits is a necessary (but not sufficient) condition for not having any characteristic patterns. Consequently, we determine how close the distribution of single bits in the encoded linkage data can get to the uniform distribution in the best case. For the sake of simplicity, we base our analysis on the assumption that the Bloom filter bits are pairwise independent. While this is not mathematically true, we consider it as a reasonable approximation as these bits are computed from keyed hash-functions (which are independent) and multiple q-grams (which are probably not independent, but where the distribution is unknown). Moreover, this approximation is also backed by our experiments.

In the following, let B denote the Bloom filter of length ℓ_{BF} that has been computed at the beginning of Alg. 2 and E the encoded linkage data of length ℓ_{ELD} that has been derived from B. We assume that for any $i \in \{1, \ldots, \ell_{\text{BF}}\}$, there exists a bias ε_i such that

$$\Pr(B[i] = 0) = \frac{1}{2} + \varepsilon_i. \quad (13)$$

The reason for introducing ε_i is to be able to discuss later on how much the distribution of bits deviates from the uniform distribution. Note that we do not make any assumptions on ε_i.

Now, let $j \in \{1, \ldots, \ell_{\text{ELD}}\}$ be an arbitrary but fixed index of a bit in the encoded linkage data. From the definition of $E[j]$ and the piling-up lemma (Lemma 1), it follows that

$$\Pr(E[j] = 0) = \frac{1}{2} + \varepsilon_{\mathcal{I}_j} \quad (14)$$

where $\varepsilon_{\mathcal{I}_j}$ is defined as explained in Lemma 2. The higher the bias $\varepsilon_{\mathcal{I}_j}$, the better for an attacker. Thus, we are interested into an upper bound of this value. Recall that for any $J \subseteq \mathcal{I}_j$, it holds that $|\varepsilon_{\mathcal{I}_j}| \leq |\varepsilon_J|$ (cf. Lemma 2). In particular, it follows that $|\varepsilon_{\mathcal{I}_j}| \leq \varepsilon_{\min}|$ with $\varepsilon_{\min} := \min_{i \in I}\{|\varepsilon_i|\}$. That is, the bias of $\Pr(E[j] = 0)$ is at most as high as the smallest bias ε_i of all bits indexed by $\mathcal{I}_j$ (and also any product of selected biases). That means, unless for the extreme case that all biases have an absolute value of 1/2, the bias will go down. In particular, this effect increases with increasing index list size t.

To validate this and get an idea of specific values, we measured these values experimentally. In our experiment, we set the length ℓ_{BF} of the Bloom filters to 500,1000 and the number k of hash functions to 5,10. Then using the database described at the beginning of Sec. 3 with 2000 randomly sampled records, we observed the maximum bias $\varepsilon := \max_{j \in \ell_{\text{ELD}}}\{|\varepsilon_{\mathcal{I}_j}|\}$ for all bits in the encoded linkage data as t increases. The results are shown in Fig. 4. As one

can see, the maximum $|\varepsilon_{\mathcal{I}_j}|$ decreases quite sharply as expected. For example, for $t \geq 20$, the maximums are all less than 0.1.

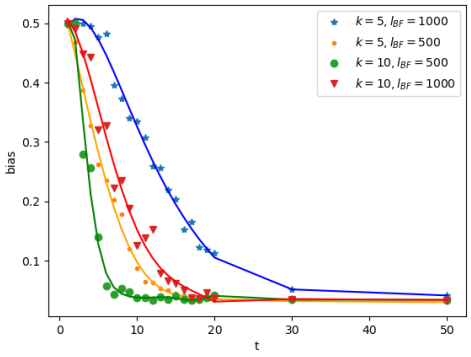

Figure 4: Observed maximum bias $\varepsilon := \max_{j \in \ell_{\text{ELD}}}\{|\varepsilon_{\mathcal{I}_j}|\}$ of all bits in the encoded linkage data with increasing t

5.2.2 Conditional Distribution. Even though the previous analysis showed that the distribution of individual bit values gets close to uniform, it does not exclude the existence of characteristic patterns in the encoded linkage data if the underlying BF B exhibits such a pattern already.

Therefore, we will consider to what extent a pattern in the Bloom filter may result in a pattern in the encoded linkage data. We assume in the following that the underlying Bloom filters exhibit a certain pattern, i.e., a set of indices $\mathcal{I}^*$ where the Bloom filter bits are equal to 1. Let $\mathbb{E}_{\mathcal{I}^*}$ denote the event for an encoded linkage data that the underlying Bloom filter shows this pattern. That is, we have

$$\Pr(B[i] = 1 | \mathbb{E}_{\mathcal{I}^*}) = 1 \quad \forall i \in \mathcal{I}^* \tag{15}$$

We restrict to the case that the pattern has been induced by a single q-gram as this is the approach used in pattern mining attacks [26] and represents patterns with the highest frequency. This means that the size of $\mathcal{I}^*$ is bounded by the number of hash functions k used for the Bloom filter encoding.

Now let us fix some bit index $j \in \{1, \ldots, \ell_{\text{ELD}}\}$ in the encoded linkage data. By definition of the bits, it holds

$$\Pr(E[j] = 0) = \Pr(\bigoplus_{i \in \mathcal{I}_j} B[i] = 0) \tag{16}$$

Obviously, if $\mathcal{I}_j \cap \mathcal{I}^* = \varnothing$, then the pattern has no impact on the value of $E[j]$, that is

$$\Pr(E[j] = 0) = \Pr(E[j] = 0 | \mathbb{E}_{\mathcal{I}^*}) \tag{17}$$

Hence, we restrict on the case that $\mathcal{I}_j \cap \mathcal{I}^* \neq \varnothing$ and define

$$\mathcal{I}^+ := \mathcal{I}_j \cap \mathcal{I}^* \quad \text{and} \quad \mathcal{I}^- := \mathcal{I}_j \setminus \mathcal{I}^*. \tag{18}$$

This allows to rewrite $E[j]$ as follows:

$$E[j] = \bigoplus_{i \in \mathcal{I}^+} B[i] \oplus \bigoplus_{i \in \mathcal{I}^-} B[i] \tag{19}$$

As $B[i] = 1$ for all $i \in \mathcal{I}^*$, the value $\bigoplus_{i \in \mathcal{I}^+} B[i]$ is equal to 0 if and only if $|\mathcal{I}^+|$ is even. As we are only interested into the absolute value of the bias of $\Pr(E[j] = 0)$ and not its concrete value, we can assume without loss of generality that $\bigoplus_{i \in \mathcal{I}^+} B[i] = 0$. Considering $\bigoplus_{i \in \mathcal{I}^+} B[i] = 1$ instead would result in the same bias but with inverted sign.

