Efficient Cryptanalysis of Bloom Filters for Privacy-Preserving Record Linkage [⋆]

Peter Christen[1], Rainer Schnell[2], Dinusha Vatsalan[1], and Thilina Ranbaduge[1]

[1] Research School of Computer Science, The Australian National University, Canberra ACT 0200, Australia. Contact: `peter.christen@anu.edu.au`
[2] Institute for Sociology, University Duisburg-Essen, Duisburg, 47057, Germany.

Abstract. Privacy-preserving record linkage (PPRL) is the process of identifying records that represent the same entity across databases held by different organizations without revealing any sensitive information about these entities. A popular technique used in PPRL is Bloom filter encoding, which has shown to be an efficient and effective way to encode sensitive information into bit vectors while still enabling approximate matching of attribute values. However, the encoded values in Bloom filters are vulnerable to cryptanalysis attacks. Under specific conditions, these attacks are successful in that some frequent sensitive attribute values can be re-identified. In this paper we propose and evaluate on real databases a novel efficient attack on Bloom filters. Our approach is based on the construction principle of Bloom filters of hashing elements of sets into bit positions. The attack is independent of the encoding function and its parameters used, it can correctly re-identify sensitive attribute values even when various recently proposed hardening techniques have been applied, and it runs in a few seconds instead of hours.

Keywords: Privacy; re-identification; frequency analysis; data linkage.

1 Introduction

Integrating data from different sources with the aim to remove duplicates, enrich data, and correct errors and inconsistencies is a crucial data pre-processing task for many data mining and analytics applications [4]. Example applications include healthcare, business analytics, national censuses, population informatics, fraud detection, government services, and national security.

However, growing concerns about privacy and confidentiality increasingly preclude the exchange or sharing of personal identifying attributes, such as names, dates of birth, and addresses, which are generally required for linking

[⋆] The authors would like to thank the Isaac Newton Institute for Mathematical Sciences, Cambridge, for support and hospitality during the programme *Data Linkage and Anonymisation* where this work was conducted (EPSRC grant EP/K032208/1). Peter Christen was also supported by a grant from the Simons Foundation. The work was also partially funded by the Australian Research Council (DP130101801).

databases due to the non-existence of common unique entity identifiers [4,18]. Work in privacy-preserving record linkage (PPRL) aims to develop techniques for identifying records that correspond to the same entity across several databases while not compromising the privacy and confidentiality of the entities [18].

PPRL is achieved by conducting linkage on the encoded (masked) values of the identifying attributes of records across two or more databases. Several data encoding techniques for PPRL have been developed. These can be categorized into cryptographic secure multi-party computation (SMC) and perturbation-based techniques [18]. The former are accurate and provably secure, but they incur expensive computation and communication costs. Most PPRL techniques are therefore based on perturbation-based techniques that provide adequate privacy protection while achieving acceptable linkage quality [11,18].

Bloom filter (BF) encoding is one such perturbation-based technique that has successfully been used in several recent practical PPRL applications [2,12]. A BF is a binary vector with bits initially set to 0. A value can be encoded into a BF using a set of hash functions by setting corresponding bits to 1 [13], and the approximate similarity between two BFs can be calculated by counting the number of positions where both BFs have 1-bits in common.

As we discuss in detail in the next section, BFs can be susceptible to crypt-analysis attacks that aim to re-identify the encoded sensitive attribute values [7,8,9,10]. Using frequency counts and patterns in a set of BFs, these attacks iteratively map bit patterns to known attribute values. These existing attacks are however not practical as they require knowledge of certain parameters used during the BF encoding phase, and they have high computational costs.

Our contribution in this paper is an efficient frequency-based approach for attacking BFs that exploits the fundamental property of how the elements of sets are hashed into BFs, as we describe in Sect. 3. In contrast to existing attack methods, our novel approach does not require any assumption on the BF parameters used when sensitive attribute values were encoded, and it is significantly faster, making it a viable attack on large sets of BFs to evaluate whether they provide adequate privacy protection. We experimentally evaluate our attack on two real-world data sets, showing its efficiency and effectiveness.

Given BF encoding is now being employed in real-world PPRL applications [2,12], it is crucial to study possible attacks on BFs to ensure their security and to make users of such systems aware of the weaknesses of BF encoding. Our novel attack method allows data custodians to identify such weaknesses of BF encoding that otherwise could be exploited by an attacker.

2 Prior Attacks on Bloom Filter based PPRL

Schnell et al. [13] were the first to introduce an approximate matching approach for PPRL using Bloom filters (BFs), as we describe in detail in Sect. 3. A recent study by Randall et al. [12] has shown that PPRL based on BF encoding can achieve similar linkage quality as can be achieved with traditional linkage methods on the unencoded attribute values. As a result, BF encoding has been the

PPRL technique of choice in several recent practical PPRL applications [2,12]. However, BFs are prone to different attacks [7,8,9,10]. Therefore, a comprehensive analysis of the weaknesses of BFs in the PPRL context is required. We now describe the few studies that have been conducted on attacks on BFs.

