FreqyWM: Frequency Watermarking for the New Data Economy

Devriş İşler[1,2], Elisa Cabana[3§], Alvaro Garcia-Recuero[4§], Georgia Koutrika[5], and Nikolaos Laoutaris[1]
[1]*IMDEA Networks Institute,* [2]*UC3M,* [3]*CUNEF University,* [4]*Independent Researcher,* [5]*Athena Research Center*

***Abstract*—We present a novel technique for modulating the appearance frequency of a few tokens within a dataset for encoding an invisible watermark that can be used to protect ownership rights upon data. We develop optimal as well as fast heuristic algorithms for creating and verifying such watermarks. We also demonstrate the robustness of our technique against various attacks and derive analytical bounds for the false positive probability of erroneously "detecting" a watermark on a dataset that does not carry it. Our technique is applicable to both single dimensional and multidimensional datasets, is independent of token type, allows for a fine control of the introduced distortion, and can be used in a variety of use cases that involve buying and selling data in contemporary data marketplaces.**

***Index Terms*—Intellectual property, digital rights management, watermarking, ownership rights, data economy**

I. Introduction

Data-driven decision making powered by Machine Learning (ML) algorithms is changing how society and the economy work. ML is driving up the demand for data in what has been called the fourth industrial revolution. To satisfy this demand, several data marketplaces (DMs), which are mediation platforms aiming to connect the two primary stakeholders of the data value chain, namely the data providers/sellers and the data buyers [1], have appeared in the last few years.

The problems: Unfortunately, as with all digital assets, being able to copy/store/transmit datasets with close to zero cost makes creating illegal copies very easy. Even worse, unlike media content and software, the issue of ownership is less obvious when it comes to datasets. Any movie, song, e-book, or software can usually be attributed to a director, musician, author, or company, respectively, but this is hard to do for large datasets. These large datasets in data economy are traded in a wholesale manner that involves large numbers of tuples/rows. Consider an anonymised mobility dataset logging the movement of people in a city. Such a dataset may have been produced by collecting GPS readings from the smartphones of individuals using a map application, or it may be deduced by analysing cell phone traces [2] or Call Description Records (CDRs) maintained by mobile operators. Deployment of advanced privacy enhancing technologies (PETs) such as multiparty computation [3], (fully) homomorphic encryption [4], functional encryption [5], and trusted execution environments [6] can protect data from leaking in the first place and allow (pre-agreed) computations on data without hampering the functioning of the data-driven economy, e.g., private set computation [7], encrypted databases [8], secure computation [9], secure data aggregation [10], and verifiable databases [11]. However, most such approaches face serious scalability challenges that hamper their deployment in real-world use-cases. An alternative to deploying PETs solutions, is to rely on purely legal tools and terms and conditions to protect data ownership in the context of the new data economy [12]. In fact, most DMs do exactly that – trade plaintext versions of entire data [1, 13, 14] assuming that the different parties will abide to pre-agreed terms and conditions. With weak to nonexistent ownership guarantees by technical means, it is difficult to imagine that the data economy will ever flourish and reach its projected potential [15]. Indeed, any sold copy of a dataset can be 'pirated' by a buyer-turned-seller that can then resell the same dataset in a DM thereby undercutting the rightful owner and rendering its investment useless.

Watermarking is a well-known technique for protecting ownership upon copying and unauthorized distribution, initially proposed for protecting digital media [16, 17] and software [18]. Watermarking techniques for datasets [19, 20, 21] and machine learning models [22] have been proposed recently. Watermarking generally consists of two algorithms: *generation* (or embedding) and *detection*. The generation allows an owner to embed an invisible (or visible) watermark into their data using a high entropy (watermarking) secret and produces a watermarked version of the data introducing tolerable distortion without degrading the data utility. During the detection algorithm, the owner proves its ownership on the suspected data (even if it is modified) using the same watermarking secret generated during the watermark generation. If the result of the detection is 1 (or *accept*), the owner can use it to prove their ownership on the (suspected) watermarked data. A watermarking scheme is assumed to be secure against the guess attack (where an attacker tries to expose the watermarking secret) and robust against (un)intentional alterations/-modifications (i.e., a watermark should be still detectable even under attacks such as [20, 23, 24, 25, 26, 27, 28]).

Limitations of existing watermarking techniques: Watermarking techniques, depending on the nature of their application, may have very different objectives, e.g., numerical database watermarking controlling the distortion on mean and standard deviation [21], reversible watermarking allowing owners to reconstruct the original data [29], watermarking text datasets preserving the meaning of a text [30] and/or the frequencies of the words [31], categorical watermarking preserving the (predefined) categories (e.g., gender) of a

§Work done while the author was affiliated with IMDEA Networks Institute.

dataset [32]. All these solutions focus on a specific data type in a specific domain [23, 33]. Another limitation of theirs relates to the level of control they offer to the user in terms of controlling the distortion introduced upon the original data due to the watermark. There are, for example, techniques that maintain the mean and the standard deviation of a numerical field [20, 34, 35] but, as we will show later, this can lead to arbitrary large distortion between the original and the watermarked data when considering the entire distribution of values that goes beyond the mean and the standard deviation. To address these limitations, *we introduce a novel watermarking technique that can be implemented over a wide range of data types and structures* (with some constraints that will be explained later) while *giving the data owner very precise control over the introduced distortion*.

A novel watermarking technique for data: In this paper, we present a novel *Frequency Watermarking* technique, henceforth *FreqyWM*, [1] for hiding a secret within a dataset in a manner that makes the said secret indistinguishable from the data that it protects. The main idea behind *FreqyWM* is *to modify slightly the appearance frequency of existing tokens* within a dataset in order *to create a secret in the form of a complex relationship* between the frequencies of different tokens. By making this relationship complex enough, we can reduce the probability that it appeared by chance close to zero. Therefore, by revealing knowledge of such secret relationship, a party can claim ownership over a dataset because the only practical way of knowing such a secret is to have inserted it in the data in the first place. A token may be a word, a database record, a URL, or any repeating value within a structured or semi-structured commercial dataset. Our secret is created by first selecting a number of token pairs. Then, for each pair, we slightly modulate the frequency counts of its tokens in order to make their difference yield zero under modulo N arithmetic. This can be easily done by adding or removing some instances of one, the other, or both tokens. By increasing the number of selected pairs we can make our watermark more resistant to attacks, as well as less likely to have appeared by chance.

FreqyWM can achieve several things. First and foremost, by revealing knowledge of the secret encoded by the watermark, a data seller can prove rightful ownership of a dataset to a third party, such as a DM. This can be used to distinguish a rightful owner from a pirate that may attempt to monetize a pirated dataset in a DM. If the DM, or the rightful owner detects such an event, the dataset can be removed and the pirate be banned. This would mimic what web-sites like YouTube do to protect copyrighted content. Detecting the presence of pirated copies can be achieved using content similarity [36], locality sensitive hashing [37, 38] and even hashing similarity [39] that go beyond the scope of watermarking.

In addition to proving ownership, our watermarking technique can also reveal who may have leaked (copied/pirated) a dataset in the first place. A dataset seller or a DM may create a different watermark for every buyer and in addition to encoding it into the data, store also a description of it in some immutable index (e.g., a blockchain). Then, if an unauthorized copy of the dataset is found at a latter point, the culprit can be identified by looking up its watermark against this index.

[1]Freqy pronounced as *freaky*.

Our major contributions are as follows:

• Our first contribution is the idea of using the appearance frequency of tokens to encode invisible watermarks upon datasets traded in DMs. We establish a family of such watermarks using frequency pairs and modulo arithmetic and prove that creating an optimal *FreqyWM* reduces to solving a Maximum Weighted Matching (MWM) problem [40, 41] combined with a polynomial special version of the 0/1 Knapsack problem [42] involving items of equal value but different weights.

• We extend frequency-based watermarking to make it resilient against a series of attacks. In particular, we protect our technique against a *Guess Attack* attempting to identify our watermarked pairs and secrets to impersonate the rightful owner. We make such an attack computationally hard by introducing a high-entropy secret while generating the watermark. We also protect against a *Re-watermarking Attack* mounted by having a pirate inject its own watermark upon an already watermarked dataset, and then present the former as a false proof of ownership. We thwart such an attack by describing a simple protocol capable of ordering chronologically multiple watermarks that may be carried by different versions of the same dataset. We protect against a *Destroy Attack* attempting to destroy our watermark by changing the frequency of different tokens in the dataset. By relaxing our modulo arithmetic rule used during the verification of a particular watermark pair, as well as the percentage of pairs to be detected before the entire watermark is verified (accepted), we oblige the attacker to effectively also destroy the actual data in the process of destroying the watermark. Finally, we show that our technique is robust to a *Sampling Attack* in which the attacker attempts to pirate only a random sample of the watermarked data.

• Our final contribution is an extensive performance evaluation study aiming to explain the impact of the main parameters of *FreqyWM* on major performance metrics under different attack scenarios using synthetic and real world datasets.

The main **findings** of our evaluation are as follows:

• We show that as long as there exists sufficient variation in the frequencies of different tokens, *FreqyWM* can encode robust watermarks with minimal distortion on the initial data. Our technique does not apply to uniform token appearance frequencies, because in this case there does not exist sufficient gap between different frequencies for encoding a watermark.

• Regarding the false positive probability, i.e., "detecting" a watermark on a dataset that does not carry it, our analytical bounds (in the form of closed form expressions) show that it quickly goes to zero as we increase the number of pairs.

• We demonstrate that a Guess Attack has negligible probability of success, thereby making it impossible for almost all practical cases. On the up side, the rightful owner or any party, that is given the watermarking secret for verifying the watermark, can do that very fast in linear time complexity.

