A Graph Traversal Attack on Bloom Filter Based Medical Data Aggregation

William Mitchell, Rinku Dewri, Ramakrishna Thurimella
Max Roschke
Department of Computer Science
University of Denver, Denver, CO 80210. USA.

Abstract

We present a novel cryptanalytic method based on graph traversals to show that record linkage using Bloom Filter Encoding does not preserve privacy in a two-party setting. Bloom Filter Encoding is often suggested as a practical approach to medical data aggregation. This attack is stronger than a simple dictionary attack in that it does not assume knowledge of the universe. The attack is very practical and produced accurate results when experimented on large amounts of name-like data derived from a North Carolina voter registration database. We also give theoretical arguments that show that going from bigrams to n-grams, $n > 2$, does not increase privacy; on the contrary it actually makes the attack more effective. Finally, some ways to resist this attack are suggested.

1 Introduction

When aggregating medical data for research, it is necessary to link data on the same person, but originating from different sources. The techniques for linking data involve ways to match pairs of data records based on the value of personally identifying information (commonly referred to as quasi-identifiers or QIDs), such as names, birth dates, addresses, and national or local identifying codes. Even when free access to personally identifying information is available, incompleteness and errors in the data make matching problematic. If data from the same person is not linked, statistical studies will suffer from inflated sample sizes and unknown repeated measures bias. If data from two different persons are linked, their combined data represent a fiction that would contaminate study conclusions, and would deflate actual sample sizes. Therefore, accuracy is critical in this aggregation.

For research, the privacy rules of the Health Insurance Portability and Accountability Act (HIPAA) in the US and the Data Protection Directive of the European Union, severely restrict researcher access to QIDs, given the nearly universal lack of patient authorization to use Protected Health Information.

Hence, along with accuracy, equally important consideration is a privacy guarantee during the linkage process.

A final factor that determines the usefulness of a linkage algorithm is whether it can scale to large datasets. Recent results that are based on new cryptographic methods, including partially homomorphic encryption [13], do not sufficiently demonstrate their implementability on large, realistic datasets.

The combining of databases such that similar records are linked together where the linking characteristics are kept private is known as *Privacy Preserving Record Linkage* (PPRL) or alternatively *Private Record Linkage* (PRL). The process of linkage consists of matching on encoded data, with an understanding by the data providers that other selected attributes will be revealed on matched data.

Note that linking data from different sources in a privacy preserving manner has many more applications than just linking health records. A commonly cited example [22] involves the study of finding correlation between certain kind of automobile accidents and resulting injuries. In order to conduct such a study, data is needed from police records, insurance companies and hospitals. These entities are not going to share data unless there are strong privacy guarantees. There are many other applications for PPRL, but our focus in this paper is limited to linking health records.

While there are multiple approaches to PPRL[22], one approach that has received considerable attention is Bloom Filter Encoding (BFE) [19]. First proposed by Schnell, Bachteler and Reiher[19], it splits a word to encode into equal-length, overlapping pieces (so called n-grams, for some $n > 1$), inserts the pieces into a Bloom filter, and determines if two records match by comparing the resulting filters. Similarity of two Bloom filters is measured by the number of bits that are common between the two filters in relation to the total number of bits. Based on this similarity, a match or non-match is declared between the two corresponding words. Other approaches to PPRL use secure multi-party computations, which are computationally expensive, or require exact matching of identifiers. In contrast, BFE is computationally reasonable to apply to large datasets and allows for approximate matching. Accordingly, interest in BFE has been seen for usage in PPRL of medical records [4, 3].

The linkage can be performed in a two-party setting or by employing a third party–a semi-honest broker. Schnell et al. observed the obvious weakness in the former approach: an attacker can launch a dictionary attack to recover the data. Such an attack assumes knowledge of the universe from which the words are drawn. The main result of this paper is to show that a non-semantic attack, one that does not assume knowledge of the universe, yet recovers the encoded data with a high rate of success is possible. The cryptanalytic method employed is novel, very efficient and resulted in recovering a high percentage of plain text. Our experimental results were carried out on a half million name-like words derived from the North Carolina voter registration database, ten thousand random ten letter words, and finally ten thousand random nine digit numbers.

The rest of the paper is organized as follows. Section 3 covers related work.

Details of Bloom filter encoding are given in Section 4. The main result is presented in Section 5. Section 6 gives some scenarios when this attack is not effective. Section 7 presents an argument detailing why it does not help to go to higher n-grams. Experimental results are given in Section 8. Finally, some implications and countermeasures are discussed in Section 9.

2 Contributions

This paper presents a novel cryptanalytic method based on graph traversals to show that record linkage using Bloom Filter Encoding does not preserve privacy in a two-party setting. Bloom Filter Encoding is often suggested as a practical approach to medical data aggregation. For a detailed discussion on various linkage approaches, including Bloom Filter Encoding, see [22]. In [19], a dictionary attack is proposed against Bloom Filter Encoding. Our attack is stronger than a simple dictionary attack in that it does not assume knowledge of the universe. The attack runs quickly and produced accurate results when experimented on large amounts of realistic name-like data derived from a North Carolina voter registration database, synthetically generated random strings, and synthetically generated random numbers. We also give theoretical arguments that show that going from bigrams to n-grams, $n > 2$, does not increase privacy; on the contrary it actually makes the attack more effective. Finally, some ways to resist this attack are suggested.

