NEPAL: Climbing Beyond the Limits of Graph Matching Attacks in Privacy-Preserving Record Linkage

Marcel Mildenberger[0009−0005−9362−2005] ✉, Jochen Schäfer[0009−0008−4396−6186], and Frederik Armknecht[0009−0003−9935−8095]

University of Mannheim, Mannheim, Germany
{marcel.mildenberger, jochen.schaefer, armknecht}@uni-mannheim.de

Abstract. Privacy-Preserving Record Linkage (PPRL) allows matching data from different sources without revealing sensitive identifiers. In practice, most PPRL schemes apply similarity-preserving encodings to plaintext records to obfuscate the underlying identifiers while allowing the linkage procedure to be outsourced to third parties.
Currently, the most effective class of attacks against PPRL are Graph Matching Attacks (GMAs). They assume that an attacker has access to encoded records and a separate database of plaintext records (e.g., a phone book). Although most practical PPRL schemes are vulnerable to GMAs, these attacks are inherently limited, as they can only re-identify encoded records that are also present in the plaintext database.
In this paper, we overcome this limitation by introducing the Neural Pattern Learning (NEPAL) attack. At its core, NEPAL is a Pattern Mining Attack that learns characteristic patterns from encoding–plaintext pairs and utilizes them to (partially) recover the plaintexts of other encoded records.
While the attack functions independently, it is particularly powerful in the GMA+NEPAL setting. There, GMA derived matches serve as training data, allowing NEPAL to extend the attack beyond the database overlap. For instance, using the *Euro Person* dataset, we demonstrate that even if only 20% of the encoded records appear in the plaintext database, the combination of GMA+NEPAL enables the near-complete recovery of all encoded records. Consequently, through NEPAL records outside the original overlap are also at risk.

Keywords: Privacy-preserving record linkage, de-anonymization, neural pattern learning attack, pattern mining attack

1 Introduction

The integration of data across institutional boundaries is important in domains such as healthcare, finance, and social sciences, where comprehensive insights rely on linking records that refer to the same individual. The overall goal of a record linkage scheme is to identify tuples of records from two or more databases

that refer to the same individual. In its simplest form, record linkage relies on unique identifiers, such as social security numbers, shared across all databases. In practice, however, such universal unique identifiers are often unavailable. Consequently, Personally Identifiable Information (PII) such as names, addresses, and birthdates serve as quasi-identifiers: If the quasi-identifiers of two records are sufficiently similar, the records are considered a match and their data is merged [5, 20].

However, calculating the similarities between plaintext PII poses a serious privacy risk, as it inevitably reveals the presence of a particular individual to the entity performing the similarity calculation. Depending on the nature of the database, this information itself can be highly sensitive, as is the case for databases of prison inmates or hospital patients [5]. To address this, Privacy-Preserving Record Linkage (PPRL) schemes were developed to enable record linkage without revealing plaintext PII. While this is achievable through cryptographic protocols such as Secure Multi-Party Computation (SMC), these protocols incur significant computational and communication overhead, rendering them impractical for large-scale linkage tasks involving millions of records [20, 8]. Therefore, the prevalent approach in practice is *non-interactive*. More precisely, these schemes transform quasi-identifiers using a similarity-preserving encoding scheme. The resulting encoded records are outsourced to a third party, the Linkage Unit (LU), which determines matches based solely on encoding similarity.

PPRL is actively used in public administration and health research: For instance, the *Lumos* project in the Australian state of New South Wales utilized Bloom Filter (BF) based PPRL to link the medical records of over 1.3 million patients across primary care and hospital systems [7]. Similarly, a large-scale initiative in Brazil, called the *100 Million Cohort*, employed similar BF-based techniques to merge administrative records of 114 million individuals to study the effects of social policies [14]. In the United States, distributed linkage systems have been deployed to integrate electronic health records across medical institutions in the Chicago area [10].

1.1 Security of PPRL

An encoding procedure for PPRL must satisfy two competing requirements.

On one hand, similar records should be transformed into similar encodings to preserve the structure of the underlying plaintext record. PPRL encoding schemes frequently represent quasi-identifiers as sets of q-grams, as this facilitates the use of common set similarity metrics such as the Dice coefficient[1] or symmetric difference to represent similarity between records [15, 18, 19]. Recall that q-grams are substrings of size q derived using a sliding window approach (e. g. , for $q = 2$, the bigram set of "ANNA" is {"AN", "NN", "NA"}).

On the other hand, the encodings must obfuscate the plaintexts to prevent re-identification. To evaluate this, a variety of attacks have been developed, with

[1] See Eq. (12) for a definition.

the two most relevant classes being Pattern Mining Attacks (PMAs) and Graph Matching Attacks (GMAs) [22, 16, 21].

In PMAs, an attacker aims to identify correlations between plaintext q-grams and the occurrence of specific patterns in the encodings. If suitable patterns are identified, the attacker can search for them within given encoded records, thereby inferring a set of q-grams likely present in the plaintext. As more q-grams are identified, the leakage of information regarding the plaintext record increases, potentially allowing for the reconstruction of the full q-gram set. A common example of a PMA is frequency analysis [22]. Crucially, the effectiveness of such attacks depends heavily on the identification of appropriate characteristic patterns.

In GMAs, an attacker typically possesses a plaintext database that overlaps with the target encoded database. That is, a subset of plaintexts known to the attacker exists in encoded form within the target database. Crucially, however, the attacker does not initially know which plaintexts correspond to which encodings. Inferring this mapping is the primary objective of a GMA [16, 21].

To achieve this, the attack exploits the fact that, for similarity-preserving encodings, the pairwise similarities of encoded records are strongly correlated with the pairwise similarities of their underlying plaintexts. Consequently, the attacker constructs *similarity graphs* for both the encoded and the plaintext datasets. In these graphs, nodes represent records and edges represent their pairwise similarities. The underlying idea is that if an encoded record and a plaintext record exhibit similar local subgraph structure, they refer to the same individual. Consequently, the attack can be formulated as a graph-alignment problem [16, 21].

Since GMAs exploit a fundamental property of similarity-preserving encodings, they are broadly applicable to various encoding schemes. However, GMAs suffer from a significant limitation: they can only re-identify records which are also represented in the plaintext database. On their own, GMAs provide no mechanism for re-identifying encodings outside the intersection of the two databases.

1.2 Contribution

In this work, we overcome the aforementioned shortcoming of GMAs by presenting a novel attack that enables the reconstruction of plaintexts from encodings for which GMAs fail to find a match.

We term this new attack Neural Pattern Learning (NEPAL), as it relies on Artificial Neural Networks (ANNs). By design, NEPAL is a form of PMA, as it aims to learn correlations between the occurrence of characteristic features in the encodings and the presence of certain q-grams in the underlying plaintexts. Our methodological contribution lies not in introducing a fundamentally new machine-learning framework, but in showing that the attack can be framed as a supervised-learning problem. To achieve this, we deliberately employ an established and well-suited technique, namely an Multi-Layer Perceptron (MLP), as the most straightforward way to operationalize and scale the analysis in practice. In contrast to prior PMAs, which rely on the manual analysis of encoding

schemes to derive appropriate patterns, NEPAL frames this task as a supervised machine learning problem. More precisely, NEPAL identifies characteristic patterns on its own by training a Multi-Layer Perceptron (MLP) on known encoding–plaintext pairs. To ensure generalizability, the network architecture and hyperparameters are automatically optimized for the specific target encoding. This makes NEPAL applicable to a variety of different encoding schemes without the need to adapt it manually to the encoding scheme. In particular, we successfully apply NEPAL against BFs, Two-Step Hashing (TSH), and Tabulation MinHash (TMH).