• Regarding Sampling Attacks, we show that with the exception of very small samples, our detection algorithm is capable of detecting our watermark. Achieving this requires using the relaxed detection algorithm that trades robustness to attacks with false positives. For example, on a sample of 20% and with thresholds that impose tiny false positive probability, the detection probability exceeds 90%.
• In terms of Destroy Attacks, we show that a watermark that imposes (costs) a tiny 0.0002% distortion on the original data, remains detectable even under attacks that add random noise that imposes a 90% modification.
• Compared to existing solutions from the literature [30, 35] that are applicable only to numerical data and preserve only the mean value of the watermarked data, *FreqyWM* allows a data owner to control the exact amount of distortion introduced by the watermark in terms of cosine or other similarity metrics which, under [30, 35] may become unbounded. For example, a *FreqyWM* watermark that imposes only 0.0002% distortion in terms of cosine similarity, is stronger than watermarks from [35] and [30] that impose 46.72% and 4% distortion, respectively under the same metric.

II. Related Work

Database watermarking is the closest type of watermarking to our work. There are of course other types of watermarking and fingerprinting (when an owner generates a unique watermark for each intended party, e.g., buyers/data marketplaces), for example, for sequential [43] and genomic datasets [44]. However, as they focus on specialized types of data, we do not go into more details about them. Survey papers such as [23, 33, 45, 46] compare database watermarking techniques in terms of verifiability, distortion, supported data types, and other aspects. Many of these solutions are applicable only to numerical data and thus cannot be applied to a range of commercial datasets, e.g., to web-browsing click-streams.

The first known watermarking technique for relational data is a *numerical database watermarking* approach [20]. The watermark information is normally embedded in the Least Significant Bit (LSB) of features of relational databases to minimize distortion. Other numerical database watermarking solutions introduce distortion by considering the statistics of numeric values [34, 35, 47, 48, 49]. The proposed solutions in [20, 35] focus on keeping the change at minimum (i.e., median and standard deviation). *However*, numerical database watermarking unfortunately cannot be applied to datasets composed of string and numerical values (e.g., CDRs, web-browsing history) that we handle in our work.

Distortion-free database watermarking schemes have also been proposed [50, 51] that introduce fake tuples or columns in the original database. The fake tuples or columns are created based on a watermark secret by computing a secret function which makes watermarking visible and easy to remove. *However, an attacker can remove the watermark with minimum computational power*, making these approaches inapplicable to our case. *Reversible watermarking* allows owners to reconstruct the original data used for watermarking on the top of watermark verification [29, 30, 49, 52, 53, 54, 55, 56, 57]. They have similar properties as other relational watermarking techniques (e.g., private key based, robust, introducing distortion).

Categorical watermarking [32] is another watermarking approach that replaces tokens in a dataset with another token in the same category. However, this causes an undesired distortion and requires predefined categories (e.g., gender, clothing size) in the data. Consequently, its applicability on datasets consisting of different data types is limited. *Text watermarking* [30, 31] is for text files where it changes a token (e.g., by replacing a word with another similar word) trying to preserve the meaning of a text [30] and/or the frequencies of the words [31]. However, assume the dataset is a list of URLs visited by the owner, then this (insecure) change/replacement may invalidate a token (e.g., causing an invalid URL).

In the context of datasets in our use case, while prior works try to minimize the amount of distortion on median, average, or first moments of the distribution of a feature, the owner can limit the exact distortion between the original and the watermarked dataset as reflected by distance metrics that capture the shape of the entire distribution of a feature. Our results in Section IV-D have shown that the latter can deviate arbitrarily if an owner tries to control only the first most important moments.

III. Frequency-based Watermarking

In this section we provide an overview of *FreqyWM* and the notations used throughout the paper in Table I.

D_o	The original data to watermark.
D_w	Watermarked (data) version of D_o.
tk_i	ith token.
f_i^o	Frequency of ith token in D_o.
f_i^w	Frequency of ith token in D_w.
R	A high entropy secret.
L_{wm}	A list of chosen token pairs for watermarking.
L_{sc}	A list of secrets required for watermark detection.
L_e	A list of eligible token pairs for watermarking.
k	Threshold for detecting a watermark.
t	Threshold to accept a pair as watermarked.
b	A budget threshold for distortion that watermarking can introduce.

TABLE I: Notation.

Running Example. To provide the intuition behind our watermarking approach, assume a scenario where an owner holds a real click-stream dataset consisting of visited URLs (e.g., the dataset by [58]). Such datasets are desired by modern data analytic-based applications [59] where their frequency histograms (e.g., the number of clicks/visits, popularity of likes in social networks) are used as an essential source of information. For instance, assuming the appearance frequencies (histogram) visualized in Figure 1 via a tabular form, the most frequent token is `youtube.com`, the second one is `facebook.com`, and so forth. After watermarking, it is important that the *ranking* of the tokens based on the frequency shall not change while the frequency appearances can be modified. For instance `youtube.com` shall be the most frequent URL (token) visited in the watermarked dataset. Another important distortion metric on the histogram is *similarity*. It

is important that owners shall have control over the change in similarity. Since the similarity metric can be varied depending on the application that a dataset will be used, owners can assign a *budget* to determine the minimum similarity desired on the frequency distribution after watermarking. Based on the above, we derive two natural constraints on the data utility to allow an owner to control distortion, without limiting watermarking to a specific data type:

• *Ranking Constraint:* Watermarking should preserve the ranking of token frequency distribution (histogram). Preservation of ranking does not of course imply that frequencies of individual tokens need to remain intact. • *Similarity Constraint:* The similarity between original and watermarked frequency distributions (histograms) should not be any less than $(100 - b)\%$ where b is a budget. Input b is determined by the owner to keep distortion due to watermark generation within a given budget. [2]

To satisfy these constraints and overcome the shortcoming of existing watermarking techniques, we introduce a new *private-key based* watermarking scheme, *FreqyWM*, that is *blind* (does not require the original data), *primary-key free* (does not need attributes that uniquely specify a tuple in a relation in a dataset), *robust*, and *secure* against *guess*, *sampling*, *destroy*, and *false-claim* attacks with a high utility and a good trade-off between the complexity of the transformation and algorithmic efficiency of the solution.

A. Overview of our Approach

FreqyWM consists of two main algorithms: the watermark generation algorithm, $WMGenerate$, and the watermark detection algorithm, $WMDetect$. $WMGenerate$ generates watermarked data based on a *budget* b capturing how much the watermarked data may differ from the original one, e.g., in terms of cosine (or other) similarity metrics of their corresponding token frequency distributions. By calling $WMGenerate$, the owner creates a watermarked version of their data consisting of tokens such that ownership can be proved. $WMDetect$ detects if a suspected dataset holds the watermark of the owner using the owner secrets produced by $WMGenerate$ and two thresholds (k and t). If $WMDetect$ outputs *accept/verified*, this evidence would prove that the owner can claim ownership of the watermark and thus the data. By nature, $WMDetect$ can be computed as many times as desired in private while it can be computed *only once* in public, because it would mean that the potential data owner shall reveal the secret leading to such watermark to the public (or whomever must verify it, e.g., a judging third party). As part of our future work, we are also looking at public verifiability without revealing the private key (Section VII).

We describe the general idea behind *FreqyWM*, illustrated in Figure 1. We use our running example. Of course, our technique is general and can be applied to any repeating token beyond just URLs, as we explain in Section IV-C.

[2]Although in our experiments we use cosine similarity, any similarity metrics can be deployed without any loss of security and change in *FreqyWM*.

Watermark Generation. Assume that the data owner holding a list of URLs visited creates a dataset D_o using the *domain* of each URL in the list as a token and sets a budget b for the similarity constraint. $WMGenerate$ has the following steps:

• *Histogram Generation.* Since *FreqyWM* aims to preserve the appearance frequency of tokens, it first creates a histogram of the original dataset D_o such that it sorts all unique tokens in descending order of their frequency (e.g., YouTube is the most visited, Facebook is the second, and so on).

• *Generation of Eligible Tokens. FreqyWM* cannot modify the frequencies randomly because of the ranking constraint. Therefore, it identifies a list L_e of eligible pairs of tokens that are candidates to be watermarked using some secret R.

• *Optimal Selection*. With the identification of eligible pairs, *FreqyWM* ensures that the ranking is preserved after watermarking. However, the similarity constraint is yet to be satisfied. To keep the similarity at least at $(100 - b)\%$, *FreqyWM* selects pairs of tokens from eligible pairs for watermarking, denoted by L_{wm}, based on the budget constraint b. For this purpose, *FreqyWM* benefits from solving two well-known problems: *Maximum Weight Matching* (MWM) and *Equally Valued 0/1 Knapsack problem* (QKP). To do so, eligible pairs are converted to a graph representation where vertices represent a token, and an edge represents a pair. *FreqyWM* applies Maximum Weight Matching to the graph representation (discussed in detail later). By applying MWM, *FreqyWM* selects the pairs from eligible pairs requiring the minimum change in total; however, it does not necessarily mean that the similarity between the original histogram and watermarked histogram will be at least $(100-b)\%$. To choose another set of pairs satisfying the Ranking Constraint from the pairs derived after MWM, an *Equally Valued 0/1 Knapsack problem* needs to be solved. The more the token pairs are selected to watermark, the more robust *FreqyWM* is, since the number of tokens to attack (e.g., remove/identify) increases. To fulfill the budget b, QKP selects a maximum number of pairs such that the similarity between the original frequency histogram and the watermarked one is *at least* $(100 - b)\%$.