3 Related Work

There have been attacks on BFE directly. First, as remarked in Schnell et al.'s original paper on BFE, a dictionary attack can be mounted in a two-party scenario. Second, a constraint satisfaction attack can be mounted on the three-party case[12]. But it was of limited effect in practice[11]. Finally, a frequency analysis attack has been proposed by [10, 15]. The attack requires knowledge of the universe, and particular choices to be made regarding the hash functions used.

Various enhancements to the basic BFE approach have been proposed. These include calculating adaptive parameters and compositing the result [3] and even adding homomorphic encryption [17].

The privacy-preserving aspect of record linkage plays a central role when architecting a federated system to perform medical data aggregation. Two such projects are SAFTINet[18] in North America and VPH-Share[7] in Europe.

A key building block of PPRL is related to privacy-preserving set intersection. This problem was initially considered by [6]. Algorithms for other set operations in a privacy-preserving setting were presented by [9]. For more recent work see [21].

4 Bloom Filter Encoding

Using BFE for PPRL was first proposed by [19]. BFE is used because similar strings encode similarly. This similarity is what is used to perform RL.

At its most basic, BFE requires only three parameters: m bits for the size of the filter, k hash functions, and the size of n–grams to use. The first two parameters are the same as a standard Bloom Filter, whereas the third is specific to BFE.

It has been shown by [8] that the k hash functions can be replaced by linearly combining the results of two independent hash functions. This computationally decreases the cost of encoding, without sacrificing any utility of the filter. The k hash functions are created from two independent hash functions H_1 and H_2 by

$$H_i(x) \equiv H_1(x) + i \cdot H_2(x) \pmod{m} \tag{1}$$

where $0 \leq i < k$ and $i \in \mathbb{Z}$. It is this computational enhancement that [10, 15] exploit to mount their frequency analysis attack.

To obtain the required two hash functions for BFE, it is common to use HMAC, specified in FIPS-198[14], with a cryptographic hash and two distinct keys. HMAC creates a new hash from an existing unkeyed hash H and key K as

$$\text{HMAC}_{H,K}(x) = H\big(K \oplus \texttt{opad} || H(K \oplus \texttt{ipad} || x)\big) \tag{2}$$

where $\oplus$ is exclusive–or, $||$ is concatenation, and `ipad` and `opad` are fixed padding constants. FIPS-198 specifies specific hashes and padding constants to use, however other carefully selected hashes and constants can be used.

After these parameters are selected, the item to encode is split into n–grams. First, $n-1$ start sentinel characters are prepended and $n-1$ stop sentinel characters are appended. These sentinel characters need not be the same, but most of the literature in this area use an underscore for both. To emphasize the distinction, we will use "^" for the start sentinel and "$" for the stop sentinel. Since an n–gram of all sentinels is not possible, the choice of sentinel is therefore arbitrary. Next, the result is split into groups of n consecutive letters. The number of groups created will be $n-1$ more than the length of the word. Each individual letter group is inserted into the Bloom Filter.

Suppose we wished to encode "SMITH" using bigrams. Then the bigrams would be computed as

$$\texttt{SMITH} \rightarrow \texttt{\^{}SMITH\$} \rightarrow \{\texttt{\^{}S}, \texttt{SM}, \texttt{MI}, \texttt{IT}, \texttt{TH}, \texttt{H\$}\},$$

with the resulting six bigrams inserted into the Bloom Filter.

If trigrams were in use, "SMITH" would be computed as

$$\texttt{SMITH} \rightarrow \texttt{\^{}\^{}SMITH\$\$} \rightarrow \{\texttt{\^{}\^{}S}, \texttt{\^{}SM}, \texttt{SMI}, \texttt{MIT}, \texttt{ITH}, \texttt{TH\$}, \texttt{H\$\$}\}.$$

4.1 Example Encoding

Suppose we wish to encode "SMITH" into a Bloom Filter with $m = 35$, $k = 3$, $n = 2$. The hash functions will be derived from HMAC with SHA-256. The

key lengths should match the block size of the selected hash. If they do not, a key derivation algorithm as described in FIPS-198 is used to make them so. SHA-256 produces a hash 256 bits in length, so the keys need to also be 256 bits in length. We will use keys $k_1 = \texttt{0x11...1}$ and $k_2 = \texttt{0x22...2}$, each of which are specified in hexadecimal and 64 characters in length.

The data to encode, "SMITH," is split up into six bigrams $\{\texttt{\^{}S}, \texttt{SM}, \texttt{MI}, \texttt{IT}, \texttt{TH}, \texttt{H\$}\}$. Each bigram is hashed by the two HMAC functions.

$$\begin{aligned}
&\text{HMAC}_{\text{SHA-256},k_1}(\texttt{\^{}S}) \equiv 21 \pmod{35} && \text{HMAC}_{\text{SHA-256},k_2}(\texttt{\^{}S}) \equiv 10 \pmod{35} \\
&\text{HMAC}_{\text{SHA-256},k_1}(\texttt{SM}) \equiv 23 \pmod{35} && \text{HMAC}_{\text{SHA-256},k_2}(\texttt{SM}) \equiv 2 \pmod{35} \\
&\text{HMAC}_{\text{SHA-256},k_1}(\texttt{MI}) \equiv 25 \pmod{35} && \text{HMAC}_{\text{SHA-256},k_2}(\texttt{MI}) \equiv 26 \pmod{35} \\
&\text{HMAC}_{\text{SHA-256},k_1}(\texttt{IT}) \equiv 29 \pmod{35} && \text{HMAC}_{\text{SHA-256},k_2}(\texttt{IT}) \equiv 14 \pmod{35} \\
&\text{HMAC}_{\text{SHA-256},k_1}(\texttt{TH}) \equiv 4 \pmod{35} && \text{HMAC}_{\text{SHA-256},k_2}(\texttt{TH}) \equiv 19 \pmod{35} \\
&\text{HMAC}_{\text{SHA-256},k_1}(\texttt{H\$}) \equiv 12 \pmod{35} && \text{HMAC}_{\text{SHA-256},k_2}(\texttt{H\$}) \equiv 31 \pmod{35}
\end{aligned}$$