While NEPAL can be used as an attack on its own, it is designed to be executed in conjunction with a GMA. Specifically, a GMA first produces a set of re-identified encoding–plaintext pairs. Subsequently, NEPAL is used to infer plaintext q-grams for encodings for which the GMA did not yield a match. To this end, the encoding–plaintext pairs identified by the GMA serve as training data for an ANN to learn characteristic patterns in encoded records caused by the occurrence of specific q-grams in the plaintexts. Given these, the ANN predicts the plaintext q-grams in other encoded records. This extends the impact of GMAs beyond the overlap between plaintext and encoded databases and therefore puts even records outside the original overlap at risk. For example, when such pairs are available for 20% of the records in the *Euro Person* dataset, GMA+NEPAL can reconstruct all of the remaining 80% with Dice scores above 0.92. The slightly lower accuracy in this setting is expected, since the training pairs obtained from the GMA may contain mismatches. The fact that reconstruction nevertheless remains strong highlights the robustness of the approach.

In summary, the contributions of this work are threefold:

- We show that BF, TSH, and TMH encodings are in principle vulnerable to PMAs. To this end, we introduce a formal notion of *patterns*, demonstrate their existence in all three schemes, and use the associated likelihood ratio as a theoretical proxy for exploitability. Previously, only BFs were known to exhibit exploitable patterns [22].
- We introduce NEPAL, a novel attack belonging to the class of PMAs. NEPAL can be executed in conjunction with GMAs to compromise records unaffected by the GMA alone. Unlike traditional PMAs tailored to specific schemes, NEPAL is capable of learning patterns from encoding–plaintext pairs regardless of the specific scheme. Once learned, these patterns enable the attacker to predict q-grams for previously unseen encoded records.
- We evaluate the NEPAL attack in three settings: First, as a standalone attack in which correct encoding–plaintext pairs are given a priori. Second, as a standalone attack with deliberately introduced errors in these pairs. And third, in conjunction with a GMA, where encoding–plaintext pairs are inferred from the GMA output. All approaches are assessed across the three similarity-preserving encoding schemes susceptible to GMAs: Bloom Filters (BFs), Two-Step Hashing (TSH), and Tabulation MinHash (TMH). Note that the previously determined patterns mentioned above were not provided to NEPAL to validate that it can identify such patterns on its own.

The remainder of this paper is structured as follows: Section 2 introduces the notion of patterns and demonstrates their existence in the three encoding schemes analyzed in this work. While these findings may be of independent interest, they also serve as an exploitability analysis for the three encodings studied here. Section 3 presents the proposed NEPAL attack, detailing the attacker model, and outlining the methodology. Section 4 describes the experimental setup and empirically validates that these patterns can be learned and exploited in practice. Finally, Section 5 summarizes the key findings and outlines directions for future research toward more secure PPRL protocols.

2 Patterns in Existing Encoding Schemes

This section formally defines the concept of patterns and demonstrates their presence in the PPRL encoding schemes BF, TMH, and TSH. We use the Likelihood Ratio (LR) induced by a pattern as a theoretical proxy for how informative and therefore exploitable an encoded record is for PMAs. If a simple, efficiently computable pattern π yields $\mathrm{LR}(\pi) > 1$, then observing that pattern increases an attacker's confidence that a corresponding plaintext q-gram is present.

Since all three encoding schemes considered in this analysis rely on hash functions, we make the following standard assumptions regarding their behaviour. Each hash function $\boldsymbol{h}$ is sampled uniformly at random from a family of hash functions $\mathcal{H}$ and is modelled as an independent, uniform random oracle. Specifically, for any input r, the value $\boldsymbol{h}(r)$ is uniformly distributed over the output range of $\boldsymbol{h}$. Furthermore, the outputs of different hash functions, as well as the outputs of the same hash function on different inputs, are mutually independent.

In current PPRL schemes, plaintext records are represented as sets R of q-grams, where each q-gram is a sequence of q characters obtained from the plaintext identifiers via a sliding window approach. We denote the encoding of a record by $[R]$ and define the shorthand $[r] = [\{r\}]$ to refer to the encoding of a single q-gram r.

Formally, a pattern is an efficiently computable, deterministic function π that takes an encoded record $[R]$ as input and outputs either 0 or 1. Intuitively, a pattern for a target set $R^* \subseteq \mathcal{U}$ (where $\mathcal{U}$ denotes the universe of all possible q-grams) allows determining, for a given encoded record $[R]$, whether $R^* \subseteq R$ holds. The effectiveness of π is captured by the *Likelihood Ratio (LR)*:

$$\mathrm{LR}(\pi) = \frac{\Pr(\pi([R]) = 1 \mid R^* \subseteq R)}{\Pr(\pi([R]) = 1 \mid R^* \not\subseteq R)} \tag{1}$$

The likelihood ratio $\mathrm{LR}(\pi)$ quantifies how much more likely the pattern observation $\pi([R]) = 1$ is when $R^* \subseteq R$ compared to when $R^* \not\subseteq R$. By applying Bayes' theorem, this relationship can be equivalently expressed as the multiplicative factor between the prior and posterior odds:

$$\frac{\Pr(R^* \subseteq R \mid \pi([R]) = 1)}{\Pr(R^* \not\subseteq R \mid \pi([R]) = 1)} = \mathrm{LR}(\pi) \cdot \frac{\Pr(R^* \subseteq R)}{\Pr(R^* \not\subseteq R)} \tag{2}$$

A value of $\mathrm{LR}(\pi) > 1$ therefore indicates that observing the pattern increases the posterior odds of $R^* \subseteq R$ relative to the prior odds, implying a higher likelihood that $R^* \subseteq R$. The magnitude of $\mathrm{LR}(\pi)$ reflects the strength of the evidence provided by the pattern, while values close to one indicate that the observation yields negligible information gain. In the remainder of this section, we present simple patterns with $\mathrm{LR}(\pi) > 1$ for all three schemes. Later, the empirical evaluation confirms that such theoretically identifiable patterns translate into practical NEPAL success, with stronger $\mathrm{LR}(\pi)$ corresponding to more successful attacks.

2.1 Bloom Filters

Bloom Filters (BFs) [3] have been widely adopted in PPRL applications [18, 17, 13, 7, 14]. Research has demonstrated their vulnerability to PMAs. Notably, [22] showed that correlations between plaintext q-grams and their encoded representations can be systematically exploited by identifying characteristic bit patterns. These patterns reveal structural dependencies within the encoding, thereby enabling the reconstruction of original plaintext values. BFs encode records into bit vectors in $\{0,1\}^l$, using $\boldsymbol{k} \geq 1$ independent, secret hash functions $\mathcal{H} = \{\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_{\boldsymbol{k}}\}$ with $\boldsymbol{h}_i : \mathcal{U} \to \{1, \ldots, l\}$ The j-th bit of $[R]$ is set to 1 if and only if there exists at least one hash function $\boldsymbol{h}_t$ and one $r \in R$ such that $\boldsymbol{h}_t(r) = j$.

