Precise and Fast Cryptanalysis for Bloom Filter Based Privacy-Preserving Record Linkage

Peter Christen, Thilina Ranbaduge, Dinusha Vatsalan, and Rainer Schnell

Abstract—Being able to identify records that correspond to the same entity across diverse databases is an increasingly important step in many data analytics projects. Research into privacy-preserving record linkage (PPRL) aims to develop techniques that can link records across databases such that besides the record pairs classified as matches no sensitive information about the entities in these databases is revealed. A popular technique used in PPRL is to encode sensitive values into Bloom filters (bit vectors), which has the advantage of allowing approximate matching using character q-grams. PPRL based on Bloom filter encoding has shown to be accurate and scalable to large databases, and is thus now being used in real-world PPRL systems in Australia, Canada and the UK. However, recent studies have shown that Bloom filters used for PPRL are vulnerable to cryptanalysis attacks that can re-identify some of the sensitive values encoded in these Bloom filters. While previous such attack methods were slow and required knowledge of various encoding parameters, we present a novel efficient attack which exploits how attribute values are encoded into Bloom filters. Our attack method does not require knowledge of the encoding function or its parameter settings used. It is able to correctly re-identify with high precision q-grams that could not have been hashed to certain Bloom filter bit positions, and using these re-identified q-grams it can then re-identify attribute values with high precision. Our method is significantly faster than earlier PPRL cryptanalysis attacks, and in our experimental evaluation it is able to successfully re-identify attribute values from large real-world databases in a few minutes.

Index Terms—Re-identification, frequency analysis, entity resolution, privacy evaluation, privacy attack.

1 Introduction

There is an increasing need in many data engineering and analytics projects to integrate data from multiple sources to enable more sophisticated analysis, to detect and correct inconsistencies in data, or to enrich individual data sources. Record linkage is the task of identifying records that correspond to the same real-world entities across databases when unique entity identifiers (such as social security numbers) are not available [1]. Instead, approximate matching techniques have to be used to calculate the similarities between records, followed by a classification of the compared record pairs into *matches* (assumed to refer to the same entity) and *non-matches* (assumed to refer to different entities). Applications of record linkage range from the health sector and business analytics to national security [1].

In many applications the records to be linked refer to people, for example patients or tax payers, and the attributes required to calculate the similarities between such records include names, addresses, dates of birth, and so on. These quasi-identifiers in many applications cannot be exchanged or shared across organizations due to privacy laws and regulations, or for commercial reasons [2], [3].

Research in the area of *privacy-preserving record linkage* (PPRL) [4], [5], [6] over the past two decades has aimed to develop techniques that allow the linkage of databases without the need of any private or confidential information to be shared between the organizations involved in a linkage project. PPRL is conducted such that only limited information about the record pairs classified as matches (such as their record identifiers or values from selected non-identifiable attributes) is revealed to the organization(s) that require(s) the linkage results. The techniques used in PPRL must also guarantee that no internal or external attacker (adversary) can compromise any sensitive information about the entities in the databases that are being linked [3], [5].

The general idea of PPRL is to encode (or encrypt) the attribute values required for linkage at the source databases such that approximate similarities between records can still be calculated [4]. Various encoding techniques for PPRL have been developed over the past decades. These can be categorized [5] into secure multi-party computation (SMC), perturbation, and hybrid techniques [7]. SMC based techniques generally employ cryptographic operations and they are provably secure [8]. However, to allow for approximate matching, many SMC based techniques have high computation and communication costs, making them currently not scalable to linking large databases [5].

Perturbation based techniques generally provide a trade-off between linkage quality, efficiency, and privacy. Practical PPRL applications [2], [9], [10] are mostly based on perturbation techniques that provide adequate privacy protection while achieving acceptable linkage quality and performance. It is however vital that practical PPRL applications are secure and cannot be attacked as otherwise sensitive information can potentially be revealed [4], [5].

One popular perturbation based technique for PPRL is Bloom filter (BF) encoding [11]. A BF is a bit vector into which elements of a set are hashed into. As we describe in

- *P. Christen and T. Ranbaduge are with the Research School of Computer Science, The Australian National University, Canberra ACT 2601, Australia. E-mail: peter.christen@anu.edu.au, thilina.ranbaduge@anu.edu.au*
- *D. Vatsalan is with the Networks Research Group, Data61-CSIRO, Sydney NSW 2015, Australia. E-mail: dinusha.vatsalan@data61.csiro.au*
- *R. Schnell is with the Methodology Research Group, University Duisburg-Essen, 47057 Duisburg, Germany. E-mail: rainer.schnell@uni-due.de*

Section 3, textual values can be encoded into BFs by converting a string into its set of q-grams (character strings of length q). Techniques to encode numerical values into BFs have also been proposed [12], [13]. The similarity between two BFs can be calculated using a set-based similarity measure [1].

BF based PPRL techniques have the advantage of being efficient and facilitate the linkage of large real-world databases, where approximate matching is crucial due to errors and variations in attribute values [2], [10]. Recent research has however shown that BF based techniques can be vulnerable to cryptanalysis attacks that aim to re-identify the sensitive attribute values encoded in BFs [14], [15], [16], [17], [18]. These attacks use the frequency counts and bit patterns in a set of BFs to map frequent bit patterns to frequent plain-text values. Existing cryptanalysis attacks are however not practical because they assume an attacker has knowledge of certain parameter settings used during BF encoding, and/or they have high computational costs.

In this paper we extend our recently proposed attack method [14] which exploits how q-grams are hashed into BFs. For each bit position, this method identifies a set of *possible* and a set of *not possible* q-grams, and then re-identifies attribute values using only the sets of possible q-grams. However, as our experiments show, these sets have low precision (i.e. they contain q-grams that were not hashed to a certain bit position) which leads to low re-identification accuracy. In contrast, our novel attack method uses the sets of not possible q-grams which have higher precision.

Contributions: We contribute an efficient cryptanalysis attack on BF encoding for PPRL which exploits bit patterns in a set of frequent BFs that are frequency aligned with a set of frequent plain-text attribute values. We present a novel approach to identify q-grams that must have been hashed to certain bit positions in a BF (we name these *assigned* q-grams), and two new methods to refine and expand the sets of possible, not possible, and assigned q-grams. We then propose a novel approach to re-identify attribute values based on the sets of not possible q-grams. We experimentally evaluate our attack method on several real-world databases, confirming its ability to re-identify q-grams at BF bit positions with high precision. Our attack can re-identify sensitive attribute values from databases that contain hundreds of thousands of records in minutes.

Outline: We next provide the required background on PPRL and describe existing attacks on BF based PPRL. In Section 3 we discuss BF encoding for PPRL as well as techniques that aim to make BFs more resilient to attacks. We describe our attack method in Section 4, and analyze its complexity and limitations in Section 5. Using real-world databases, in Section 6 we experimentally show how our attack method is able to re-identify attribute values with high precision, and we provide recommendations on how to use BF encoding securely for practical PPRL. We conclude our work in Section 7 with possible research directions.

2 Background and Related Work

Privacy in record linkage was first investigated in the mid 1990s and since then PPRL has been an active research field. A variety of techniques have been developed to allow linkage using encoded or encrypted values of quasi-identifying attributes across two or more databases [4], [5], [6].

Because real-world data are often dirty, exact matching is not sufficient to achieve high linkage quality in most PPRL applications [5]. This is especially the case when personal identifying attributes, such as names and addresses, are used for linkage because these are prone to variations and typographical errors [1]. Any encoding used for PPRL needs to preserve the original similarities between values to allow approximate matching between encoded values.

PPRL based on SMC techniques encrypt values using, for example, homomorphic encryption [8]. Because these techniques are often computationally expensive [8], recent research in PPRL has developed perturbation based techniques using, for example, noise addition using differential privacy [7], generalization such as k-anonymity, or embedding spaces [5], [6]. These techniques facilitate scalable PPRL at the cost of a trade-off with privacy and linkage quality.

BF encoding is a widely used perturbation technique for PPRL that can support approximate matching by preserving the similarities in the BF space [5], [11], [19]. Schnell et al. [11] introduced an approximate matching approach for string values encoded into BFs, as we describe in Section 3, while Vatsalan and Christen [12] and Karapiperis et al. [13] recently proposed two approaches for encoding numerical attributes. A recent study showed that BF encoding can achieve similar linkage quality compared to traditional record linkage techniques on unencoded values [10].

The efficiency and effectiveness of BFs has sparked wide ranging research using this technique, and first practical applications of BF based PPRL are now being implemented in several countries [2], [9], [10]. However, as we describe next and summarize in Table 5, BFs are prone to cryptanalysis attacks that aim to re-identify encoded values based on frequency information and background knowledge.

A first such attack using a constraint satisfaction solver was studied by Kuzu et al. [16], where plain-text attribute values are mapped to BF bit patterns from an encoded database that meet constraints such as frequency alignments. Kuzu et al. [17] then investigated the accuracy of their attack using two real-world databases. A medical database was used as the encoded database and a public voter registration database as the plain-text database. The experiments showed that the attack is less likely to be accurate in many cases and required significant computational resources with a larger number of frequent names.

Niedermeyer et al. [18] proposed a cryptanalysis attack on BF encodings using filtering and statistical analysis techniques that exploits the linear combination of the values used to generate hash functions [11]. The attack generates a set of possible bit patterns (so called *atoms*) that are aligned with plain-text values according to their frequencies. The experiments conducted showed that the proposed attack was able to re-identify 934 frequent values from $7,580$ surnames before it was stopped. Kroll and Steinmetzer [15] extended this attack for the case where values from several attributes are encoded into one BF. Their attack was able to correctly re-identify 44% of $100,000$ attribute values. Both these attack methods, [18] and [15], assume the double hashing (DH) scheme [20] used by Schnell et al. [11] for BF encoding, which we describe in the following section.

We recently proposed an efficient attack method [14] that works independently of the BF encoding method used and does not require any knowledge about the encoding parameter settings applied. Based on frequent attribute values that are aligned with frequent BFs, sets of *possible* and *not possible* q-grams are identified for each BF bit position. Using the possible q-gram sets, attribute values are re-identified based on BF bit patterns. While this attack is fast, for many BFs it re-identifies wrong attribute values because the sets of not possible q-grams are not fully used. In this paper we substantially improve this attack as we describe in Section 4.

3 Bloom Filter Encoding and Hardening

BFs were proposed by Bloom in 1970 as an efficient way to represent sets [21]. A BF $\mathbf{b}$ is a bit vector of length l bits where initially all bits are set to 0. A set of k independent hash functions, $h_1, \ldots, h_k$, each with range $[1, \ldots, l]$, is used to hash each element s of a set $\mathbf{s}$ into a BF by setting corresponding bit positions to 1: $\mathbf{b}[h_j(s)] = 1$, with $1 \leq j \leq k$. For a given query element s_q, if any of the k hashed positions for s_q is 0 (formally: $\exists j : 1 \leq j \leq k : \mathbf{b}[h_j(s_q)] = 0$), then s_q cannot be in the set $\mathbf{s}$. While the hashing of elements into BFs cannot lead to false negatives [21], false positives are possible due to collisions, as shown in Figure 1.

In PPRL, the (sensitive) values in the attributes used to compare records first need to be converted into sets that can be hashed into BFs, where string values are commonly converted into sets of character q-grams [11]. For each record in a database to be linked, either one BF is generated per attribute used for the linkage (*attribute-level* BF), or one combined single BF is generated where all linkage attributes are hashed into (*record-level* BF) [11], [19]. Using record-level BFs will increase privacy, however it will likely lead to reduced linkage quality because only a single similarity can be calculated between two records (compared to one similarity per attribute when attribute-level BFs are used) [5].

The BFs generated by a database owner for each record in their database are then sent to a linkage unit (LU) that calculates the similarities for pairs of BFs (as described below) and classifies these pairs as matches or non-matches [11], [19]. PPRL can also be conducted without a LU by exchanging BF segments among the database owners to distributively calculate the similarities between BFs [22].

A set-based similarity function such as the Jaccard or Dice coefficient [1] can be used to calculate the similarity between two BFs. As shown in Figure 1, for two BFs, $\mathbf{b}_1$ and $\mathbf{b}_2$, the Dice coefficient is calculated as $sim_D(\mathbf{b}_1, \mathbf{b}_2) = 2c/(x_1 + x_2)$, where c is the number of bit positions that are set to 1 in both BFs (the common 1-bits), and x_1 and x_2 are the number of 1-bits in $\mathbf{b}_1$ and $\mathbf{b}_2$, respectively [5], [11].

The initial proposal of BF encoding for PPRL used a double hashing scheme (DH) [11], [20], where the k bit positions for an element to be hashed are determined by the sum of the integer representation of two independent hash functions. However, DH has a weakness in that it generates a much reduced number of bit patterns that can be exploited by a cryptanalysis attack [15], [18]. As an alternative, random hashing (RH) has recently been proposed [23], where the random seeds for the k hash functions are based on the actual q-grams in the value to be encoded, leading to more diverse bit patterns that could not be successfully attacked by some previous attack methods [15], [18]. However, as our experiments in Section 6 show, our proposed attack can be successful on both the DH and RH encoding methods.