Linear combinations of these values are taken to set $k = 3$ bits per bigram. For the case of $\texttt{\^{}S}$, this leads to the following values.

$$\begin{aligned}
&\text{HMAC}_{\text{SHA-256},k_1}(\texttt{\^{}S}) + 0 \cdot \text{HMAC}_{\text{SHA-256},k_2}(\texttt{\^{}S}) \equiv 21 + 0 \cdot 10 \equiv 21 \pmod{35} \\
&\text{HMAC}_{\text{SHA-256},k_1}(\texttt{\^{}S}) + 1 \cdot \text{HMAC}_{\text{SHA-256},k_2}(\texttt{\^{}S}) \equiv 21 + 1 \cdot 10 \equiv 31 \pmod{35} \\
&\text{HMAC}_{\text{SHA-256},k_1}(\texttt{\^{}S}) + 2 \cdot \text{HMAC}_{\text{SHA-256},k_2}(\texttt{\^{}S}) \equiv 21 + 2 \cdot 10 \equiv 6 \pmod{35}
\end{aligned}$$

Therefore after inserting $\texttt{\^{}S}$, bits 6, 21, and 31 will all be set. Continuing this logic for each bigram will result in the contents of the filter shown in Figure 1.

5 Attack

Assuming we have access to the complete encoding information, the following attack is possible. Complete information in this case would include knowledge of any pre–processing steps, and the setup of the Bloom Filter including HMAC keys if applicable. From this information, a reasonable alphabet, Σ, can be inferred.

The attack proceeds on a single record in four phases:

1. Determine the potential n-grams in the Bloom Filter by brute force.
2. Construct a directed graph from the potential n-grams.
3. Traverse the graph noting the walks traveled.
4. Filter the paths.

The first phase of determining the potential n-grams is done by simple brute force. With the sentinels, there will be $(|\Sigma| + 1)^n - 1$ distinct n-grams. Each n-gram is tested for membership in the filter. Despite this being brute force, it is still reasonable to perform this step because n needs to be small to allow for approximate matching in record linkage, and the cryptographic hashes used can be quickly computed.

Next, a directed graph is constructed. The vertices are the n-grams found with two additional vertices labeled `source` and `sink`. There is a directed edge from u to v if the last $n-1$ characters of u's label are the same as the first $n-1$ characters of v's label. There is an edge from `source` to v if the first $n-1$ characters of v's label are all start sentinels. There is an edge from u to `sink` if the last $n-1$ characters of u's label are all stop sentinels.

Then, all walks from `source` to `sink` are considered. Such a walk represents a word that could have been inserted in the filter. Walks can be easily found by variants of a depth first search looking for either all simple paths, all edge–disjoint walks, or all walks of a maximum length.

Finally, filters can be applied. For example, if the Bloom filter is known to encode only a single word, then a potential word can be encoded into a new filter and compared to the original. Only if they match exactly could the word have been the actual word inserted into the bloom filter. Additional filters can be devised based upon the known contents of the bloom filter, such as those based on expected word length, or even semantic filters based upon dictionaries or other sources.

5.1 Attack Example

Suppose that we have a filter with $m = 200$, $k = 6$, using two HMAC based hashes using SHA-256 whose keys are $11\ldots1$ and $22\ldots2$, respectively, where both keys are 64 hexadecimal digits in length.

After inserting "`WILLIAM`", we are left with the filter shown in Figure 2.

Using an alphabet of "`ABCDEFGHIJKLMNOPQRSTUVWXYZ`", we test each of the $(26 + 1)^2 - 1 = 728$ possible bigrams, including those using start and stop sentinels. In this case we find {`^W`, `AM`, `EC`, `IA`, `IL`, `JQ`, `LI`, `LL`, `M$`, `WI`} are present. Note that there are two bigrams extracted that do not derive from `WILLIAM`, `EC` and `JQ`. From these bigrams we derive the directed graph shown in Figure 3.

Note how the vertices `EC` and `JQ`, which were the result of false positives from the Bloom filter, do not meaningfully contribute to the walks formed from the graph. They are just isolated vertices. Only if they shared enough of their label with another vertex could they potentially contribute.

The words derived from the set of all simple paths in this graph are `WIAM`, `WILIAM`, and `WILLIAM`. If we have the additional knowledge that only one word was inserted, we can filter these paths by requiring the bigrams from a walk correspond to the exact bit pattern of the Bloom filter being attacked. After such a filter only `WILLIAM` remains.

The words derived from the set of all edge–disjoint walks in this graph are `WIAM`, `WILIAM`, `WILILLIAM`, `WILILLLIAM`, `WILLIAM`, `WILLILIAM`, `WILLLIAM`, and `WILLLILIAM`. Applying the same exact match filter removes `WIAM` and `WILIAM` since they do not have the `LL` bigram from the original word.

Observe that in both cases, the correct word was recovered from the filter, and the set of possibilities is relatively small. After filtering, fewer possibilities remain, with the correct word still present.