Recall from Algorithm 1 that $|\mathcal{I}_j| = t = |\mathcal{I}^+| + |\mathcal{I}^-|$ and from the assumptions we have $|\mathcal{I}^+| \leq |\mathcal{I}^*| = k$. Thus, if $t \leq k$, it may happen that $\mathcal{I}_j = \mathcal{I}^*$ and $\mathcal{I}^- = \varnothing$. In this case, the value of $E[j]$ is fixed by the event $\mathbb{E}_{\mathcal{I}^*}$ and one cannot exclude the existence of an observable pattern in E. Therefore, we assume $t \geq k$ from now on.

Under this condition, also using the same line of arguments as before, we get

$$\Pr(E[j] = 0) = \Pr(\bigoplus_{i \in \mathcal{I}^-} B[i] = 0) = \frac{1}{2} + \varepsilon_{\mathcal{I}^-}. \tag{20}$$

Also here, $|\varepsilon_{\mathcal{I}^-}|$ is upper bounded by the smallest bias $\varepsilon_{\min} := \min_{i \in \mathcal{I}^-}\{|\varepsilon_i|\}$ and in general is expected to decrease significantly when the size of $\mathcal{I}^-$ increases. In particular, the size of $\mathcal{I}^-$ is at least $t - k$ and increases with increasing t (and hence most likely will decrease $|\varepsilon_{\mathcal{I}^-}|$).

Using the same parameter settings as previously discussed, we studied the maximum of $|\varepsilon_{\mathcal{I}^-}|$ as $t - k$ increases. The results are shown in Fig. 5. In general, the experiments support the theoretical result that increasing t will decrease $|\varepsilon_{\mathcal{I}^-}|$. For example, for $k = 5, \ell_{\text{BF}} = 500$, and from $t - k \geq 5$ which means that $t \geq 10$, the resulting $|\varepsilon_{\mathcal{I}^-}|$ will be less than 0.1.

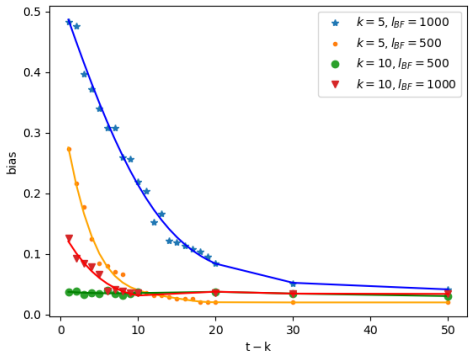

Figure 5: Observed maximum of all encoded linkage data bits under the assumption that the Bloomfilter exhibits a certain pattern. Only bits that depend on pattern bits are considered.

5.2.3 Conditional Co-Occurrence. The previous analysis shows that if t is sufficiently high, the bias of $\Pr(E[j] = 0)$ decreases significantly. That is, a single bit cannot be used for recognizing the presence of a certain pattern in the Bloom filter. This, however, does not exclude the case that the values of two or more bits are correlated. For any choice of distinct indices $i, j \in \{1, \ldots, \ell_{\text{ELD}}\}$, we therefore investigate now

$$\Pr(E[i] = E[j] | \mathbb{E}_{\mathcal{I}^*}), \tag{21}$$

that is the probability that both bit values in the encoded linkage data are equal under the condition $\mathbb{E}_{\mathcal{I}^*}$, i.e., the event that the underlying Bloom filter exhibits a pattern given by an index set $\mathcal{I}^*$.

From now on, we assume that $\mathbb{E}_{\mathcal{I}^*}$ does hold and aim to estimate the bias of $\Pr(E[i] = E[j]) = \Pr(E[i] \oplus E[j] = 0)$. Similarly as discussed before, we can re-write this as

$$\Pr(E[i] \oplus E[j] = 0) = \Pr(\bigoplus_{a\in\mathcal{I}_j} B[a] = \bigoplus_{a\in\mathcal{I}_i} B[a]) \quad (22)$$

$$= \bigoplus_{a\in\mathcal{I}^+} B[a] \oplus \bigoplus_{a\in\mathcal{I}^-} B[a] \quad (23)$$

where

$$\mathcal{I}^+ := \underbrace{(\mathcal{I}_i \cup \mathcal{I}_j \backslash \mathcal{I}_i \cap \mathcal{I}_j)}_{\mathcal{I}} \cap \mathcal{I}^* \quad \text{and} \quad \mathcal{I}^- := \mathcal{I} \setminus \mathcal{I}^*. \quad (24)$$

Similar to the analysis conducted in Sec. 5.2.2, we assume that $\bigoplus_{a\in\mathcal{I}^+} B[a] = 0$ and get

$$\Pr(E[i] \oplus E[j] = 0) = \Pr(\bigoplus_{a\in\mathcal{I}^-} B[a] = 0) = \frac{1}{2} + \varepsilon_{\mathcal{I}^-}. \quad (25)$$

Note that as opposed to the previous situation, it may happen that $\mathcal{I}^- = \varnothing$. This is possible if $\mathcal{I}_i \backslash \mathcal{I}^* = \mathcal{I}_j \backslash \mathcal{I}^*$ happens by coincidence: then all the non-pattern bits would cancel out. If we assume again that $|\mathcal{I}_i| = |\mathcal{I}_j| = t \geq k = |\mathcal{I}^*|$, then the probability of this event decreases with increasing t.

Results of the largest bias $|\varepsilon_{\mathcal{I}^-}|$ are shown in Fig. 6. The probability of co-occurrence under the condition of $\mathbb{E}_{\mathcal{I}^*}$ decreases as expected and when $t \geq 10$, ε will be less than 0.1 except for $k = 5, \ell_{\text{BF}} = 1000$.

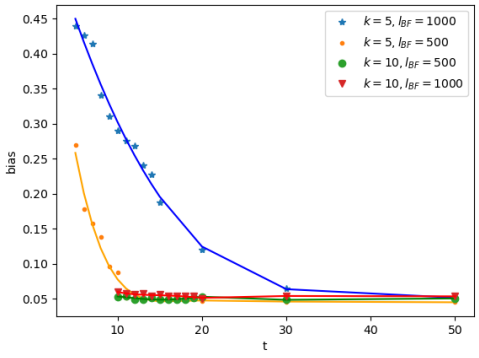

Figure 6: Observed maximum of the bias of co-occurrence under the condition that $\mathbb{E}_{\mathcal{I}^*}$. Only bits $E[i], E[j]$ where $\mathcal{I}_i \cap \mathcal{I}^* \neq \varnothing$ and $\mathcal{I}_j \cap \mathcal{I}^* \neq \varnothing$ are considered and $t \geq k$.