Let $r \in \mathcal{U}$ be fixed, and let $I_r = \{\boldsymbol{h}_i(r)\}_{i=1}^{\boldsymbol{k}}$ denote the set of bit positions produced by hashing r with all $\boldsymbol{k}$ hash functions. We define a pattern π as follows: given a bit vector $\boldsymbol{b} = (\boldsymbol{b}[1], \ldots, \boldsymbol{b}[l])$, the pattern outputs 1 if and only if $\boldsymbol{b}[i] = 1$ for all $i \in I_r$. By construction, $\Pr(\pi([R]) = 1 \mid r \in R) = 1$ holds. However, the pattern may also hold even when r is *not* present in the record, namely when other q-grams set all positions in I_r.

To obtain a simple lower bound, consider a fixed position $i \in I_r$ (corresponding to a pattern of size one). The probability that no hash function maps any $r^* \in R$ to position i is $\left(1 - \frac{1}{l}\right)^{\boldsymbol{k}|R|}$. Consequently, $1 - \left(1 - \frac{1}{l}\right)^{\boldsymbol{k}|R|}$ represents the probability that position i is set to 1 by *at least one* hash function of some $r^* \in R$. It follows that the likelihood ratio is bounded by:

$$\mathrm{LR}(\pi) \geq \frac{1}{1 - \left(1 - \frac{1}{l}\right)^{\boldsymbol{k}|R|}}. \tag{3}$$

For typical parameters such as $l = 1024$, $\boldsymbol{k} = 10$, and $|R| = 20$, we obtain a likelihood ratio of $\mathrm{LR}(\pi) \approx 5.63 > 1$. This indicates that observing the pattern is over five times more likely when the respective q-gram is present in the plaintext compared to when it is absent. This value constitutes a *lower bound* for the LR, as it reflects only a single observed bit position. Considering multiple matching positions would further increase the ratio.

2.2 Two-Step Hashing

As an attempt to mitigate the vulnerability to pattern mining described in Section 2.1, [15] proposed the following two-step approach for encoding a record R. In the first step, a $\boldsymbol{k} \times l$ bit matrix is created, where each row i corresponds to a BF encoding of R utilizing a *single* hash function $\boldsymbol{h}_i$:

$$\boldsymbol{B} = \begin{pmatrix} \boldsymbol{b}_1[1] & \dots & \boldsymbol{b}_1[l] \\ \vdots & \ddots & \vdots \\ \boldsymbol{b}_{\boldsymbol{k}}[1] & \dots & \boldsymbol{b}_{\boldsymbol{k}}[l] \end{pmatrix} = (\boldsymbol{B}[1], \dots, \boldsymbol{B}[l]) \tag{4}$$

In the second step, all non-zero columns $\boldsymbol{B}[j]$ of this matrix are hashed to integers using a unique hash function $\boldsymbol{H}_j$ for each column j, forming the final encoding.

We define a pattern π to test whether the intersection $[r] \cap [R]$ is sufficiently large, i. e. , whether a substantial number of integers from $[r]$ also appear in $[R]$. To establish a *lower bound* for $\mathrm{LR}(\pi)$, we consider a simple pattern consisting of exactly one integer. Specifically, for some $r \in \mathcal{U}$, we select one integer $c \in [r]$ and define $\pi([R]) = 1 \iff c \in [R]$. Thus, the pattern is observed when this integer is present in the encoding.

By construction, $c = \boldsymbol{H}_j(V)$ for some vector V of length $\boldsymbol{k}$. Assuming $\boldsymbol{H}_j$ is a cryptographically secure hash function, the probability of collisions is negligible. Hence, $c \in [R]$ if and only if $\boldsymbol{B}[j] = V$. Let $I_r = \{\boldsymbol{h}_i(r) \mid \boldsymbol{h}_i(r) = j\}_{i=1}^{\boldsymbol{k}}$ denote the indices of the rows in $\boldsymbol{B}[j]$ that are set by r. Then $\boldsymbol{B}[j] = V$ if and only if all other positions of $\boldsymbol{B}[j]$ (those not in I_r) are 0.

By construction, the positions of $\boldsymbol{B}[j]$ corresponding to I_r are always set to one whenever $r \in R$. Therefore, the probability that $\boldsymbol{B}[j] = V$ for record R is equivalent to the probability that all other bit positions in that column are zero:

$$\Pr\big(\pi([R]) = 1 \mid r \in R\big) = \left(1 - \frac{1}{l}\right)^{|R|(\boldsymbol{k} - |I_r|)} \tag{5}$$

Conversely, when $r \notin R$, the value $c \in [r]$ can only occur by coincidence: the positions in I_r must be set in column j, while all other entries of $\boldsymbol{B}[j]$ remain zero. The probability of a specific bit being set in column j by any q-gram in R is $1 - (1 - 1/l)^{|R|}$. Thus:

$$\Pr\big(\pi([R]) = 1 \mid r \notin R\big) = \left(1 - \left(1 - \frac{1}{l}\right)^{|R|}\right)^{|I_r|} \cdot \left(1 - \frac{1}{l}\right)^{|R|(\boldsymbol{k} - |I_r|)} \tag{6}$$

When calculating $\mathrm{LR}(\pi)$, the term from Eq. (5) cancels out. Since we are interested in a lower bound, we assume the minimal overlap $|I_r| = 1$, yielding the following inequality:

$$\mathrm{LR}(\pi) \geq \frac{1}{1 - \left(1 - \frac{1}{l}\right)^{|R|}} \tag{7}$$

For typical parameters such as $l = 1024$ and $|R| = 20$, we obtain a lower bound of $\mathrm{LR}(\pi) \approx 51.68 > 1$. Even this single-integer pattern therefore provides strong evidence of exploitability through PMAs, and using multiple integers would further increase the LR.

2.3 Tabulation MinHashing

Tabulation MinHash (TMH) [19] is an application of the MinHash principle [4] to the field of PPRL. As in the previously discussed approaches, TMH relies on $\boldsymbol{k}$ independent hash functions $\mathcal{H} = \{\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_{\boldsymbol{k}}\}$. Given these, we define:

$$\boldsymbol{h}_i^*(R) := \mathrm{LSB}(\min\{\boldsymbol{h}_i(r) \mid r \in R\}) \tag{8}$$

Here, $\mathrm{LSB}(\cdot)$ denotes the Least Significant Bit (LSB) of the hash value. The final encoding is defined as:

$$[R] = \left(\boldsymbol{h}_1^*(R), \ldots, \boldsymbol{h}_{\boldsymbol{k}}^*(R)\right) \tag{9}$$

For a q-gram r, the relevant pattern is whether the encodings $[R]$ and $[r]$ match in one or more positions.