• *Frequency Modification*. Until now, *FreqyWM* determines the final pairs of tokens for watermarking but frequency appearances are yet to be modified to create the watermarked histogram. Therefore, *FreqyWM* modifies the frequencies of the selected tokens where the frequencies of a pair of tokens would be equal to 0 (as a watermark embedding rule) in some modulo that is calculated based on secrets and tokens in the pair. To make it more comprehensible and show how the modifications occur, let us assume that the frequencies of a chosen pair, e.g., `youtube.com` and `instagram.com`, are 1098 and 537, respectively. Assume also that a modulo value, say 129, is computed based on the secrets and the tokens (e.g., `youtube.com` and `instagram.com`). The difference between the two frequencies in modulo 129 is 45. To set the difference to 0, we need to change the appearance frequencies for Youtube and Instagram in the dataset. 45 is divided (by 2) as 23 (by ceiling) and 22 (by flooring). The new frequencies of `youtube.com` and `instagram.com`

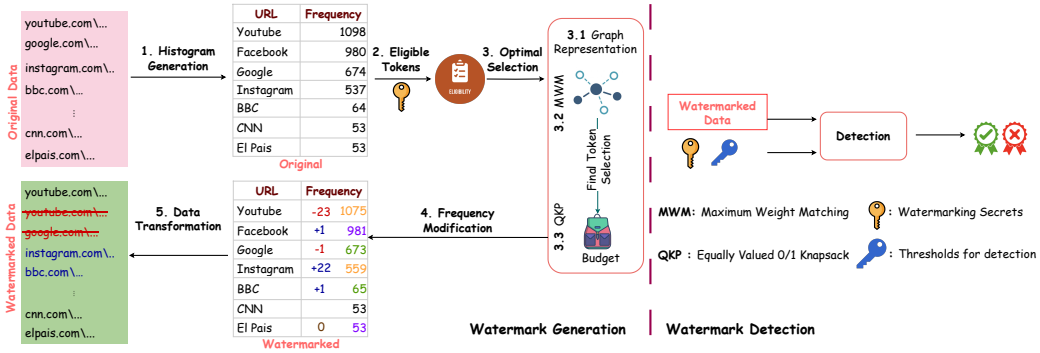


Fig. 1: *FreqyWM* illustrated based on a (Top Level Domain, TLD) URL dataset. URLs chosen as a pair for watermarking are represented with the same colored frequencies (e.g., `Youtube` and `Instagram`) while the ones not selected are colored black (e.g., `CNN`).

need to become $1098 - 23 = 1075$ and $537 + 22 = 559$ such that $(1075 - 559) \mod 129 \equiv 0$. We can do that by removing 23 instances of Youtube from the dataset, and adding 22 more instances of Instagram. However, when the remainder (i.e., $(1098 - 537) \mod 21 \equiv 16$) is greater than half of the modulo, we add the modulo result calculated as $(\lceil (1098 - 537) \div 21 \rceil) \times (1098 - 537)$ to the difference. This way, we never have to eliminate remainders that exceed half of the modulo. As it will be evident in the next section, this observation enables us to determine eligible tokens.

• *Data Transformation. FreqyWM* adds/removes tokens based on the frequencies and produces a watermarked dataset D_w.

Watermark Detection. An owner wishes to verify if a (watermarked) dataset D'_w (a modified version of D_w) is watermarked by using the secrets stored from the watermark generation. To determine the confidence level in the detection (e.g., the minimum number of detected watermarked tokens), the owner provides some threshold values (k and t). With the watermarking secret and the thresholds, the detection returns *accept/verified* or *reject*.

B. Detailed Description of FreqyWM

Algorithm I: Watermark Generation

Input: D_o, b
Output: D_w, L_{sc}
$D_o^{hist} = \textbf{Preprocess}(D_o)$
$R \leftarrow \{0,1\}^{\lambda}$, $z \leftarrow Z^+$
foreach $\{tk_i, tk_j\}_{i \neq j} \in D_o$ **do**
  $s_{ij} = H(tk_i || H(R || tk_j)) \mod z$
end
$L_e \leftarrow \textbf{Eligible}(D_o^{hist}, \{s_{ij}\})$
$L_{wm} \leftarrow \textbf{OptMatch}(D_o^{hist}, L_e, \{s_{ij}\}, b)$
foreach $\{tk_i, tk_j\} \in L_{wm}$ **do**
  $D_o^{hist}.\textbf{Update}(f_i^w, f_j^w, s_{ij})$
end
$D_w \leftarrow \textbf{Create}(D_o^{hist}, D_o)$
$L_{sc} = \{L_{wm}, R, z\}$
Result: $D_w, L_{sc} = \{L_{wm}, R, z\}$

1) Watermark Generation: The data owner holds the original data D_o and defines a budget b that decides how much distortion a watermark can introduce. For comprehensibility, assume that D_o is a single-dimensional dataset, e.g., a dataset with one attribute (see Section IV-C for how to apply *FreqyWM* to multi-dimensional datasets). D_o consists of repeating values called *tokens* that can be of *any* data type, which enables *FreqyWM* to be data-type agnostic. The goal of watermarking is to generate the optimal watermark, i.e., *with the largest number of watermarked pairs* within the given budget b. The generation algorithm (Algorithm I) follows these steps:

Histogram Generation: It pre-processes D_o to generate a histogram $D_o^{hist} = \textbf{Preprocess}(D_o)$. D_o^{hist} consists of a set of tokens, $\{tk_0, \ldots, tk_{|D_o^{hist}|}\}$ (e.g., tk_0 =`youtube.com`) where each tk_i has an (original) appearance frequency f_i^o (e.g., there are 1098 YouTube visits). The histogram D_o^{hist} is sorted in a descending order of frequency. To keep the distortion introduced by the watermark at minimum (e.g., after watermarking, YouTube is still the most visited, followed by Facebook, although their frequencies may have changed), we calculate two boundaries for each token tk_i: an upper boundary u_i and a lower boundary l_i. The boundaries allow us to determine how much change we can introduce to the token and whether a token pair is eligible as explained later. Naturally, for the token with the highest frequency in the histogram, it is $u_0 = \infty$ because we can increase the frequency of tk_0 as much as we want, while the lower boundary of the last token, $tk_{|D_o^{hist}|-1}$, is set to its frequency as $l_{|D_o^{hist}|-1} = f_{|D_o^{hist}|-1}$ because we can remove at so many appearances. For the rest of the boundary calculations of each token tk_i, u_i is defined as the difference between $f_{(i-1)}^o$ and f_i^o, while l_i is assigned as $f_i^o - f_{(i+1)}^o$. Note that once the boundaries are set, they remain same until frequency modification.[3]

Generation of Eligible Tokens: In cryptography, $\lambda \in \mathbb{N}$ is a security parameter, i.e., a variable measuring the probability with which an adversary can break a cryptographic scheme [60]. In other words, λ provides a way of measuring how difficult it is for an adversary to break a cryptographic scheme. *FreqyWM* requires randomization to be secure by ensuring that an attacker has only negligible advantage to recover the watermark and create collision for false claim (e.g., coming

[3]The frequencies of some tokens may have high importance. An owner can filter the dataset and exclude them from watermarking.

up with another watermarking secret which returns accept on data not watermarked by it). Thus, we choose a hash function to overcome the collision. In detail, a hash function H (chosen from a family of such functions) is a deterministic function from an arbitrary size input to a fixed size output, denoted $H : \{0,1\}^* \rightarrow \{0,1\}^\lambda$. The hash function [60] is *collision resistant* if it is hard to find two different inputs $m_0 \neq m_1$ that hash to the same output $H(m_0) = H(m_1)$.

Based on the above, to determine token pairs for watermarking, *FreqyWM* first generates a high entropy random number, i.e., secret, $R \leftarrow \{0,1\}^\lambda$ and an integer $z \in \mathbb{Z}^+$. Then, it uses R and z to compute s_{ij} values for modulo operation as: $s_{ij} = H(tk_i||H(R||tk_j)) \mod z$, where $||$ denotes concatenation. A set L_e of all eligible pairs is generated by an algorithm $Eligible$ based on given pairs $\{tk_i, tk_j\}$ and corresponding s_{ij} values as $L_e \leftarrow Eligible(D_o^{hist}, \{s_{ij}\})$. A pair is accepted as eligible if it satisfies that the boundaries of each token in the pair are at least $\lceil s_{ij}/2 \rceil$ where $s_{ij} \geq 2$. s_{ij} cannot be 0 or 1 because of modulo operation since modulo 0 is undefined and modulo 1 is 0. Note that the size of L_e is bounded by $[0, \binom{|D_o^{hist}|}{2}]$ where 0 means that there is *no eligible pair* while $\binom{|D_o^{hist}|}{2}$ means that *all the possible pairs* of tokens are eligible. After the eligible pairs are constituted, the boundary check is not necessary anymore since whichever set of pairs (that does not have a common token among) is chosen, the ranking will be preserved.

Optimal Selection: The eligible pairs are defined by ensuring the ranking constraint. However, to determine which subset of eligible pairs shall be selected such that chosen optimal number of pairs of tokens, denoted by a set L_{wm}, respect the budget constraint, it runs optimal matching algorithm from the eligible pairs L_e using the frequencies and s_{ij} values as $L_{wm} \leftarrow OptMatch(D_o^{hist}, L_e, \{s_{ij}\}, b)$. In Section III-B2, we show that for our optimal selection solution, we acutely reduce our problem to *Maximum Weight Matching (MWM)* and *Equally Valued 0/1 Knapsack problem (QKP)* problems to solve. We also devise two heuristics: *greedy* and *random*.

Frequency Modification: Based on L_{wm}, the algorithm creates new frequencies of tokens chosen from the optimal matching algorithm $(f_i^w - f_j^w) \mod s_{ij} \equiv 0$. This, of course, changes the boundaries of tokens; however, we do not need to update the boundaries as they are not needed anymore.