5.2 Computational Cost of Attack

To conduct this attack there are two distinct phases of the attack. First, the n–grams present are recovered. This is done in $\Theta(|\Sigma|^2)$ time. Second, the graph is constructed and walks on the graph are considered. The traversal time, as for all depth first searches, is $\Theta(|V| + |E|)$. Practically this is proportional to the length of the word in the filter.

In contrast, the constraint satisfaction attack, described in [12], works by creating an instance of the Constraint Satisfaction Problem. This is a well-known NP-Complete problem, which at the present means that attack has exponential complexity.

6 Limitations of Attack

This attack has two significant limitations. First, it relies upon complete knowledge of how a string is encoded in the Bloom Filter. This is a strong assumption, but is still realistic for some scenarios. See §9.2 for further details. Second, the structure of the word inserted greatly impacts its recoverability. In the preceding example of `WILLIAM`, we had words that were recovered that could have been formed from the n–grams, but they were not correct.

This occurs when words have duplicate $(n-1)$–grams in them. This leads to extra edges that are not necessarily present in the word. In addition if there are duplicate n–grams, despite being present multiple times in a word, they only contribute to bits being turned on once in the Bloom Filter. A word exhibiting both of these situations is `MISSISSIPPI`. Consider Figures 4 and 5 that show the structure of `MISSISSIPPI` when split up as bigrams and trigrams. It should be obvious from the figures that there are many walks from `source` to `sink`. However, there are fewer in the case of trigrams.

7 Moving from n–grams to $(n+1)$–grams

Moving to longer n–grams does not make this attack any more difficult. To the contrary, it actually makes it easier. The core idea is that the graph becomes more constrained. We start by demonstrating with two examples why increasing to longer n–grams does not help, `WILLIAM` and `MISSISSIPPI`, followed by proofs of the increase in effectiveness.

For the word WILLIAM, first consider the graph derived from it as bigrams. This is shown in Figure 6. By observation, the (source, sink)–walks correspond to the words WIAM, WILIAM, WILLIAM, WILLLIAM, ... Alternatively, these walks can be described by the regular expression WI(LL*I)*AM. Now consider the graph derived from WILLIAM as trigrams. This is shown in Figure 7. It has only one (source, sink)–walk, WILLIAM. Clearly this is many fewer walks, and in both cases the correct word is among the possible choices.

For the word MISSISSIPPI, the situation is a bit more complicated. In the bigram case, figure 4, there are the words MI, MISI, MIPI, MISISI, MISIPI, MIPISI, MIPIPI, ... = MI((S+|P+)I)*. In the trigram case, figure 5, there are the words MISSIPPI, MISSISSIPPI, MISSISSISSIPPI, ... = MIS(SIS)*SIPPI.

Observe that in both cases, every walk in the trigram graph has a corresponding walk in the bigram graph, but the converse is not necessarily true.

7.1 Theoretical Proof for increasing attack effectiveness by going to larger n grams

We will show that moving from n–grams to $(n+1)$–grams, does not weaken this attack, and in fact strengthens it.

Definition 7.1. A *phantom vertex* is a vertex in the derived graph that does not correspond to a n–gram that was inserted into the filter.

Phantom vertices arise from a query that is a false positive in the Bloom Filter. For example, the vertices EC and JQ from Figure 3 are phantom vertices.

Definition 7.2. For a word w, the *graph derived from w as n–grams* is the graph created by splitting up w into n–grams as described in §4 and creating a graph from it according to the rules described in §5.

Note that this definition does not consider phantom vertices because those only occur when considering the elements derived from a Bloom Filter. The graphs of Figures 4 and 5 are both examples of graphs derived from a word. The structure of a graph derived from a word indicates the relative difficulty of recovering the word from the set of n–grams.

Definition 7.3. For a word w, the *graph derived from a Bloom Filter containing w as n–grams* is the graph created by recovering all n–grams present in the Bloom Filter and created a graph according to the rules described in §5.

This second definition is broader because it allows for phantom vertices to be in the graph. Observe that trivially, the graph derived from w as n–grams is always a subgraph of the graph derived from a Bloom Filter containing w as n–grams. Therefore results on the difficulty of the attack on the graph derived from w as n–grams are relevant to attacking the Bloom Filter. A high false positive query rate to the Bloom Filter would clearly degrade record linkage performance, so in practice phantom vertices are not a large concern.

For the following theorems, it is important to keep the following in mind. If a word has a repeated n–gram, it also has at least two repeated $(n-1)$–grams. Similarly there are repeated $(n-2)$–grams as so forth. The intuition here is that as n increases, the repeated n–grams decrease and vice versa.

Definition 7.4. In a graph derived from a word as n–grams, each $(\texttt{source}, \texttt{sink})$–walk has a *corresponding word* created from the path by taking the first character of each vertex label and then removing the start and stop sentinel characters.

For example, the walk (`source`, `^W`, `WI`, `IL`, `LL`, `LI`, `IA`, `AM`, `M$`, `sink`) has `WILLIAM` as a corresponding word.

Theorem 7.5. For a word w, the graph derived from w as n–grams will contain a $(\texttt{source}, \texttt{sink})$–walk that has w as a corresponding word.

Proof. Let G be the graph derived from the word w as n–grams. By contradiction, suppose that G does not contain a $(\texttt{source}, \texttt{sink})$–walk with corresponding word w. Then there exists at least one edge required for the walk that is not present in G. Call it e. By construction, the label of e must be a sequence of n consecutive letters in w, with possible sentinels on both ends of the label. All such labels would have been constructed when creating G, thus the edge must be present in G. Therefore all edges necessary for a $(\texttt{source}, \texttt{sink})$–walk with corresponding word w will be present, and thus the walk will exist. □

Theorem 7.6. If a word w has at most a repeated $(n-2)$–gram, then there exists only one $(\texttt{source}, \texttt{sink})$–walk in the graph derived from w as n–grams.