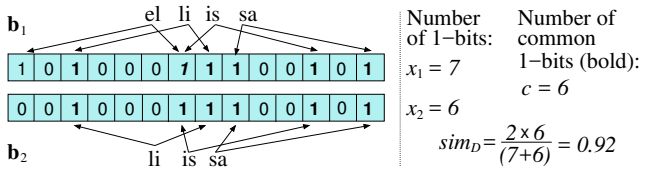

Fig. 1. The Dice coefficient similarity calculation between the names 'elisa' and 'lisa', converted into bigrams ($q = 2$) and encoded into two Bloom filters $\mathbf{b}_1$ and $\mathbf{b}_2$ of length $l = 14$ bits using $k = 2$ hash functions, as described in Section 3. The 1 shown in italics at position $p = 7$ in $\mathbf{b}_1$ is a hash collision, because both 'el' and 'is' are hashed to this position.

The vulnerability of BFs used in PPRL with regard to cryptanalysis attacks has recently been addressed by the development of BF *hardening* techniques [23]. A first approach to make BF encoding more secure is to use record-level instead of attribute-level BFs. With record-level BFs much less frequency information is available that could be exploited by a cryptanalysis attack [16], [18].

Another proposed hardening approach for BFs is XOR-folding, where a BF of length l bits is split into two halves of length $l/2$ each, and then bit-wise exclusive OR (XOR) is applied on these two shorter BFs to combine them into one new BF [24]. The XOR operation makes it impossible to reconstruct the actual 0 or 1 bit values in the original BF. A different hardening approach is balancing [23], where a BF of length l bits is concatenated with its negated copy (all bits flipped) followed by a permutation of the $2l$ bits. As a result, all BFs generated from a database will have exactly half of their bits set to 1 (i.e. have a uniform Hamming weight of l) and thus less frequency information is available that could be exploited by an attack. In our experiments in Section 6 we show that our attack can be successful even when XOR-folding or balancing has been applied.

4 Precise and Efficient Q-gram Based Cryptanalysis on Bloom Filters

Our proposed attack consists of four steps as we describe in detail in this section, and illustrate in Figures 2 to 6. Compared to our previous attack [14] (described in Section 4.1), our new methods substantially improve the quality of the re-identified attribute values, as confirmed in the experimental evaluation in Section 6.

As with existing attacks on BFs for PPRL [14], [15], [16], [17], [18], we assume an attacker has access to a set of BFs, $\mathbf{B}$, and their frequencies. However, unlike earlier attacks, our approach does not require any knowledge about the parameters used in the encoding process, such as the actual hash functions and their number k, nor the hashing approach (such as DH or RH) used.

We assume $\mathbf{B}$ represents a set of BFs that encode sensitive values from one or a few attributes. These are the values we aim to re-identify, i.e. we want to assign plaintext values to BFs. Based on the frequency distribution of the Hamming weights (number of 1-bits) of the BFs in $\mathbf{B}$, an attacker can guess which attribute(s) have been encoded

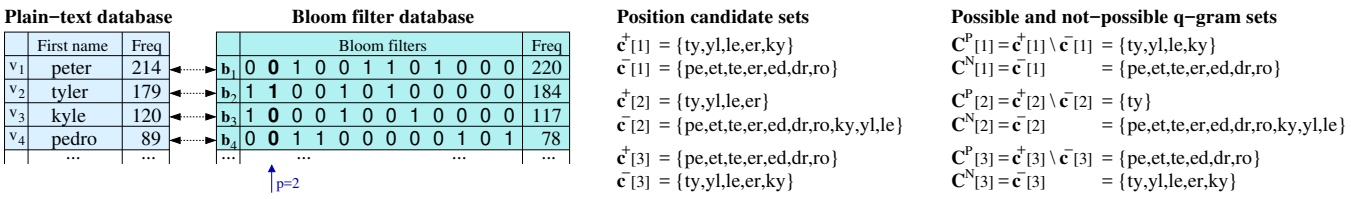

Fig. 2. Example of the candidate q-gram generation step, as described in Section 4.1. Based on a set of plain-text values, $\mathbf{V}$, and a set of BFs, $\mathbf{B}$, that are pair-wise aligned based on their frequencies (left side of the figure), for each BF bit position p and the 0- and 1-bit patterns and generated q-grams from aligned attribute values, the candidate sets of possible, $\mathbf{c}^+[p]$, and *not* possible, $\mathbf{c}^-[p]$, q-grams are generated (middle of figure). After analyzing all pairs of frequent attribute values and BFs, the final set of q-grams possible at any position p, $\mathbf{C}^P[p]$, is then the set of not possible q-grams removed from the set of possible q-grams, $\mathbf{C}^P[p] = \mathbf{c}^+[p] \setminus \mathbf{c}^-[p]$, while the final not possible q-gram set is the not possible candidate set, $\mathbf{C}^N[p] = \mathbf{c}^-[p]$ (right side of figure). For example, for the BF bit position $p = 2$ (shown in bold), ty is the only possible q-gram ($\mathbf{C}^P[2] = \{ty\}$) because the value 'kyle' (with q-grams $\{ky, yl, le\}$) has a 0-bit at position 2, which means the q-grams ky, yl, and le (as well as all q-grams generated from the attribute values 'peter' and 'pedro', that also have a 0 at bit position 2) could not have been hashed to this position.

into these BFs, because different attributes (such as first name, surname, or postcode) have distinctive distributions of Hamming weights [23]. The attacker can sample values from a large public database (such as a telephone directory) and select a set of plain-text values, $\mathbf{V}$, from a single or a few attributes where their corresponding q-gram sets have a frequency distribution that is similar to the distribution of Hamming weights of the BF set $\mathbf{B}$ to be attacked.

As for notation, we use bold letters for BFs, sets and lists (with upper-case bold letters for sets or lists of BFs, sets and lists) and normal type letters for integer or string values. We denote sets with curly and lists with square brackets (where lists have an order while sets do not), the frequency of how often a specific BF $\mathbf{b}_i \in \mathbf{B}$ occurs with $\mathbf{b}_i.f$, and for each plain-text value $v_i \in \mathbf{V}$ its frequency with $v_i.f$.

4.1 Q-gram Candidate Set Generation

In this first step of our attack method we identify for each BF bit position p the sets of q-grams that are possible and not possible, respectively, at that position, as shown in Figure 2 and detailed in Algorithm 1 [14]. Our approach exploits the construction principle of BFs as follows.

***Theorem 1.* Hashing q-grams into Bloom filters**: For a given BF, $\mathbf{b}$, and set of q-grams, $\mathbf{q}$, extracted from an attribute value, v, a 1-bit in a certain position, p, in $\mathbf{b}$ implies that *at least one q-gram* $q_i \in \mathbf{q}$ from v must hash to p. On the other hand, a 0-bit at a certain position, p, implies *no q-gram* $q_i \in \mathbf{q}$ from v could have been hashed to p.

Proof: As described in Section 3, the elements s of a set $\mathbf{s}$ are hashed into a BF of length l bits using $k \geq 1$ hash functions $h_1, \ldots, h_k$. Assuming a BF is initialized with all bits set to 0, the bit at position p (with $1 \leq p \leq l$), $\mathbf{b}[p]$, is set to 1 if and only if $\mathbf{b}[p] = 1 \Leftrightarrow (\exists s \in \mathbf{s} : \exists h_j, 1 \leq j \leq k : h_j(s) = p)$. A bit at position p can only be 0 if none of the k hash functions 'hits' position p for any of the elements in $\mathbf{s}$: $\mathbf{b}[p] = 0 \Leftrightarrow (\forall s \in \mathbf{s} : \forall h_j, 1 \leq j \leq k : h_j(s) \neq p)$. □

Algorithm 1 starts by initializing two empty candidate sets for each BF position p: $\mathbf{c}^+[p]$ of possible q-grams and $\mathbf{c}^-[p]$ of *not* possible q-grams. In lines 2 and 3 we identify the BFs in $\mathbf{B}$ and the attribute values in $\mathbf{V}$ that occur at least f_m times and include them in two lists $\mathbf{B}_F$ and $\mathbf{V}_F$. In line 4 we then sort both these lists according to their frequencies in reverse order (most frequent first), and in line 5 we align a BF $\mathbf{b}_i$ and corresponding attribute value v_i into the sorted list $\mathbf{A}$ of pairs $(\mathbf{b}_i, v_i)$. We only consider such pairs as long as both $\mathbf{b}_i$ and v_i have a unique frequency compared to the next, less frequent, $\mathbf{b}_j$ and v_j. This criterion ensures we do not have an attribute value that could correspond to two or more BFs and vice versa, as this would likely lead to an incorrect alignment of a BF with an attribute value and thus incorrect mappings of q-grams to BF bit positions.

Algorithm 1: ***Possible and not possible q-gram set generation*** [14]

Input:
- $\mathbf{V}$: Attribute values and their frequencies from a public database
- $\mathbf{B}$: BFs and their frequencies from the sensitive database
- l: BF length
- q: Length of substrings to extract from attribute values
- f_m: Minimum frequency for BFs and attribute values

Output:
- $\mathbf{C}^P$: List of possible q-gram sets at each BF position
- $\mathbf{C}^N$: List of not possible q-gram sets at each BF position
- $\mathbf{A}$: Aligned tuples of frequent BFs and frequent attribute values

1: $\mathbf{c}^+[p] = \{\}, \mathbf{c}^-[p] = \{\}, 1 \leq p \leq l$ // *Initialize candidate q-gram sets*
2: $\mathbf{V}_F = \{v_i \in \mathbf{V} : v_i.f \geq f_m\}$ // *Get freq attribute values*
3: $\mathbf{B}_F = \{\mathbf{b}_i \in \mathbf{B} : \mathbf{b}_i.f \geq f_m\}$ // *Get freq BFs*
4: $revFreqSort(\mathbf{B}_F), revFreqSort(\mathbf{V}_F)$
5: $\mathbf{A} = [(\mathbf{b}_i, v_i) : \mathbf{b}_i \in \mathbf{B}_F, v_i \in \mathbf{V}_F : \mathbf{b}_i.f > \mathbf{b}_j.f \wedge v_i.f > v_j.f : 1 \leq i < j \leq min(|\mathbf{V}_F|, |\mathbf{B}_F|)]$
6: **for** $(\mathbf{b}_i, v_i) \in \mathbf{A}$ **do**: // *Loop over pairs of freq BF and attr value*
7: $\mathbf{q}_i = genQGramSet(v_i, q)$ // *Convert attr value into q-gram set*
8: **for** $1 \leq p \leq l$ **do**: // *Loop over BF bit positions*
9: **if** $(\mathbf{b}_i[p] == 1)$ **then**:
10: $\mathbf{c}^+[p] = \mathbf{c}^+[p] \cup \mathbf{q}_i$ // *Add to set of possible q-grams*
11: **else**:
12: $\mathbf{c}^-[p] = \mathbf{c}^-[p] \cup \mathbf{q}_i$ // *Add to set of not possible q-grams*
13: $\mathbf{C}^P[p] = \{\}, \mathbf{C}^N[p] = \{\}, 1 \leq p \leq l$ // *Initialize final q-gram sets*
14: **for** $1 \leq p \leq l$ **do**:
15: $\mathbf{C}^P[p] = \mathbf{c}^+[p] \setminus \mathbf{c}^-[p]$ // *Final set of possible q-grams*
16: $\mathbf{C}^N[p] = \mathbf{c}^-[p]$ // *Final set of not possible q-grams*
17: **return** $\mathbf{C}^P, \mathbf{C}^N, \mathbf{A}$

Starting in line 6, we iterate over pairs of aligned BFs and attribute values, $(\mathbf{b}_i, v_i) \in \mathbf{A}$, and in line 7 we convert v_i into its set $\mathbf{q}_i$ of character q-grams of length q. Then we loop over all BF bit positions, p, in line 8 and if the bit in BF $\mathbf{b}_i$ at position p is 1 then in line 10 we add the q-grams in set $\mathbf{q}_i$ to the set $\mathbf{c}^+[p]$ of possible candidate q-grams at that position because this 1-bit means at least one q-gram from $\mathbf{q}_i$ was hashed to position p (following Theorem 1). Conversely, if the bit in BF $\mathbf{b}_i$ at position p is 0 then we add all q-grams in $\mathbf{q}_i$ to $\mathbf{c}^-[p]$ in line 12 because no q-gram from the set $\mathbf{q}_i$ could have been hashed to that position.

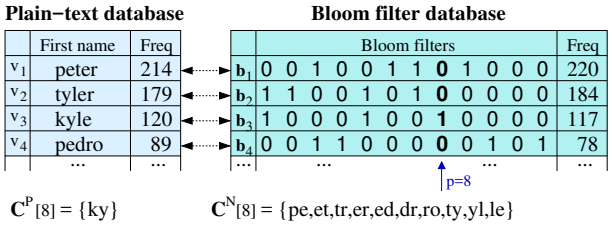

Fig. 3. Example q-gram assignment based on four attribute values aligned with four BFs, and the set of possible and not possible q-gram sets at position $p = 8$ (shown in bold) as generated by Algorithm 1. For value 'kyle', the only possible q-gram that could have been hashed to position 8 is 'ky' because all other q-grams of 'kyle' are in the set $\mathbf{C}^N[8]$. We can therefore *assign* the q-gram 'ky' to position 8 as $\mathbf{C}^A[8] = \{ky\}$.

After initializing in line 13 the final lists of possible, $\mathbf{C}^P[p]$, and not possible, $\mathbf{C}^N[p]$, q-gram sets, in lines 14 to 16 we iterate over all bit positions p and generate the final set of possible q-grams at each position as $\mathbf{C}^P[p] = \mathbf{c}^+[p] \setminus \mathbf{c}^-[p]$. The final set of not possible q-grams at position p is equal to the candidate set of not possible q-grams: $\mathbf{C}^N[p] = \mathbf{c}^-[p]$.