Data Transformation: It generates or removes tokens based on new frequencies. Note that the position of where to add tokens is important for security of *FreqyWM* against guess attack. Therefore, new tokens should be added in random positions (see Section IV-C for more discussion). As a final step, it returns the list of tokens D_w and stores L_{wm}, z value, and the random value R as a list L_{sc}.

2) Optimal and Heuristic Approaches : Given that all watermarked pairs have equal value in terms of proving ownership of the data, an optimal watermark is just a watermark of maximum size in terms of watermarked pairs, within the defined constraints (*similarity* and *ranking*). *Optimal Matching*. Let us now define our optimal watermarking. Let $G = \{V, E\}$ be a connected undirected graph which is the representation of frequencies driven from eligible pairs L_e. $V = \{v_1, v_2, \ldots, v_{|V|}\}$ where v_i represents tk_i and $E = \{e_1, e_2, \ldots, e_{|E|}\}$ where $e(v_i, v_j)$ is the edge between v_i and v_j. The weight of an edge $e(v_i, v_j)$, $w(e_i)$, is equal to $T - ((f_i^o - f_j^o) \mod s_{ij})$ where T is a big value (e.g., $T > C$ where C is the highest difference between two frequencies in the eligible pairs). Then, our optimal watermarking problem reduces to finding the maximum number of edges (pairs) such that *no edge* has a common vertex and b is not exceeded.

Definition 1 (Optimal Watermarking)**.** *Let $OptWM(G(V,E), b)$ be the optimal watermarking with a budget of b among an eligible set of items L_e represented as a connected undirected graph $G(V,E)$. The optimal watermarking produces the maximum number of edges (pairs) while not exceeding the budget b defined below:*

$$MAX\ |M^w|,\ \ M^w = \{e_1, ..., e_{|M|}\}\ \ s.t.\ \ sim(D_o^{hist}, D_w^{hist}) \geq (100 - b)$$

where M^w denotes the chosen pairs for watermarking.

The solution of the pairing problem is reduced to two well-known problems with polynomial time solutions: *Maximum Weight Matching* (MWM) and *Equally Valued 0/1 Knapsack problem* (QKP) (which we have a special case where all values are equal). Note that while the general $0/1$ Knapsack problem is known to be NP-Hard [42], this special equally valued $0/1$ Knapsack problem would have a polynomial time (greedy) solution. In particular, our optimal pairing problem is reduced and solved as follows:

- Find the maximum weight matching $M = e_1, e_2, \ldots, e_{|M|}$ as $M = MWM(G(V,E))$. Notice that M includes the edges such that the sum gives the maximum weight. It refers to minimum weight for us since the weights are defined as $T - (f_i - f_j \mod s_{ij})$ which makes the highest frequency difference have the smallest weight and the smallest one have the highest weight. With this conversion, we identify the edges distorting the histogram minimally.
- After finding the edges via MWM, we have one more constraint which is the budget b. The matching algorithm has to return the maximum number of matchings for which the distortion (e.g., based on cosine similarity) does not exceed b which can be solved via QKP where the value of each item is 1, and the weight is recomputed as $T - w(e_i)$. Recomputation is necessary because for the QKP we want to add as many items as possible that will be bounded by b. Therefore, it finds the set of edges L_{wm} in M such that the selected edges do not exceed the budget b by employing the QKP as $L_{wm} = QKP(M, b)$ where $L_{wm} = e_1, e_2, \ldots, e_{|L_{wm}|}$ and value of each e_i is 1 ($val(e_i) = 1$). Showing the optimality of the resulting watermark according to Definition 1 is straightforward and can be proven via proof-by-contradiction. In a nutshell, if our solution is not optimal, it means that one of the solutions produced by MWM and QKP cannot be optimal. However, since MWM and QKP are both assumed

to be optimal, this contradicts with our statement and thus our solution is optimal.

Heuristic Matching Algorithms. We define two heuristic algorithms: 1) *greedy*; and 2) *random*. In the *greedy* algorithm, all the eligible token pairs are sorted in an ascending order by their remainders as $rm_{ij} \equiv (f_i^o - f_j^o) \mod s_{ij}$. The algorithm starts selecting a pair respectively for watermarking where b would not be exceeded when it is chosen (i.e., comparing the similarities of original and watermark histograms). This continues until b is exhausted or there is no more item to visit. The *random* matching algorithm follows the same steps as the greedy algorithm except it does not sort the eligible pairs but rather selects a pair randomly from L_e.

3) Watermark Detection: In detection, the data owner wishes to know if there is a watermark of its in a token dataset D'_w to claim ownership. The owner holds its secret list $L_{sc} = \{L_{wm}, R, z\}$ where L_{wm} is the list of watermarked token pairs, R is the high entropy value, and z is the (modulo) integer, all generated by the watermark generation, along with two thresholds: (1) t, a *threshold* to decide if a certain pair is watermarked; and (2) k, the *minimum number of watermarked pairs* required to conclude whether D'_w is a watermarked dataset. How to set t and k depends on the robustness an owner wants (see Sections III-B4 and IV-A2). If the owner wants to prove ownership to a third party, it has to reveal its secrets to that party. This causes to prove the ownership once in public (see Section V-D). Our watermark detection algorithm (Algorithm II) proceeds as follows:

(1) It builds the histogram list D_w^{hist} of the suspected dataset D'_w as in the watermark generation algorithm. The algorithm does not calculate the boundaries, just the token frequencies.

(2) For each token pair $\{tk_i, tk_j\}$ in L_{wm}, if the pair exists in D_w^{hist}, the algorithm generates s_{ij} values as $s_{ij} = H(tk_i||H(R||tk_j)) \mod z$.

(3) Then, it decides whether it will accept a given token pair (tk_i, tk_j), as watermarked or not by checking if the following statement holds: $(f_i - f_j) \mod s_{ij} \leq t$.

(4) After finding which pairs are watermarked, it checks whether their number is over the *minimum number of pairs*, k, needed to conclude that D'_w is watermarked by the owner, and returns $accept$ (verified) or $reject$, accordingly.

4) Probabilistic Analysis of False Positives: In this section, we develop a statistical bound in the form of the closed form expression derived from Markov's inequality theorem, to demonstrate that the false positive probability (i.e., accepting a dataset as watermarked when it is not) goes to zero if we increase the minimum number of pairs k that has to be accepted, or if we decrease the threshold t to accept a pair as watermarked.

Recall that the m-th token pair $\{tk_i, tk_j\} \in L_{wm}$ is accepted as watermarked, if $(f_i - f_j) \mod s_{ij} \leq t$. We represent the probability that this "watermarking statement" holds as $P(X_m = 1) = p_m$, for $m = 1, ..., n$. The value of p_m depends on t. The logic is that if t is zero, the false negatives will increase. Let us assume that p_m's follow a `Uniform[0,1]` distribution. The probability of having at least k successes in n trials can be written as $P(S_n \geq k) = \sum_{i=k}^{n} P(S_n = m)$. We now study the behavior of $P(S_n \geq k)$ depending on the behavior of t and k by using the *Sandwich Rule* and Markov's upper bound obtained by its inequality theorem $P(S_n \geq k) \leq \frac{\mu}{k}$, where $\mu = \sum_{m=1}^{n} p_m$ is the mean of S_n. Our analysis shows that if we decrease t, the probability of accepting a dataset as watermarked will decrease to zero and if we increase k, it will be hard to "accept" a dataset as watermarked. For further details, see the full version [61].

Algorithm II: Watermark Detection

Input: $D'_w, L_{sc} = \{L_{wm}, R, z\}, k, t$
Output: $accept/reject$
$D_w^{hist} =$ **Preprocess**(D'_w) $count = 0$, $result = reject$
foreach $\{tk_i, tk_j\} \in L_{wm}$ **do**
  if ***Found***(tk_i, tk_j, D_w^{hist}) **then**
    $s_{ij} = H(tk_i||H(R||tk_j)) \mod z$
    if $(f_i - f_j) \mod s_{ij} \leq t$ **then**
      $count++$
    end
  end
end
if $count \geq k$ **then**
  $result = accept$
end
Result: result

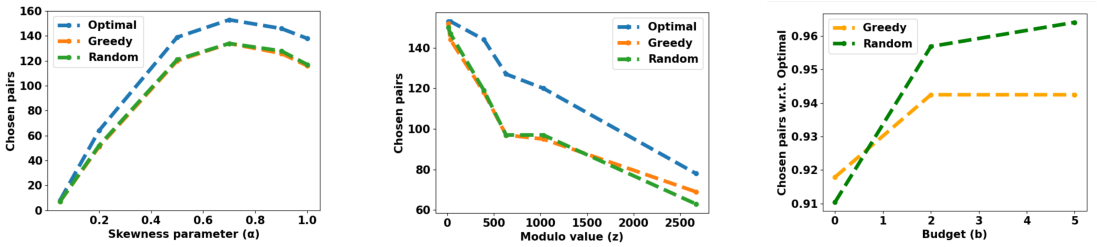
IV. Experimental Evaluation

All of our experimental results are produced on a standard laptop machine with dual-core Intel Core(TM) i7 − 5600U CPU 2.5GHz, 16.00 GB RAM, 64-bit OS, and implemented in Python language. We deployed SHA256 as a hash function.