Proof. By contradiction, suppose that there exist two distinct $(\texttt{source}, \texttt{sink})$–walks in the graph, $\mathcal{W}_1$ and $\mathcal{W}_2$. Since they both start at the `source` vertex, there must be a vertex, t, where the two walks diverge. For the walks to diverge at t, the out–neighborhood of t must contain at least two vertices, u and v. By construction, the last $n-1$ characters of t's label must match the first $n-1$ characters of u's label and v's label. The labels on u and v must be distinct for them to be distinct vertices. This is a contradiction because u and v's labels imply a repeated $(n-1)$–gram. □

Definition 7.7. A walk $\mathcal{W}_1$ is a *shortcut* of a walk $\mathcal{W}_2$ if $\mathcal{W}_1$ and $\mathcal{W}_2$ share the same source and destination, the length of $\mathcal{W}_2$ is longer than the length of $\mathcal{W}_1$, every vertex in $\mathcal{W}_1$ is also a vertex in $\mathcal{W}_2$, and every edge less one in $\mathcal{W}_1$ is present in $\mathcal{W}_2$.

For example, in Figure 3, the walk (`source`, `^W`, `WI`, `IA`, `AM`, `M$`, `sink`) is a shortcut of the walk (`source`, `^W`, `WI`, `IL`, `LL`, `LI`, `IA`, `AM`, `M$`, `sink`) because the $(\texttt{WI}, \texttt{IA})$ edge is replaced by the walk $(\texttt{WI}, \texttt{IL}, \texttt{LL}, \texttt{LI}, \texttt{IA})$.

Theorem 7.8. If a word w has a repeated $(n-1)$–gram, then there exist two walks in the graph derived from w as n–grams such that one is a shortcut of the other.

Proof. Let the repeated $(n-1)$–gram start at the i^{th} and j^{th} characters of w. Consider the walk $\mathcal{W}$ with corresponding word w. By Theorem 7.5, it is a walk present in the graph derived from w as n–grams. Along the walk, there must be a vertex labeled with a suffix starting at character i. Let the first such encountered vertex along the walk be denoted u. Similarly there must be a vertex labeled with the prefix starting at character j. Let the last such encountered vertex along the walk be denoted v. In $\mathcal{W}$, there must be vertices between the first occurrence of u and the last occurrence of v, else the $(n-1)$–gram would not have been repeated. The graph construction also puts a (u, v) edge in the graph. Therefore there exists a walk $\mathcal{W}'$ that is a shortcut of $\mathcal{W}$ that uses the (u, v) edge to shortcut between the first occurrence of u and the last occurrence of v. □

Theorem 7.9. If a word w has a repeated n–gram, then the graph derived from w as n–grams will have a cycle.

Proof. Consider the walk $\mathcal{W}$ of the word w. By Theorem 7.5, it is a walk present in the graph derived from w as n–grams. Let u be the vertex labeled with a repeated n–gram. Along $\mathcal{W}$, the vertex u must appear at least twice since the n–gram is repeated. Therefore there is a walk starting at u that also ends at u. This walk either is a cycle, or contains a cycle. Thus the graph derived from w as n–grams contains a cycle. □

Theorem 7.10. The graph derived from w as n–grams will have at least as many $(\texttt{source}, \texttt{sink})$–walks as the number of $(\texttt{source}, \texttt{sink})$–walks in the graph derived from w as $(n+1)$–grams.

Proof. Let the graph derived from w as n–grams be G_1 and the graph derived from w as $(n+1)$–grams be G_2. Consider a $(\texttt{source}, \texttt{sink})$–walk $\mathcal{W}$ in G_2. It has a corresponding word x. Let the walk created by considering x split up as n–grams be $\mathcal{W}'$. Each vertex encountered along $\mathcal{W}'$ must also be present in G_1 since the n–grams are portions of $(n+1)$–grams derived from w, and thus are n–grams derived from w. By the construction of edges in G_1, each edge present in $\mathcal{W}'$ must also be present in G_1. Therefore the walk $\mathcal{W}'$ is present in the graph G_1. Hence each $(\texttt{source}, \texttt{sink})$–walk in G_2 corresponds to a $(\texttt{source}, \texttt{sink})$–walk in G_1. □

Corollary 7.11. Going from n–grams to $(n+1)$–grams will never increase and possibly decrease the number of candidate words derived from the graph.

Proof. Theorem 7.10 directly shows that the number of candidate words, which are necessarily $(\texttt{source}, \texttt{sink})$–walks, never increases. Theorems 7.6, 7.8 and 7.9 show how the $(\texttt{source}, \texttt{sink})$–walks become more constrained as n increases, leading to fewer distinct walks in the graph. However, by Theorem 7.5, the actual word we wish to extract will always be present. □

The overall intuition one should gain from the preceding theorems is that by increasing the length of the n–grams, the effectiveness of this attack will only increase because the constraints put upon the derived graph increases.