We first consider the case $r \in R$. Let $\mathcal{E}_{\min}(r, R)$ denote the event that r yields the smallest hash value in R for a given hash function $\boldsymbol{h}_i$, i. e. , $\min\{\boldsymbol{h}_i(r^*) \mid r^* \in R\} = \boldsymbol{h}_i(r)$. If $\mathcal{E}_{\min}(r, R)$ occurs, then $\boldsymbol{h}_i^*(r) = \boldsymbol{h}_i^*(R)$ follows directly. Otherwise, $\min\{\boldsymbol{h}_i(r^*) \mid r^* \in R\}$ and $\boldsymbol{h}_i(r)$ are effectively independent, uniformly distributed bit strings (assuming a sufficiently large hash range), so equality of their LSBs occurs with probability 1/2. Applying the law of total probability, it follows that:

$$\begin{aligned} &\Pr(\boldsymbol{h}_i^*(r) = \boldsymbol{h}_i^*(R) \mid r \in R) \\ &= 1 \cdot \frac{1}{|R|} + \frac{1}{2} \cdot \left(1 - \frac{1}{|R|}\right) \\ &= \frac{1}{2} + \frac{1}{2\,|R|} \end{aligned} \tag{10}$$

Conversely, if $r \notin R$, then $\min\{\boldsymbol{h}_i(r^*) \mid r^* \in R\}$ and $\boldsymbol{h}_i(r)$ are independent. Consequently, $\boldsymbol{h}_i^*(r) = \boldsymbol{h}_i^*(R)$ holds with probability 1/2.

We therefore consider the simplest pattern, which checks a single fixed index i: $\pi([R]) = 1 \iff [R]\,[i] = \boldsymbol{h}_i^*(r)$. For this single-bit pattern, the lower bound for the likelihood ratio is:

$$\mathrm{LR}(\pi) \geq \frac{\frac{1}{2} + \frac{1}{2|R|}}{\frac{1}{2}} = 1 + \frac{1}{|R|} \tag{11}$$

For a typical record size of $|R| = 20$, this simple pattern yields $\mathrm{LR}(\pi) = 1.05 > 1$. The pattern is weaker than for BF or TSH, but it is still informative enough to be exploitable through PMAs. Considering multiple matching positions would further increase the LR.

3 Attack Description

NEPAL adopts the general methodology of PMAs [22] but reframes the cryptanalysis of encodings as a supervised machine-learning task. The attack proceeds in two stages. In the first stage, a machine-learning model is trained to predict the set of plaintext q-grams present in a record based on its encoded representation. In the second stage, the trained model is applied to target encodings—generated using the same parameters as the training set—to predict their corresponding plaintext q-gram sets. Optionally, these predicted q-grams can be re-assembled into coherent strings.

Formally, let $\mathcal{C}$ denote the universe of all possible encoded records (i. e. , the set of possible encoding outputs), and let $\mathcal{U}$ denote the universe of all possible q-grams constructible from a given alphabet. The primary objective of NEPAL is to approximate a function $f : \mathcal{C} \to \mathcal{U}^*$, where $\mathcal{U}^*$ denotes the power set of $\mathcal{U}$ (i. e. , the set of all possible subsets of q-grams). For any encoding $[R] \in \mathcal{C}$, the output $f([R])$ is a predicted q-gram set R^* that closely approximates or equals the true q-gram set R of the plaintext. In NEPAL, the function f is approximated by a neural network.

It is important to note that the NEPAL attack is agnostic regarding the origin of the training data. While its most direct application is an extension to GMAs, it can also function as a standalone attack if the attacker acquires encoding–plaintext pairs from alternative sources, such as data leaks or collusion with a data custodian.

3.1 Attacker Model

The attacker model for NEPAL aligns with that of GMAs and adheres to widely accepted assumptions regarding attacker capabilities in PPRL research [16, 21]. Specifically, the attacker has access to two datasets:

1. A set of known encoding–plaintext pairs $\mathcal{K} = \{(R_1, [R_1]), \ldots, (R_n, [R_n])\}$.
2. A set of target encoded records with unknown plaintexts, denoted by $\mathcal{T} = \{[R_{n+1}], \ldots, [R_l]\}$.

The attacker's objective is to reconstruct the plaintext identifiers for as many records in $\mathcal{T}$ as possible, leveraging only the encodings in $\mathcal{T}$ and the known pairs in $\mathcal{K}$.

Under Kerckhoffs' principle, GMA+NEPAL inherits the standard attacker model from prior GMAs: the attacker is assumed to know the encoding scheme and its non-secret parameters, whereas specific secrets such as random seeds or salt values remain unknown. For standalone NEPAL, however, even this knowledge is not required once encoding–plaintext pairs are available. These assumptions should therefore be understood as the minimum requirements for GMA+NEPAL, not for NEPAL in isolation.

In practice, this model corresponds to an adversarial LU, typically characterized as semi-honest (or honest-but-curious) in standard PPRL threat models [16, 21, 22].

3.2 Stage 1: Pattern Mining

The first stage of the NEPAL attack aims to learn a mapping from encoded records to the set of q-grams contained in the corresponding plaintext records. We model this as a multi-label classification task in which the network predicts, for each possible q-gram, the probability of its presence given an encoded input.

For this purpose, we employ a MLP, a type of ANN, chosen for its ability to capture complex, non-linear relationships within high-dimensional data [9, 11]. The MLP is parameterized by θ and defines a function $f_\theta : \mathcal{C} \to [0,1]^{|\mathcal{U}|}$, mapping each encoded record $[R]$ to a probability vector $\hat{y} = f_\theta([R])$. Each component $\hat{y}[j]$ represents the estimated probability that the j-th q-gram $r_j \in \mathcal{U}$ is present in the original plaintext set R.

The model parameters θ are optimized by minimizing a suitable loss function over the training subset of the known encoding–plaintext pairs $\mathcal{K}$, thereby aligning the network's predictions with the true q-gram memberships. The specific loss function, optimizer, and other model characteristics, such as the number and width of hidden layers and learning rates, are treated as tunable hyperparameters. This setup enables the model to autonomously adapt to the specific characteristics of the encoding problem (e. g. , BF length, number of hashes) without manually redesigning the attack for each scheme.

We employ the *Optuna* [1] framework within *Ray Tune* [12] to perform hyperparameter optimization, selecting the configuration that maximizes the Dice coefficient between the predicted and true q-gram sets, denoted as $\mathrm{Dice}(f_\theta([R]), R)$, evaluated on a 20% validation split of $\mathcal{K}$. Given two sets X and Y, the Dice coefficient is defined as:

$$\mathrm{Dice}(X, Y) = \frac{2\,|X \cap Y|}{|X| + |Y|} \in [0, 1] \tag{12}$$

A value of 0 indicates that the sets X and Y share no common elements, whereas a value of 1 indicates that the two sets are identical. The full search space for the hyperparameter optimization is detailed in Table 1.

3.3 Stage 2: Plaintext Reconstruction

Following training, the model f_θ is applied to each target encoding $[R] \in \mathcal{T}$ to obtain a prediction vector $\hat{y} = f_\theta([R])$. The final set of predicted q-grams is derived by applying a confidence threshold $0 \leq \tau \leq 1$:

$$R^* = \{r_j \in \mathcal{U} \mid \hat{y}_j \geq \tau\} \tag{13}$$

The optimal value for τ is determined during hyperparameter optimization. This process effectively acts as an inversion of the encoding at the level of q-gram-set membership.