Kuzu et al. [8] formulated a cryptanalysis attack on BFs as a constraint satisfaction problem (CSP). CSP is characterized by a set of variables and a set of constraints on these variables. The aim of this attack is to identify a set of values from a given domain that can be assigned to each variable such that the constraints are satisfied. This is achieved by a frequency analysis of the sensitive attribute values and the BF encodings of the records in a database. However, this attack requires that the attacker has access to a global database where the encoded records are drawn from. This is unlikely in practical applications.

In 2013, Kuzu et al. [9] investigated the accuracy of their CSP attack with two real-world databases. The authors tried to re-identify the personal details of patients in an encoded medical database by using a frequency analysis of BF encodings of a public voter registration database that has a different frequency distribution to the medical database. Their results indicated that although the CSP attack might be feasible in such situations, it is less likely to be accurate in identifying original attribute values and it requires more computational resources. The attack re-identified four out of 20 frequent names correctly.

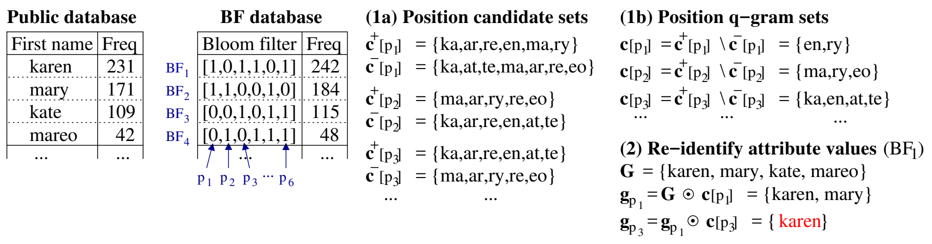
Niedermeyer et al. [10] more recently proposed an attack on BFs built from German surnames. The attack was based on the frequencies of sub-strings of length 2 extracted from frequent surnames. Of $7,580$ surnames, the authors re-identified the 934 most frequent ones (about 12%) before stopping the attack. In contrast to the approach in [9], this attack only depends on the availability of a list of the most common surnames. This work was extended by Kroll and Steinmetzer [7] into a cryptanalysis on BF encodings of several attributes, which was able to re-identify 44% of all attribute values correctly. However, both attacks are based on the specific double hashing scheme used by Schnell at al. [13].

Existing cryptanalysis attacks are feasible only for certain settings and assumptions used in the BF encoding phase. They also require excessive computational resources making them not practical in real settings. Our novel attack method, described next, improves on both drawbacks of existing methods.

3 Overview and Preliminaries

We now provide an overview of our attack on BFs, as illustrated in Fig. 1. As with other attacks on BFs [7,8,9,10], our approach exploits the frequency distribution of a set of BFs that were generated from a large database. As for notation, we use bold letters for sets (with upper-case bold letters for lists or sets of sets) 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.

We assume the attacker has access to a set of encoded BFs, $\mathbf{B}$, and their frequencies, but he does not know anything about the parameters used in the encoding process (such as the number of hash functions used, or the actual hash-



Public database

First name	Freq
karen	231
mary	171
kate	109
mareo	42
...	...

BF database

	Bloom filter	Freq
BF_1	[1,0,1,1,0,1]	242
BF_2	[1,1,0,0,1,0]	184
BF_3	[0,0,1,0,1,1]	115
BF_4	[0,1,0,1,1,1]	48
	...	...

(1a) Position candidate sets

$\mathbf{c}^+[p_1] = \{ka,ar,re,en,ma,ry\}$
$\mathbf{c}^-[p_1] = \{ka,at,te,ma,ar,re,eo\}$

$\mathbf{c}^+[p_2] = \{ma,ar,ry,re,eo\}$
$\mathbf{c}^-[p_2] = \{ka,ar,re,en,at,te\}$

$\mathbf{c}^+[p_3] = \{ka,ar,re,en,at,te\}$
$\mathbf{c}^-[p_3] = \{ma,ar,ry,re,eo\}$
... ...

(1b) Position q−gram sets

$\mathbf{c}[p_1] = \mathbf{c}^+[p_1] \setminus \mathbf{c}^-[p_1] = \{en,ry\}$
$\mathbf{c}[p_2] = \mathbf{c}^+[p_2] \setminus \mathbf{c}^-[p_2] = \{ma,ry,eo\}$
$\mathbf{c}[p_3] = \mathbf{c}^+[p_3] \setminus \mathbf{c}^-[p_3] = \{ka,en,at,te\}$
... ...

(2) Re−identify attribute values (BF_1)

$\mathbf{G} = \{karen, mary, kate, mareo\}$
$\mathbf{g}_{p_1} = \mathbf{G} \odot \mathbf{c}[p_1] = \{karen, mary\}$
$\mathbf{g}_{p_3} = \mathbf{g}_{p_1} \odot \mathbf{c}[p_3] = \{karen\}$

Fig. 1. Outline of the proposed cryptanalysis attack, which is based on a set of BFs and a set attribute values, both sorted according to their frequencies. In steps (1a) and (1b), the attack exploits the bit patterns in BFs to identify the sets of q-grams that are possible, $\mathbf{c}^+[p]$, and *not* possible, $\mathbf{c}^-[p]$, respectively, for each bit position p. In step (2), the re-identification of a set of (sensitive) attribute values, $\mathbf{G}$, is conducted by intersecting the q-gram sets of values in $\mathbf{G}$ with the sets $\mathbf{c}$ (illustrated using $\odot$).

ing mechanism). We assume these BFs represent a set of records that encode sensitive values from one or a few attributes. Based on the frequency distribution of the Hamming weights (number of 1-bits) in $\mathbf{B}$, the attacker can guess which attribute(s) have been encoded, because different attributes (such as first name, surname, city name, or postcode) have distinctive distributions of Hamming weights [15]. The distribution of Hamming weights is independent of the (unknown) secret key. Therefore, an attacker can sample attribute values from a publicly available population database (such as a telephone directory) and select a set of frequent values, $\mathbf{V}$, from an attribute that has a frequency distribution that matches with the distribution of the set of BFs $\mathbf{B}$ to be attacked.