8 Empirical Results

The attack presented is very practical. We have implemented the attack using the libraries Crypto++[1], GMP[5], and a custom Bloom Filter in C++. It can reasonably run on an Intel i7-4810MQ processor in a laptop with the attack running on a single BFE record in under 50ms on average for the bigram case, which includes the time to calculate all of the bigram hashes. In all of the following trials, the words were encoded into 1000–bit Bloom filters with bigrams using 30 hashes computed as linear combinations from SHA-256 hashes with HMAC keys of all 1s and all 2s. Additionally a filter was applied to the recovered words to ensure that when coded, the resulting Bloom filter matched the original.

For comparison with other attacks[11, 15], we ensured a minimum of 10,000 records for each data source to allow reasonable comparison of our attack to other attacks.

8.1 Name–Like Data

Using the North Carolina Voter Registration database, we derived name–like data by taking the first names and last names, normalizing the case, stripping the non-alphabetic characters and finally removing duplicates. This resulted in 474,319 distinct words.

When considering only simple paths, 364,450 of the words resulted in only one guess which was always correct, that is 76.8% of the words. The mean number of guesses was 1.32. The correct word was among the set of guesses for 442,619 of the words or 93.3%.

When considering edge disjoint paths, 285,454 resulted in only one guess which was always correct, that is 60.1% of the words. The mean number of guesses was 23.34, but for words of length less than 15, which is 465,94 or 98.2% of the words, the mean number of guesses was just 4.71. The correct word was among the set of guesses for 471,522 of the words or 99.4%.

8.2 Random Alphabetic Strings

A list of 10,000 words of length ten, where each letter was uniformly drawn from the standard 26 character alphabet was generated.

When considering only simple paths, 7,974 resulted in only one guess which was correct in 7,887 of those cases or 98.9% of the words. The mean number of guesses was 1.21. The correct word was among the set of guesses for 9,625 or 96.25% of the words.

When considering edge disjoint paths, 6,013 resulted in only one guess which was correct in 6,009 of those cases or 99.9% of the words. The mean number of guesses was 2.47. The correct word was among the set of guesses for 9,989 of the words or 99.8%.

8.3 Random Numeric Strings

A list of 10,000 words of length nine, where each letter was uniformly drawn from the ten digits was generated. This simulates US Social Security Numbers.

When considering only simple paths, 3,121 resulted in only one guess which was correct in 2,663 cases or 85.3% of the words. The mean number of guesses was 2.14. The correct word was among the set of guesses for 7,716 of the words or 77.1%.

When considering edge disjoint paths, 47 resulted in only one guess which was correct in every case. The mean number of guesses was 52.1, but when a word results in over 1,000 guesses are tossed out, leaving 9,951 words, the mean number of guesses was 18.7. The correct word was among the set of guesses for 9,823 of the words or 98.2%.

The relative difficulty in recovering these numeric words appears to be due to the word length relative to the alphabet size. In particular as the word length approaches the alphabet size, the likelihood of repeated bigrams and trigrams increases. As seen in the preceding sections this results in a more complicated graph. For example, in this specific case the word 486884464 resulted in 196,076 guesses which as one word alone increased the mean number of guesses by 19.6 for the whole set of words.

9 Discussion

9.1 Medical Data Aggregation

The medical community is having to integrate data from disparate sources to conduct their research, particularly longitudinal studies. This integration requires data sharing mechanisms that address both record linkage quality and adherence to regulatory frameworks such as HIPPA and the EU's Data Protection Directive. One of the key challenges to this is to perform the integration while preserving privacy. Many of the methods based upon various cryptographic primitives do not scale to realistic data sizes. Recent work, such as [13], leave practical implementation for future work. In SAFTINet[18] and other similar frameworks, the problem of linkage is performed very frequently.

While blocking and similar implementation techniques have proven useful[16], the problem of PPRL on large data sets remains open. To compound the technical difficulty, governmental and institutional demands make medical data aggregation a challenging area.

9.2 Implications of Attack

One essential requirement of our attack is complete knowledge of the parameters used for the Bloom Filter. Some of these parameters should be treated as public information. For instance the parameters m and k can be determined exactly, or with very high confidence, respectively, by simple inspection of the data. The choice of the hash functions can not be as easily determined. Following

Kerckhoff's principle, the choice of the hash function should be public, but any keys used should remain private. Practically this means using some kind of keyed hash, such as HMAC, where the keys used are kept private.

If the keys are known to the attacker, our attack shows that the system can be broken by any honest, but curious, attacker that obtains the data. That is any attacker that obtains BFE encoded data and has knowledge of all the parameters used in the Bloom Filter can decode a significant portion of the data without any other party being able to detect this.

A situation where this will always be possible is any two-party system. Both parties will by necessity have access to all the information necessary to perform our attack. Therefore no two party system can use BFE and ensure privacy under even the relatively weak assumption of an honest-but-curious attacker.

This also means that any protocol built upon BFE must have the party that performs the record linkage unable to access the particular Bloom Filter parameters used. It also means that any party that contributes data to the protocol cannot be allowed to see the data contributed by another party because they could perform this attack.

9.3 Potential Counter–measures

We are aware of a couple simple counter–measures that make this attack a little harder to perform.

First, putting multiple words into a single Bloom filter. Having more than one word increases the likelihood that the walks from two or more words will intersect. This makes the path traversal harder because there will then be more guesses for the correct words. However this will only work if the alphabets for the two words is the same. If they are different, the resulting graph will be completely disjoint except for the edges from source to sink. This may have implications to record linkage because it would force the mixing of individual attributes that without mixing could be individually weighted. This is the approach explored in [3], but consideration towards the alphabets used was not one of their criteria. Additionally their suggestions include adaptively sizing filters on a per item basis and compositing the result. If the adaptive sizes of two items match and they are composited at the same offset, then the net effect is putting two items into the same filter which increases the difficulty as noted. Otherwise, the only effect on this attack is potentially adding phantom vertices to the graph.