At the end of Algorithm 1, based on the frequency aligned frequent BFs and frequent attribute values in $\mathbf{A}$, for each BF bit position p we have the two sets of q-grams that could have been hashed to position p, $\mathbf{C}^P[p]$, and that cannot have been hashed to that position, $\mathbf{C}^N[p]$.

Note that this candidate generation step is the main step used in our previous cryptanalysis attack method [14], where (as we discuss in Section 4.4) the attack only makes use of the q-gram sets in $\mathbf{C}^P$. However, as our experiments in Section 6 show, the precision of these sets is low because many of the q-grams in a set $\mathbf{C}^P[p]$ were in fact not hashed to position p. On the other hand, for basic BFs the sets $\mathbf{C}^N[p]$ are of high precision because a 0-bit specifies all those q-grams that could not have been hashed to a certain position. The BF hardening techniques described in Section 3 however invalidate this property of 0-bits.

4.2 Q-gram Position Assignment

In this section we propose a method that uses the list $\mathbf{C}^P$ of possible q-gram sets, as generated in Algorithm 1, to identify q-grams that must have been hashed to a certain bit position (we name these as *assigned* q-grams). While $\mathbf{C}^P[p]$ contains the q-grams that *possibly* have been hashed to bit position p, so far we have not validated if any of them actually has been hashed to this position. Our method identifies q-grams where for a given frequent BF and attribute value pair $(\mathbf{b}_i, v_i) \in \mathbf{A}$ (as generated in Algorithm 1) there is only one single q-gram that could have generated the 1-bit for v_i at a certain position p, as illustrated in Figure 3.

Algorithm 2, which identifies assigned q-grams, starts by initializing one empty set of assigned q-grams, $\mathbf{C}^A[p]$, per BF bit position p. The loop in line 2 then iterates over the aligned pairs of frequent BFs $\mathbf{b}_i$ and frequent attribute values v_i in $\mathbf{A}$ as generated by Algorithm 1. For each BF bit position p, if the bit $\mathbf{b}_i[p]$ is set to 1 then in line 6 we intersect the q-gram set $\mathbf{q}_i$ (as generated from v_i in line 3) with the set of possible q-grams at that position, $\mathbf{C}^P[p]$. If the result of this intersection is a single q-gram (the test in line 7, where $|\cdot|$ denotes the cardinality of a set), then this

Algorithm 2: *Q-gram position assignment*

Input:
- $\mathbf{A}$: List of aligned frequent attribute values and BFs (from Algo. 1)
- $\mathbf{C}^P$: List of possible q-grams for BF positions (from Algo. 1)
- l: Length of BFs
- q: Length of substrings to extract from attribute values

Output:
- $\mathbf{C}^A$: List of assigned q-gram sets at each BF position

1: $\mathbf{C}^A[p] = \{\}, 1 \leq p \leq l$ // *Initialize assigned q-gram sets*
2: **for** $(\mathbf{b}_i, v_i) \in \mathbf{A}$ **do**: // *Loop over pairs of freq BF and attr value*
3: $\mathbf{q}_i = genQGramSet(v_i, q)$ // *Convert attr value into q-grams*
4: **for** $1 \leq p \leq l$ **do**: // *Loop over BF bit positions*
5: **if** $(\mathbf{b}_i[p] == 1)$ **then**:
6: $\mathbf{q}_p = \mathbf{q}_i \cap \mathbf{C}^P[p]$ // *Get set of possible q-grams at position*
7: **if** $(|\mathbf{q}_p| == 1)$ **then**: // *Only one possible q-gram*
8: $\mathbf{C}^A[p] = \mathbf{C}^A[p] \cup \mathbf{q}_p$ // *Add to set of assigned q-grams*
9: **return** $\mathbf{C}^A$

q-gram is added to the set of assigned q-grams at position p in line 8. At the end of Algorithm 2 we have the new list of q-gram sets $\mathbf{C}^A[p]$ that contain those q-grams that must have been hashed to a certain position p.

4.3 Q-gram Set Refinement and Expansion

A drawback of the candidate set generation method described in Section 4.1 is that only a small number of frequent attribute values, v_i, can accurately be aligned in list $\mathbf{A}$ to the corresponding frequent BFs, $\mathbf{b}_i$, that encode these values. The reason for this is that the frequencies of the attribute values encoded into BFs are likely somewhat different from the frequencies in a plain-text database that an attacker has access to, as illustrated in the left side of Figure 2.

A frequency based ordering of values in the same attribute from two similar databases will likely lead to the same ordering of the top most frequent values. However, for less frequent values the ordering at some point will likely become different. For example, if 'smith' is the most frequent surname in a population and it is considerably more frequent than 'miller', then it is likely that 'smith' is also more frequent than 'miller' in two large databases that are samples of this population. However, in one of these databases the surname 'williams' might be ranked 20th and 'rendell' 21st, while in another database their order is reversed. Note that most existing attacks on BF encoding for PPRL make this assumption of aligning frequent BFs and frequent plain-text values [15], [16], [17], [18].

Therefore, as frequencies become lower and the differences between the respective frequencies of two BFs or two attribute values become smaller, wrong assignments of attribute values to BFs (that encode a different value) can happen. Because the list $\mathbf{A}$ in Algorithm 1 likely only includes a small number of attribute values, the number of q-grams in those values is likely only a fraction of all possible q-grams that occur in a domain. In our experiments only around 5% of all possible q-grams for first names and surnames are included in the values $v_i \in \mathbf{A}$.

Our first improvement identifies new not possible positions for q-grams that are in the sets of possible q-grams, while the second improvement expands the sets of possible, not possible and assigned q-grams with additional q-grams. Both improvements are based on analyzing the q-gram sets

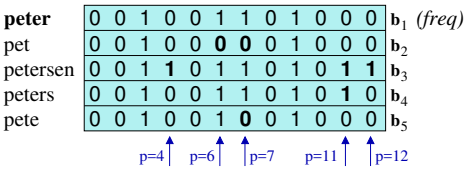

Fig. 4. Example q-gram refinement and expansion for the frequent attribute value 'peter' (in bold) and four not frequent shorter and longer values. For the shorter values, from the bit patterns in BFs $\mathbf{b}_2$ and $\mathbf{b}_5$, it becomes clear that position $p = 7$ must encode q-gram 'er' because both $\mathbf{b}_2[7]$ and $\mathbf{b}_5[7]$ are 0 and only 'pet' and 'pete' do not contain 'er'. Similarly, $p = 6$ must encode 'te' because only $\mathbf{b}_2$ with value 'pet' does not contain 'te'. For the longer values, $p = 4$ and $p = 12$ can only encode the q-grams 'se' and 'en' because only $\mathbf{b}_3$ with value 'petersen' (containing both 'se' and 'en') has a 1-bit in these two positions, while $p = 11$ can also encode 'rs' as it occurs in both 'peters' and 'petersen'.

and bit patterns between frequent (as included in the list $\mathbf{A}$ in Algorithm 1) and not frequent attribute values and BFs, respectively, as detailed in Algorithm 3.

We use the example shown in Figure 4 to illustrate our refinement and expansion methods. For each frequent value $v_i \in \mathbf{A}$ (as returned by Algorithm 1), we identify the not frequent values $v_j \in \mathbf{V} \setminus \mathbf{V}_F$ (with $\mathbf{V}_F$ the set of all frequent attribute values in $\mathbf{A}$) where their q-gram sets $\mathbf{q}_j$ are either a proper subset of the q-gram set $\mathbf{q}_i$ of the frequent value v_i, or a proper superset of $\mathbf{q}_i$. For example, in Figure 4, the q-gram sets of 'pet' and 'pete' are subsets of the q-gram set of 'peter', while the q-gram sets of 'peters' and 'petersen' are supersets of the q-gram set of 'peter'.

This refinement and expansion process depends upon the availability of sub- and supersets of q-gram sets as generated from attribute values; which is dependent on data characteristics such as the size of alphabets, the length of attribute values, and their frequency distributions. As Table 1 shows, in our experimental data sets such sub- and supersets seem to be quite common even in attributes where all values are of the same length or where attribute values have a large variety. For example, even when all Zipcodes in a data set have the same length, some have a q-gram set that is a sub- or superset of others. When using $q = 2$, Zipcode '28725', {28,87,72,25}, is a superset of '28728', {28,87,72}.

Let us denote the set of all not frequent attribute values with $\mathbf{V}_N$, with $\mathbf{V}_N = \mathbf{V} \setminus \mathbf{V}_F$. For a given frequent attribute value $v_i \in \mathbf{V}_F$, we find the set of all shorter attribute values $\mathbf{V}_i^S \subset \mathbf{V}_N$ where their q-gram sets are proper subsets of the q-gram set $\mathbf{q}_i$ of v_i (with $\mathbf{q}_i = genQGramSet(v_i, q)$), and the set of all longer attribute values $\mathbf{V}_i^L \subset \mathbf{V}_N$ where their q-gram sets are proper supersets of $\mathbf{q}_i$. More formally:

$$\mathbf{V}_i^S = \{v_j \in \mathbf{V}_N : genQGramSet(v_j, q) \subset \mathbf{q}_i\}$$
$$\mathbf{V}_i^L = \{v_j \in \mathbf{V}_N : genQGramSet(v_j, q) \supset \mathbf{q}_i\}$$

Similarly, for each frequent BF $\mathbf{b}_i \in \mathbf{A}$, we can identify candidate BFs in the set of not frequent BFs, $\mathbf{B}_N$ (where $\mathbf{B}_N = \mathbf{B} \setminus \mathbf{B}_F$ and $\mathbf{B}_F$ is the set of all frequent BFs in $\mathbf{A}$), that potentially can encode a value $v_j \in \mathbf{V}_i^S$ as those BFs $\mathbf{b}_j \in \mathbf{B}_N$ that have less 1-bits than $\mathbf{b}_i$, and all 1-bits of $\mathbf{b}_j$ are also set to 1 in $\mathbf{b}_i$. If a BF $\mathbf{b}_j$ has even a single 1-bit in a position where $\mathbf{b}_i$ has a 0-bit then $\mathbf{b}_j$ cannot encode a value in $\mathbf{V}_i^S$. Similarly, candidate BFs $\mathbf{b}_j$ that could encode a value $v_j \in \mathbf{V}_i^L$ are those BFs that have 1-bits in all the positions

TABLE 1
The number and percentage of attribute values (based on their q-gram sets, with $q = 2$) in the NCVR data set (as described in Section 6.1) that are a sub- and superset of at least one other attribute value.

Attribute	Unique values	Subsets	Supersets
First name	22,101	5,321 / 24.08%	9,104 / 41.19%
Surname	40,089	8,707 / 21.72%	15,459 / 38.56%
Address	210,154	3,436 / 1.64%	3,551 / 1.69%
City	735	27 / 3.67%	14 / 1.90%
Zipcode	824	24 / 2.91%	111 / 13.47%

where $\mathbf{b}_i$ also has a 1-bit, plus they need to have at least one extra 1-bit (but likely more) at a position where $\mathbf{b}_i$ has a 0-bit. If there is a single position where $\mathbf{b}_j$ has a 0-bit but $\mathbf{b}_i$ has a 1-bit then $\mathbf{b}_j$ cannot encode an attribute value in $\mathbf{V}_i^L$ because this 0-bit in $\mathbf{b}_j$ means the corresponding value v_j does not contain all q-grams that v_i contains.

For a given frequent BF $\mathbf{b}_i \in \mathbf{A}$, we define $\mathbf{B}_i^S$ and $\mathbf{B}_i^L$ to be the candidate sets of not frequent BFs that can encode shorter and longer values in $\mathbf{V}_i^S$ and $\mathbf{V}_i^L$, respectively:

$$\mathbf{B}_i^S = \{\mathbf{b}_j \in \mathbf{B}_N : (\mathbf{b}_i \otimes \mathbf{b}_j = \mathbf{b}_j) \wedge hw(\mathbf{b}_i) > hw(\mathbf{b}_j)\}$$
$$\mathbf{B}_i^L = \{\mathbf{b}_j \in \mathbf{B}_N : (\mathbf{b}_i \otimes \mathbf{b}_j = \mathbf{b}_i) \wedge hw(\mathbf{b}_i) < hw(\mathbf{b}_j)\}$$

The $\otimes$ operator denotes the bit-wise logical AND, and the function $hw(\mathbf{b})$ counts the number of 1-bits in a BF $\mathbf{b}$, i.e. returns its Hamming weight.

Because of collisions, it might be possible that an extra q-gram that only occurs in a value $v_j \in \mathbf{V}_i^L$ is hashed to the same bit positions as the q-grams in v_i. Given $k > 1$ hash functions are used this is highly unlikely. The expected fraction of 1-bits in a BF of length l after $n = |\mathbf{q}|$ q-grams have each been hashed k times is $f_1 = 1 - (1 - 1/l)^{kn}$ (see page 111 in [25]). The probability P for one extra q-gram to be hashed k times to bit positions that have already been set to 1 by other q-grams is therefore $P = f_1^k$. For example, for a BF that has 60% 1-bits and $k = 30$ hash functions are used, this probability becomes $P = 2.21 * 10^{-7}$. Even if this would happen, it would only mean that a BF $\mathbf{b}_j \in \mathbf{B}_N$ would not be included into $\mathbf{B}_i^L$, however it would not lead to a wrong refinement or expansion of q-gram sets.