A. Synthetic Experiments

For our synthetic experiments, we generated synthetic datasets using a $power - law$ distribution [64] with different skewness values α as $[0.05, 0.2, 0.5, 0.7, 0.9, 1]$. The sample size is $1M$ and the number of tokens is $1K$ for each different α value. When α is 0, it is a uniform distribution in which there are no eligible tokens to watermark. When α is 1, the original dataset D_o is skewed with a very long tail with almost equal values. In this setting, we evaluate how the parameters (a modulo value z, a budget b, and skewness parameter α) are affecting the number of chosen pairs for watermarking and the performance of *optimal*, *greedy*, and *random* approached in terms of number of chosen pairs.

Figure 2a shows the correlation between skewness of a dataset α and the size of chosen pairs when budget $b = 2$ and modulo value $z = 1031$. When a dataset is almost uniform (i.e., $\alpha = 0.05$), the solutions can select very few pairs since there are not many eligible items (i.e., the upper and lower boundaries are not enough, in fact many of them are 0). When α starts increasing, the differences between the frequencies of tokens increase. Thus, the number of eligible items increases which also affects the number of chosen pairs under a given budget. However, at some point (i.e., $\alpha = 0.7$), the number of chosen pairs decreases due to the tail of (histogram) frequencies becoming uniform. The same figure



(a) Effect of different skewness parameters (α) on chosen pairs by *Optimal*, *Greedy*, and *Random*.

(b) Effect of different modulo values (z) on chosen pairs by *Optimal*, *Greedy*, and *Random*.

(c) Chosen pairs by *Greedy* and *Random* with respect to *Optimal*.

Fig. 2: Effects of parameters on the size of chosen pairs for watermarking.

Dataset	Size	Token	Distinct Tokens	$\lvert L_e \rvert$	Optimal	Greedy	Random	Gen (sec)	Detect (sec)
Chicago Taxi [62]	9.68GB	`Taxi ID`	6573	33308	805	770	773	182.51	0.609
eyeWnder [58]	247MB	`URL`	11479	257	38	33	31	420.81	0.053
Adult [63]	4MB	`Age`	73	72	21	20	17	0.03	0.001

TABLE II: Validation results on real world datasets. **Dataset**: Dataset used **Size**: The size of original dataset. **Token**: Definition of the token (e.g., the name(s) of the attributes). **Distinct Token**: The number of distinct tokens. $|L_e|$: The number of eligible pairs. **Optimal**: The number of chosen pairs by the optimal matching. **Greedy**: The number of chosen pairs by the greedy matching. **Random**: The number of chosen pairs by the random matching. **Gen**: Running time of watermark generation. **Detect**: Running time of watermark detection.

shows the superior performance of the optimal solution. The gap between optimal and the heuristics is around 20% for most α values while the two heuristics perform similar the one with the other (0.02% in average).

Figure 2b illustrates how the modulo value z affects the size of chosen pairs. When we pick smaller modulo value z, we would have a higher number of chosen pairs. The reason is that a smaller z leads to smaller remainders s_{ij} that need to be eliminated, thereby yielding more selected pairs within a given budget b. When z is very small (i.e., 10), the three approaches are very close (also see Section V-A for the effect of z in terms of security). However, when z increases, our optimal approach selects many more pairs than greedy and random. Figure 2c shows how the budget selection affects the performance comparison between the heuristics and the optimal. We set modulo value $z = 1031$ and use the dataset with the skewness $\alpha = 0.7$. When we increase the budget b, the heuristics get closer to the optimal performance. This is expected since even the optimal algorithm cannot select more than all the eligible pairs and with a large budget even the heuristics can approach that.

1) Limit of z: We stated that z is selected from Z^+; however, by analyzing the frequency histogram we can derive the upper and lower boundaries. Since the minimum value (lower bound) z can take is 2, we delve into investigation of the upper bound of z. Note that since the token with the highest frequency has the upper bound of infinity, there will be *at least* one pair that could be used for watermarking. Assume a watermarking pair candidate (tk_i, tk_j). Their frequencies, f_i and f_j, are changed such that $(f_i^w - f_j^w) \mod s_{ij} \equiv 0$. To have an upper bound for z, let us investigate which pair of tokens results in the highest difference. If we can determine the highest difference, say r_{max}, then r_{max} can be assumed as the upper bound for z since it is the highest remainder. Now, considering D_o^{hist}, the highest difference is between tk_0 (the token having the highest frequency) and $tk_{|D_o^{hist}|-1}$ (the token having the lowest frequency). That means that the largest remainder for any pair is $r_{max} = (f_0^0 - f_{|D_o^{hist}|-1}^0)$. Thus it is natural to accept that the upper bound of z is r_{max}. To conclude, z can be chosen from $(2, r_{max})$. Overall, r_{max} can be calculated as $\forall f_i, f_j \in D_o^{hist}$ s.t. $f_i \geq f_j$; $r_{max} = \max(\{f_i - f_j\})$. Hence, the upper bound for z is calculated. However, note that this value can be small and can be an advantage to an attacker. As discussed in Section IV-A, z affects the number of chosen pairs; thus, it correlates with the mix of possible attacks and is use-case scenario dependent. We plan to investigate this observation theoretically and experimentally in terms of security, robustness, and utility in the future.

2) Limit of t: Another critical parameter is $t \in [0, s_{ij} - 1]$. Note that since s_{ij} has an upper bound as $z - 1$, the highest value assigned to t is $z - 1$. While in our experimental study we chose t as a constant value, t could be also a percentage. Assume that an owner wishes to state that it wants 50% of error tolerance. Now, setting $t = s_{ij} \times 0.5$ states that a pair, say tk_i and tk_j, will be accepted as a watermarked if $(f_i - f_j) \mod s_{ij} \leq s_{ij}/2$. Thus, t represents the robustness level an owner desires. For instance, if $t = 0$ then the watermark becomes fragile since it cannot tolerate any changes in D_w, thus missing watermarked pairs (i.e., high false negatives). On the other hand, when $t = 100$, it is more robust and can tolerate modifications in D_w; however, it also means that it accepts more false positives (see also Section III-B4).

B. Validation Using Real World Datasets

Next we apply *FreqyWM* to three real world datasets from different domains: (1) `Chicago Taxi` dataset [62]; (2) A real click-stream dataset logging the URLs visited by a group of users of the `eyeWnder` advertisement detection add-on [58]; (3) `Adult` dataset [63]. Our intention is to validate our main conclusion using real data from the previous evaluation with synthetic data, i.e., that the heuristic approaches perform well enough compared to the optimal. A second evaluation objective is to report the real processing time on an ordinary machine for generating and detecting the watermark using these real datasets.

For our watermark generation, we set the modulo value $z = 131$ and the budget $b = 2$. We run our algorithm 30 times and take the mean of total computations. Table II presents our validation results. `Taxi ID`, `URL`, and `Age` were chosen as tokens for `Chicago Taxi`, `eyeWnder`, and `Adult`, respectively. After generation, for `Chicago Taxi`, `eyeWnder`, and `Adult` datasets, our optimal solution chose 805, 38, and 21 pairs, respectively. Considering the heuristic approaches, greedy chose 770 pairs for `Chicago Taxi`, 33 pairs for `eyeWnder`, and 20 pairs `Adult` while random chose 773, 31, and 17 pairs, for `Chicago Taxi`, `eyeWnder`, and `Adult`, respectively. Running times of computations for watermark generation on the `Chicago Taxi`, `eyeWnder`, and `Adult` datasets were 182.51 secs, 420.81 secs, and 0.03 secs, respectively (where we exclude histogram and watermarked data generations). For watermark detection, the total detection time for each watermarked datasets was less than 1 sec. As it can be interpreted from Table II, the number of chosen pairs increases when the number of eligible pairs increases. For instance, while `eyeWnder` has more distinct tokens (11479) than `Chicago Taxi` has (6573), `eyeWnder` has fewer eligible pairs (257) than `Chicago Taxi` has (33308). Thus, *FreqyWM* has selected more pairs for `Chicago Taxi` (805) than it selected for `eyeWnder` (38).

C. Watermarking Multi-Dimensional Data

During our discussion so far, we set the token as a single attribute. However, as we previously stated, a token does not necessarily need to be restricted to a single attribute of a multi-dimensional dataset. Therefore, a token can be also defined as combination of more than one attributes (e.g., `[Age, WorkClass]`) in the `Adult` dataset. We ran *FreqyWM* on such token represented as `[Age, WorkClass]` with the same parameter setting in Section IV-B and the number of tokens (i.e., distinct `[Age, WorkClass]` attributes in the real dataset) were 481. The size of pairs chosen for watermarking was 20. With multi-dimensional data removing a token appearance is as simple as with single-dimensional data. Increasing, however, a token's frequency is more involved. The reason is that just repeating the value of the token would leave other fields not being part of the token with a value to be set, e.g., all the other fields beyond Age and WorkClass in the `Adult` dataset. A naive solution would be select a random appearance of the token and copy its other fields every time that an additional instance of the token needs to be added to the watermarked dataset. This, however, could create semantic inconsistencies if there are constraints to be respected for individual attributes or combinations of them. Making sure that added appearances of a token do not lead to semantic inconsistencies that, in addition to degrading the quality of the data, could also give away the existence of a watermarked pair to an attacker. This analysis requires domain knowledge about what each dataset represents. Such knowledge, however, is orthogonal to all previous steps of our algorithms and, thus, can be appended as a last step based on one's domain knowledge of the data whenever a token's frequency needs to be increased. We are currently investigating them and the effect of *FreqyWM* on data utility of such dataset with unique attributes as it is difficult to determine as addressed by [23].