5.3 Graph Matching Attack

5.3.1 Breaking Similarity Relationship. Recall that the graph matching attack (cf. Sec. 3.3) is based on two conditions: (i) a high overlap rate and (ii) $sim(\lambda, \lambda')$ correlates with $\text{Dice}([\lambda], [\lambda']) = \text{Dice}(B, B')$ for any pair of plaintext records with linkage data λ and λ' and let B and B' be the corresponding Bloom filter encodings (see also (4)).

In [1], it was shown that the attack is quite fragile with respect to the overlap rate. As soon as it deviates from 100%, the success rate drops sharply. However, the overlap rate depends on the use case and is out of the control of the PPRL scheme. Therefore, to protect against graph matching attacks, we aim to weaken condition (ii) significantly. That is, the values $sim(\lambda, \lambda')$ and $\text{Dice}(E, E')$ should not be strongly correlated anymore where E is the encoding of λ and analogously E' of λ'.

In fact, there is a trade-off that needs to be taken into account. For preserving linkage quality, i.e., correctness of the scheme, it is important that a high value of $sim(\lambda, \lambda')$ has to imply a (relatively) high value $\text{Dice}(E, E')$. However, when $sim(\lambda, \lambda')$ decreases, i.e., the less similar the records are, the value $\text{Dice}([\lambda], [\lambda'])$ should not decrease analogously.

In the following, we provide a theoretical analysis that this property is given for the BFD scheme for appropriate choices of t. This is also backed by experiments. As an example, the similarity relation between $sim(\lambda, \lambda')$ and $\text{Dice}(E, E')$ in our experiments is shown in Fig. 7. The three graphs show the relationship for $t \in \{5, 10, 20\}$. We see that for increasing value of t, the correlation between $sim(\lambda, \lambda')$ and $\text{Dice}(E, E')$ gets weaker. More precisely, except for the case of $sim(\lambda, \lambda')$ being high, the value of $\text{Dice}(E, E')$ is close to 0.5. That is, when an attacker observes $\text{Dice}(E, E')$, she cannot deduce the value of $sim(\lambda, \lambda')$.

As the BFD scheme deploys the Bloom filter encoding as intermediate step and as we know that $sim(\lambda, \lambda')$ and $\text{Dice}(B, B')$ are strongly correlated, we investigate in the following the relation between $\text{Dice}(B, B')$ and $\text{Dice}(E, E')$. Recall that

$$\text{Dice}(B, B') = \frac{2\,|supp(B) \cap supp(B')|}{|supp(B)| + |supp(B')|} \quad (26)$$

$$\text{Dice}(E, E') = \frac{2\,|supp(E) \cap supp(E')|}{|supp(E)| + |supp(E')|} \quad (27)$$

with $supp(B) = \{i | B[i] = 1\}$. Without loss of generality, we can assume that $|supp(B)| \geq |supp(B')|$. We have

$$\text{Dice}(B, B') = \frac{2\,|supp(B) \cap supp(B')|}{|supp(B)| + |supp(B')|} \quad (28)$$

$$\leq \frac{2\,|supp(B) \cap supp(B')|}{|supp(B')| + |supp(B')|} \quad (29)$$

$$= \frac{|supp(B) \cap supp(B')|}{|supp(B')|} \quad (30)$$

$$= \Pr(B[i] = 1 | B'[i] = 1). \quad (31)$$

Likewise, it holds that

$$\text{Dice}(B, B') \geq \Pr(B'[i] = 1 | B[i] = 1). \quad (32)$$

That is, $\max\{\Pr(B'[i] = 1|B[i] = 1), \Pr(B[i] = 1|B'[i] = 1)\}$ is an upper bound on $\text{Dice}(B, B')$ and $\min\{\ldots\}$ a lower bound, where equality is achieved if $|supp(B)| = |supp(B')|$. In particular, if both $\Pr(B'[i] = 1|B[i] = 1)$ and $\Pr(B[i] = 1|B'[i] = 1)$ are large, then $\text{Dice}(B, B')$ is large as well. On the other hand, if both probabilities are low, then this holds for the Dice coefficient also. Of course, the same observations hold with respect to E and E'. We will make use of these to estimate the impact of $\text{Dice}(B, B')$ on $\text{Dice}(E, E')$.

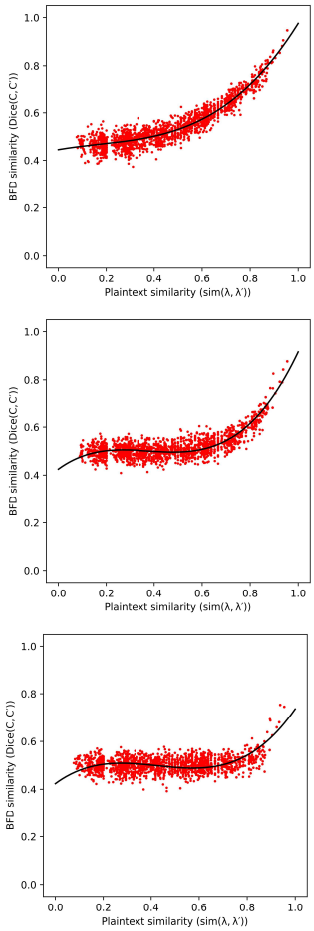

Figure 7: Similarity relation between $sim(\lambda, \lambda')$ (plaintext records) and $\mathrm{Dice}(E, E')$ (encoded records in BFD) for different values of t (top: $t = 5$, middle: $t = 10$, bottom: $t = 20$)

Let us start with two Bloom filters B and B' that are similar, being expressed by the fact that for all $i \in \{1, \ldots, \ell_{\mathrm{BF}}\}$, it holds that

$$\min\{\Pr(B'[i] = 1 | B[i] = 1), \Pr(B[i] = 1 | B'[i] = 1)\} \geq \delta \tag{33}$$

for a sufficiently large value δ.

First, we show that under this condition, the probability of

$$\Pr(B[i] = B'[i]) \tag{34}$$

is also lower bounded by δ. We know already that

$$\Pr(B'[i] = 1 | B[i] = 1) \geq \delta. \tag{35}$$

In the following, we assume that

$$\Pr(B'[i] = 1) \leq \Pr(B[i] = 0). \tag{36}$$

This is given in practice as the number k of hash functions is significantly smaller than the length ℓ_{BF} of the Bloom filter. This means that the probability of a randomly position bit to be set to 1 is smaller than the probability that is has been left to zero. Experiments confirm this: Using our parameter choices, we observed that $\max_i \Pr(B'[i] = 1) \approx 0.35$ which means that $\Pr(B[i] = 0) \geq 0.65$.