If the alignment in Algorithm 1 of frequent attribute values with frequent BFs into the list $\mathbf{A}$ is correct, and if there are shorter ($\mathbf{V}_i^S$) and longer ($\mathbf{V}_i^L$) not frequent attribute values for a given frequent value $v_i \in \mathbf{A}$, then the BFs identified in the sets $\mathbf{B}_i^S$ and $\mathbf{B}_i^L$ must encode such shorter and longer attribute values, respectively.

In our refinement and expansion approach detailed in Algorithm 3, in lines 4 to 9, for all frequent BF and attribute value pairs $(\mathbf{b}_i, v_i) \in \mathbf{A}$ we first identify all the shorter and longer attribute values, $\mathbf{V}_i^S$ and $\mathbf{V}_i^L$, and the BFs $\mathbf{B}_i^S$ and $\mathbf{B}_i^L$. Note that if there are several shorter or longer values and several corresponding BFs then we do not know which shorter value $v_j \in \mathbf{V}_i^S$ corresponds to which BF $\mathbf{b}_j \in \mathbf{B}_i^S$, and which longer value $v_j \in \mathbf{V}_i^L$ corresponds to which BF $\mathbf{b}_j \in \mathbf{B}_i^L$. The larger the sets $\mathbf{V}_i^S$, $\mathbf{B}_i^S$, $\mathbf{V}_i^L$, and $\mathbf{B}_i^L$ are, the less accurate the refinement and expansion process will become because an increasing number of q-grams and bit positions will differ between v_i and the v_j's and $\mathbf{b}_i$ and the $\mathbf{b}_j$'s, respectively. We therefore use a parameter m to limit the size of these sets in Algorithm 3 (lines 10 and 21).

Algorithm 3: *Q-gram set refinement and expansion*

Input:
- $\mathbf{V}$: Attribute values and their frequencies from a public database
- $\mathbf{B}$: BFs and their frequencies from the sensitive database
- $\mathbf{A}$: List of aligned frequent attribute values and BFs (from Algo. 1)
- l: Length of BFs
- q: Length of substrings to extract from attribute values
- m: Maximum size of attribute value and BF sets

Output:
- $\mathbf{C}_r^P$: List of possible q-gram sets at each BF position
- $\mathbf{C}_r^N$: List of not possible q-gram sets at each BF position
- $\mathbf{C}_r^A$: List of assigned q-gram sets at each BF position

1: $\mathbf{C}_r^P[p] = \{\}$, $\mathbf{C}_r^N[p] = \{\}$, $\mathbf{C}_r^A[p] = \{\}$, $1 \leq p \leq l$
2: $\mathbf{V}_F = \{v_i : (v_i, \mathbf{b}_i) \in \mathbf{A}\}$, $\mathbf{B}_F = \{\mathbf{b}_i : (v_i, \mathbf{b}_i) \in \mathbf{A}\}$
3: $\mathbf{V}_N = \{v_j \in \mathbf{V} : v_j \notin \mathbf{V}_F\}$, $\mathbf{B}_N = \{\mathbf{b}_j \in \mathbf{B} : \mathbf{b}_j \notin \mathbf{B}_F\}$
4: **for** $(\mathbf{b}_i, v_i) \in \mathbf{A}$ **do**: *// Loop over pairs of freq BF and attr value*
5: $\mathbf{q}_i = genQGramSet(v_i, q)$ *// Convert attr value into q-grams*
6: $\mathbf{V}_i^S = \{v_j \in \mathbf{V}_N : genQGramSet(v_j, q) \subset \mathbf{q}_i\}$
7: $\mathbf{V}_i^L = \{v_j \in \mathbf{V}_N : genQGramSet(v_j, q) \supset \mathbf{q}_i\}$
8: $\mathbf{B}_i^S = \{\mathbf{b}_j \in \mathbf{B}_N : (\mathbf{b}_i \otimes \mathbf{b}_j = \mathbf{b}_j) \wedge hw(\mathbf{b}_i) > hw(\mathbf{b}_j)\}$
9: $\mathbf{B}_i^L = \{\mathbf{b}_j \in \mathbf{B}_N : (\mathbf{b}_i \otimes \mathbf{b}_j = \mathbf{b}_i) \wedge hw(\mathbf{b}_i) < hw(\mathbf{b}_j)\}$
10: **if** $(0 < |\mathbf{V}_i^S| \leq m) \wedge (0 < |\mathbf{B}_i^S| \leq m)$ **then**: *// **Refinement***
11: $\mathbf{q}_u = \cup\, genQGramSet(v_j, q)\ \forall\ v_j \in \mathbf{V}_i^S$
12: $\mathbf{q}_d = \mathbf{q}_i \setminus \mathbf{q}_u$ *// Q-grams that only occur in v_i*
13: $\mathbf{b}_o = \oplus\, \mathbf{b}_j\ \forall\ \mathbf{b}_j \in \mathbf{B}_i^S$ *// Bit-wise OR of BFs*
14: **if** $(\mathbf{q}_d \neq \emptyset) \wedge (\mathbf{b}_o \neq \mathbf{b}_i)$ **then**:
15: **for** $1 \leq p \leq l$ **do**: *// Loop over BF bit positions*
16: **if** $(\mathbf{b}_o[p] == 0) \wedge (\mathbf{b}_i[p] == 1)$ **then**:
17: $\mathbf{C}_r^P[p] = \mathbf{C}_r^P[p] \cup \mathbf{q}_d$ *// Add to possible q-grams*
18: $\mathbf{C}_r^N[p] = \mathbf{C}_r^N[p] \cup \mathbf{q}_u$ *// Add to not possible q-grams*
19: **if** $(|\mathbf{V}_i^S| == 1) \wedge (|\mathbf{B}_i^S| == 1) \wedge (|\mathbf{q}_d| == 1)$ **then**:
20: $\mathbf{C}_r^A[p] = \mathbf{C}_r^A[p] \cup \mathbf{q}_d$ *// Add to assigned q-grams*
21: **if** $(0 < |\mathbf{V}_i^L| \leq m) \wedge (0 < |\mathbf{B}_i^L| \leq m)$ **then**: *// **Expansion***
22: $\mathbf{q}_u = \cup\, genQGramSet(v_j, q)\ \forall\ v_j \in \mathbf{V}_i^L$
23: $\mathbf{q}_s = \cap\, genQGramSet(v_j, q)\ \forall\ v_j \in \mathbf{V}_i^L$
24: $\mathbf{q}_d = \mathbf{q}_s \setminus \mathbf{q}_i$ *// Q-grams that only occur in all $v_j \in \mathbf{V}_i^L$*
25: $\mathbf{b}_o = \oplus\, \mathbf{b}_j\ \forall\ \mathbf{b}_j \in \mathbf{B}_i^L$
26: $\mathbf{b}_a = \otimes\, \mathbf{b}_j\ \forall\ \mathbf{b}_j \in \mathbf{B}_i^L$
27: **if** $(\mathbf{q}_d \neq \emptyset) \wedge (\mathbf{b}_a \neq \mathbf{b}_i)$ **then**:
28: **for** $1 \leq p \leq l$ **do**: *// Loop over BF bit positions*
29: **if** $(\mathbf{b}_a[p] == 1) \wedge (\mathbf{b}_i[p] == 0)$ **then**:
30: $\mathbf{C}_r^P[p] = \mathbf{C}_r^P[p] \cup \mathbf{q}_d$ *// Add to possible q-grams*
31: **if** $(|\mathbf{V}_i^L| == 1) \wedge (|\mathbf{B}_i^L| == 1) \wedge (|\mathbf{q}_d| == 1)$ **then**:
32: $\mathbf{C}_r^A[p] = \mathbf{C}_r^A[p] \cup \mathbf{q}_d$ *// Add to assigned q-grams*
33: **else if** $(\mathbf{b}_o[p] == 0) \wedge (\mathbf{b}_i[p] == 0)$ **then**:
34: $\mathbf{C}_r^N[p] = \mathbf{C}_r^N[p] \cup \mathbf{q}_u$ *// Add to not possible q-grams*
35: **return** $\mathbf{C}_r^P$, $\mathbf{C}_r^N$, and $\mathbf{C}_r^A$

A small value of m will lead to smaller refined and expanded q-gram sets $\mathbf{C}_r^P$, $\mathbf{C}_r^N$ and $\mathbf{C}_r^A$ (as generated by Algorithm 3) that are of higher quality, while increasing m will lead to larger refined and expanded q-gram sets that however might contain more incorrectly identified q-grams as m gets larger. We will evaluate the quality of re-identified q-gram sets with different values of m in Section 6.

4.3.1 Refinement

Independent of the sizes of $\mathbf{V}_i^S$ and $\mathbf{B}_i^S$, we observe that a BF bit position that is 0 in all the $b_j \in \mathbf{B}_i^S$ but 1 in $\mathbf{b}_i$ can only encode a q-gram that occurs in v_i but in none of the $v_j \in \mathbf{V}_i^S$. Based on this, in lines 11 and 12 in Algorithm 3 we get the union $\mathbf{q}_u$ of all q-grams from the attribute values $v_j \in \mathbf{V}_i^S$, and then extract the set of differing q-grams, $\mathbf{q}_d$, as those q-grams that only occur in v_i but in none of the $v_j \in \mathbf{V}_i^S$. In line 13 we generate the bit-wise OR ($\oplus$) of all BFs in $\mathbf{B}_i^S$ as $\mathbf{b}_o$, because any position that is 0 after this operation (but is 1 in $\mathbf{b}_i$) can encode only q-grams in $\mathbf{q}_d$.

If $\mathbf{q}_d$ is not empty (i.e. there are differing q-grams) and $\mathbf{b}_o$ differs from $\mathbf{b}_i$, then we loop over all bit positions in line 15 and if for a certain position p the bit in $\mathbf{b}_o[p] = 0$ but $\mathbf{b}_i[p] = 1$ then we know that this position can only encode the differing q-grams in $\mathbf{q}_d$. In line 17 we therefore add the q-grams in $\mathbf{q}_d$ to the set of possible q-grams at that position, $\mathbf{C}_r^P[p]$, and in line 18 we add the q-grams in $\mathbf{q}_u$ (that occur in any of the shorter attribute values) to the set of not possible q-grams at that position, $\mathbf{C}_r^N[p]$.

A special case to consider is if there is only one shorter attribute value, v_j, and one corresponding BF, $\mathbf{b}_j$, and only one q-gram differs between v_i and v_j (line 19). In this case we know that this q-gram must have been hashed to a certain position, and therefore in line 20 we add the q-gram to the set of assigned q-grams at that position, $\mathbf{C}_r^A[p]$.

4.3.2 Expansion

Similar to the refinement approach, independent of the sizes of $\mathbf{V}_i^L$ and $\mathbf{B}_i^L$, a BF bit position that is 0 in $\mathbf{b}_i$ and 1 in all $\mathbf{b}_j \in \mathbf{B}_i^L$ must encode a q-gram that occurs in all $v_j \in \mathbf{V}_i^L$. On the other hand, if none of the $\mathbf{b}_j \in \mathbf{B}_i^L$ have a 1-bit at a given position then none of the q-grams in any of the $v_j \in \mathbf{V}_i^L$ could have been hashed to this position. Based on this, in lines 22 and 23 in Algorithm 3 we generate the union and intersection of all q-gram sets as $\mathbf{q}_u$ and $\mathbf{q}_s$, respectively, from the $v_j \in \mathbf{V}_i^L$. In line 24 we identify all q-grams that occur in all longer (not frequent) attribute values but not in the frequent value as $\mathbf{q}_d$. In lines 25 and 26 we calculate the bit-wise OR ($\oplus$) of all BFs in $\mathbf{B}_i^L$ as $\mathbf{b}_o$, and the bit-wise AND ($\otimes$) of all BFs in $\mathbf{B}_i^L$ as $\mathbf{b}_a$. The former is required to identify bit positions where no q-gram in $\mathbf{q}_d$ could have been hashed to, while the latter is needed to identify positions where the extra q-grams in $\mathbf{q}_s$ could have been hashed to.

In line 27, if both $\mathbf{q}_d$ is not empty (i.e. there are differing q-grams) and $\mathbf{b}_a$ differs from $\mathbf{b}_i$, then we loop over all bit positions (line 28) and for any position p where $\mathbf{b}_a[p] = 1$ and $\mathbf{b}_i[p] = 0$ in line 30 we add the set of extra q-grams, $\mathbf{q}_d$ (that occur in all longer values but not the frequent value) to the set of possible q-grams at that position, $\mathbf{C}_r^P[p]$. Additionally, if there is only one longer attribute value, v_j, and one BF, $\mathbf{b}_j$, and only one q-gram differs between v_i and v_j (line 31), then we know this single q-gram in $\mathbf{q}_d$ must have been hashed to this position and so we add it in line 32 to the set of assigned q-grams, $\mathbf{C}_r^A[p]$, at this position.

As described above, if a BF position p is 0 in all the BFs $\mathbf{b}_j \in \mathbf{B}_i^L$ and also 0 in the frequent BF $\mathbf{b}_i$ (the test in line 33) then we know that no q-gram in any longer value $v_j \in \mathbf{V}_i^L$ could have been hashed to this position. In line 34 we therefore add the q-grams in $\mathbf{q}_u$ to the set of not possible q-grams, $\mathbf{C}_r^N[p]$, at that position.