We align BFs and attribute values according to their frequencies, and consider the set of most frequent values in both. For each bit position p in the BFs, for all corresponding attribute values that have this bit set to 1 we add their q-grams (character sub-strings of length q generated from attribute values) to the set $\mathbf{c}^+[p]$ of possible q-grams for that position. The reasoning is that a 1-bit means at least one q-gram of an attribute value was hashed to this position. Similarly, for all attribute values with a value of 0 at bit position p we add their q-grams to the set $\mathbf{c}^-[p]$ of *not* possible q-grams for that position, because a 0-bit means no q-gram of an attribute value could have been mapped to this position.

At the end of this process, for each bit position p we obtain the set $\mathbf{c}[p] = \mathbf{c}^+[p] \setminus \mathbf{c}^-[p]$ of q-grams that potentially could have been hashed to position p. Based on the list $\mathbf{C} = [\mathbf{c}[1], \ldots, \mathbf{c}[l]]$, where l is the length of the BFs, and a set $\mathbf{G}$ of attribute values we aim to re-identify (i.e. learn which BF possibly encodes which value in $\mathbf{G}$), we can now analyze each BF in $\mathbf{B}$ and remove those attribute values from $\mathbf{G}$ that are not possible matches according to $\mathbf{C}$ because they do not contain any q-grams that would have been hashed to a certain 1-bit.

For example, in Fig. 1, for the most frequent BF_1, 'kate' is not a possible value because in order to obtain a 1-bit in position p_1, it would have to contain either the q-gram 'en' or 'ry'; while 'mary' is also not possible because it would need to contain one of the q-grams 'ka', 'en', 'at', or 'te' in position p_3.

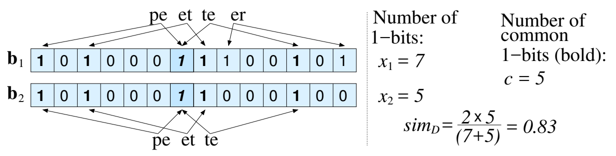

pe et te er
$\mathbf{b}_1$ 1 0 1 0 0 0 1 1 1 0 0 1 0 1
$\mathbf{b}_2$ 1 0 1 0 0 0 1 1 0 0 0 1 0 0
pe et te
Number of 1−bits:
$x_1 = 7$
$x_2 = 5$
Number of common 1−bits (bold):
$c = 5$
$sim_D = \frac{2 \times 5}{(7+5)} = 0.83$

Fig. 2. An example Dice coefficient similarity calculation of the two first names 'peter' and 'pete' encoded in BFs, as described in Sect. 3. The dark bit shows a hash collision.

Before we formalize and present our approach in detail in Sect. 4, we first describe BF encoding, as well as some recent approaches to harden them.

Bloom Filter Encoding: Proposed by Bloom [1] in 1970 for the space and time efficient representation of sets, a BF $\mathbf{b}$ is a bit vector of length l where all bits are initially set to 0. k independent hash functions, $h_1, \ldots, h_k$, each with range $1, \ldots, l$, are used to map the elements s in a set $\mathbf{s}$ into the BF by setting the bit positions $\mathbf{b}[h_j(s)] = 1$, with $1 \leq j \leq k$.

For PPRL, the set $\mathbf{s}$ of q-grams generated from string values [13], or neighboring values for numerical values [17], can be hash-mapped into a BF. These BFs are then either sent to a linkage unit (LU, an external party that conducts the linkage) to calculate the similarity between BFs in order to classify them as matches or non-matches [13], or they are partially exchanged among the database owners to distributively calculate the similarities between BFs [16].

The Dice coefficient has been used for comparing BFs since it is insensitive to many matching zeros in long BFs [13]. For two BFs, $\mathbf{b}_1$ and $\mathbf{b}_2$, the Dice coefficient similarity is: $sim_D(\mathbf{b}_1, \mathbf{b}_2) = 2c/(x_1 + x_2)$, where c is the number of bit positions that are set to 1 in both BFs (common 1-bits), and x_1 and x_2 are the number of bit positions set to 1 in $\mathbf{b}_1$ and $\mathbf{b}_2$, respectively. Figure 2 shows the encoding of bigrams ($q = 2$) of two string values into $l = 14$ bits long BFs using $k = 2$ hash functions, and their Dice coefficient similarity calculation.