Recovering the set of plaintext q-grams already constitutes a meaningful privacy breach even if the exact identifier cannot always be reconstructed. For example, a highly accurate recovered q-gram set can already support membership

Table 1. Search space for hyperparameter optimization.

Hyperparameter	**Search Space**
Number of layers	1, 2
Hidden layer size	512, 1024, 2048, 4096
Dropout rate	Uniform between 0.1 and 0.4
Activation function	ELU, SELU, Tanh
Optimizer	Adam, AdamW, RMSprop
Loss function	BCEWithLogitsLoss, MultiLabelSoftMarginLoss, SoftMarginLoss
Filter Threshold τ	Uniform between 0.2 and 0.7
Learning rate scheduler	ReduceLROnPlateau, CosineAnnealingLR, CyclicLR, None
Batch size	8, 16, 32, 64

testing or substantially narrow candidate identities. Exact re-identification is only required when the attacker must recover the original identifier itself, for example to enable downstream attacks such as impersonation.

If exact re-identification is required, the attacker can attempt to reconstruct a coherent string from the predicted q-grams, which likely represents the original plaintext identifier (e. g. , given name, surname, or date of birth). It is important to note that encoding schemes typically operate on q-gram *sets* rather than strings. Consequently, exact string reconstruction may not always be feasible, even if NEPAL perfectly inverts the encoding. In particular, if a string contains a specific q-gram multiple times, this multiplicity is lost when the string is converted to a set prior to encoding.

String reconstruction is formulated as finding the longest path in a directed graph constructed from the predicted q-grams. The approach is deterministic and proceeds as follows: For each set of predicted q-grams R^*, we construct a directed graph $G_{R^*} = (\mathcal{V}, E)$:

- **Vertices ($\mathcal{V}$):** The vertex set $\mathcal{V}$ comprises all q-grams in R^*.
- **Edges (E):** For every pair of q-grams $r_j, r_k \in R^*$, a directed edge (r_j, r_k) exists if and only if the suffix of r_j matches the prefix of r_k. Formally:

$$E = \{(r_j, r_k) \in \mathcal{V} \times \mathcal{V} \mid \text{suff}_{q-1}(r_j) = \text{pref}_{q-1}(r_k)\}$$

  where $\text{pref}_{q-1}(r)$ and $\text{suff}_{q-1}(r)$ denote the prefix (first $q-1$ characters) and suffix (last $q-1$ characters) of the q-gram r, respectively.

The final plaintext candidate is obtained by traversing the longest path in the graph, starting from the initial vertex and iteratively appending the last character of each subsequent q-gram.

If the graph is acyclic, the longest path can be found in linear time using dynamic programming with a time complexity of $O(|\mathcal{V}|+|E|)$ [6]. However, if the

graph is cyclic, the problem becomes computationally expensive. In such cases, we employ a depth-first search that explores all possible paths without reusing edges, returning the longest path found. Although an exhaustive search has a worst-case runtime of $O(2^{|E|})$, this approach remains computationally efficient in practice because the search space is substantially constrained by the sparsity and structure of the graph.

This reconstruction method is straightforward and deterministic, relying on neither probabilistic inference nor external knowledge sources. However, it cannot resolve ambiguities when multiple paths of equal length or plausibility exist, nor does it include semantic validation to ensure that reconstructed strings correspond to valid real-world entities.

Other string reconstruction methods, such as dictionary-based approaches or entity-validation heuristics, could be employed to further improve the accuracy of exact reconstruction.

4 Evaluation

We evaluated the efficacy of our proposed NEPAL attack[2] on the same eight datasets utilized by [16] for their GMA. The first six datasets are subsets of the *FakeName* dataset, generated by randomly sampling 1000, 2000, 5000, 10000, 20000, and 50000 records from the American name set of the *Fake Name Generator*[3]. Each record comprises randomly generated but realistic combinations of given name, surname, and date of birth. Additionally, we utilized the Euro Person dataset[4], originally created by the UK Office for National Statistics specifically for PPRL applications, which contains 26625 simulated records of personal identifiers. Following the methodology of [16], we selected given names, surnames, and dates of birth as attributes for our attack. Finally, the attacks were executed on a curated version of the Titanic passenger manifest[5], consisting of 891 unique records including the first and last names of passengers.

All datasets were attacked in three distinct configurations: (1) standalone NEPAL, where correct encoding–plaintext pairs are provided a priori, (2) standalone NEPAL trained on partially erroneous encoding–plaintext pairs and (3) GMA+NEPAL, where encoding–plaintext pairs are derived from the output of a GMA. For the latter, we employed the GMA reference implementation[6] from [16].

We do not claim that any particular overlap level is universally realistic. Rather, we study how attack effectiveness scales with the amount and quality of the available encoding–plaintext pairs, and prior work on GMAs shows that useful re-identification can already be achieved at comparatively low overlap levels [16, 21].

[2] `https://github.com/marcelmildenberger/neural-pattern-learning-attack`
[3] `https://www.fakenamegenerator.com`
[4] `https://ec.europa.eu/eurostat/cros/content/job-training\_en`
[5] `https://github.com/pandas-dev/pandas/blob/main/doc/data/titanic.csv`
[6] `https://github.com/SchaeferJ/graphMatching`

The performance of NEPAL is evaluated using the Dice coefficient, which represents the harmonic mean of precision and recall. In this context, precision measures the proportion of correctly predicted q-grams relative to all predictions, while recall quantifies the proportion of true q-grams successfully recovered. The Dice coefficient is calculated on a per-record basis and subsequently averaged across all records in the dataset to provide a comprehensive performance metric.

We consider the attack successful if it significantly outperforms a naive baseline strategy defined as follows: For each record, the baseline predicts the t most frequent q-grams observed across the entire dataset, where t denotes the average number of q-grams per record. Among the simple baselines we considered under the same attacker knowledge as NEPAL, this was the strongest directly comparable one. In particular, simple distribution-based guessing performed worse. The average baseline Dice coefficient is ≈ 0.237 across all datasets.

The success of reconstructing the original strings is measured using the Perfect Re-Identification Rate (PRR). The PRR is defined as the fraction of records in the target set for which the deterministically reconstructed plaintext string exactly matches the original identifier.

The encoding parameters and GMA settings used during evaluation align with those described in [16]. For the hyperparameter optimization of NEPAL, 125 trials are sampled from the search space, with the objective of maximizing the average Dice coefficient on the validation set. We employ a training–validation split of 0.8, consistent with standard supervised learning practices. Early stopping is applied with a patience of five epochs, requiring a minimum improvement of $\Delta = 1 \times 10^{-4}$ to continue training, with the maximum number of epochs capped at 25.

All experiments were conducted on a virtual machine running Ubuntu 24.04, equipped with 20 AMD Epyc 9254 CPU cores, 176 GB of RAM, and an NVIDIA GeForce RTX 3090 Ti GPU with 24 GB of VRAM.

4.1 Standalone NEPAL

We first evaluated the NEPAL attack using correct encoding–plaintext pairs sampled uniformly at random from each dataset. We varied the proportion of known data from 20% to 80% of the entire dataset in increments of 20 percentage points, with the remaining encodings constituting the target set for reconstruction.