D. Comparison to Related Works

As stated previously, we cannot directly apply (numerical) database watermarking to datasets similar to the ones we used for validation. However, one naive approach would be to convert a dataset to a numerical representation (e.g., a histogram) and watermark this numerical representation. In a nutshell, the histogram of a given dataset based on a predefined token is generated and then, the histogram is treated as a two dimensional database consisting of primary keys which are the tokens and an attribute consisting of integer values which are the frequencies. Later, a database watermarking is employed on this histogram. Then as in *FreqyWM* data transformation (e.g., removing/adding tokens) occurs according to the (new) watermarked histogram computed by the database watermarking. However, applying a numerical database watermarking is not really straightforward since it will distort the underlying dataset unexpectedly as a result of change in histogram data (e.g., cosine similarity) and would require modification in their watermarking techniques (e.g., how to create a watermarked dataset from the watermarked numerical representation). However, since this is the closest and simplest approach, we compare against it.

To actualize the above approach, we considered two numerical database watermarkings: 1) Shehab et al. [35] (referred as WM-OBT) due to partitioning approach (i.e., grouping tokens before watermarking) similar to *FreqyWM*; 2) Li et al. [30] (referred as WM-RVS) due to being one of the most recent reversible watermarking schemes introducing very small distortion compared to other same family of watermarkings.

More specifically, WM-OBT follows a data partition approach in which a watermark, defined as a bit sequence, is inserted on a group of partitions. Each data partition is filled by tokens and the frequencies of the tokens in each partition are modified/distorted by solving a minimization (if a watermark bit is 0) or maximization (if a watermark bit is 1) problem via a genetic algorithm [65], in which the objective function is in the form of a sum of sigmoid functions. WM-RVS treats each numeric value individually and changes its decimal part by selecting the random least significant position based on the watermarking key/bit and attributes. Furthermore,

to apply WM-OBT and WM-RVS on a histogram generated from a dataset, we had to adjust them such that their solutions produced are integers since a frequency count cannot be a decimal value.

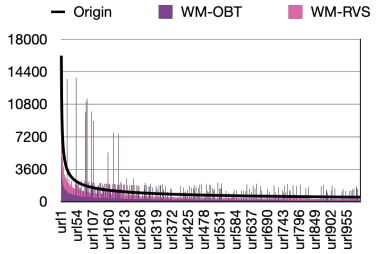


Fig. 3: Comparisons of the watermarked histograms generated from WM-OBT (purple color) and WM-RVS (fuchsia color) w.r.t. the original data histogram (black color) for the synthetic dataset with dummy token names.

For comparison, we investigate them based on two constraints: 1) change in the original histogram after watermarking (i.e., cosine similarity with watermarked histogram), and 2) the ranking of the tokens after watermarking.

We ran *FreqyWM*, WM-OBT, and WM-RVS on our synthetic data with skewness parameter 0.5 (with $1K$ distinct tokens and $1M$ sample size) where we set $b = 2$, and $z = 131$ for *FreqyWM*. We set parameters for WM-OBT and WM-RVS such that the parameters are proportional to the experimental settings of Shehab et al. [35] and Li et al. [30]. For WM-OBT, we use genetic algorithm (GA) technique for optimization [65] where we fix the number of partitions as 20 (where each partition has around 50 tokens), watermark bit sequence as $[1, 1, 0, 1, 0]$, condition as 0.75, and we allow the change (constraint) between $[-0.5, 10]$. The decoding threshold minimizing the probability of decoding error is calculated as 0.0966. For WM-RVS, we use the same bit sequence as in WM-OBT without creating it from the chaotic encryption. Also, let us note that WM-OBT took more than 30 minutes to run for such a small size dataset due to its optimization while WM-RVS was in the order of seconds. Figure 3 visualizes how the watermarked data histograms look like with respect to the original data histogram after applying WM-OBT and WM-RVS based on the experiments.

Similarity. In *FreqyWM*, even with 2% budget, the similarity between the original histogram and the watermarked histogram is 99.9998%, indicating that not all the budget was exhausted. On the other hand, for WM-OBT and WM-RVS, the similarities are 54.28% and 96%, respectively. The mean and standard deviation of the changes introduced to the histogram by WM-OBT are 444 and 855.91, respectively while they are -69.43 and 414.10 for WM-RVS, respectively.

Ranking. Another important evaluation is to compare the ranking of the tokens in the histograms under WM-OBT and *FreqyWM*. *FreqyWM* by definition maintains the ranking of tokens. Preserving the ranking allows us not to sacrifice the utility of a dataset, e.g., preserving the popularity of URLs. However, after our analysis, we observed that WM-OBT and WM-RVS changed the ranking of 998 and 987 out of the total 1000 tokens, respectively!

The results on similarity and ranking support our claim that applying a numerical database technique on histogram data would result in unexpected and uncontrolled distortion that seriously undermines the utility of the original data.

V. Security and Robustness Analysis

This section discusses the security and robustness of our *FreqyWM* method against four attacks: **guess**, **sampling**, **destroy**, and **re-watermarking (false-claim)** attacks which are well-known attacks in watermarking as studied by [23].In order to measure the robustness against sampling and destroy attacks, we run our optimal solution on a dataset where the skewness parameter $\alpha = 0.5$ (with $1K$ distinct tokens and $1M$ sample size), unless stated otherwise, the modulo value $z = 131$, and the budget $b = 2$ and it selected 139 pairs for watermark. We run the experiments for 100 times and compute the average accepted pairs over all repetitions.

A. Guess (Brute-Force) Attack

In the guess attack, the probabilistic polynomial time adversary tries to guess the watermark, i.e., the secret embedded in the data. This is possible only if it can figure out a subset of token pairs $\{tk_i, tk_j\}_l$ (where $\binom{|D_w^{hist}|}{2} \geq l \geq k$) based on the watermarked data D_w, the random value R, and the modulo value z where the watermark detection algorithm based on these inputs (for some fixed k and t) returns `accept`. Assuming that the hash function is collision resistant, R is random, and z is an integer, the probability of the attacker being successful can be formally defined as:

$$Pr[R \leftarrow \{0,1\}^{\lambda}; (D_w, L_{sc} = \{\{tk_i, tk_j\}_{|L_{wm}|}, R, z\}) \leftarrow WmGenerate(D_o, b) : \mathcal{A}(D_w) \rightarrow L'_{sc} = \{\{tk_i, tk_j\}_l, R^*, z^*\}| WmDetect(D_w, L'_{sc}, k, t) = 1] \leq \mathsf{negl}(\lambda)$$

Considering the typical parameter values, the probability of success becomes negligible.

B. Sampling Attack

In this attack, $\mathcal{A}$ copies a random subsample from the watermarked dataset D_w in an attempt to exploit (pirate/steal) it while hoping that the watermark won't be detectable within the extracted sample. The attack is run for different sample sizes from 1% to 90%, extracted from the original watermarked dataset D_w. For each percentage and subsample we apply the detection algorithm and compute the percentage of accepted pairs. Also, for each subsample detection experiment, we deploy different values of the threshold t for accepting a pair as watermarked as $t = \{0, 1, 2, 4, 10\}$. The attack scenario is as follows: $\mathcal{A}$ randomly selects $x\%$ of D_w where x defines the percentage for the sampling attack (e.g., 1) as a subsample size of $1M \times \frac{x}{100}$. When the owner suspects the dataset (possible subsampled), it scales it up to the size of D_w by multiplying the frequency counts by $\frac{100}{x}$ by using its info from the (original) watermarked dataset (e.g., via info

added to its metadata). For instance, for 1% sampling attack, a subsample would have total of $1M \times 0.01 = 10K$ where each f_i is multiplied by approximately 0.01. Note that if the sample size is greater than the number of distinct tokens, which is the number of items in D_w^{hist}, the sample will have all the distinct tokens with a high chance. This also means that all the chosen watermarked pairs are in the subsample. Our results show that the size of the extracted subsample does not greatly affect the number of accepted pairs if it is greater than the number of unique tokens ($1K$). Since the frequencies of the tokens vary, the value of t does affect the result of the detection. For example, with $t = 0$ the detection algorithm can detect around 36% (in average) of the watermarked pairs. When t increases from 1 to 10, the performance of the detection increases (in average) from 72% to 99.5%.

Let us now see the results when the size of the extracted subsample is very low, so that it might not contain at least 1 token from the total $1K$ of unique tokens that the original watermarked dataset has. Figure 4 shows the results for sample size proportions between 0.0007% and 0.5%. Observe that if the sample size is greater than $5\times$ the number of unique tokens ($1K$), the detection algorithm stabilizes its performance for detecting the watermark. Below $2\times$ ($2K$), the performance starts to decrease with higher velocity. In these extreme cases, the detection algorithm will have more difficulties to detect the data as watermarked. However, the utility of the data is highly degraded since the subsample sizes are very small compared to the original size of $1M$ tokens. This causes a small number of distinct tokens to be found in the subsample.

Effect of modulo bases. As seen previously, t is crucial for detecting whether a pair is watermarked. For small values of t to be sufficient to fend off sampling attacks, the remainders need to be small numbers that are covered by t. One way to achieve this is by ensuring that the modulo bases used (i.e., the s_{ij}'s) are relatively small numbers when compared to the actual appearance frequencies of watermarked pairs. When this does not apply, the method will of course fail. For instance, assume a watermarked pair involving frequencies $f_i = 540, f_j = 440$ which under base $s_{ij} = 100$ leave a remainder of 0. W.l.o.g, lets assume that a 50% frequency attack leads to a dataset with $f_i = 270, f_j = 220$ which leads to a remainder of $(270 - 220) \mod 100 \equiv 50$. Now if t is chosen smaller than 50 then the watermarked pair will not be detected. The reason is that mod 100 leaves large remainders when applied to f_i and f_j that are in the same order with 100. In our experimental results f_i's were always much larger numbers than the employed s_{ij}, thereby, even small t's would detect a pair under a sampling attack. To determine the optimal t and how robust it is against attacks, a further investigation is needed as it depends on various parameters such as z, s_{ij} as well on the mix of expected attacks as discussed later. Note that our experimental results show that s_{ij} values are ~ 2 order of magnitude smaller than z. Furthermore, we also tested the sampling attack in other watermarked datasets with different values of the skewness parameter and obtained similar results.