4.3.3 Merging Q-gram Sets

At the end of the refinement and expansion step we have new sets of q-grams that can be merged with the corresponding sets from Algorithms 1 and 2. If we denote the sets of *basic* possible, not possible and assigned q-grams from Algorithms 1 and 2 with $\mathbf{C}_b^P$, $\mathbf{C}_b^N$ and $\mathbf{C}_b^A$, and the *refined and expanded* q-gram sets generated by Algorithm 3

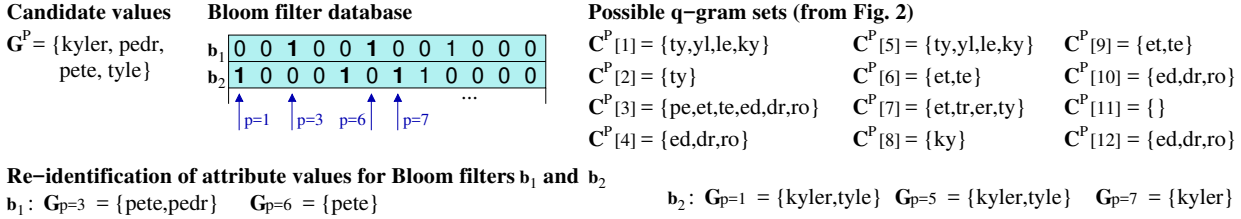

Fig. 5. Example re-identification using the sets of possible q-grams, $\mathbf{C}^P$, as generated in Figure 2. For a given set of candidate attribute values which we aim to re-identify (guess), $\mathbf{G}^P$, and a given BF $\mathbf{b}_i$, for each 1-bit position p in $\mathbf{b}_i$ we check if at least one q-gram in a candidate value $g_j \in \mathbf{G}^P$ occurs in the set $\mathbf{C}^P[p]$ at that position. If no q-gram of g_j occurs in $\mathbf{C}^P[p]$ then we can remove g_j as a candidate that could be encoded in BF $\mathbf{b}_i$ because at least one q-gram from g_j must be in $\mathbf{C}^P[p]$ to have generated the 1-bit at this position. For example, for BF $\mathbf{b}_1$ and the 1-bit at position $p = 3$ (shown in bold), the two candidate values 'kyler' and 'tyle' could not have been encoded into $\mathbf{b}_1$ because none of their q-grams (that could have generated the 1-bit at position $p = 3$) occurs in $\mathbf{C}^P[3]$. We therefore have the set of candidate values for BF $\mathbf{b}_1$ at position $p = 3$ as $\mathbf{G}_{p=3} = \{\text{pete}, \text{pedr}\}$. For the next 1-bit in $\mathbf{b}_1$ at $p = 6$ only 'pete' is left as a candidate ($\mathbf{G}_{p=6} = \{\text{pete}\}$) because 'pedr' does not contain any q-grams that occur in $\mathbf{C}^P[6]$ ('et' or 'te'). Similarly, for BF $\mathbf{b}_2$ and $p = 1$ we have the set of possible candidates $\mathbf{G}_{p=1} = \{\text{kyler}, \text{tyle}\}$ because 'pete' and 'pedr' contain no q-grams that occur in $\mathbf{C}^P[1]$. Only when we consider the 1-bit at position $p = 7$ can we also eliminate 'tyle' as a possible candidate for $\mathbf{b}_2$ (and so $\mathbf{G}_{p=7} = \{\text{kyler}\}$) because $\mathbf{C}^P[7]$ does not contain any of the q-grams that occur in 'tyle'.

with $\mathbf{C}^P_r$, $\mathbf{C}^N_r$ and $\mathbf{C}^A_r$, then for each bit position $1 \leq p \leq l$ the *merged* q-gram sets are generated as:

$$\begin{aligned} \text{Not possible: } \mathbf{C}^N_m[p] &= \mathbf{C}^N_b[p] \cup \mathbf{C}^N_r[p] \\ \text{Possible: } \mathbf{C}^P_m[p] &= (\mathbf{C}^P_b[p] \cup \mathbf{C}^P_r[p]) \setminus \mathbf{C}^N_m[p] \\ \text{Assigned: } \mathbf{C}^A_m[p] &= (\mathbf{C}^A_b[p] \cup \mathbf{C}^A_r[p]) \setminus \mathbf{C}^N_m[p] \end{aligned}$$

For each bit position, the merging first takes the union of the not possible q-gram sets, followed by the union of the possible q-gram sets minus the just merged not possible q-gram sets. This ensures any q-gram that was identified to be not possible at a certain position is not in the set of possible q-grams at that position. The sets of assigned q-grams are merged similarly, where no q-gram in an assigned set for a position can occur in the not possible set at that position.

Based on these merged sets, the final step of our attack is to re-identify attribute values, as we describe next.

4.4 Attribute Value Re-identification

Given sets of possible, $\mathbf{C}^P$, and not possible, $\mathbf{C}^N$, q-grams at BF bit positions (as generated by Algorithms 1 to 3), we can now try to re-identify attribute values from a set of plain-text values $\mathbf{V}$ and a set of BFs $\mathbf{B}$ (the sets used in Algorithms 1 and 3). Our aim is to re-identify attribute values with high precision, i.e. values where we are certain they must have been encoded into a given BF. Note that we currently do not exploit the assigned q-grams in the sets $\mathbf{C}^A$ because these are subsets of the possible sets ($\mathbf{C}^A \subset \mathbf{C}^P$).

In the following subsections we present two re-identification approaches. The first one is our recently presented cryptanalysis attack method [14] that only exploits the sets of possible q-grams, $\mathbf{C}^P$. Due to the low precision of the q-gram sets in $\mathbf{C}^P$, as shown in our experimental evaluation in Section 6, for many BFs more than one attribute value is re-identified resulting in low re-identification accuracy. In Section 4.4.2 we then propose a new approach that exploits the sets of not possible q-grams, $\mathbf{C}^N$, which are of high precision, leading to improved re-identification accuracy.

A requirement of the candidate attribute values in $\mathbf{V}$ that we can re-identify is that at least one of their q-grams must occur in either of the sets $\mathbf{C}^P$ or $\mathbf{C}^N$, respectively. Otherwise there would be no information about possible and not possible BF bit positions that could be used in the re-identification process. In our experiments we found that the majority of attribute values in a domain contain at least one q-gram that occurs in one of the sets in $\mathbf{C}^P$ or $\mathbf{C}^N$.

To generate the set of candidate attribute values which we aim to re-identify (i.e. guess), named $\mathbf{G}^P \subset \mathbf{V}$ (possible) and $\mathbf{G}^N \subset \mathbf{V}$ (not possible), we first identify all q-grams in the possible and not possible sets as $\mathbf{Q}^P = \{q_j \in \mathbf{C}^P[p] : 1 \leq p \leq l\}$ and $\mathbf{Q}^N = \{q_j \in \mathbf{C}^N[p] : 1 \leq p \leq l\}$. Based on those two sets of q-grams we can extract the possible, $\mathbf{G}^P$, and not possible, $\mathbf{G}^N$, candidate attribute values we aim to re-identify as $\mathbf{G}^P = \{v_i \in \mathbf{V} : \exists\, q_j \in genQGramSet(v_i, q) : q_j \in \mathbf{Q}^P\}$ and $\mathbf{G}^N = \{v_i \in \mathbf{V} : \exists\, q_j \in genQGramSet(v_i, q) : q_j \in \mathbf{Q}^N\}$.

4.4.1 Re-identification using Possible Q-gram Sets

The re-identification step based on the possible q-gram sets, $\mathbf{C}^P$, is detailed in Algorithm 4 and illustrated in Figure 5 [14]. The idea is that for a 1-bit to occur at a certain position in a BF, at least one q-gram of an attribute value must have been hashed to that position (see Theorem 1). Therefore, for a given candidate attribute value $g_j \in \mathbf{G}^P$ and BF $\mathbf{b}_i \in \mathbf{B}$, if the set of possible q-grams at position p, where $\mathbf{b}_i[p]$ is 1, does not include any q-gram of g_j then g_j could not have been encoded in $\mathbf{b}_i$.

The inputs to Algorithm 4 are the same set of BFs, $\mathbf{B}$, used in Algorithms 1 and 3, the set of candidate attribute values we aim to re-identify (guess), $\mathbf{G}^P$ (generated as described above), the list of possible q-gram sets at bit positions, $\mathbf{C}^P$ (as generated in Algorithms 1 or 3), and the BF length, l. The algorithm returns a set of re-identified attribute value(s), $\mathbf{G}_i \subset \mathbf{G}^P$, for each BF $\mathbf{b}_i \in \mathbf{B}$, collected in the result list $\mathbf{R}$.

After initializing $\mathbf{R}$, from line 2 onwards Algorithm 4 loops over all BFs $\mathbf{b}_i \in \mathbf{B}$, and for each $\mathbf{b}_i$ it initializes the set of possible candidate attribute values $\mathbf{G}_i$ (that potentially have been hashed into this BF) as all values in $\mathbf{G}^P$ in line 3. We then loop over all BF bit positions p in line 4, and for any position that has a 1-bit ($\mathbf{b}_i[p] = 1$) we retrieve the set of possible q-grams $\mathbf{q}_p = \mathbf{C}^P[p]$ at that position in line 6.

We now check for each value $g_j \in \mathbf{G}_i$ if *at least one of the q-grams* in $\mathbf{q}_p$ occurs in g_j (lines 7 and 8). If a g_j does not contain any q-gram from $\mathbf{q}_p$ then it could not have generated the 1-bit at position p in $\mathbf{b}_i$, and so in line 9 we

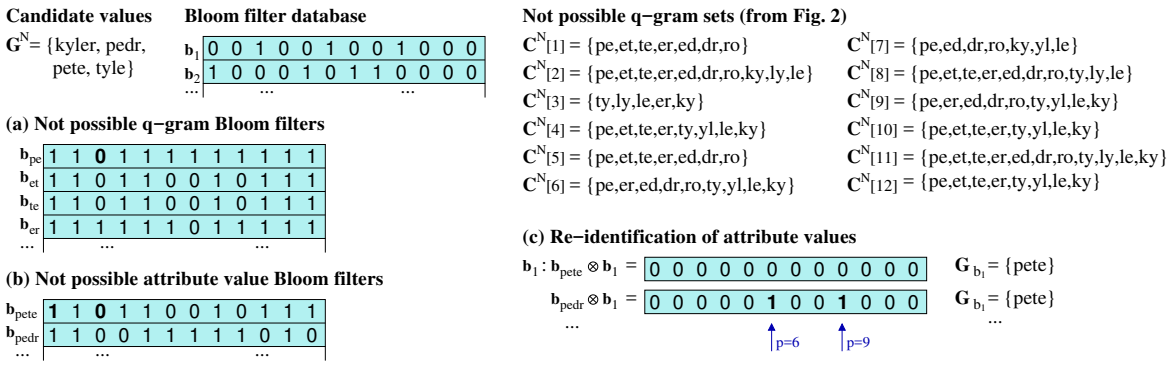

Fig. 6. Example re-identification using the sets of not possible q-grams, $\mathbf{C}^N$, as generated in Figure 2. (a) For each q-gram $q_j \in \mathbf{C}^N$ we generate a BF $\mathbf{b}_{q_j}$ that has 1-bits in all positions where q_j cannot be hashed to based on the sets in $\mathbf{C}^N$. For example, $p = 3$ (bold in BF $\mathbf{b}_{pe}$) is the only position where q-gram 'pe' possibly can be hashed to (and thus this bit is 0) based on the sets $\mathbf{C}^N[1]$ to $\mathbf{C}^N[12]$ (only $\mathbf{C}^N[9]$ does not contain 'pe'). (b) We then combine these q-gram BFs into attribute value BFs using bit-wise AND (conjunction, $\otimes$), leading to BFs that have 1-bits in all positions where no q-gram of a given attribute value could have been hashed to. For example, the BF $\mathbf{b}_{pete}$ has a 1-bit (in bold) in position $p = 1$ because all its q-grams occur in $\mathbf{C}^N[1]$, and a 0-bit at $p = 3$ (in bold) because 'et' and 'te' are possible at that position. (c) For a given set of candidate attribute values which we aim to re-identify (guess), $\mathbf{G}^N$, and a given BF $\mathbf{b}_i$, a value $g_j \in \mathbf{G}^N$ can only be encoded in $\mathbf{b}_i$ if the conjunction between $\mathbf{b}_i$ and the value's BF $\mathbf{b}_{g_j}$ results in only 0-bits. For example, the conjunction of BFs $\mathbf{b}_1$ and $\mathbf{b}_{pete}$ leads to only 0-bits, and thus $\mathbf{b}_1$ can encode 'pete'. On the other hand, $\mathbf{b}_1$ conjuncted with $\mathbf{b}_{pedr}$ results in two 1-bits (in bold) at positions $p = 6$ and $p = 9$ and therefore $\mathbf{b}_1$ cannot encode 'pedr'.