Figure 1 reports the Dice coefficients versus training proportion, averaged across datasets by encoding scheme. BF increases from 0.79 at 20% training proportion to 0.93 at 80%, while TMH follows from 0.73 to 0.88. TSH exhibits behavior very similar to BF, starting at 0.8 and reaching 0.93, only slightly surpassing BF. Higher training proportions improve the reconstruction of plaintext q-grams, and all methods remain above the baseline Dice coefficient of 0.24, even at low training proportions.

Figure 2 illustrates the growth of the PRR with training proportion. For BF encoding, the rate increases from approximately 4.02% at 20% training proportion to 11.63% at 80%; TMH rises from 2.31% to 6.87%; and TSH achieves the

Fig. 1. Dice coefficient by encoding scheme as a function of training proportion. Dashed lines represent noisy datasets. The graph for BF is almost fully masked by the TSH graph.

highest PRR, climbing from 4.47% to 13.12%. Small improvements in the Dice coefficient can lead to larger gains in PRR. This effect arises from the deterministic reconstruction step, which is sensitive to missing q-grams and may fail to recover the correct identifier even when the predicted q-gram set is highly accurate. Consequently, the gap between Dice and PRR is expected.

Figure 3 demonstrates that dataset size significantly influences Dice coefficients. Large datasets (Euro Person, FakeName 50k) maintain values near 0.97–0.99 even at 20% training proportion, whereas the smallest datasets (FakeName 1k/2k) reach only 0.74 and 0.86, respectively, at 80%. Mid-sized datasets climb to values exceeding 0.9 by the 80% training proportion. Consistently, larger amounts of training data improve reconstruction quality. The Euro Person dataset is particularly noteworthy. Although it was designed specifically for PPRL evaluation and is therefore among the most realistic datasets considered here, it is also among the most vulnerable in Figure 3. Notably, it already reaches Dice coefficients above 0.9 at a training proportion of 20%.

In summary, our evaluation demonstrates that the proposed NEPAL attack achieves consistently strong results across a wide range of settings. On average, the NEPAL attack attains a Dice coefficient of 0.852, with a maximum of 0.997, highlighting its effectiveness in recovering plaintext q-grams from encoded data. Similarly, the average PRR reaches 7.56%, with peaks of up to 33.05%, underscoring the re-identification risks inherent in these encoding schemes. Among the schemes, BF encoding shows the highest vulnerability in terms of Dice coefficient, with an average of 0.868 and an average PRR of 8.26%. TMH encoding exhibits the lowest re-identification risk, with an average PRR of 4.89%, and the lowest reconstruction quality, with an average Dice coefficient of 0.824. In contrast, TSH encoding presents the highest re-identification risk, averaging 9.53%,

Fig. 2. PRR as a function of training proportion. The plot compares three approaches: BF (blue), TMH (orange), and TSH (green).

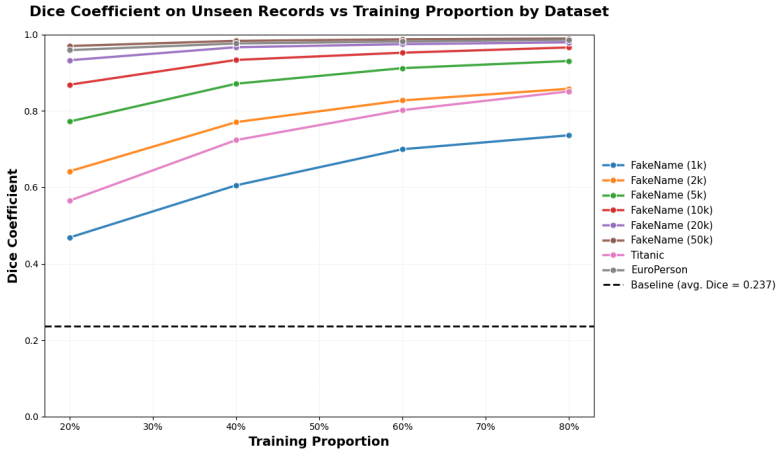

Fig. 3. Dice coefficient as a function of training proportion. The plot compares all eight datasets.

while achieving reconstruction performance comparable to BF (average Dice coefficient of 0.863) and slightly surpassing it at training proportions of 40% and above. Overall, our results indicate that TMH is the most resilient encoding scheme, whereas BF and TSH are more vulnerable. While BF and TSH exhibit similar vulnerabilities in terms of Dice coefficient, the consistently higher PRR of TSH indicates a greater record-level re-identification risk. This finding is noteworthy, as TSH was explicitly proposed as a more secure alternative to BF-based encodings [15], claiming improved resistance to cryptanalysis attacks. Our results demonstrate that, contrary to this design objective, TSH does not provide improved protection against PMAs and may even increase re-identification risk. This observation aligns with our analysis in Section 2, which shows that specific structural properties of TSH and BF induce high patterns, rendering both schemes highly susceptible to PMAs.

4.2 Standalone NEPAL on Noisy Data

To further assess the robustness of our attack under conditions characterized by suboptimal data quality, we introduced various types of errors into the datasets prior to executing the attack. These errors reflect common identifier imperfections, including typographical variations, inconsistent formatting, missing values, and processing errors. For each dataset, encodings were initially generated from the original, unaltered plaintext identifiers.

Subsequently, an erroneous plaintext dataset was generated as follows: For each plaintext record, we iterated over its attributes and independently and randomly determined whether each attribute should be altered. If selected, the attribute was subjected to one of the following randomly chosen mutations: deletion of the attribute, substitution of a single character, case change, swapping of two characters, insertion of whitespace, addition of suffixes (name fields only), date shifting by up to 12 days, modification of date formatting (e. g. , Day-Month-Year vs. Month-Day-Year), or replacement of dates with text tokens. Crucially, at most one mutation was applied to any given attribute.

Following these field-level mutations, given names and surnames were subject to potential swapping within a record. Furthermore, encodings between randomly selected record pairs were swapped, thereby introducing complete mismatches between plaintext identifiers and their corresponding encodings.

The impact of these errors on the attack's success is illustrated in Fig. 1 (dashed lines). Although noise reduces reconstruction quality, the overall trend persists. Across encoding schemes, the Dice coefficient increases from 0.65 to 0.75 for BF, 0.60 to 0.72 for TMH, and 0.65 to 0.75 for TSH as the training proportion moves from 20% to 80%. Relative to the clean setting (solid lines), the performance curves for noisy data are consistently lower, exhibiting an absolute drop of approximately 0.16–0.18 at high training proportions. For instance, at an 80% training proportion, BF and TSH reach ≈ 0.93 in the clean setting but drop to ≈ 0.75 under noise, while TMH decreases from ≈ 0.88 to ≈ 0.72.

4.3 NEPAL in Conjunction with GMA

We also evaluated GMA+NEPAL using encoding–plaintext pairs generated by first executing a GMA. The GMA was conducted across overlap scenarios of 20%, 40%, 60%, and 80%, considering both DropFrom strategies described in [16]. Figure 4 reports the Dice coefficients versus overlap, computed between predicted and ground-truth q-gram sets across the three encoding schemes.