Under this assumption, it follows

$$\begin{aligned}
& \Pr(B'[i] = 0 | B[i] = 0) && (37)\\
= \;& 1 - \Pr(B'[i] = 1 | B[i] = 0) && (38)\\
= \;& 1 - \Pr(B[i] = 0 | B'[i] = 1) \cdot \frac{\Pr(B'[i] = 1)}{\Pr(B[i] = 0)} && (39)\\
\overset{(36)}{\geq} \;& 1 - \Pr(B[i] = 0 | B'[i] = 1) && (40)\\
\geq \;& 1 - (1 - \delta) = \delta. && (41)
\end{aligned}$$

We can now deduce the following lower bound:

$$\begin{aligned}
& \Pr(B[i] = B'[i]) && (42)\\
= \;& \Pr(B[i] = 0 \wedge B'[i] = 0) + \Pr(B[i] = 1 \wedge B'[i] = 1) && (43)\\
= \;& \Pr(B'[i] = 0 | B[i] = 0) \cdot \Pr(B[i] = 0) && (44)\\
& + \Pr(B'[i] = 1 | B[i] = 1) \cdot \Pr(B[i] = 1) && (45)\\
\geq \;& \delta \cdot \Pr(B[i] = 0) + \delta \cdot \Pr(B[i] = 1) && (46)\\
= \;& \delta \cdot (\Pr(B[i] = 0) + \Pr(B[i] = 1)) && (47)\\
= \;& \delta. && (48)
\end{aligned}$$

Next, we pick some arbitrary index $i \in \{1, \ldots, \ell_{\mathrm{ELD}}\}$ and aim to derive bounds for $\Pr(E'[i] = 1 | E[i] = 1)$ in dependence of δ. We define

$$\Delta := \{j \in \mathcal{I}_i | B[j] \neq B'[j]\}. \tag{49}$$

One can see that

$$\begin{aligned}
E'[i] \;=\; & \bigoplus_{j \in \mathcal{I}_i} B'[i] = \bigoplus_{j \in \mathcal{I}_i \setminus \Delta} B[i] \oplus \bigoplus_{j \in \Delta} (B[i] \oplus 1) && (50)\\
= \;& \bigoplus_{j \in \mathcal{I}_i} B[i] \oplus \bigoplus_{j \in \Delta} 1 = E[i] \oplus \bigoplus_{j \in \Delta} 1 && (51)\\
= \;& 1 \oplus \bigoplus_{j \in \Delta} 1 && (52)
\end{aligned}$$

Obviously, $E'[i] = 1$ if and only if $|\Delta|$ is even.

Let $p := \Pr(B[i] = B'[i])$, assuming that this probability does not depend on i. Then:

$$\begin{aligned}
\Pr(E'[i] = 1 | E[i] = 1) \;=\; & \Pr(|\Delta| \text{even}) && (53)\\
= \;& \sum_{k=0}^{\lfloor \frac{t}{2} \rfloor} \binom{t}{2k} \cdot (1-p)^{2k} \cdot p^{t-2k} && (54)
\end{aligned}$$

Table 1 displays the values of $\Pr(E'[i] = 1 | E[i] = 1)$ for different choices of t and p. Note that the same formula can be derived for $\Pr(E[i] = 1 | E'[i] = 1)$ and hence can be used for estimating the Dice coefficient.

As expected, the value increases or decreases depending on p for a fixed t. However, the more t increases, the more does $\Pr(E'[i] = 1 | E[i] = 1)$ tend to 0.5 except for the case that $p := \Pr(B[i] = B'[i])$ is very close to 1, i.e., the case that B and B' are very similar. Therefore, it shows the behaviour that we aimed for, as described at the beginning.

To further support this view, we can derive some lower and upper bound on $\Pr(E'[i] = 1 | E[i] = 1)$ and hence on the Dice coefficient. For a lower bound, observe that

$$\Pr(E'[i] = 1 | E[i] = 1) \overset{(54)}{\geq} p^t \overset{(48)}{\geq} \delta^t.$$

This shows that the Dice coefficient increases for increasing δ and fixed t.

Table 1: Computation of $\Pr(E'[i] = 1|E[i] = 1)$ using (54) for different choices of t (=size of the index list) and p (= probability that the two different Bloom filters show the same value at an arbitrary position).

t \ p	0.8	0.9	0.95	0.99
1	0.8	0.9	0.95	0.99
2	0.64	0.81	0.90	0.98
3	0.61	0.76	0.86	0.97
4	0.56	0.70	0.83	0.96
5	0.54	0.66	0.80	0.95
6	0.52	0.63	0.77	0.94
7	0.51	0.60	0.74	0.93
8	0.51	0.58	0.72	0.93
9	0.51	0.57	0.69	0.92
10	0.50	0.55	0.67	0.91
11	0.50	0.54	0.66	0.90
12	0.50	0.53	0.64	0.89
15	0.50	0.52	0.60	0.87
20	0.50	0.51	0.56	0.83

To give an upper bound on the Dice coefficient, let us define the characteristic function even : $\mathbb{N} \to \{0, 1\}$ by $even(x) = 1$ if and only if x is even. Then, it holds

$$\begin{aligned}\Pr(E'[i] = 1|E[i] = 1) &= \sum_{k=0}^{\lfloor \frac{t}{2} \rfloor} \binom{t}{2k} \cdot (1-p)^{2k} \cdot p^{t-2k} \\ &= \sum_{k=0}^{t} \text{even}(k) \cdot \binom{t}{k} \cdot (1-p)^{k} \cdot p^{t-k} \\ &\leq \sum_{k=0}^{t} \binom{t}{k} \cdot (1-p)^{k} \cdot p^{t-k} \qquad (55)\end{aligned}$$

The expression in (55) is actually the cumulative distribution function for the Binomial distribution. In particular, (55) tends towards 0.5 for $p \to 0.5$.

5.3.2 Correlation and Re-identification Rate. To further analyze the security of the BFD scheme, we conducted several experiments on two features: correlation and re-identification rate.

In statistics, correlation refers to the degree to which a pair of variables are related. For the standard Bloom filter based schemes without diffusion, the correlation between the similarity of plaintext records and the similarity between encoded records is large (see also Fig. 2). For our analysis, we computed the sample correlation coefficient.

Definition 3 (Sample Correlation Coefficient). *Given a series of n measurements of the pair (X_i, Y_i) indexed by $i = 1, \ldots, n$, the 'sample correlation coefficient' can be used to estimate the population Pearson correlation $\rho_{X,Y}$ between X and Y. The sample correlation coefficient is defined as:*

$$\rho^{Est}_{X,Y} \stackrel{\text{def}}{=} \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}},$$

where $\overline{x}$ and $\overline{y}$ are the sample arithmetic mean of X and Y.