Different encoding methods have been proposed for BFs. Hashing several attributes of a record into one BF is a method known as cryptographic long term key (CLK). It is used to improve privacy [14]. Another record-level BF (RBF) encoding was proposed to improve linkage quality [5]. In RBF, attribute values are first hashed into different BFs and then bits are selected from each attribute-level BF into a RBF according to attribute weights.

The initial proposal of BFs for PPRL used a double hashing scheme [13], where the k individual bit positions for an element s to be hashed are determined by the sum of the integer representation of two independent hash functions that are mapped into the range $1 - l$. Random hashing has recently been proposed as an improvement over double hashing to prevent against cryptanalysis attacks, where k random numbers are drawn for every element s to be hashed [10,15].

Bloom Filter Hardening: Several BF hardening methods have been studied in recent times to reduce the vulnerability of BFs against cryptanalysis attacks [15]. Compared to attribute-level BFs, record-level BF encodings such as CLK and RBF reduce the risk of re-identification by such attacks [8,10].

Exploiting the fact that data sets with (near) uniform Hamming weight distribution of BFs are more difficult to attack (with existing attack methods) than data sets with non-uniform distributions, balancing BFs with constant Hamming weight has been proposed [15]. Balanced BFs can be constructed by concatenating a BF of length l with its negated copy (all bits flipped) and then permuting the $2l$ bits. Another proposed approach to harden BFs is XOR-folding, where a BF of length l is split into two halves of length $l/2$ each, and then bit-wise exclusive OR is applied to combine the two shorter BFs [15].

While balancing and XOR-folding are easy and data independent hardening techniques, salting with record-specific values has been suggested as an alternative hardening method where an additional (record-specific) value is concatenated with attribute values before being hashed into the BF [10]. A cryptanalysis attack is unlikely to be successful without knowing the salting key, however attributes suitable for salting might not be available in a database. Other, more experimental hardening techniques, include random bits and fake record injection [5,18], as well as BLIP (BLoom-and-flIP) which flips bits (noise addition) in a BF according to a differential privacy model [15]. Many of these hardening techniques improve security against attacks at the cost of a reduction in linkage quality [15]. In the experiments in Sect. 5 we will investigate if balancing and XOR-folding make BFs more resistant to our proposed attack.

4 Frequency-based Bloom Filter Cryptanalysis

We now describe our frequency-based attack on BFs in detail. As shown in Fig. 1, the attack consists of two main steps. First, for each BF position we find its set of possible and *not* possible q-grams (steps (1a) and (1b) in Fig. 1). Next, we re-identify for each BF the set of attribute values that possibly were hashed into this BF (step (2) in Fig. 1). Algorithms 1 and 2 show the details of steps (1a) and (1b), and (2), respectively, which we describe in the next two sub-sections. We then provide an analysis of our attack and discuss its limitations.

4.1 Candidate Q-gram Set Generation

For our attack we require as input a set of BFs, $\mathbf{B}$, that we assume to come from a sensitive database (the one from which we aim to re-identify its sensitive values), and a set of attribute values, $\mathbf{V}$, assumed to come from a public database. Each BF $\mathbf{b}_i \in \mathbf{B}$ and each attribute value $v_i \in \mathbf{V}$ has a frequency attached to it, denoted with $\mathbf{b}_i.f$ and $v_i.f$, respectively. Unlike previous attacks on BFs [8,9,10], we do not require any other information about how the BFs were encoded.

Algorithm 1 starts by initializing two empty sets for each BF position p: $\mathbf{c}^+[p]$ of possible q-grams at that position, and $\mathbf{c}^-[p]$ of *not* possible q-grams at that position. Next, in lines 2 and 3, we find all BFs and attribute values that occur at least m times. In line 4 we sort both the BFs and attribute values according to their frequencies in reverse order (most frequent first), and then we align a BF $\mathbf{b}_i$ and an attribute value v_i into the sorted list $\mathbf{A}$ of pairs $(\mathbf{b}_i, v_i)$.

Algorithm 1: ***Candidate q-gram set generation***

Input:
- $\mathbf{V}$: Set of attribute values and their frequencies from a public database
- $\mathbf{B}$: Set of BFs and their frequencies from the sensitive database
- l: Length of Bloom filters
- q: Length of sub-strings to extract from attribute values
- m: Minimum frequency for BFs and attribute values