Algorithm 4: ***Re-identification based on possible q-gram sets***

Input:
- $\mathbf{B}$: Set of BFs from the sensitive database (same as in Algo. 1)
- $\mathbf{G}^P$: Set of candidate attribute values to be re-identified
- $\mathbf{C}^P$: List of possible q-gram sets at BF position (from Algo. 1 or 3)
- l: Length of BFs

Output:
- $\mathbf{R}$: List of possible attribute values re-identified for each BF in $\mathbf{B}$

1: $\mathbf{R} = []$
2: **for** $\mathbf{b}_i \in \mathbf{B}$ **do**:
3: $\quad \mathbf{G}_i = \mathbf{G}^P$ *// Initialize set of candidate attribute values*
4: $\quad$ **for** $1 \leq p \leq l$ **do**: *// Loop over BF bit positions*
5: $\quad\quad$ **if** $(\mathbf{b}_i[p] == 1)$ **then**:
6: $\quad\quad\quad \mathbf{q}_p = \mathbf{C}^P[p]$ *// Set of possible q-grams at this position*
7: $\quad\quad\quad$ **for** $g_j \in \mathbf{G}_i$ **do**: *// Check all candidate attribute values*
8: $\quad\quad\quad\quad$ **if** $(\forall q \in \mathbf{q}_p : q \notin g_j)$ **then**: *// At least one q-gram must occur*
9: $\quad\quad\quad\quad\quad \mathbf{G}_i = \mathbf{G}_i \setminus \{g_j\}$ *// Value cannot have been hashed into* $\mathbf{b}_i$
10: $\quad \mathbf{R}.append(\mathbf{G}_i)$ *// Append possible attribute values for BF* $\mathbf{b}_i$
11: **return** $\mathbf{R}$

Algorithm 5: ***Re-identification based on not possible q-gram sets***

Input:
- $\mathbf{B}$: Set of BFs from the sensitive database (same as in Algo. 1)
- $\mathbf{G}^N$: Set of candidate attribute values to be re-identified
- $\mathbf{C}^N$: List of not possible q-gram sets at BF pos. (from Algo. 1 or 3)
- l: Length of BFs
- q: Length of substrings to extract from attribute values

Output:
- $\mathbf{R}$: List of possible attribute values re-identified for each BF in $\mathbf{B}$

1: $\mathbf{R} = [], \mathbf{B}^Q = \{\}, \mathbf{B}^G = \{\}$
2: $\mathbf{b}_0 = [0]^l$ *// A BF with only 0-bits*
3: $\mathbf{Q}^N = \{q_j \in \mathbf{C}^N[p] : 1 \leq p \leq l\}$ *// All q-grams that occur in* $\mathbf{C}^N$
4: $\mathbf{B}^Q[q_i] = \mathbf{b}_0, \forall q_i \in \mathbf{Q}^N$ *// A 0-bit BF for each q-gram in* $\mathbf{Q}^N$
5: **for** $1 \leq p \leq l$ **do**: *// Loop over BF bit positions*
6: $\quad$ **for** $q_i \in \mathbf{C}^N[p]$ **do**:
7: $\quad\quad \mathbf{B}^Q[q_i][p] = 1$ *// Set positions to 1 where q-gram is not possible*
8: **for** $g_j \in \mathbf{G}^N$ **do**:
9: $\quad \mathbf{q}_j = genQGramSet(g_j, q)$ *// Convert attr value into q-grams*
10: $\quad$ **if** $(\forall q_k \in \mathbf{q}_j : q_k \in \mathbf{Q}^N)$ **then**: *// All q-grams known*
11: $\quad\quad \mathbf{B}^G[g_j] = \otimes\, \mathbf{b}_j \; \forall\, \mathbf{B}^Q[q_k] : q_k \in \mathbf{q}_j$ *// Attribute value BF*
12: **for** $\mathbf{b}_i \in \mathbf{B}$ **do**:
13: $\quad \mathbf{G}_i = \{\}$ *// Initialize set of candidate attribute values*
14: $\quad$ **for** $g_j \in \mathbf{G}^N$ **do**:
15: $\quad\quad$ **if** $(\mathbf{B}^G[g_j] \otimes \mathbf{b}_i == \mathbf{b}_0)$ **then**: *// All bits are 0*
16: $\quad\quad\quad \mathbf{G}_i = \mathbf{G}_i \cup \{g_j\}$ *// A re-identified attribute value*
17: $\quad \mathbf{R}.append(\mathbf{G}_i)$ *// Append possible attribute values for BF* $\mathbf{b}_i$
18: **return** $\mathbf{R}$

remove g_j from the set of candidates $\mathbf{G}_i$ for this BF. At the end of this loop (line 10), the set $\mathbf{G}_i$ will contain only those values from $\mathbf{G}^P$ that possibly have generated the BF $\mathbf{b}_i$.

In this re-identification approach those positions with 0-bits do not help us to remove values $g_j \in \mathbf{G}_i$ because any q-gram in a g_j has likely been hashed to other positions. We next present a more accurate re-identification approach that exploits the not possible q-gram sets, $\mathbf{C}^N$.

4.4.2 Re-identification using Not Possible Q-gram Sets

As illustrated in Figure 6, for each q-gram there are bit positions where this q-gram cannot have been hashed into based on the sets in $\mathbf{C}^N$. We combine these not possible positions for an attribute value $g_j \in \mathbf{G}^N$ by taking the intersection of the sets of positions for all q-grams in g_j. This results in a BF $\mathbf{b}_{g_j}$ which has 1-bits in all positions where a BF that potentially can encode g_j needs to have a 0-bit. A conjunction of $\mathbf{b}_{g_j}$ with any $\mathbf{b}_i \in \mathbf{B}$ must therefore result in a BF that only has 0-bits for $\mathbf{b}_i$ to be able to encode g_j.

Algorithm 5 details the steps of this approach, where the inputs are similar to Algorithm 4, except the not possible q-gram sets, $\mathbf{C}^N$, and a corresponding set of candidate attribute values, $\mathbf{G}^N$, are provided. As the algorithm requires q-grams to be generated, the length q of these q-grams is also needed. The algorithm returns in the list $\mathbf{R}$ for each BF $\mathbf{b}_i \in \mathbf{B}$ a set of re-identified attribute values, $\mathbf{G}_i \subset \mathbf{G}^N$.

In lines 1 and 2, Algorithm 5 initializes the result list, $\mathbf{R}$, the sets of BFs for q-grams, $\mathbf{B}^Q$, and candidate values, $\mathbf{B}^G$, as well as a BF $\mathbf{b}_0$ of length l that only contains 0-bits. Next, in line 3, all unique q-grams in $\mathbf{C}^N$ are collected into the set $\mathbf{Q}^N$, and for each q-gram $q_i \in \mathbf{Q}^N$ one BF that only contains 0-bits is created in line 4, collected into the set $\mathbf{B}^Q$.

In the first phase of the algorithm (lines 5 to 7), for each bit position p and for each q-gram $q_i \in \mathbf{C}^N[p]$ at that position, we set the bit at position p in the q-gram's corresponding BF $\mathbf{B}^Q[q_i]$ to 1, leading to one BF per q-gram with 1-bits in all positions where this q-gram cannot be hashed to.

In the second phase, in lines 8 to 11, for each candidate attribute value $g_j \in \mathbf{G}^N$ that has all its q-grams in the set $\mathbf{Q}^N$ (of q-grams where we know their not possible positions, the test in line 10), we generate a BF for value g_j in line 11 as the conjunction (bit-wise AND, $\otimes$) of the BFs in $\mathbf{B}^Q$ for all of the value's q-grams, $q_k \in \mathbf{q}_j$, as generated in line 9. We collect all candidate attribute value BFs in the set $\mathbf{B}^G$.

In lines 12 to 17, we loop over all BFs $\mathbf{b}_i \in \mathbf{B}$ and for each in line 13 we initialize an empty set of candidate values. In line 14 we then loop over all candidate attribute values $g_j \in \mathbf{G}^N$, and if the conjunction between the BF $\mathbf{b}_i$ and the candidate's BF $\mathbf{B}^G[g_j]$ results in a BF that contains only 0-bits then g_j could have been encoded into BF $\mathbf{b}_i$ and therefore in line 16 we add g_j to the set $\mathbf{G}_i$. We then add the candidate set for $\mathbf{b}_i$ to the result list $\mathbf{R}$ in line 17.

5 Complexity Analysis and Limitations

We now discuss the complexity and limitations of our efficient cryptanalysis attack on BFs.

Complexity: To simplify our analysis, we assume $n = |\mathbf{B}| = |\mathbf{V}|$ is the number of BFs and attribute values, with n_Q the average number of q-grams per $v \in \mathbf{V}$, k the number of hash functions used, and l the BF length.

In Algorithm 1, the initialization in lines 2 and 3 is of $O(n)$, sorting in line 4 is $O(n \log n)$, and generating $\mathbf{A}$ is $O(n_A)$ assuming $n_A = |\mathbf{A}| = min(|\mathbf{B}_F|, |\mathbf{V}_F|)$. The complexity of lines 7 to 12 is $O(n_A\, n_Q\, l)$ as a v_i in each $(v_i, \mathbf{b}_i) \in \mathbf{A}$ is converted into a q-gram set and all l bit positions are checked for the corresponding $\mathbf{b}_i$. In lines 14 to 16, assuming $\mathbf{Q}$ is the set of all unique q-grams generated from all values $v_i \in \mathbf{V}$, then on average $t = k/l\,|\mathbf{Q}|$ q-grams are hashed into one bit position, leading to a complexity of $O(t)$ for one set difference and $O(t\,l)$ for all l bit positions.

Algorithm 2 loops over all $(\mathbf{b}_i, v_i) \in \mathbf{A}$, converts each v_i into its q-gram set and then iterates over all l BF bit positions where a set intersection is conducted for each 1-bit. Assuming that each $\mathbf{C}^P[p]$ contains t q-grams then the complexity of Algorithm 2 is $O(n_A(n_Q + l\,t))$.

The initialization in lines 2 and 3 in Algorithm 3 is $O(n_A + n)$ as it requires loops over $\mathbf{A}$, and $\mathbf{V}$ and $\mathbf{B}$, respectively. The first loop in lines 4 to 9 has a complexity of $O(n_A\, n(n_Q + l))$ because for each $(v_i, \mathbf{b}_i) \in \mathbf{A}$ q-gram sets are generated and compared with q-gram sets from all $v_j \in \mathbf{V}_N$ and bit-wise AND operations are performed between BFs. The refinement and expansion phases in lines 18 to 42 have a complexity of $O(n_A(m\, n_Q + l))$ as they require the union and intersection of small numbers (m) of q-gram sets as well as bit-wise AND and OR of BFs of length l for each tuple $(v_i, \mathbf{b}_i) \in \mathbf{A}$.

Algorithm 4 is of $O(n\,l\,t|\mathbf{G}^P|)$ as it loops over all BFs of length l in $\mathbf{B}$, and for each checks if any of the candidate values in $\mathbf{G}^P$ can be encoded into a certain BF, where we assume t is the average number of q-grams in a $\mathbf{C}^P[p]$.

Finally, in Algorithm 5, the initialization in lines 3 and 4, and the loop in lines 5 to 7, are of $O(l\,u)$, where $u = |\mathbf{Q}|\,(1 - k/l)$ is the average number of q-grams that are not hashed to a BF bit position, as they iterate over all l bit positions and sets of not possible q-grams in $\mathbf{Q}^N$. The second loop (lines 8 to 11) iterates over all candidate values $g_j \in \mathbf{G}^N$ and for each generates a BF (line 11). As we assume an attribute value g_j in average contains n_Q q-grams then this loop is of $O(|\mathbf{G}^N|\, n_Q)$. The loop in lines 12 to 17 is of $O(n\,|\mathbf{G}^N|)$ as it iterates over all BFs in $\mathbf{B}$ and for each checks if any value $g_j \in \mathbf{G}^N$ could possibly have been encoded in the BF.

TABLE 2
The number of unique values in the attributes used in the experimental evaluation. For NCVR and UKCD we used two files, where one was the sensitive database encoded into BFs and the other the plain-text database available to the attacker, while in a cross-state experiment NCVR was the encoded and Michigan the plain-text database.

Data set	First name	Surname	City / Address
NCVR	22,117 / 22,282	40,113 / 40,513	735 / 745
UKCD	624 / 813	994 / 1,638	414 / 529
Michigan	195,431	362,807	–

Limitations: Our attack method on BFs first assumes an attacker has access to a population database from where the set $\mathbf{V}$ of plain-text attribute values and their frequencies can be extracted. Second, we assume that the BF database $\mathbf{B}$ contains a subset of BFs that occur multiple times, and that the frequency distribution of BFs in $\mathbf{B}$ is similar to the frequency distribution of a single or a subset of attribute values in $\mathbf{V}$. These two assumptions are common to earlier published attacks on BFs [14], [15], [16], [17], [18].

6 Experimental Evaluation

We evaluated our attack method with several data sets, as we describe next together with the experimental setup used. We then discuss the results obtained, and finally provide recommendations on how to securely employ BFs for PPRL.

6.1 Data Sets and Experimental Setup

We conducted experiments using three data sets from two countries, as summarized in Table 2. The first is a pair of North Carolina Voter Registration (named 'NCVR') data sets (from: ftp://alt.ncsbe.gov/data/) collected in June 2014 (the database to be attacked) and October 2016 (the plain-text database). We extracted two subsets containing 224,073 (2014) and 224,061 (2016) records, respectively. The second pair of data sets (named 'UKCD') are census records collected from the years 1851 (the sensitive database) and 1861 (the plain-text database) for the town of Rawtenstall in England [26]. These data sets contain 17,034 (1851) and 22,430 (1861) records. For both data sets we use the *First name*, *Surname*, and *City* (NCVR) or *Address* (UKCD) attributes, respectively, as well as the concatenation of pairs of these attributes. A third data set was a voter database from Michigan (from: http://michiganvoters.info) containing 7,408,330 records that we used as the plain-text database available to an attacker in a cross-state attack experiment.