It holds for the sample correlation coefficient that $\rho^{Est}_{X,Y} \in [-1, 1]$ where $|\rho| \approx 1$ indicates a high correlation and $\rho \approx 0$ a low correlation.

In a graph matching attack, the success of the attack is measured by the *re-identification rate*. It's defined as the number of correctly re-identified records divided by the size of the encoded database. Recall from Section 3.3 that the attacker is assumed to have access to a largely overlapping pair of databases, namely D and $[D']$. In [25], the *overlap rate* of two databases is defined as two times the number of common records divided by the sum of the sizes of two databases.

We conducted experiments to study the correlation between similarities and re-identification rates for varying overlap rates and sizes t of the index lists. The results are shown in Fig. 8 and Fig. 9. The first point on the left shows the results for the scheme with Bloom filters but without diffusion, while the remaining ones show the results for the BFD scheme for varying t. Moreover, the four graphs display the results for different overlap rates: 100%, and 95%. As observed in [1], the re-identification rate decreases sharply when the overlap between the encoded database and plaintext database drops. Our experiments confirm this effect where the decrease is stronger for the BFD scheme than for the original BF-based PPRL scheme.

5.4 Other Attacks

Our analysis indicates a stronger resilience of BFD compared to the plain BF-based PPRL against pattern mining attacks and graph matching attacks. More precisely, we identified a necessary condition for each attack in Sec. 3.2 and Sec. 3.3, respectively. Afterwards, we showed in Sec. 5.2 and Sec. 5.3, respectively, that the introduction of a diffusion layer helps that these conditions do not hold anymore (or at least to weaken these). Therefore, we conclude that BFD is (more) secure against these attacks.

Naturally, two questions arise:

(1) Is BFD also secure against variants of these attacks?
(2) Is BFD also secure against other (possibly unknown) types of attacks?

For the reasons explained in Sec. 3.1, we do not aim for a provably secure PPRL scheme. Thus, we cannot give any definitive answer to these two questions. However, for any variant of a pattern mining and graph matching attack that relies on the same condition as identified in Sec. 3.2 and Sec. 3.3, respectively, the same analysis applies. In this sense, we conjecture that the diffusion layer should also protect against these attack variants.

Moreover, one can prove that BFD does not introduce any new weaknesses. More precisely, any attack that is possible against BFD is likewise possible against the underlying BF-based PPRL scheme as well. This can be shown with a standard proof by reduction. Let $\mathcal{S}_{\text{BF}}$

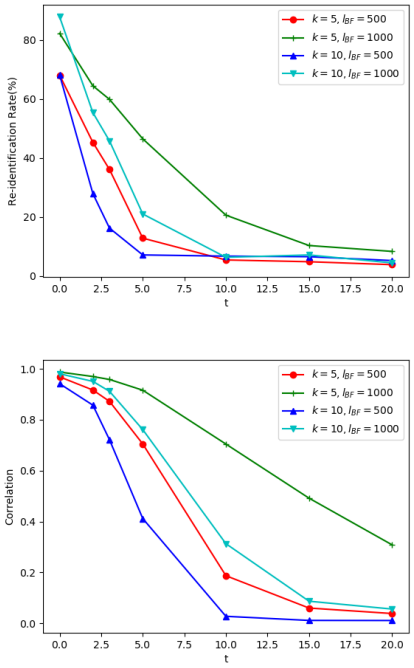

Figure 8: Sample correlation coefficient and re-identification rate of the graph matching attack for varying index set sizes t (overlap: 100%). The first point on the left side shows the result for the plain BF-based PPRL scheme (without diffusion) and the remaining points for the proposed BFD scheme (with diffusion).

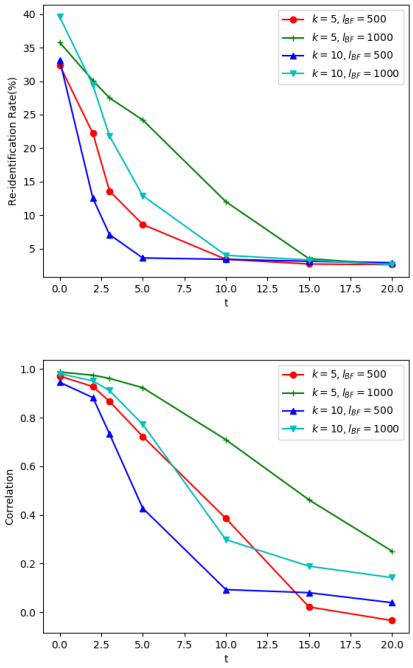

Figure 9: Sample correlation coefficient and re-identification rate of the graph matching attack for varying index set sizes t (overlap: 95%).

denote a plain BF-based PPRL scheme and $\mathcal{S}_{\text{BFD}}$ the BFD scheme that results from $\mathcal{S}_{\text{BF}}$ by applying a diffusion layer (computed with Alg. 1) to the BF filters. Furthermore, let $\mathcal{A}_{\text{BFD}}$ denote any attacker against $\mathcal{S}_{\text{BFD}}$. We use $\mathcal{A}_{\text{BFD}}$ to construct an attacker $\mathcal{A}_{\text{BF}}$ against $\mathcal{S}_{\text{BF}}$ that has exactly the same effort and success probability.

$\mathcal{A}_{\text{BF}}$ works as follows. It gets as input one or several encoded databases, where each entry is a Bloom filter B. First, it runs Alg. 1 to create a diffusion layer. Second, it applies the diffusion layer to each Bloom filter B, getting the corresponding encoded linkage data E. These linkage data are given $\mathcal{A}_{\text{BFD}}$. Note that $\mathcal{A}_{\text{BF}}$ perfectly simulated $\mathcal{S}_{\text{BFD}}$ based on $\mathcal{S}_{\text{BF}}$. Thus, $\mathcal{A}_{\text{BFD}}$ can normally operate on these inputs and return some result (to $\mathcal{A}_{\text{BF}}$). Attacker $\mathcal{A}_{\text{BF}}$ simply uses this result as its own result and is finished.

The consequence is that if there exists an efficient attacker against $\mathcal{S}_{\text{BFD}}$, then this is likewise true for $\mathcal{S}_{\text{BF}}$. Thus, $\mathcal{S}_{\text{BFD}}$ cannot be less secure than $\mathcal{S}_{\text{BF}}$.

6 Evaluation

In the following, we evaluate the BFD scheme for practical applications by investigating its efficiency (Sec. 6.1) and linkage quality (Sec. 6.2).