Output:
- $\mathbf{C}$: List of possible q-grams for each BF position

```
1:  $\mathbf{c}^+[p] = \{\}$, $\mathbf{c}^-[p] = \{\}$, $1 \leq p \leq l$     // Initialize list of candidate q-gram sets
2:  $\mathbf{V}_F = \{v \in \mathbf{V} : v.f \geq m\}$                  // Attribute values with frequency of at least $m$
3:  $\mathbf{B}_F = \{\mathbf{b} \in \mathbf{B} : \mathbf{b}.f \geq m\}$                  // BFs with frequency of at least $m$
4:  $revSort(\mathbf{B}_F)$, $revSort(\mathbf{V}_F)$          // Sort according to frequencies, highest first
5:  $\mathbf{A} = [(\mathbf{b}_i, v_i) : \mathbf{b}_i \in \mathbf{B}_F, v_i \in \mathbf{V}_F : \mathbf{b}_i.f > \mathbf{b}_j.f \wedge v_i.f > v_j.f : 1 \leq i < j \leq min(|\mathbf{V}_F|, |\mathbf{B}_F|)]$
                                  // Align $\mathbf{V}_F$ and $\mathbf{B}_F$ as long as their frequencies are unique
6:  for $(\mathbf{b}_i, v_i) \in \mathbf{A}$ do:                  // Step (1a): Get candidate sets of q-grams
7:      $\mathbf{q}_i = genQGramSet(v_i, q)$          // Convert attribute value into its q-gram set
8:      for $1 \leq p \leq l$ do:                      // Loop over all BF positions
9:          if $\mathbf{b}_i[p] == 1$ then:             // Bit at position p is 1
10:             $\mathbf{c}^+[p] = \mathbf{c}^+[p] \cup \mathbf{q}_i$             // Add to set of possible q-grams
11:         else:                                 // Bit at position $p$ is 0
12:             $\mathbf{c}^-[p] = \mathbf{c}^-[p] \cup \mathbf{q}_i$             // Add to set of *not* possible q-grams
13: $\mathbf{C} = [\,]$                                 // Step (1b): Initialize empty list of q-gram sets
14: for $1 \leq p \leq l$ do:                          // Loop over all BF positions to combine q-gram sets
15:     $\mathbf{C}.append(\mathbf{c}^+[p] \setminus \mathbf{c}^-[p])$              // Remove not possible from possible q-grams
16: return $\mathbf{C}$
```

We do this as long as both the ith BF $\mathbf{b}_i$ and ith attribute value v_i have a unique frequency compared to the next, less frequent, BF or attribute value, respectively. This stopping criterion ensures that we do not have a BF that could correspond to two or more attribute values, and vice versa, as this would lead to more uncertainty in the mapping of q-grams into BF positions.

Step (1a) of our approach starts in line 6 and loops over pairs of aligned BFs and attribute values, $(\mathbf{b}_i, v_i) \in \mathbf{A}$. First, we convert v_i into its set of q-grams, $\mathbf{q}_i$, in line 7. Then we loop over all BF positions, $1 \leq p \leq l$, in line 8, and if the bit at position p in BF is 1 (i.e., $\mathbf{b}_i[p] = 1$), we add the q-gram set $\mathbf{q}_i$ to the set $\mathbf{c}^+[p]$ of possible q-grams at that position (line 10) because a 1-bit means any q-gram from $\mathbf{q}_i$ could have been hashed to that position. Conversely, if the BF bit at position p is 0, then no q-gram from $\mathbf{q}_i$ could have been hashed to that position and so we add the q-grams in $\mathbf{q}_i$ to $\mathbf{c}^-[p]$ (line 12).

In step (1b), line 13 onwards in Algo. 1, we get the final set of q-grams for each BF position p as the set of possible q-grams ($\mathbf{c}^+[p]$) minus the set of not possible q-grams ($\mathbf{c}^-[p]$), and add these sets to the list $\mathbf{C}$ in line 15.

4.2 Attribute Value Re-identification

The re-identification step, shown in Algo. 2, has as input the same set of BFs, $\mathbf{B}$, as used in the first step, as well as the list of candidate q-grams sets, $\mathbf{C}$, generated in Algo. 1. A set of frequent attribute values, $\mathbf{G}$, also needs to be provided. These are the values we aim to re-identify (guess) in $\mathbf{B}$ using $\mathbf{C}$. The output of the algorithm is a set of one or more re-identified attribute value(s), $\mathbf{g}_i \subset \mathbf{G}$, for each BF $\mathbf{b}_i \in \mathbf{B}$, collected in the result list $\mathbf{R}$.

Algorithm 2: ***Attribute value re-identification***

```
Input:
- $\mathbf{B}$: Set of BFs from the sensitive database (same as in Algo. 1)
- $\mathbf{C}$: List of possible q-grams for each BF position (from Algo. 1)
- $\mathbf{G}$: Set of attribute values we aim to re-identify in the set of BFs $\mathbf{B}$
Output:
- $\mathbf{R}$: List of possible attribute values re-identified for each BF in $\mathbf{B}$
1:  $\mathbf{R} = [\,]$                  // Initialize empty list of re-identified attribute values
2:  for $\mathbf{b}_i \in \mathbf{B}$ do:          // Loop over all BFs
3:      $\mathbf{g}_i = \mathbf{G}$              // Initialize set of candidate attribute values as all possible values
4:      for $1 \leq p \leq l$ do:                  // Loop over all BF positions
5:          if $\mathbf{b}_i[p] == 1$ then:           // Bit at position $p$ is 1
6:              $\mathbf{c} = \mathbf{C}[p]$                   // Set of possible q-grams at this bit position
7:              for $v_j \in \mathbf{g}_i$ do:            // Check all candidate attribute values
8:                  if $(\forall q \in \mathbf{c} : q \notin v_j)$ then:  // Check if no q-gram from $\mathbf{c}$ in attribute value
9:                      $\mathbf{g}_i = \mathbf{g}_i \setminus v_j$       // Value could not have been hashed into this BF
10:     $\mathbf{R}.append(\mathbf{g}_i)$                  // Append possible attribute values for this BF
11: return $\mathbf{R}$
```

The algorithm loops over all BFs $\mathbf{b}_i$ in $\mathbf{B}$ (line 2 onwards), 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 $\mathbf{G}$ (line 3). Next we loop over all BF positions p (line 4), and for any position that has a 1-bit ($\mathbf{b}_i[p] = 1$) we retrieve the set of possible q-grams $\mathbf{c} = \mathbf{C}[p]$ at that position (line 6).