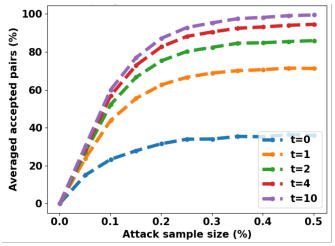


Fig. 4: Sampling attack results with very low sample size and $\alpha = 0.5$.

C. Destroy Attack

In this case the attacker $\mathcal{A}$ tries to damage the watermark. The no-security-by-obscurity principle [66] allows $\mathcal{A}$ to know that it can destroy the watermark in a way that it cannot be detected by the owner. $\mathcal{A}$ computes the histogram of watermarked data D_w. $\mathcal{A}$ modifies the frequencies of tokens as it pleases by allowing re-ordering (changing the popularity/rank of the tokens) or without allowing re-ordering. We define these two attacks and discuss *FreqyWM*'s robustness against them.

1) Destroy Attack without re-ordering: In this attack type, $\mathcal{A}$ can modify the frequencies without changing the order of frequencies. We introduce two types: (1) attacker changes the frequencies randomly by the given boundaries and (2) attacker changes the frequencies by (at most) some percentage.

Changing the frequencies randomly within the boundaries. $\mathcal{A}$ calculates the boundaries for each token. Then, $\mathcal{A}$ chooses a random value r_i for each tk_i as $r_i \leftarrow (-l_i, u_i)$. $\mathcal{A}$ changes the frequency of tk_i and updates u_{i+1} of tk_{i+1} by r_i.

Changing the frequencies by (at most) some percentage. $\mathcal{A}$ changes the frequencies of tokens up to some percentage (e.g., 1%). To illustrate, $\mathcal{A}$ calculates the boundaries as u_i and l_i for each tk_i where it sets the percentage to 1%. It calculates $u_i' = floor(u_i \times 0.01)$ and $l_i' = floor(l_i \times 0.01)$. Then it gets a random value r_i between $(-l_i', u_i')$. It hereby changes tk_i by at most $\pm 1\%$. After every change ($f_i' = f_i + r_i$), the boundary of the next element is updated. Thus, the attack never changes the ranking/ordering since l_i' and u_i' are already in the boundaries.

Figure 5 shows how robust *FreqyWM* is against these two destroy attacks. We compare the success rate (the percentage of accepted token pairs given threshold for accepting a pair t) of detection algorithm with respect to modified watermarked data after the attacks. We also include in the figure a second dataset of skewness $\alpha = 0.7$ *that does not carry the watermark*, and report on how many of its pairs would be falsely verified for different values of t. For an attack in which the frequencies are changed by (at most) some percentage (represented by the red line in the figure), *FreqyWM* can detect around 90% of the pairs when $t = 0$. When t is increased, after a point where $t \geq f_i - f_j \mod s_{ij}$, the success rate converges at around 90%. For an attack where the frequencies are changed randomly within the upper and lower frequency boundaries (green line in Figure 5), *FreqyWM* can detect more than 35% of the pairs when $t = 0$. Note that the latter is more powerful than the former attack. There is a direct proportion

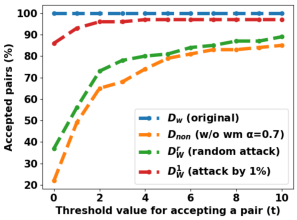


Fig. 5: Percentage of verified pairs for the following datasets: (1) D_w : the original watermarked dataset $\alpha = 0.5$ without any attack/modification, (2) D_{non} : a non-watermarked dataset defined over the same token space but with $\alpha = 0.7$, (3) D^r_w : D_w after attacked by random attack without reordering, (4) D^1_w : D_w after attacked by changing frequencies at most 1%.

between t and the success rate. As shown, the success rate reaches to 90% when t goes to 10.

From Figure 5, we can interpret in what parameter setting false negative (rejecting a watermarked pair) and false positive (accepting a pair as watermarked while it is not) can be avoided. Thus, the watermarking detection algorithm can successfully detect a watermarked dataset attacked and reject a dataset that was not watermarked. For instance, the rate of false positive increases when the threshold for accepting a pair t increases while the minimum number of accepted pairs for detection k decreases which is the area under the results of the dataset (not watermarked) with a different skewness parameter (the area under the orange line). On the other hand, the rate of false negative increases when the threshold accepting a pair t decreases while threshold for detecting a watermark k increases which is the area above the results of the attack without re-ordering (the area above the green line) if we consider a very strong attack. To avoid false negatives/positives, convenient parameter settings (i.e., t and k) for detecting a watermark lie between these two areas (between the orange and the green lines in Figure 5). However, if a weaker attack (changing the frequencies by some percentage) is considered, the range of these parameters increases (the area between the red and orange line). Hence, the detection algorithm can detect a watermarked dataset and reject a dataset not watermarked by the owner with a careful parameter setting. For instance, adjusting t (and k) based on the nature of the data and the specific application context can enable us to reduce the false positives/negatives. This is an interesting future work.

2) Destroy Attack with re-ordering: In this attack type, an attacker $\mathcal{A}$ can modify the frequencies as it pleases without observing any ordering restrictions. Note that this attack introduces more noise than the attack without re-ordering which reduces the usability of watermarked data D_w. $\mathcal{A}$ modifies the frequencies with various percentages $[10\%, 30\%, 50\%, 60\%, 80\%, 90\%]$ where the success rates are $[94\%, 88\%, 82\%, 79\%, 78\%, 76\%]$ respectively. *FreqyWM* can detect the watermark with 76% chance up to modifications of 90% in frequencies approximately (where $t = 4$).

D. Re-watermarking/ False-Claim Attack

This attack is mounted by an attacker $\mathcal{A}$ creating a new watermark on the watermarked data D_w, generated by an honest owner. $\mathcal{A}$ generates its own watermarked data by simply inserting D_w into the watermark generation algorithm as data to produce $D^{\mathcal{A}}_w$. Then $\mathcal{A}$ can present $D^{\mathcal{A}}_w$ and claim the ownership of $D^{\mathcal{A}}_w$ (since $\mathcal{A}$ can prove its ownership claim by introducing its watermarking secret list $L^{\mathcal{A}}_{sc}$). This attack creates a dispute since both the real owner, who created D_w, and $\mathcal{A}$ have proofs of their ownerships. The dispute can be arbitrated by introducing a judge (a trusted third party as suggested by [67]) to the watermarking scheme. Both parties, $\mathcal{A}$ and the real owner, introduce their secrets and their watermarked data. $\mathcal{A}$ sends its secrets $L^{\mathcal{A}}_{sc}$ and its watermarked data $D^{\mathcal{A}}_w$. The real owner sends its secrets L_{sc} and its watermarked data D_w. The judge computes watermark detection algorithm on each received data for each secret which creates four outputs. The judge compares these results and identifies the real owner since only the secret of the real owner can produce accept on both data. To show practicality of our defense against the re-watermarking attack, we implemented the attack above. Our results show that the first watermark is detected with 92% on $D^{\mathcal{A}}_w$ under $t = 0$. The attacker's only way to succeed is to perform successful guess or destroy attack which it cannot perform as shown previously.

VI. Discussion

We propose possible adjustments to *FreqyWM* for more sophisticated properties and discuss some corner cases below:

• *Incremental FreqyWM.* In the literature, there exist watermarking techniques that allow to update a watermark on a dataset without computing insertion from scratch [57]. We believe that an incremental FreqyWM can be built on top of dynamic maximum weighted matching [68, 69] works but we leave such investigations to future work.

• *Multi-watermarks.* There are at least two reasons that someone may want to watermark a file multiple times: 1) a legitimate one, e.g., to track the provenance of dataset as it passes from a distributed pipeline, in which a watermark can be added to signify the completion of each one of the processing stages, or to have a chronological order in the versions; 2) a malicious one, to falsely claim ownership via a re-watermarking attack as already discussed in Section V-D.

Non-withstanding the motivation, we have run an experiment by running 10 watermark insertions assuming a budget $b = 2$ for each iteration on a sample dataset with skewness $\alpha = 0.5$. to calculate three effects: 1) the discrepancy between the original asset and a final one; 2) dataset feature analysis; and 3) the effect of successive watermarking on machine learning model accuracy.

Discrepancy. The resulting similarity between the original (histogram) and the latest watermarked version is 0.003%. As it is evident, FreqyWM did not introduce 20% but rather very tiny distortion.

Feature. We analyze the change of various features of the `eyeWnder` dataset that commonly deployed after the water-

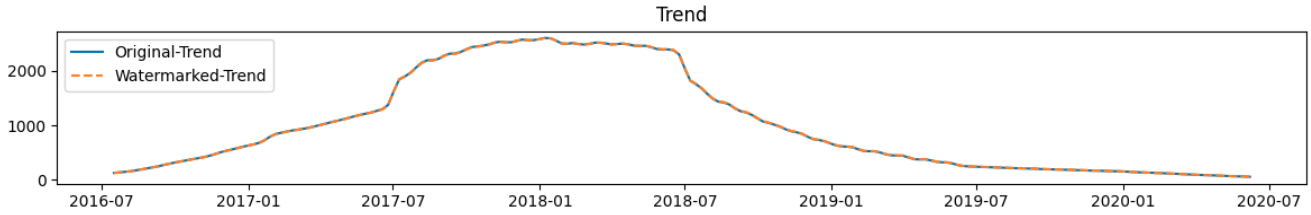


Fig. 6: Analysis of the effect of multi-watermarks (i.e., 10 watermarks) on the trend analysis of `eyeWnder`.