6.1 Efficiency

Bloom filter based PPRL schemes are highly efficient, with only modest time requirements for encoding and linking records. Therefore, large real-world applications linking millions of records are possible. As our proposed BFD scheme builds on that scheme by adding a diffusion layer, the efficiency of our scheme can be evaluated by analyzing the run time of the three procedures described in Sec. 4 and comparing them with the BF-based PPRL scheme. The results of our experiments can be found in Table 2 for a Bloom filter length $\ell_{\text{BF}} = 500$ and $\ell_{\text{BF}} = 1000$.

The first thing to observe is that BFD requires in addition the generation of a diffusion layer (Alg. 1). In comparison to the two other procedures, it is significantly more time consuming. However, while encoding and linking needs to be done for each record, the diffusion layer needs to be computed only once. Moreover, as the overhead is significantly less than a second, we consider this to be acceptable.

Note that the size of data that needs to be transmitted here is rather modest. A straightforward approach would be to send the ℓ_{ELD} index sets where each index set contains t values from $\{1, \ldots, \ell_{\text{BF}}\}$, that is $\ell_{\text{ELD}} t \cdot \lceil \log_2(\ell_{\text{BF}}) \rceil / 8$ bytes. For example, if $\ell_{\text{ELD}} = \ell_{\text{BF}} = 500$ and $t = 10$ as suggested in Sec. 7, this would sum up to less than 6 kB. If the random choices made in Alg. 1 are derived from some seed, one can save even more data by sending the seed only.

As expected, the effort to encode a record increases linearly with the index size t, i.e., the number of BF bits used to compute one

bit in the encoded linkage data. However, as the time effort is still in the range of very few milliseconds and as rather small values for t provide the best trade-off between feasibility and security (cf. Sec. 7), we consider this to be acceptable.

Finally, one can see that the time required for linking two records of the BFD scheme is approximately the same as the BF scheme as they are in the same magnitude, namely, 10^{-5} milliseconds. This is also anticipated as $\ell_{\text{BF}} = \ell_{\text{ELD}}$ and as the linkage procedure is essentially the same as before.

Note that another factor that impacts the run time is the length ℓ_{BF} of the underlying Bloom filter. However, as we discuss in Section 7, setting $\ell_{\text{ELD}} = \ell_{\text{BF}}$ is a reasonable choice. This has been adopted here as well.

Table 2: Average execution time of the diffusion, encoding and linking procedures for the plain BF-based PPRL scheme (BF) and the BFD scheme for $\ell_{\text{ELD}} = 500, 1000$

	t	Diffusion (ms)	Encoding (ms)	Linking (10^{-5} ms)
BF			0.10	6.43
BFD, $\ell_{\text{ELD}} = 500$	1	24.03	0.21	6.32
	5	25.07	0.40	6.32
	10	25.53	0.60	6.82
	15	26.07	0.86	6.75
	20	26.94	1.10	6.69
	30	26.75	1.46	6.90
	50	31.48	2.33	6.66
BFD, $\ell_{\text{ELD}} = 1000$	1	103.44	0.36	6.50
	5	109.94	0.65	6.72
	10	117.89	0.96	6.65
	15	128.14	1.37	7.10
	20	127.06	1.94	6.78
	30	126.42	2.30	6.52
	50	128.90	3.57	6.60

6.2 Linkage Quality

We analyze the linkage quality of the BFD scheme by comparing it with the plain BF-based PPRL scheme, as the latter is known to provide good linkage quality. To this end, we adopt the suggestion from [11] to express the linkage quality by MPR, being the mean of precision and recall:

$$\text{MPR} = \frac{1}{2} \cdot (\text{Precision} + \text{Recall}) = \frac{1}{2} \cdot \left(\frac{\text{tp}}{\text{tp} + \text{fp}} + \frac{\text{tp}}{\text{tp} + \text{fn}} \right) \quad (56)$$

Here, tp refers to the number of true positives, fp the number of false positives, and fn the number of false negatives.

For comparing the linkage qualities of the two schemes, we consider their relative MPR:

$$\text{Relative MPR} = \frac{\text{MPR}_{BFD}}{\text{MPR}_{BF}}. \quad (57)$$

The linkage quality of BFD is the same or better than the plain BF schemes if the relative MPR is ≥ 1 and comparable if this value is close to 1.

Figure 10 shows the relative MPR for varying the index set size t and threshold τ values of 0.6., 0.7, and 0.8. Note that the threshold for the plain BF scheme is set to $\tau_{BF} = 0.8$. However, as discussed in Sec. 5.3.1, the Dice coefficient of encoded records can differ before and after applying diffusion. These differences require the modification of the threshold τ accordingly.

As expected, the relative MPR decreases for increasing t and/or increasing threshold τ. However, apart from the case where $k = 10$ and $\ell_{\text{BF}} = 500$, the relative MPR is above 0.9 when $t \leq 10$. As we are going to discuss into more detail in Sec. 7, this allows for a parameter selection that offers a good trade-off between security and feasibility.

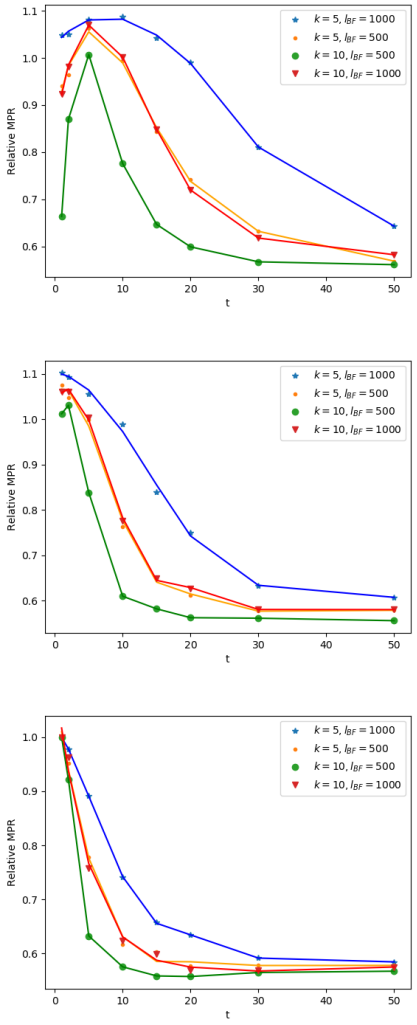

Figure 10: Relative MPR in dependence of t for different choices for the threshold τ (top: $\tau = 0.6$, middle: $\tau = 0.7$, bottom: $\tau = 0.8$)

7 Parameter Selection

A record linkage process is usually organized by an authority to fulfill tasks like granting permissions or organizing the agreement on secrets between the database holders (like the deployed hash functions in the case of BF-based PPRL). Generally, this authority is also responsible for recommending specific parameter choices. In practice, these recommendations are based on theoretical analysis and experiments.