Using $\mathbf{c}$, we now check for each attribute value v_j in $\mathbf{g}_i$ if *at least one of the q-grams* in $\mathbf{c}$ occurs in that value (lines 7 and 8). If v_j does not contain at least one q-gram from $\mathbf{c}$, it could not have generated the 1-bit at position p. If this is the case, in line 9 we remove v_j from the set of candidates $\mathbf{g}_i$. At the end of this loop (line 10), the set $\mathbf{g}_i$ will contain all attribute values from $\mathbf{G}$ that possibly have generated the BF $\mathbf{b}_i$, and we append this set to the results list $\mathbf{R}$. Note that positions with 0-bits do not help us to remove values $v_j \in \mathbf{g}_i$ because any q-gram in a value v_j is potentially hashed into other positions.

4.3 Analysis and Limitations

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

Complexity: We assume $N = |\mathbf{B}|$ is the number of BFs coming from the sensitive database, Q is the average number of q-grams per value in the set of attribute values $\mathbf{V}$ from the public database, k is the number of hash functions used to hash q-grams into BFs, and l is the length of a BF.

In step (1) of our attack, we first find all frequent BFs and attribute values that occur at least m times. This requires a linear scan through $\mathbf{B}$ and $\mathbf{V}$, respectively, which has a complexity of $O(N)$ and $O(|\mathbf{V}|)$. In line 4 of Algo. 1, the sorting function used on values in $\mathbf{B}_F$ and $\mathbf{V}_F$ has a complexity of $O(M \cdot log\, M)$, where $M = min(|\mathbf{B}_F|, |\mathbf{V}_F|)$. In step (1a), each attribute value in $\mathbf{A}$ is converted into its set of q-grams which are then added to the sets $\mathbf{c}^+[p]$ and $\mathbf{c}^-[p]$ for all BF bit positions $1 \leq b \leq l$. This step has a computational complexity of $O(M \cdot Q \cdot l)$. In step (1b), the sets of possible q-grams are added to list $\mathbf{C}$ by performing a set difference operation between $\mathbf{c}^+[p]$ and $\mathbf{c}^-[p]$. Assuming $\mathbf{Q}$ is the set of all unique q-grams generated from all values $v_i \in \mathbf{V}$, then on average $t = |\mathbf{Q}| * k/l$ q-grams are hashed into each bit position. Therefore each set difference is of $O(t)$, and so step (1b) has an overall complexity of $O(t \cdot l)$ for l bit positions.

In step (2) of our attack, Algo. 2 iterates through each BF $\mathbf{b}_i \in \mathbf{B}$ to re-identify the possible attribute values $\mathbf{g}_i \in \mathbf{G}$ that map to $\mathbf{b}_i$. For each bit position p that is set to 1, we check if any q-gram in $\mathbf{C}[p]$ occurs in a value $v_j \in \mathbf{g}_i$. This leads to a total complexity of $O(N \cdot |\mathbf{G}| \cdot l \cdot t)$, assuming the average size of a $\mathbf{C}[p]$ is t. Finally, the worst case space complexity of the list $\mathbf{R}$ returned by Algo. 2 is $O(N \cdot |\mathbf{G}|)$ if every value in $\mathbf{G}$ is mapped to each $\mathbf{b}_i \in \mathbf{B}$.

Limitations: Our frequency-based cryptanalysis attack on BFs depends on several assumptions. First, we assume the attacker has access to a publicly available population database from where the set $\mathbf{V}$ of attribute values and their frequencies can be extracted. We also assume that $\mathbf{B}$ does contain a sub-set of BFs that occur several times, and that the frequency distribution of BFs in $\mathbf{B}$ can be correlated by an attacker to the distribution of a single or a sub-set of attribute values in $\mathbf{V}$. Without such frequency information, our attack (like previous cryptanalysis attacks on BFs [7,8,9,10]) would not be possible.

5 Experiments and Results

We conducted our experimental study using real data sets from two domains. The first are a pair of North Carolina Voter Registration (NCVR) data sets (`ftp://alt.ncsbe.gov/data/`) collected in June 2014 (the database to be attacked) and October 2016 (the public database). These data sets contain over five million records of voters including their first names, surnames, and addresses. We present results of our attack individually on the first name attribute, as well as on the concatenation of the first name and surname attributes.

The second are a pair of census data sets (named UKCD) collected from the years 1851 and 1901 (used as the sensitive and public databases, respectively) for the town of Rawtenstall in England [6]. These data sets contain around 50,000 records with personal details of individuals. We again use the first name attribute, and the concatenation of the first name and surname attributes.

The parameter settings we use in our experiments are $q = [2, 3, 4]$ (length of sub-strings used in BF encoding), BF length $l = [250, 500, 1000, 2000]$, and either the double or random hashing method [15] as described in Sect. 3. We calculate the number of hash functions k based on l and q such that it minimizes the false positive rate [16]. We also investigate the discussed BF hardening techniques balancing and XOR-folding [15]. Finally, we use different numbers for the most frequent attribute values to be re-identified, i.e. $|\mathbf{G}| = [10, 20, 50, 100]$.