For BF encoding, the Dice coefficient improves from 0.48 at 20% overlap to 0.85 at 80%. TMH follows from 0.47 to 0.81; and TSH climbs from 0.49 to 0.88, leading at higher overlap levels. Higher overlap yields a larger volume of training data via the GMA, which in turn enhances the reconstruction of plaintext q-grams.

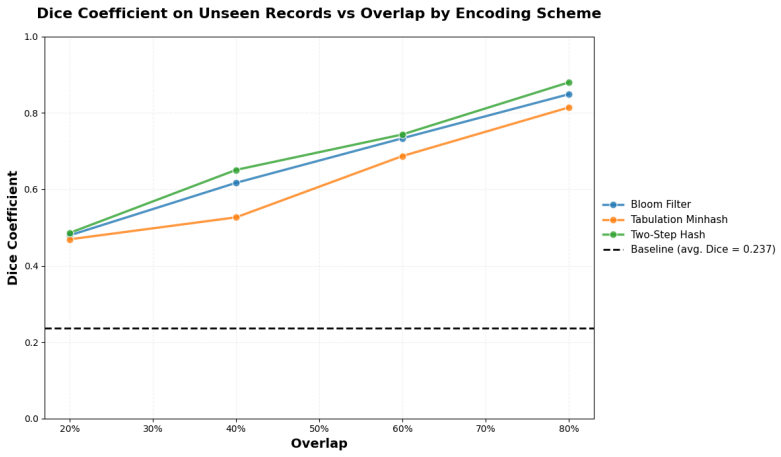

Fig. 4. Dice coefficient as a function of overlap. The plot compares three approaches: BF (blue), TMH (orange), and TSH (green).

Figure 5 illustrates how the PRR rises with overlap. For BF, it increases from 1.88% at 20% overlap to 11.82% at 80%. TSH starts at 3.29% and reaches 11.51%, yielding the highest rates across most levels. TMH moves from 0.99% to 6.32%, indicating stronger resilience against GMA+NEPAL.

Figure 6 details the Dice coefficients by dataset. Large datasets (Euro Person, FakeName 50k) exceed 0.95 by 60% overlap, while FakeName 20k rises from 0.75 at 20% to 0.98 at 80%. Mid-sized datasets (10k and 5k) improve from 0.53 and 0.13 to approximately 0.95 and 0.9, respectively. The smallest datasets (1k and 2k) climb more slowly, reaching 0.51 and 0.73 at 80%, while Titanic reaches 0.74. Increased overlap corresponds to more training data obtained through the GMA, which directly improves the reconstruction quality of the NEPAL attack.

Overall, the results demonstrate that the proposed NEPAL attack, in combination with the GMA, achieves high Dice coefficients across diverse overlap

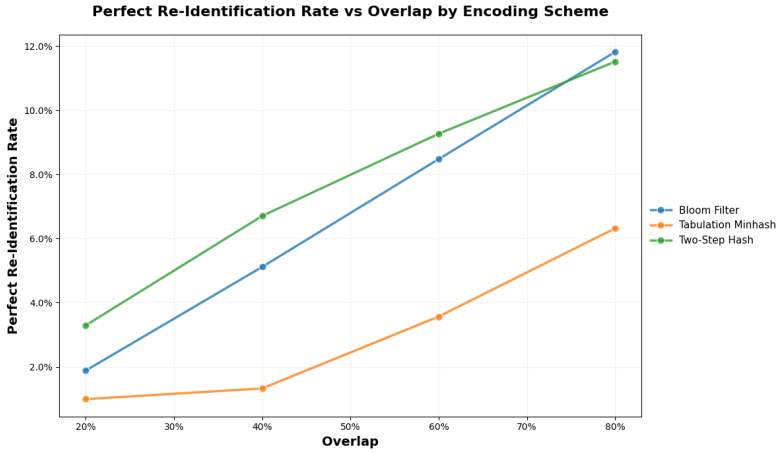

Fig. 5. PRR as a function of overlap. The plot compares three approaches: BF (blue), TMH (orange), and TSH (green).

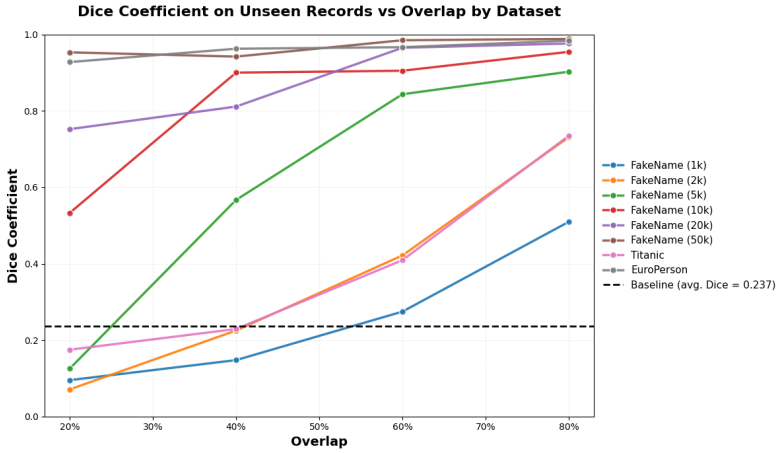

Fig. 6. Dice coefficient as a function of overlap. The plot compares all eight datasets.

settings and encoding schemes. On average, the attack attains a Dice coefficient of 0.664, peaking at 0.997, which confirms its strong capacity to recover plaintext q-grams from encoded representations. The corresponding average PRR of 5.93%, with maximum values reaching 28.39%, further highlights the significant privacy risks. Among the encodings, TSH achieves the highest Dice coefficient with an average of 0.69 and also the greatest re-identification risk (average of 7.69%), particularly at overlap levels above 40%. The BF achieves a slightly lower average Dice coefficient of 0.673, accompanied by a comparable PRR of 6.9%. In contrast, TMH remains the most resilient encoding, exhibiting both the lowest reconstruction performance (Dice coefficient of 0.629) and the lowest PRR (3.12%). These results are consistent with the exploitability analysis in Section 2. Compared with TMH, BF and TSH show stronger and more easily exploitable patterns. They also demonstrate that once a GMA provides encoding–plaintext pairs, NEPAL can compromise additional records beyond the original overlap.

5 Conclusion

In this paper, we introduced NEPAL, a novel machine learning-based attack capable of reconstructing plaintexts from data encoded for PPRL. The attack consistently and significantly outperforms baselines, achieving Dice coefficients exceeding 0.99 in certain scenarios. These results underscore the fundamental insecurity of most non-interactive PPRL schemes and suggest they should not be relied upon for privacy assurance. Across the evaluated schemes, TSH and BF exhibited the highest vulnerability, followed by TMH. These empirical findings in Section 4 align closely with the theoretical analysis presented in Section 2. NEPAL also provides a practical tool for assessing whether a similarity-preserving encoding leaks exploitable patterns to PMAs. In this sense, it can serve as a systematic stress test for learnable patterns in existing and future similarity-preserving encoding schemes.