In comparison to the plain BF-based PPRL scheme, the BFD scheme comprises a couple of new parameters, namely

(1) the length ℓ_{ELD} of the encoded linkage data (Alg. 1),
(2) the size t of the index sets, being the number of BF bits used to compute one bit (Alg. 1), and
(3) the (adapted) threshold τ, used as criteria for whether two records shall be linked or not (Alg. 3).

Even if some parameters of existing BF schemes might be re-used, at least some parameters need to be chosen. In the following, we demonstrate this process in a concrete example, using the setup explained in Sec. 5.1. In particular, we build on a BF-based schemes that uses $k \in \{5, 10\}$ hash functions to compute Bloom filters of length $\ell_{\text{BF}} \in \{500, 1000\}$.

The choice of the first parameter of the new parameters, the length ℓ_{ELD} of the encoded linkage data, is a matter of finding an appropriate trade-off. Decreasing ℓ_{ELD} too much (in comparison to ℓ_{BF}) may result in a too substantial loss of information, negatively impacting the linkage quality. On the other hand, choosing ℓ_{ELD} too large may render the diffusion layer useless, as an attacker may reconstruct the BF bits by solving a system of linear equations. Therefore, to make both schemes comparable in efficiency and linkage quality, we suggest $\ell_{\text{ELD}} = \ell_{\text{BF}}$.

A wide range of possible choices exist for the other two parameters, the number t of BF bits that define one bit in the encoded linkage data and the threshold τ. In fact, by defining the index sets by $\mathcal{I}_j := \{j\}$, i.e., $t = 1$, and by keeping the same threshold as before, the BFD scheme is just the original plain BF scheme. Of course, the question is if better variants can be found.

Note that the value t impacts several properties such as security and linkage quality while τ influences just the latter. Therefore, we suggest fixing first t and then adapting τ. In the following, we discuss the choice of t for the case of $k = 5$ hash functions and a Bloom filter length of $\ell_{\text{BF}} = 500$. The process works analogously for the other settings as well.

For security, we always want t to be as large as possible. In the security analysis regarding pattern mining attacks in Section 5.2, we assumed that $t \geq k$ which, in our scenario, results in a requirement of $t \geq 5$. However, we stress that this assumption was only made to ensure that certain potential security risks cannot occur. In this sense, it represents a sufficient but not a necessary condition. Our experiments (cf. Fig. 6) indicate that setting $t \geq 10$ provides good resistance against pattern mining attacks.

With respect to security against graph matching attacks, our experiments (cf. Fig. 7) show that $t = 5$ does not sufficiently weakens the relation between $sim(\lambda, \lambda')$ and $\text{Dice}(E, E')$). From Fig. 8, we see that $t \geq 10$ will reduce the re-identification rate of graph matching attack and correlation to half of those of BF.

While security benefits from large values of t, the opposite is true with respect to the computational effort and linkage quality. Recall that the purpose of diffusion (see Sec. 4.1) is that small changes in the input result into large changes in the output. That is, by increasing t, it may happen that the encoding of similar (but not equal) records strongly differ and hence will not be linked. That is, the number of false negatives fn will likely go up. On the other hand, the encoding on non-similar records will remain non-similar. Thus, the number of fp will either remain the same or decrease. This means that by increasing t, precision will remain as it is while recall will decrease. For health-related authorities, usually precision has more importance over recall. However, recall is often more important for criminal systems. Therefore the choices are from the interested parties. Thus, the discussion above can be summarized that t should be ≥ 10 but as small as possible. Naturally, we suggest to use $t = 10$ for a reasonable compromise of security and feasibility.

It remains to fix the threshold accordingly. Here, experiments (cf. Fig. 10) showed that good linkage quality is achieved for a threshold of $\tau = 0.6$. Concluding, the suggested parameter choices for the setting of $k = 5$ and $\ell_{\text{BF}} = 500$ are $t = 10$ and $\tau = 0.6$. For the other settings, Table 3 displays the suggested choices of t. The process was more or less the same as described above. The only exception is given in the case of $k = 10$, $\ell_{\text{BF}} = 500$. In this case, the (sufficient) condition of $t \geq k$ made for the security analysis against pattern mining attacks could not be met.

Table 3: Suggested choice of t ($\tau = 0.6$)

Optimal t — k \ ℓ_{BF}	500	1000
5	10	20
10	5*	10

*: $t = 5$ has been found by experiments only

8 Conclusion

Privacy-preserving record linkage schemes based on Bloom filters (BF-based PPRL) are popular in practice and academia due to their high efficiency and linkage quality. However, recent research showed that these schemes are not sufficiently secure. This paper proposed a new PPRL scheme, BFD, using Bloom filters extended by a linear diffusion layer. Extensive theoretical and experimental analysis showed that the BFD scheme is more secure against known attacks than the BF-based PPRL scheme while achieving similar linkage quality and running within tolerable time compared to BF-based PPRL. Thus, BFD could, in the long run, be considered as a replacement for the standard BF-based PPRL scheme.

Besides, as no rigorous security analysis has been conducted on the standard BF-based PPRL scheme (and likewise not for its variants that have been proposed), the analysis methods explained in this paper may have merit on their own. For example, as explained, there seems to be a positive relationship between the maximum re-identification rate of the graph matching attack and the correlation, which could be further investigated. Moreover, other encoding methods than Bloom filters may be analyzed by the methods proposed here.

ACKNOWLEDGMENTS

The research reported here is supported by the research grant DFG 407023611 of the German Research Foundation. The funder DFG had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

REFERENCES

[1] Authors anonymized for submission. 2022. On the effectiveness of graph matching attacks against privacy-preserving record linkage. *Accepted at PLOS one* (2022). https://doi.org/10.1371/journal.pone.0267893

[2] Manfred Antoni and Rainer Schnell. 2019. The Past, Present and Future of the German Record Linkage Center (GRLC). *Journal of Economics and Statistics* 239, 2 (2019), 319–331. https://doi.org/10.2139/ssrn.3549199

[3] Burton Bloom. 1970. Space/Time Trade-Offs in Hash Coding With Allowable Errors. *Commun. ACM* 13 (07 1970), 422–426. https://doi.org/10.1145/362686.362692

[4] Christian Borgs. 2019. *Optimal Parameter Choice for Bloom Filter-based Privacy-preserving Record Linkage.* Ph. D. Dissertation. University of Duisburg-Essen. https://doi.org/10.17185/duepublico/70274

[5] James H. Boyd, Sean M. Randall, and Anna M. Ferrante. 2015. Application of Privacy-Preserving Techniques in Operational Record Linkage Centres. In *Medical Data Privacy Handbook*, Aris Gkoulalas-Divanis and Grigorios Loukides (Eds.). Springer, Cham, 267–287. https://doi.org/10.1007/978-3-319-23633-9_11

[6] Adrian P. Brown, Christian Borgs, Sean M. Randall, and Rainer Schnell. 2017. Evaluating Privacy-Preserving Record Linkage Using Cryptographic Long-Term Keys and Multibit Trees on Large Medical Datasets. *BMC Medical Informatics and Decision Making* 17, 83 (2017), 1–7.