Second, the start and stop sentinel values provide a clear beginning and end for all the graph traversals. If the sentinels were removed, then any of the recovered n–grams could plausibly be the first n–gram in the recovered word. Again, this may have undesired implications to record linkage since the characters that make up the beginning and end of the words will be under represented in the n–grams in the filter. However it would create many more traversals which could mask the true contents of the filter, but the true contents would still be in the set of traversals.

10 Conclusion

We presented a novel cryptanalytic technique based on graph traversals to show that the Bloom filter encoding does not preserve privacy in a two-party linkage setting. We proved that it is a practical attack by running experiments on large name-like datasets derived from the North Carolina voter database. Recently several other researchers have pointed out weaknesses of this encoding by giving statistical attacks. Fundamentally, this encoding, viewed as cipher text, lacks diffusion [20], i.e. small changes to plain text *do not* result in large changes to cipher text. Therefore, it is not surprising that this encoding is susceptible to statistical attacks. On the other hand, any encoding with high diffusion is unsuitable for record linkage because *approximate* matching cannot be performed on encoded text. A conundrum! Therefore, a radically different approach is warranted to perform record linkage, one that can give high linkage accuracy, handle large datasets, at the same time preserve privacy. Commutative encryption appears to hold some promise for solving PPRL. Some recent results [2] suggest that, what always was deemed to be computationaly too expensive, efficient implementation are indeed possible for realistic size data sets.

References

[1] Wei Dai. Crypto++ Library 5.6.2 Available: http://www.cryptopp.com/, 2013.

[2] Rinku Dewri, Toan Ong, and Ramakrishna Thurimella. Linking health records for federated query processing. *Proceedings on Privacy Enhancing Technologies*, 2016(3):4–23, 2016.

[3] Elizabeth Durham, Murat Kantarcioglu, Yuan Xue, Csaba Toth, Mehmet Kuzu, and Bradley Malin. Composite bloom filters for secure record linkage. *Knowledge and Data Engineering, IEEE Transactions on*, 26(12):2956–2968, 2014.

[4] Elizabeth Ashley Durham, Yuan Xue, Murat Kantarcioglu, and Bradley Malin. Private medical record linkage with approximate matching. In *AMIA Annual Symposium Proceedings*, volume 2010, page 182. American Medical Informatics Association, 2010.

[5] Torbjrn Granlund et al. GNU multiple precision arithmetic library 6.0.0a, March 2014. `https://gmplib.org/`.

[6] Michael J Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *Advances in Cryptology - EUROCRYPT 2004*, Lecture Notes in Computer Science, pages 1–19. Springer Berlin Heidelberg, 2 May 2004.

[7] Yuriy Kaniovskyi, Siegfried Benkner, Chris Borckholder, Steven Wood, Piotr Nowakowski, Alfredo Saglimbeni, and Tomas Pariente Lobo. A semantic cloud infrastructure for data-intensive medical research. *International Journal of Big Data Intelligence*, 2(2):91–105, 2015.

[8] Adam Kirsch and Michael Mitzenmacher. Less hashing, same performance: Building a better bloom filter. In Yossi Azar and Thomas Erlebach, editors, *ESA*, volume 4168 of *Lecture Notes in Computer Science*, pages 456–467. Springer, 2006.

[9] Lea Kissner and Dawn Song. Privacy-Preserving set operations. In *Advances in Cryptology – CRYPTO 2005*, Lecture Notes in Computer Science, pages 241–257. Springer Berlin Heidelberg, August 2005.

[10] Martin Kroll and Simone Steinmetzer. Automated cryptanalysis of bloom filter encryptions of health records. *arXiv preprint arXiv:1410.6739*, 2014.

[11] Mehmet Kuzu, Murat Kantarcioglu, Elizabeth Durham, Csaba Toth, and Bradley Malin. A practical approach to achieve private medical record linkage in light of public resources. *Journal of the American Medical Informatics Association*, 20(2):285292, July 2013.

[12] Mehmet Kuzu, Murat Kantarcioglu, Elizabeth Ashley Durham, and Bradley Malin. A constraint satisfaction cryptanalysis of bloom filters in private record linkage. In S. Fischer-Hübner and N. Hopper, editors, *Privacy Enhancing Technologies*, volume 6794 of *LNCS*, page 226245. Springer, 2011.

[13] Ibrahim Lazrig, Tarik Moataz, Indrajit Ray, Indrakshi Ray, Toan Ong, Michael Kahn, Frdric Cuppens, and Nora Cuppens. Privacy preserving record matching using automated semi-trusted broker. In *Data and Applications Security and Privacy XXIX*, volume 4168, pages 103–118. Springer International Publishing, June 2015.

[14] National Institute of Standards and Technology. FIPS 198-1, The Keyed-Hash Message Authentication Code, Federal Information Processing Standard (FIPS), Publication 198-1. Technical report, Department of Commerce, July 2008.

[15] Frank Niedermeyer, Simone Steinmetzer, Martin Kroll, and Rainer Schnell. Cryptanalysis of basic bloom filters used for privacy preserving record linkage. *Journal of Privacy and Confidentiality*, 6(2):59–79, 2014.

[16] Thilina Ranbaduge, Dinusha Vatsalan, Peter Christen, and Vassilios Verykios. Hashing-Based distributed multi-party blocking for Privacy-Preserving record linkage. In *Advances in Knowledge Discovery and Data Mining*, Lecture Notes in Computer Science, pages 415–427. Springer International Publishing, 19 April 2016.