Our study also has limitations. In its current form, NEPAL deliberately relies on a conceptually simple MLP and on a deterministic string reconstruction method for exact plaintext recovery, reflecting our goal of showing that the attack is already practical with established and straightforward components. This deterministic string reconstruction step does not resolve the ambiguity inherent in predicting q-gram sets, particularly the multiplicity of repeated q-grams in the original identifier. Accordingly, the reported PRR should be understood as a lower bound for what may be achievable with different reconstruction approaches on top of the same recovered q-gram sets.

From an ethical standpoint, our objective is not to enable misuse but to support the responsible evaluation and hardening of PPRL systems. Non-interactive encoding schemes should not be assumed secure in their current form, and more robust constructions, such as BFs with diffusion [2], must be considered.

Ultimately, our findings give rise to several open research questions warranting future investigation. First, and foremost, there is a critical need for a formal and well-defined security model for PPRL. Historically, the security of

PPRL encoding schemes has been evaluated primarily by demonstrating resistance against known attacks. However, as our results demonstrate, the security of non-interactive PPRL encodings cannot be assessed solely based on resistance to previously existing threats. Our analysis reveals that TSH, despite being proposed as a more secure alternative [15], is even more vulnerable to PMAs than standard BFs. Moreover, even advanced constructions such as TMH exhibit exploitable structures and are not immune to PMAs. These findings highlight the significance of our attack methodology, particularly in light of recent work that continues to focus on optimizing BF-based encodings [8] despite their fundamental vulnerabilities. Consequently, our results call into question the suitability of such encodings for deployment in security-critical settings.

Another important direction for future work is the design of encoding schemes that are resilient to PMAs—specifically, schemes that do not exhibit learnable or exploitable patterns—while simultaneously maintaining robustness against GMAs and preserving linkage quality. Furthermore, alternative linkage protocols beyond non-interactive PPRL schemes, particularly those that do not rely on similarity-preserving encodings, should be explored and investigated in greater detail. Finally, future work should also study stronger reconstruction methods on top of the recovered q-gram sets, since improved reconstruction or entity validation could further increase exact re-identification rate beyond the PRR values reported here.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A next-generation hyperparameter optimization framework. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. pp. 2623–2631 (2019)
2. Armknecht, F., Heng, Y., Schnell, R.: Strengthening Privacy-Preserving Record Linkage using Diffusion. Proceedings on Privacy Enhancing Technologies **2023**(2), 298–311 (Apr 2023). https://doi.org/10.56553/popets-2023-0054
3. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Communications of the ACM **13**(7), 422–426 (1970)
4. Broder, A.Z.: On the resemblance and containment of documents. In: Proceedings. Compression and Complexity of SEQUENCES 1997. pp. 21–29. IEEE (1997)
5. Christen, P., Ranbaduge, T., Schnell, R.: Linking sensitive data. Methods and techniques for practical privacy-preserving information sharing. Cham: Springer (2020)
6. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to algorithms. MIT press (2022)
7. Correll, P., Feyer, A.M., Phan, T., Drake, B., Jammal, W., Irvine, K., Power, A., Muir, S., Ferdousi, S., Moubarak, S., Oytam, Y., Linden, J., Fisher, L.: Lumos: a statewide linkage programme in australia integrating general practice data to guide system redesign. Integrated Healthcare Journal **3**, e000074 (05 2021). https://doi.org/10.1136/ihj-2021-000074

8. Dritsas, E., Trigka, M., Mylonas, P.: Privacy-preserving record linkage over big data platforms. In: 2025 International Conference on INnovations in Intelligent SysTems and Applications (INISTA). pp. 1–6 (Oct 2025). https://doi.org/10.1109/INISTA68122.2025.11249628
9. Gardner, M.W., Dorling, S.R.: Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. Atmospheric environment **32**(14-15), 2627–2636 (1998)
10. Kho, A.N., Cashy, J.P., Jackson, K.L., Pah, A.R., Goel, S., Boehnke, J., Humphries, J.E., Kominers, S.D., Hota, B.N., Sims, S.A., Malin, B.A., French, D.D., Walunas, T.L., Meltzer, D.O., Kaleba, E.O., Jones, R.C., Galanter, W.L.: Design and implementation of a privacy preserving electronic health record linkage tool in chicago. J. of the American Medical Informatics Association **22**(5), 1072–1080 (2015)
11. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature **521**(7553), 436–444 (2015)
12. Liaw, R., Liang, E., Nishihara, R., Moritz, P., Gonzalez, J.E., Stoica, I.: Tune: A research platform for distributed model selection and training. arXiv preprint arXiv:1807.05118 (2018)
13. Meier, J., Jakscha, T., Schnell, R., Heller, G.: Technische Dokumentation zur Umsetzung der Pseudonymisierung der PID-Daten für die Module Geburtshilfe und Neonatologie des QS-Verfahrens Perinatalmedizin in der Vertrauensstelle. IQTIG – Institut für Qualitätssicherung und Transparenz im Gesundheitswesen (11 2017)
14. Pita, R., Pinto, C., Sena, S., Fiaccone, R., Amorim, L., Reis, S., Barreto, M.L., Denaxas, S., Barreto, M.E.: On the accuracy and scalability of probabilistic data linkage over the brazilian 114 million cohort. IEEE Journal of Biomedical and Health Informatics **22**(2), 346–353 (2018). https://doi.org/10.1109/JBHI.2018.2796941
15. Ranbaduge, T., Christen, P., Schnell, R.: Secure and accurate two-step hash encoding for privacy-preserving record linkage. In: Advances in Knowledge Discovery and Data Mining: 24th Pacific-Asia Conference, PAKDD 2020, Singapore, May 11–14, 2020, Proceedings, Part II 24. pp. 139–151. Springer (2020)
16. Schäfer, J., Armknecht, F., Heng, Y.: R+ r: Revisiting graph matching attacks on privacy-preserving record linkage. In: 2024 Annual Computer Security Applications Conference (ACSAC). pp. 699–715. IEEE (2024)
17. Schmidlin, K., Clough-Gorr, K., Spoerri, A.: Privacy preserving probabilistic record linkage (p3rl): A novel method for linking existing health-related data and maintaining participant confidentiality. BMC medical research methodology **15**, 46 (05 2015). https://doi.org/10.1186/s12874-015-0038-6
18. Schnell, R., Bachteler, T., Reiher, J.: Privacy-preserving record linkage using bloom filters. BMC medical informatics and decision making **9**, 1–11 (2009)
19. Smith, D.: Secure pseudonymisation for privacy-preserving probabilistic record linkage. Journal of Information Security and Applications **34**, 271–279 (2017). https://doi.org/10.1016/j.jisa.2017.01.002
20. Vatsalan, D., Sehili, Z., Christen, P., Rahm, E.: Privacy-preserving record linkage for big data: Current approaches and research challenges. Handbook of big data technologies pp. 851–895 (2017)
21. Vidanage, A., Christen, P., Ranbaduge, T., Schnell, R.: A graph matching attack on privacy-preserving record linkage. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management. pp. 1485–1494 (2020)
22. Vidanage, A., Ranbaduge, T., Christen, P., Schnell, R.: Efficient pattern mining based cryptanalysis for privacy-preserving record linkage. In: 2019 IEEE 35th International Conference on Data Engineering (ICDE). pp. 1698–1701 (2019). https://doi.org/10.1109/ICDE.2019.00176