[7] Peter Christen. 2014. Preparation of a real voter data set for record linkage and duplicate detection research.

[8] Peter Christen, Thilina Ranbaduge, and Rainer Schnell. 2020. *Linking Sensitive Data: Methods and Techniques for Practical Privacy-Preserving Information Sharing.* Springer, Cham. https://doi.org/10.1007/978-3-030-59706-1

[9] Peter Christen, Rainer Schnell, Dinusha Vatsalan, and Thilina Ranbaduge. 2017. Efficient Cryptanalysis of Bloom Filters for Privacy-Preserving Record Linkage. In *Lecture Notes in Computer Science.* Springer, Berlin, 628–640. https://doi.org/10.1007/978-3-319-57454-7_49

[10] Martin Franke, Ziad Sehili, Florens Rohde, and Erhard Rahm. 2021. Evaluation of Hardening Techniques for Privacy-Preserving Record Linkage. In *Advances in Database Technology – EDBT 2021*, Yannis Velegrakis, Demetris Zeinalipour, Panos K. Chrysanthis, and Francesco Guerra (Eds.). OpenProceedings, Konstanz, 289–300.

[11] David Hand and Peter Christen. 2018. A note on using the F-measure for evaluating record linkage algorithms. *Statistics and Computing* 28, 3 (2018), 539–547.

[12] Xi He, Ashwin Machanavajjhala, Cheryl Flynn, and Divesh Srivastava. 2017. Composing Differential Privacy and Secure Computation: A Case Study on Scaling Private Record Linkage. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (Dallas, Texas, USA) *(CCS '17).* Association for Computing Machinery, New York, NY, USA, 1389–1406. https://doi.org/10.1145/3133956.3134030

[13] Anushka Isuru Vidanage. 2022. *Efficient Cryptanalysis Techniques for Privacy-Preserving Record Linkage.* Ph. D. Dissertation. Australian National University.

[14] Mehmet Kuzu, Murat Kantarcioglu, Elizabeth Durham, and Bradley Malin. 2011. A Constraint Satisfaction Cryptanalysis of Bloom Filters in Private Record Linkage. In *The 11th Privacy Enhancing Technologies Symposium*, S. Fischer-Hübner and N. Hopper (Eds.). Springer, Berlin, 226–245. https://doi.org/10.1007/978-3-642-22263-4_13

[15] Rainer Lenz and Tim Hochgürtel. 2021. Random disclosure in confidential statistical databases. *Statistical Journal of the IAOS* 1 (2021), 401–413.

[16] L. Matsui. 1994. Linear Cryptanalysis Method for DES Cipher. *Advances in Cryptology—Eurocrypt '93 (LNCS 765)* (01 1994), 386–397.

[17] 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 (12 2014). https://doi.org/10.29012/jpc.v6i2.640

[18] Thilina Ranbaduge and Rainer Schnell. 2020. Securing Bloom Filters for Privacy-Preserving Record Linkage. In *CIKM '20: Proceedings of the 29th ACM International Conference on Information & Knowledge Management.* Association for Computing Machinery, New York, 2185–2188.

[19] Rainer Schnell, Tobias Bachteler, and Jörg Reiher. 2009. Privacy-preserving record linkage using Bloom filters. *BMC medical informatics and decision making* 9 (09 2009), 41. https://doi.org/10.1186/1472-6947-9-41

[20] Rainer Schnell, Tobias Bachteler, and Jörg Reiher. 2011. A Novel Error-Tolerant Anonymous Linking Code. *SSRN* (01 2011). https://doi.org/10.2139/ssrn.3549247

[21] Rainer Schnell and Christian Borgs. 2016. Randomized Response and Balanced Bloom Filters for Privacy Preserving Record Linkage. In *Proceedings of the 16th IEEE International Conference on Data Mining Workshops (ICDMW)*, Carlotta Domeniconi, Francesco Gullo, Francesco Bonchi, Josep Domingo-Ferrer, Ricardo Baeza-Yates, Zhi-Hua Zhou, and Xindong Wu (Eds.). IEEE, Los Alamitos, 218–224. https://doi.org/10.1109/ICDMW.2016.0038

[22] Rainer Schnell and Christian Borgs. 2018. Hardening Encrypted Patient Names Against Cryptographic Attacks Using Cellular Automata. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)* (Singapore). IEEE, Los Alamitos, 518–522. https://doi.org/10.1109/ICDMW.2018.00082

[23] Claude Shannon. 1945. *A Mathematical Theory of Cryptography.*

[24] Sebastian Stammler, Tobias Kussel, Phillipp Schoppmann, Florian Stampe, Galina Tremper, Stefan Katzenbeisser, Kay Hamacher, and Martin Lablans. 2020. Mainzelliste SecureEpiLinker (MainSEL): privacy-preserving record linkage using secure multi-party computation. *Bioinformatics* 38, 6 (09 2020), 1657–1668. https://doi.org/10.1093/bioinformatics/btaa764 arXiv:https://academic.oup.com/bioinformatics/article-pdf/38/6/1657/42744413/btaa764.pdf

[25] Anushka Vidanage, Peter Christen, Thilina Ranbaduge, and Rainer Schnell. 2020. A Graph Matching Attack on Privacy-Preserving Record Linkage. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management.* ACM, New York, 1485–1494. https://doi.org/10.1145/3340531.3411931

[26] Anushka Vidanage, Thilina Ranbaduge, Peter Christen, and Rainer Schnell. 2019. Efficient Pattern Mining Based Cryptanalysis for Privacy-Preserving Record Linkage. In *2019 IEEE 35th International Conference on Data Engineering ICDE 2019.* IEEE, Los Alamitos, 1698–1701. https://doi.org/10.1109/ICDE.2019.00176

[27] Anushka Vidanage, Thilina Ranbaduge, Peter Christen, and Rainer Schnell. 2022. A Taxonomy of Attacks on Privacy-Preserving Record Linkage. *Journal of Privacy and Confidentiality* 12, 1 (Jul. 2022). https://doi.org/10.29012/jpc.764

[28] S. Wolfram. 2022. *A new kind of science.* Wolfram Publishing, Champaign.