Following earlier attacks on BFs [14], [15], [16], we set encoding parameters as q-gram length $q = [2, 3, 4]$, BF length $l = [500, 1000, 2000]$, and either the double or random hashing methods [23] as described in Section 3. We set the

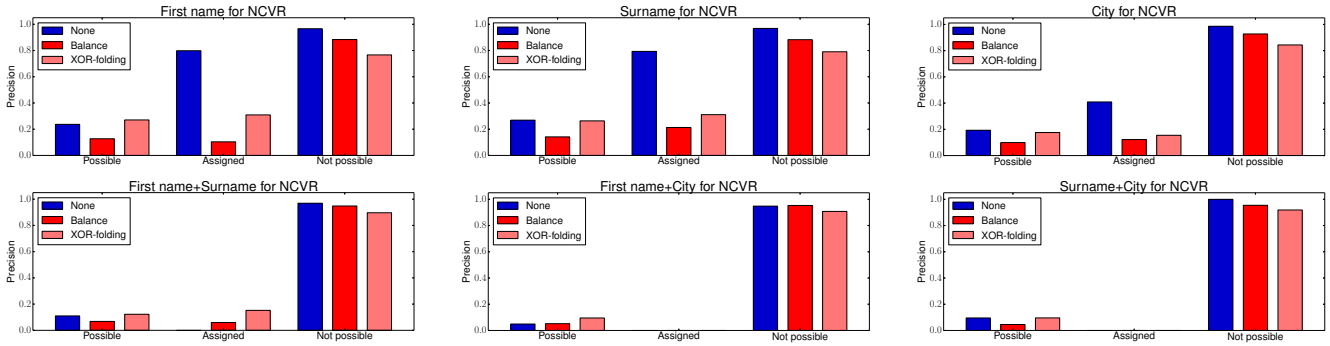

Fig. 7. Precision results of re-identified q-gram sets for the NCVR data sets for different attribute combinations, and no BF hardening as well as the 'Balancing' and 'XOR-folding' hardening techniques described in Section 3 applied. An important result is that the precision of not possible q-gram sets is high even when BF hardening techniques have been applied. Missing bars for the 'assigned' q-gram sets means there were no q-grams that could be assigned to BF positions. Similar results were obtained with the UKCD data sets (not shown due to limited space).

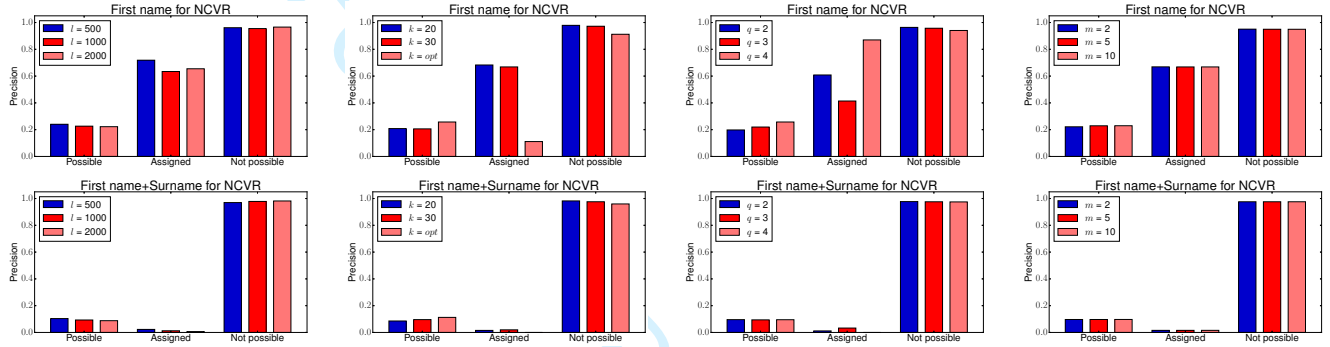

Fig. 8. Precision results of re-identified q-gram sets for the NCVR data sets for different settings of BF length l (left), number of hash functions k (middle left), length of q-grams q (middle right), and maximum size of attribute value and BF sets m as used in the refinement and expansion step in Algorithm 3 (right). The results for a given parameter were obtained by averaging over the settings of the other parameters. No hardening technique was applied in these experiments. As can be seen our proposed attack method is robust with regard to most used parameter settings.

number of hash functions $k = [20, 30]$ and also calculated an optimal number (opt) for k based on l and q such that the false positive rate is minimized [22]. We also applied the BF hardening techniques balancing and XOR-folding [23] as described in Section 3. We set the maximum size of attribute value and BF sets in Algorithm 3 as $m = [2, 5, 10]$ as these values provided good results in a set of initial experiments, and the number of candidate attribute values to re-identify (as discussed in Section 4.4) to $|\mathbf{G}^P| = |\mathbf{G}^N| = [100, 1000]$.

We implemented our attack method in Python 2.7 and conducted experiments on a server with 64-bit 2.4 GHz CPUs, 128 GBytes of memory and running Ubuntu 14.04. The programs are available from the authors.

We present the precision of the possible ($\mathbf{C}^P$), assigned ($\mathbf{C}^A$), and not possible ($\mathbf{C}^N$) q-gram sets averaged over all BF bit positions, where for each position we calculate precision as the ratio of q-grams in a set in $\mathbf{C}^P$, $\mathbf{C}^A$ or $\mathbf{C}^N$ that were truly hashed (for $\mathbf{C}^P$ and $\mathbf{C}^A$) or not hashed (for $\mathbf{C}^N$) to a position. We do not report recall since each q-gram is hashed k times, and even if not all positions of a q-gram are found (low recall), those identified can provide valuable information to allow re-identification of attribute values.

We evaluate the quality of the re-identification methods from Section 4.4 as *re-identification accuracy* by counting the number of BFs in the encoded database for which we were able to (1) correctly re-identify their attribute value (labeled 'Correct 1'), (2) correctly re-identify their attribute value but at least one other (incorrect) value was also re-identified (labeled 'Correct many'), or (3) one or several wrong attribute value(s) were re-identified (labeled 'Wrong').

6.2 Results and Discussion

In Figure 7 we show the precision of the q-gram sets $\mathbf{C}^P$, $\mathbf{C}^A$, and $\mathbf{C}^N$. As can be seen, the sets of not possible q-grams have the highest precision. Even when BF hardening techniques are applied our attack can correctly identify the positions where q-grams could not have been hashed to. This provides high quality information about q-gram assignments to bit positions that can be used in the re-identification process. The sets of possible q-grams are of lower precision because for many BF bit positions they also contain q-grams that in fact were not hashed to those positions. Using the possible sets therefore leads to more multiple re-identified attribute values, as Figure 9 shows.

The sets of assigned q-grams are of reasonable precision for certain attributes (but not all) and only when no hardening technique has been applied. One reason that some of these assigned sets have low precision is that any error in the frequency alignment of BFs and attribute values in the list $\mathbf{A}$, as described in Section 4.1, leads to wrong assignments. From Figure 7 it is clear that the best way to re-identify attribute values is to use the not possible q-gram sets, $\mathbf{C}^N$.

Figure 8 shows how different parameters affect the precision of the q-gram sets in $\mathbf{C}^P$, $\mathbf{C}^A$, and $\mathbf{C}^N$. Note that

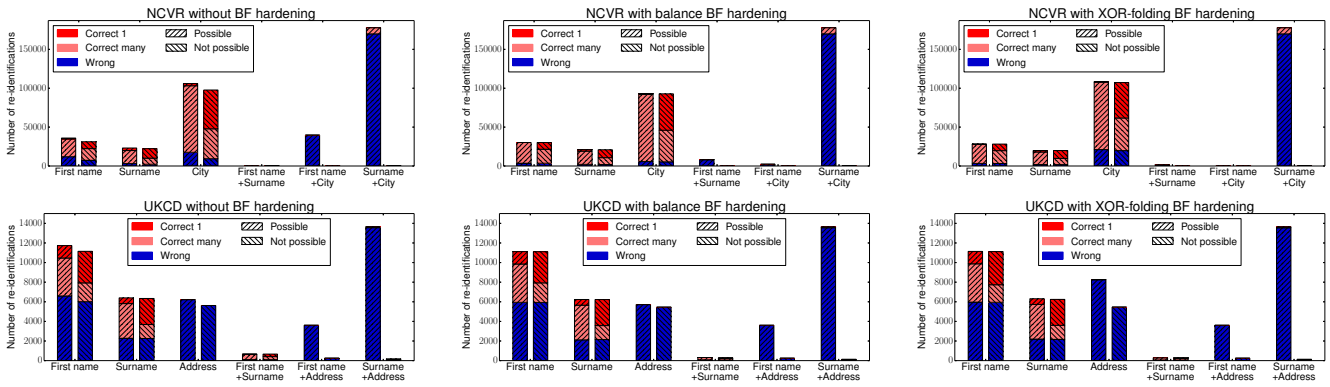

Fig. 9. Re-identification accuracy results for the NCVR (top) and UKCD (bottom) data sets based on the possible and not possible q-gram sets, where no BF hardening (left), balancing (middle) or XOR-folding (right) was applied. The bars show the number of BFs for which attribute values were re-identified either correctly (with only 1 or with several re-identified values), or where one or more wrong value(s) were re-identified.

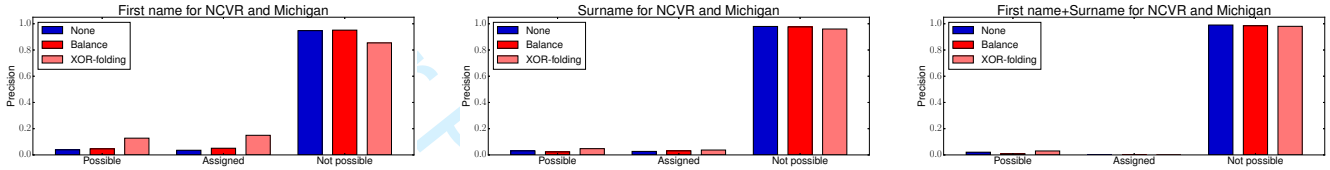

Fig. 10. Precision results of re-identified q-gram sets where the encoded database is NCVR and the plain-text database is from Michigan. As discussed in Section 6.2, both for first names and surnames the top most frequent values differ across these two data sets. As a result the possible and assigned q-gram sets are of low precision, however the not possible q-gram sets still provide information about the values encoded in BFs.

only the parameter m (maximum size of attribute value and BF sets as used in Algorithm 3) is controllable by the attacker. As can be seen, our attack is robust with regard to most used parameter settings, except for the precision of the assigned q-gram sets, $\mathbf{C}^A$, which drops for an optimal number of hash functions, $k = opt$ and q-grams of length $q = 3$. This indicates that the approach to identify assigned q-grams is data and parameter specific and might not be useful to accurately re-identify encoded values.

The values for m were selected based on the objective of our attack to identify q-grams with high precision. With an increase in m the refinement and expansion steps in Algorithm 3 will lead to lower quality of the identified q-grams and thus lower final re-identification quality. Overall, as can be seen from Figure 8, the sets of not possible q-grams, $\mathbf{C}^N$, are of much higher precision compared to the possible and assigned q-grams, $\mathbf{C}^P$ and $\mathbf{C}^A$, supporting that $\mathbf{C}^N$ should be used to re-identify attribute values.

As can be seen in Figure 9, for the larger NCVR data sets the re-identification quality is higher than for the smaller UKCD data sets. This is because a larger number of frequent BFs and attribute values are available that can be correctly aligned in the list $\mathbf{A}$. Our attack will be more accurate on larger BF databases as they can be aligned to large population databases such as telephone directories.

An example where our attack fails (as would others) is on addresses for the UKCD data set, where the alignment in the list $\mathbf{A}$ is incorrect even for the most frequent address (changing from 'Newchurch' in 1851 to 'Crawshawbooth' in 1861). This highlights, as observed earlier [17], that an attacker needs to have access to a plain-text database that has a similar frequency distribution as the BF database.

The new re-identification method based on the not possible q-gram sets is able to correctly re-identify single attribute values for more BFs compared to our previous re-identification method based on the possible q-grams sets [14]. As Table 3 shows, our new method is also more precise as the average number of 'Correct many' re-identifications is much smaller compared to our previous method. Furthermore, the new re-identification method is also faster and even on the larger NCVR data sets the attack only requires a few seconds, as shown in Table 6.

With regard to the refinement and expansion of q-gram sets (as proposed in Section 4.3), Table 4 shows that for the re-identification method based on possible q-grams the proposed methods generally increase the number of correct single re-identifications while at the same time they reduce the number of multiple re-identifications. For the re-identification method based on not possible q-grams the results are more mixed, with some increases in the number of single re-identifications and large decreases in the number of multiple re-identifications. At least for the first name attribute there is also an increase in the number of wrong re-identifications. We plan to further investigate the performance of the refinement and expansion methods.

To evaluate our attack in a scenario where an attacker does not have access to a plain-text data set that has similar frequency characteristics to the encoded BF database (a more likely scenario in a practical attack given the availability of large population databases in various domains [17]), in Figure 10 we show the precision of q-gram sets using NCVR as the encoded BF database and a voter data set from Michigan as the plain-text database. These two databases are very different with regard to their sizes, and the most frequent first name and surname values are ranked differently. For example, the three most frequent surnames in NCVR are:

TABLE 3
Average number of multiple re-identified attribute values using either the re-identification approach based on possible or not possible q-gram sets for the NCVR data sets with the number of candidate attribute values, $\mathbf{G}^P$ or $\mathbf{G}^N$, set to 100 or $1,000$, respectively. Results are averaged (with standard deviations shown) over various parameter settings. The method based on not possible q-gram sets leads to much smaller numbers of re-identified values, making the attack more accurate. Similar results were obtained with the UKCD data sets (not shown due to limited space).