We evaluate the accuracy of our attack by calculating (1) the percentage of correct guesses with 1-to-1 matching, (2) the percentage of correct guesses with 1-to-m (many) matching, (3) the percentage of wrong guesses, and (4) the percentage of no guesses, where these four percentages sum to 100. These four categories are labeled as '1-1 corr', '1-m corr', 'Wrong', and 'No' in the plots, respectively. We evaluate the efficiency of our attack using run time.

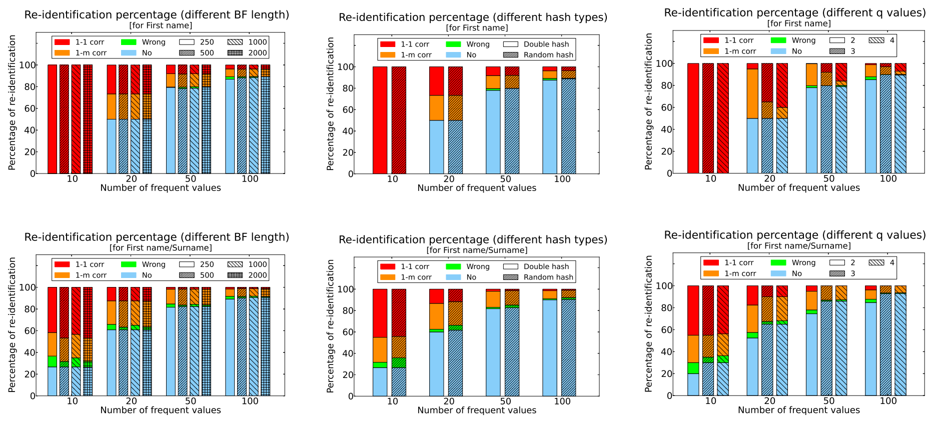


Fig. 3. Results for the NCVR data sets with first name (top) and combination of first and surname (bottom) for different BF lengths (left), different hashing methods (middle), and different values of q (right). No BF hardening was applied.

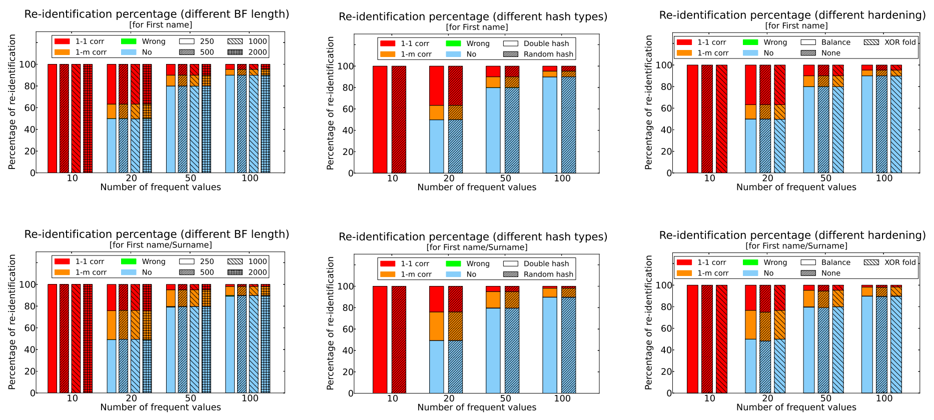


Fig. 4. Results for the UKCD data sets with first name (top) and combination of first and surname (bottom) for different BF lengths (left), different hashing methods (middle), and different BF hardening methods (right).

We implemented our attack using Python 2.7 and ran all experiments on a server with 64-bit Intel Xeon 2.4 GHz CPUs, 128 GBytes of memory and running Ubuntu 14.04. The programs and data sets are available from the authors.

Discussion: In Fig. 3 we show the results for the NCVR data sets. As can be seen, when the BF length l increases (left column) the percentage of correct re-identifications mostly increases, as with larger l the number of q-grams mapped to a certain bit position decreases. All values can be correctly re-identified for individual attributes when the number of frequent values is 10. Around half of all values can still be re-identified even when values from two attributes are

Table 1. Comparison of re-identification results with existing BF attack methods.

Publication	Data set	Num BFs	1-1 correct	Run time
Kuzu et al. (2011) [8]	NCVR first names	3,500	400	1,000 sec
Kuzu et al. (2013) [9]	Patient names	20	4	few sec
"	"	42	0	> week
Niedermeyer et al. (2014) [10]	German surnames	7,580	934	> days
Kroll and Steinmetzer (2015) [7]	German names and locations	100K	44K	> days
Our approach	NCVR first names	10–100	10–7	0.73–0.75 sec
"	NCVR first and surnames	10–100	6–3	1.5–1.9 sec

combined. The middle column shows that random hashing (which supposedly improves privacy on BFs compared to double hashing [7,10]) does not provide improved protection against our attack, as a similar percentage of values can be correctly re-identified for both hashing approaches. As for using different values of q (right column) the accuracy of an attack somewhat improves when q is increased because larger values of q result in more unique q-grams.