[17] Sean M. Randall, Adrian P. Brown, Anna M. Ferrante, James H. Boyd, and James B. Semmens. Privacy preserving record linkage using homomorphic encryption. In *First International Workshop on Population Informatics for Big Data*, 2015.

[18] Lisa M Schilling, Bethany M Kwan, Charles T Drolshagen, Patrick W Hosokawa, Elias Brandt, Wilson D Pace, Christopher Uhrich, Michael Kamerick, Aidan Bunting, Philip R O Payne, William E Stephens, Joseph M George, Mark Vance, Kelli Giacomini, Jason Braddy, Mika K Green, and Michael G Kahn. Scalable architecture for federated translational inquiries network (SAFTINet) technology infrastructure for a distributed data network. *EGEMS (Washington, DC)*, 1(1):1027, 7 October 2013.

[19] Rainer Schnell, Tobias Bachteler, and Jörg Reiher. Privacy-preserving record linkage using Bloom filters. *BMC medical informatics and decision making*, 9:41, January 2009.

[20] Claude E. Shannon. Communication Theory of Secrecy Systems. *Bell System Technical Journal*, 28(4):656715, 1949.

[21] Kenn Slagter, Ching-Hsien Hsu, Yeh-Ching Chung, and Gangman Yi. SmartJoin: a network-aware multiway join for MapReduce. *Cluster computing*, 17(3):629–641, 19 February 2014.

[22] Dinusha Vatsalan, Peter Christen, and Vassilios S Verykios. A taxonomy of privacy-preserving record linkage techniques. *Information Systems*, 38(6):946–969, 2013.

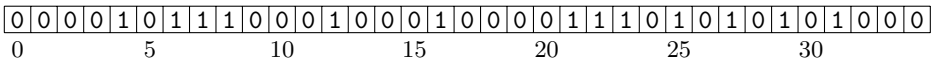

Figure 1: Contents of Bloom Filter after inserting `^S`, `SM`, `MI`, `IT`, `TH`, `H$`.

```
9 0469 0480 0E0B 2002 2102 8041 4080 02D0 1200 258A 4024 1000
```

Figure 2: Contents of Bloom Filter in hexadecimal after inserting `WILLIAM`.

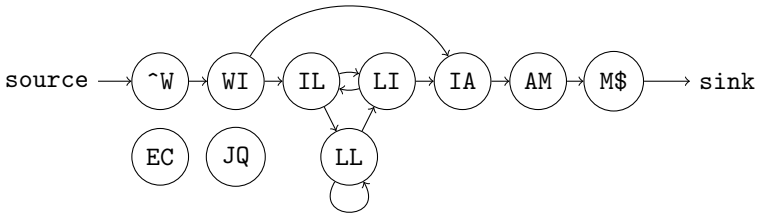

Figure 3: Graph derived from {`^W`, `AM`, `EC`, `IA`, `IL`, `JQ`, `LI`, `LL`, `M$`, `WI`}.

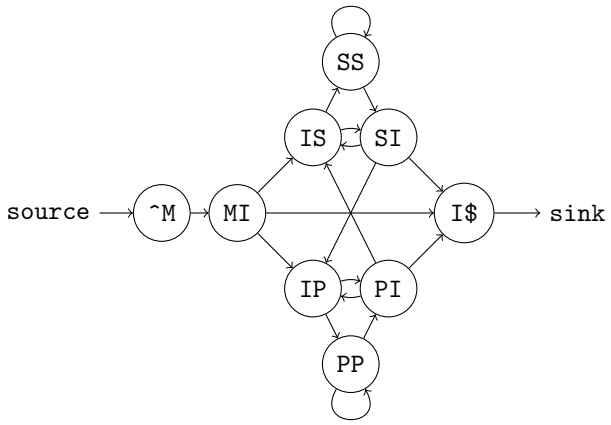

Figure 4: Graph derived from `MISSISSIPPI` as bigrams.

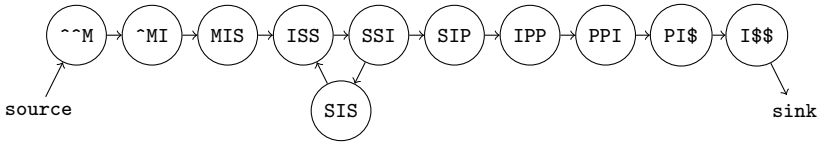

Figure 5: Graph derived from `MISSISSIPPI` as trigrams.

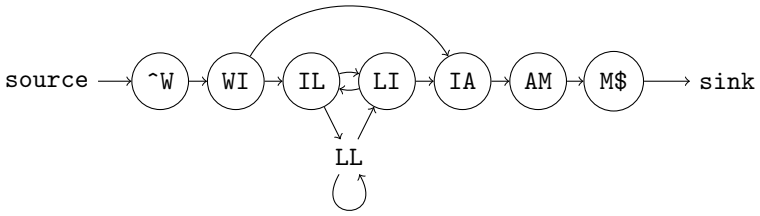

Figure 6: Graph derived from `WILLIAM` as bigrams.

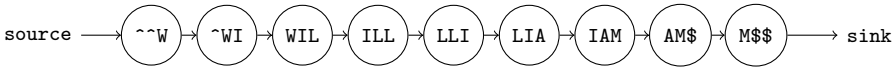

Figure 7: Graph derived from `WILLIAM` as trigrams.