Re-identify method / Size G	First name	Surname	City	First name+Surname	First name+City	Surname+City
Possible, $\|\mathbf{G}^P\| = 100$	10.38 ± 0.9	5.97 ± 2.1	8.19 ± 3.2	18.44 ± 1.5	10.98 ± 3.1	26.92 ± 6.8
Not Possible, $\|\mathbf{G}^N\| = 100$	2.13 ± 0.2	2.19 ± 0.1	1.45 ± 1.0	3.37 ± 0.6	4.67 ± 1.9	2.00 ± 0.0
Possible, $\|\mathbf{G}^P\| = 1{,}000$	93.84 ± 11.4	43.50 ± 20.6	56.83 ± 20.0	168.12 ± 24.9	122.18 ± 75.1	239.94 ± 117.4
Not Possible, $\|\mathbf{G}^N\| = 1{,}000$	5.94 ± 0.9	3.61 ± 0.5	3.66 ± 0.4	3.83 ± 0.6	5.33 ± 2.4	2.00 ± 0.0

TABLE 4
Average improvements in the number of re-identified attribute values for the NCVR data sets using the refinement and expansion methods. Smaller improvements were obtained for the UKCD data sets because these are much smaller and contain less values that could match the refinement or expansion requirements described in Section 4.3.

Attribute	Possible	Not possible
	Corr 1 / Corr M / Wrong	Corr 1 / Corr M / Wrong
First name	+94.6% / -21.6% / +15.6%	-1.6% / -23.6% / +10.0%
Surname	+55.9% / -7.1% / -1.2%	+13.3% / -16.2% / -0.2%
City	+56.0% / -2.5% / -0.2%	+3.6% / -6.9% / -3.7%

TABLE 5
Comparison of re-identification results with existing BF cryptanalysis attack methods with regard to re-identification quality and run time.

Publication	Data set and attr.	Num BFs	Corr 1	Time
Kuzu'11 [16]	NCVR first names	3,500	400	1,000 sec
Kuzu'13 [17]	Patient names	20	4	few sec
"	"	42	0	> week
Niedermeyer [18]	German surnames	7,580	934	> days
Kroll [15]	German surnames and town names	100K	44K	> days
Christen [14]	NCVR first names	100	10	0.75 sec
Our approach	NCVR diff. attr.	>200K	>49K	< 10 min

Smith (1.40%), Williams (1.05%), and Jones (0.94%), while in Michigan they are Smith (0.98%), Johnson (0.75%), and Williams (0.61%). These different rankings lead to wrong alignments of plain-text values and BFs in the list $\mathbf{A}$ in Algorithm 1 that results in mostly wrong identification of possible and assigned q-grams (as Figure 10 shows) and thus low re-identification accuracy of attribute values. However, the not possible q-gram sets identified are still highly precise because of the commonality of q-grams in these frequent attribute values. As a result, some information about q-grams encoded in the BF database can be learned.

In Table 5 we compare our attack method with previously proposed cryptanalysis attacks on BFs for PPRL in terms of accuracy and efficiency. As can be seen, our new approach is both more efficient and effective in re-identifying attribute values from large databases, achieving higher precision and much reduced computational requirements than existing attacks on BFs for PPRL.

6.3 Recommended Use of Bloom Filters for PPRL

Given BF encoding is now being employed in real-world PPRL applications [2], [9], [10], it is crucial to study possible attacks on BFs to ensure their security.

Our experiments have shown the vulnerability of basic BF encoding, as well as BFs hardened using balancing or XOR-folding, to our attack method. Our work highlights the need for the development of improved PPRL encoding and hardening techniques to overcome such attacks. Our attack provides database owners with an efficient method to evaluate the privacy of their BF encoded databases before using them for PPRL, and to identify weaknesses of BF encoding that could be exploited by an attacker.

Based on our (as well as previous) attack methods, we provide the following recommendations to mitigate the success of such attacks. (1) Existing attacks exploit frequent BFs that can be aligned with frequent plain-text values. Therefore, database owners should ensure each BF in their encoded database is unique. This can be achieved by using record-level BF encoding [11], [19], where several attribute values are encoded into one BF. (2) Employ different hash mechanisms for each attribute to further reduce frequency information (so a q-gram will be encoded differently depending upon if it, for example, occurs in a first name or a surname). (3) Apply advanced BF hardening methods [23], to distort any useful patterns that could be exploited even further. Combined, these three recommendations will mean that there are unlikely frequent BFs or even frequent bit patterns that can be exploited by an attack method.

Although database owners need to be aware that there is no guarantee that any encoding is secure against any future attack methods that are based on sophisticated cryptanalysis techniques [27], the proposed mitigation methods will make attacks even more harder. However, further extensive experiments are required to properly analyze the strengths and weaknesses of different BF encoding and hardening techniques and their potential vulnerabilities with regard to different attack methods.

7 Conclusions and Future Work

We have presented a novel efficient attack method on BFs that contain encoded sensitive attribute values intended for PPRL. Unlike earlier attacks on BFs for PPRL, our approach only requires an attacker to have access to a public database of attribute values and their frequencies, but no information about the BF encoding used. Our attack can be successful even when some BF hardening techniques have been applied. It is also much faster than earlier attacks, making it feasible for database owners to efficiently validate the security of their large encoded sensitive databases before they are being sent to other parties for conducting PPRL.

TABLE 6
Timing results in seconds for the complete attack using either the re-identification approach based on possible or not possible q-gram sets for the NCVR data sets for different attribute combinations and the number of candidate attribute values, $\mathbf{G}^P$ or $\mathbf{G}^N$, set to 100 or $1,000$, respectively. Results are averaged (with standard deviations shown) over various parameter settings as described in Section 6. The re-identification approach based on not possible q-gram sets is clearly much faster than our earlier approach which is based on possible q-gram sets [14]).

Re-identify method / Size G	First name	Surname	City	First name+Surname	First name+City	Surname+City
Possible, $\lvert\mathbf{G}^P\rvert = 100$	7.66 ± 2.7	11.22 ± 4.1	2.27 ± 1.1	30.29 ± 9.2	17.08 ± 5.0	156.31 ± 136.2
Not Possible, $\lvert\mathbf{G}^N\rvert = 100$	2.50 ± 1.1	3.97 ± 1.7	1.95 ± 1.0	7.09 ± 2.3	6.49 ± 2.4	5.71 ± 2.1
Possible, $\lvert\mathbf{G}^P\rvert = 1{,}000$	50.94 ± 17.9	78.92 ± 38.4	4.77 ± 1.9	293.09 ± 116.4	136.16 ± 49.4	$1{,}374.93 \pm 1247.6$
Not Possible, $\lvert\mathbf{G}^N\rvert = 1{,}000$	2.79 ± 1.1	5.07 ± 2.6	1.98 ± 1.0	9.57 ± 3.5	7.71 ± 3.1	6.38 ± 2.1

We believe our attack is an important component to make BF based PPRL more secure for practical applications.

As future work we aim to improve the refinement and expansion steps by using all possible q-gram subsets, including the assigned q-gram sets, $\mathbf{C}^A$. We also plan to investigate how probabilistic language models [28] can be employed to find the most likely candidate attribute value when several candidates have been re-identified for a certain BF. Finally, we plan to explore the risk of re-identification when other hardening techniques, such as BLoom-and-flIP (BLIP) or salting [18], have been applied.

Acknowledgements

The authors would like to thank the Isaac Newton Institute, Cambridge, for support and hospitality during the programme *Data Linkage and Anonymisation* (EPSRC grant EP/K032208/1). P. Christen was partially supported by the Simons Foundation. This work was also partially funded by the Australian Research Council under grant DP130101801.

References

[1] P. Christen, *Data Matching*. Springer, 2012.
[2] J. Boyd, S. Randall, and A. Ferrante, "Application of privacy-preserving techniques in operational record linkage centres," in *Medical Data Privacy Handbook*, 2015.
[3] R. Schnell, "Privacy-preserving record linkage," in *Methodological Developments in Data Linkage*, K. Harron *et al.*, Eds., 2015.
[4] R. Hall and S. Fienberg, "Privacy-preserving record linkage," in *PSD*, Corfu, Greece, 2010, pp. 269–283.
[5] D. Vatsalan, P. Christen, and V. Verykios, "A taxonomy of privacy-preserving record linkage techniques," *IS*, vol. 38, no. 6, 2013.
[6] D. Vatsalan, Z. Sehili, P. Christen, and E. Rahm, "Privacy-preserving record linkage for Big Data," in *Handbook of Big Data Technologies*. Springer, 2017, pp. 851–895.
[7] J. Cao, F.-Y. Rao, E. Bertino, and M. Kantarcioglu, "A hybrid private record linkage scheme: separating differentially private synopses from matching records," in *IEEE ICDE*, 2015.
[8] Y. Lindell and B. Pinkas, "Secure multiparty computation for privacy-preserving data mining," *JPC*, vol. 1, no. 1, p. 5, 2009.
[9] C. Pow, K. Iron, J. Boyd, A. Brown *et al.*, "Privacy-preserving record linkage: an international collaboration between Canada, Australia and Wales," *IJPDS*, vol. 1, no. 1, 2017.
[10] S. Randall, A. Ferrante, J. Boyd, J. Bauer, and J. Semmens, "Privacy-preserving record linkage on large real world datasets," *JBI*, vol. 50, pp. 205–212, 2014.
[11] R. Schnell, T. Bachteler, and J. Reiher, "Privacy-preserving record linkage using Bloom filters," *BMC MIDM*, vol. 9, no. 1, 2009.
[12] D. Vatsalan and P. Christen, "Privacy-preserving matching of similar patients," *JBI*, vol. 59, pp. 285–298, 2016.
[13] D. Karapiperis, A. Gkoulalas-Divanis, and V. S. Verykios, "FEDERAL: A framework for distance-aware privacy-preserving record linkage," *IEEE TKDE*, vol. 30, no. 2, pp. 292–304, 2017.
[14] P. Christen, R. Schnell, D. Vatsalan, and T. Ranbaduge, "Efficient cryptanalysis of Bloom filters for privacy-preserving record linkage," in *PAKDD*, Jeju, Korea, 2017, pp. 628–640.
[15] M. Kroll and S. Steinmetzer, "Who is 1011011111...1110110010? Automated cryptanalysis of Bloom filter encryptions of databases with several personal identifiers," in *BIOSTEC*, Lisbon, 2015.
[16] M. Kuzu, M. Kantarcioglu, E. Durham, and B. Malin, "A constraint satisfaction cryptanalysis of Bloom filters in private record linkage," in *PET*, Waterloo, 2011.
[17] M. Kuzu, M. Kantarcioglu, E. Durham, C. Toth, and B. Malin, "A practical approach to achieve private medical record linkage in light of public resources," *JAMIA*, 2013.
[18] F. Niedermeyer, S. Steinmetzer, M. Kroll, and R. Schnell, "Cryptanalysis of basic Bloom filters used for privacy preserving record linkage," *JPC*, vol. 6, no. 2, pp. 59–79, 2014.
[19] E. Durham, M. Kantarcioglu, Y. Xue, C. Toth, M. Kuzu, and B. Malin, "Composite Bloom filters for secure record linkage," *IEEE TKDE*, vol. 26, no. 12, pp. 2956–2968, 2014.
[20] A. Kirsch and M. Mitzenmacher, "Less hashing, same performance: building a better Bloom filter," in *ESA*, 2006, pp. 456–467.
[21] B. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Comm ACM*, vol. 13, no. 7, 1970.
[22] D. Vatsalan and P. Christen, "Scalable privacy-preserving record linkage for multiple databases," in *ACM CIKM*, Shanghai, 2014.
[23] R. Schnell and C. Borgs, "Randomized response and balanced Bloom filters for privacy preserving record linkage," in *ICDMW DINA*, Barcelona, 2016.
[24] ——, "XOR-folding for Bloom filter-based encryptions for privacy-preserving record linkage," *German Record Linkage Center*, no. WP-GRLC-2016-03, 2016.
[25] T. Mitchell, *Machine Learning*. McGraw Hill, 1997.
[26] Z. Fu, P. Christen, and J. Zhou, "A graph matching method for historical census household linkage," in *PAKDD*, Tainan, 2014.
[27] K. M. Martin, *Everyday Cryptography: Fundamental Principles and Applications*. New York: Oxford University Press, 2012.
[28] C. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

Peter Christen is a Professor at the Australian National University (ANU) Research School of Computer Science (RSCS). He graduated with a PhD in Computer Science from the University of Basel, Switzerland, in 1999. His research interests are in record linkage and data mining, with a focus on privacy and machine learning aspects of record linkage. He has published over 130 articles in these areas, including in 2012 the monograph "Data Matching" published by Springer.

Thilina Ranbaduge is a PhD post-doctoral fellow at the ANU. His research interests are in data mining, and in multi-database and privacy-preserving record linkage. He received his PhD in Computer Science from the ANU in 2018 and completed his PG.Dip and BSc (Hons) at the University of Moratuwa, Sri Lanka, in 2013 and 2009, respectively.

Dinusha Vatsalan is a Research Scientist at Data61-CSIRO, Australia, and an Honorary Lecturer in the ANU RSCS. She completed her PhD in Computer Science at the ANU in 2014 and her BSc (Hons) at the University of Colombo, Sri Lanka, in 2009. Her research interests are in privacy in data matching and mining, privacy in social media, privacy risk evaluation and prediction, and health informatics.

Rainer Schnell is Professor at the University of Duisburg-Essen and holds the Chair in Research Methodology in the Social Sciences. He graduated with a postdoctoral degree (habilitation) in Research Methodology at the University of Mannheim in 1996. He is a Survey Methodologist with a research focus on nonsampling errors, applied sampling, census operations, and privacy preserving record linkage.