The accuracy results for the UKCD data sets are shown in Fig. 4 for the same attributes as for the NCVR data sets. Similar re-identification patterns as with the NCVR data sets can be seen for different BF lengths (left) and hashing methods (middle). We also study how different BF hardening methods affect the re-identification accuracy, as shown in the right column plots in Fig. 4. For the single attribute case (top right), neither of the hardening techniques is capable of reducing the re-identification accuracy, however for the combined attribute case both hardening techniques improve the privacy of BF encoding.

In Table 1 we compare our attack method with existing approaches in terms of accuracy and efficiency (with results taken from the corresponding papers). As can be seen, our method is both more efficient and effective in re-identification, having both higher accuracy and reduced computational requirements.

These results show the vulnerability of basic BF encoding to our novel attack. They highlight the need for improved hardening techniques to overcome such attacks. Our attack provides data custodians with an efficient method to evaluate the privacy of their BF encoded databases before using them for PPRL.

Recommendations: As a set of guidelines for the practical application of BF based PPRL systems, to limit the vulnerability of such systems to known attack methods we recommend to use record-level BF encoding (CLK or RBF), apply BF hardening methods as discussed in Sect. 3, and reduce the frequency of bit patterns (for example, by salting) to prevent any frequency analysis.

6 Conclusions and Future Work

We have presented a novel efficient frequency-based attack on Bloom filters (BFs) that contain encoded sensitive attribute values intended for privacy-preserving record linkage (PPRL). Unlike earlier attacks on BFs for PPRL, our approach only requires an attacker to have access to a public database of attribute values, but no information about the BF encoding used. Our approach is faster than earlier attacks, making it feasible for database owners to efficiently validate the security of their encoded sensitive databases before they are being sent to other parties for conducting PPRL. We believe our attack is an important component to making PPRL more secure for practical applications.

As future work we will study how our approach can be modified for attacking composite and record-level BFs. We also plan to investigate the risk of re-identification when advanced hardening techniques, such as BLIP or BF salting, have been applied. Finally, our attack can be accelerated by further analyzing the re-identified attribute values, while correlations between 1-bits and q-gram sets across BFs could be identified using association rule mining techniques [3].

References

1. Bloom, B.: Space/time trade-offs in hash coding with allowable errors. Communications of the ACM 13(7), 422–426 (1970)
2. Boyd, J., Randall, S., Ferrante, A.: Application of privacy-preserving techniques in operational record linkage centres. In: Medical Data Privacy Handbook (2015)
3. Ceglar, A., Roddick, J.F.: Association mining. ACM CSUR 3(2), 5 (2006)
4. Christen, P.: Data Matching – Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection. Springer (2012)
5. Durham, E.A., Kantarcioglu, M., Xue, Y., Toth, C., Kuzu, M., Malin, B.: Composite Bloom filters for secure record linkage. IEEE TKDE 26(12), 2956–2968 (2014)
6. Fu, Z., Christen, P., Zhou, J.: A graph matching method for historical census household linkage. In: PAKDD. Tainan (2014)
7. Kroll, M., Steinmetzer, S.: Who is 1011011111...1110110010? Automated cryptanalysis of Bloom filter encryptions of databases with several personal identifiers. In: BIOSTEC. Lisbon (2015)
8. Kuzu, M., Kantarcioglu, M., Durham, E., Malin, B.: A constraint satisfaction cryptanalysis of Bloom filters in private record linkage. In: PET. Waterloo (2011)
9. Kuzu, M., Kantarcioglu, M., Durham, E., Toth, C., Malin, B.: A practical approach to achieve private medical record linkage in light of public resources. JAMIA (2013)
10. Niedermeyer, F., Steinmetzer, S., Kroll, M., Schnell, R.: Cryptanalysis of basic Bloom filters used for privacy preserving record linkage. JPC 6(2), 59–79 (2014)
11. Ranbaduge, T., Vatsalan, D., Christen, P., Verykios, V.: Hashing-based distributed multi-party blocking for privacy-preserving record linkage. In: PAKDD (2016)
12. Randall, S., Ferrante, A., Boyd, J., Bauer, J., Semmens, J.: Privacy-preserving record linkage on large real world datasets. JBI 50, 205–212 (2014)
13. Schnell, R., Bachteler, T., Reiher, J.: Privacy-preserving record linkage using Bloom filters. BMC Med Inform Decis Mak 9(1) (2009)
14. Schnell, R., Bachteler, T., Reiher, J.: A novel error-tolerant anonymous linking code. working paper, German Record Linkage Center (2011)
15. Schnell, R., Borgs, C.: Randomized response and balanced bloom filters for privacy preserving record linkage. In: ICDMW DINA. Barcelona (2016)
16. Vatsalan, D., Christen, P.: Scalable privacy-preserving record linkage for multiple databases. In: ACM CIKM. Shanghai (2014)
17. Vatsalan, D., Christen, P.: Privacy-preserving matching of similar patients. JBI 59, 285–298 (2016)
18. Vatsalan, D., Christen, P., Verykios, V.S.: A taxonomy of privacy-preserving record linkage techniques. Elsevier IS 38(6), 946–969 (2013)