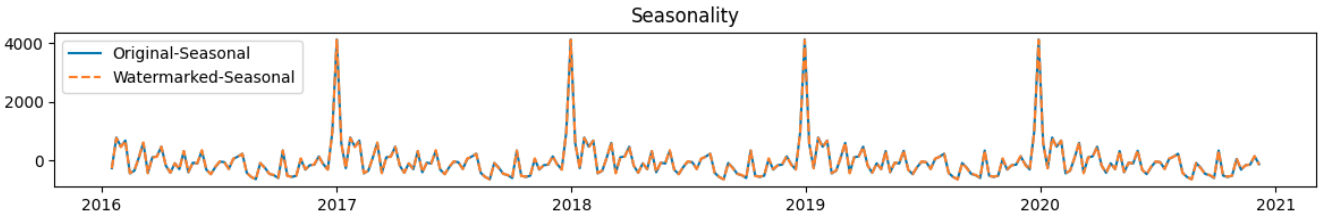


Fig. 7: Analysis of the effect of multi-watermarks (i.e., 10 watermarks) on the seasonality analysis of `eyeWnder`.

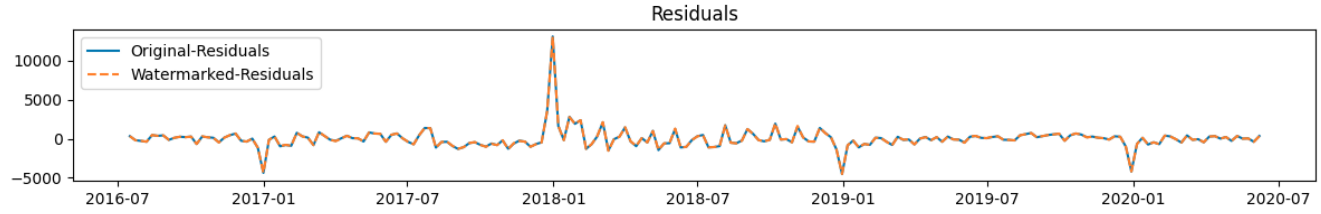


Fig. 8: Analysis of the effect of multi-watermarks (i.e., 10 watermarks) on the residual analysis of `eyeWnder`.

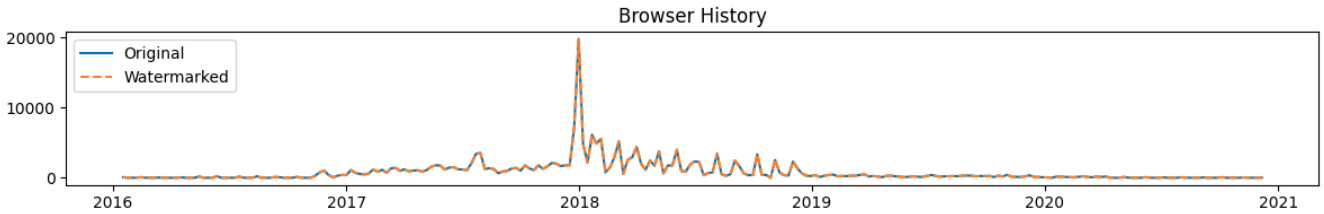


Fig. 9: Analysis of the effect of multi-watermarks (i.e., 10 watermarks) on the browser history analysis of `eyeWnder`.

marking. Figure 9 shows the change in the browser history while Figures 6, 7, and 8 show the effect of watermarking when the dataset is analyzed in terms of trends, seasonality, and residuals, respectively. As shown by the figures, multi-watermarks introduced very insignificant change to the dataset.

Accuracy. We also analyze the effect of multi-watermarks on an ML model accuracy We use the `eyeWnder` dataset. We implement a sequential data analysis approach using TensorFlow to predict the next URL in a sequence, utilizing a dataset of timestamped URLs. The model consists of an embedding layer, LSTM layers, and a sigmoid output layer. We train the model with 10 epochs and with 128 batch-size. The model achieved an accuracy of 82.33% when trained on the original `eyeWnder` dataset whilst achieving an accuracy of 82.34% when it is watermarked. The model trained on the watermarked dataset has slightly better accuracy. We suspect that this is due to increase in the size of the dataset (i.e., the watermarked one has 140 more URLs). While our initial results are promising, we plan to extensively investigate the effect of *FreqyWM* on data usability using different ML models for more concrete reductions.

• *Challenging datasets.* Apart from datasets with close to uniform frequencies, *FreqyWM* can also be challenged when the range of token values is too wide, e.g., sales' datasets with many decimal values, resulting to very few (if any) repetition of the same value. One natural solution to this is to first bucketize (cluster) the widely ranged data and then apply *FreqyWM* at the level of the bucket as opposed to the exact token value.

VII. Conclusions and Future Work

We proposed *FreqyWM*, a novel frequency watermarking technique for protecting the ownership of data in the emerging new data economy. We analysed the performance of *FreqyWM*

and showed how *FreqyWM* can encode watermarks with minimal distortion on the original data, provided that the data has sufficient variability in terms of token frequencies. We analysed *FreqyWM*'s robustness to generic attacks. *FreqyWM* is applicable to large numbers of tuples sold in wholesale manner in modern DMs. An interesting, yet challenging, research direction is to consider how to watermark small sets or even individual tuples used in distributed data operations such as replication and remote hosting and/or query execution. We are currently looking at more attack scenarios and at devising systematic procedures for optimizing the parameters and also how to apply *FreqyWM* to multidimensional datasets by overcoming the challenges mentioned in Section IV-C. We also investigate integrating data privacy (e.g., differentially-private fingerprinting [70]).

REFERENCES

[1] S. A. Azcoitia and N. Laoutaris, "A survey of data marketplaces and their business models," *SIGMOD Rec.*, vol. 51, no. 3, pp. 18–29, 2022. [Online]. Available: https://doi.org/10.1145/3572751.3572755

[2] A. Lutu, D. Perino, M. Bagnulo, E. Frias-Martinez, and J. Khangosstar, "A characterization of the covid-19 pandemic impact on a mobile network operator traffic," in *Proceedings of the ACM Internet Measurement Conference*, ser. IMC '20. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3419394.3423655

[3] D. Evans, V. Kolesnikov, and M. Rosulek, "A pragmatic introduction to secure multi-party computation," *Found. Trends Priv. Secur.*, 2018. [Online]. Available: https://doi.org/10.1561/3300000019

[4] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford University, USA, 2009. [Online]. Available: https://searchworks.stanford.edu/view/8493082

[5] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: Definitions and challenges," in *Theory of Cryptography Conference, TCC*. Springer, 2011. [Online]. Available: https://doi.org/10.1007/978-3-642-19571-6_16

[6] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted execution environment: What it is, and what it is not," in *TrustCom/BigDataSE/ISPA*. IEEE, 2015. [Online]. Available: https://doi.org/10.1109/Trustcom.2015.357

[7] Y. Li, D. Ghosh, P. Gupta, S. Mehrotra, N. Panwar, and S. Sharma, "PRISM: private verifiable set computation over multi-owner outsourced databases," in *SIGMOD: International Conference on Management of Data, Virtual*. ACM, 2021. [Online]. Available: https://doi.org/10.1145/3448016.3452839

[8] R. Poddar, T. Boelter, and R. A. Popa, "Arx: An encrypted database using semantically secure encryption," *Proc. VLDB Endow.*, 2019. [Online]. Available: http://www.vldb.org/pvldb/vol12/p1664-poddar.pdf

[9] N. Anciaux, L. Bouganim, P. Pucheral, I. S. Popa, and G. Scerri, "Personal database security and trusted execution environments: A tutorial at the crossroads," *Proc. VLDB Endow.*, 2019. [Online]. Available: http://www.vldb.org/pvldb/vol12/p1994-anciaux.pdf

[10] X. Ren, L. Su, Z. Gu, S. Wang, F. Li, Y. Xie, S. Bian, C. Li, and F. Zhang, "HEDA: multi-attribute unbounded aggregation over homomorphically encrypted database," *Proc. VLDB Endow.*, 2022. [Online]. Available: https://www.vldb.org/pvldb/vol16/p601-gu.pdf

[11] W. Zhou, Y. Cai, Y. Peng, S. Wang, K. Ma, and F. Li, "Veridb: An sgx-based verifiable database," in *SIGMOD: International Conference on Management of Data*. ACM, 2021. [Online]. Available: https://doi.org/10.1145/3448016.3457308

[12] P. Jougleux, "Data ownership (and succession law)," in *Facebook and the (EU) Law: How the Social Network Reshaped the Legal Framework*. Springer, 2022, pp. 129–143.

[13] J. Kennedy, P. Subramaniam, S. Galhotra, and R. C. Fernandez, "Revisiting online data markets in 2022: A seller and buyer perspective," *SIGMOD Rec.*, vol. 51, no. 3, pp. 30–37, 2022. [Online]. Available: https://doi.org/10.1145/3572751.3572757

[14] R. C. Fernandez, P. Subramaniam, and M. J. Franklin, "Data market platforms: Trading data assets to solve data problems," *Proc. VLDB Endow.*, vol. 13, no. 11, pp. 1933–1947, 2020. [Online]. Available: http://www.vldb.org/pvldb/vol13/p1933-fernandez.pdf

[15] F. Banterle, "Data ownership in the data economy: a european dilemma," *EU Internet Law in the Digital Era: Regulation and Enforcement*, pp. 199–225, 2020. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3277330

[16] M. Asikuzzaman and M. R. Pickering, "An overview of digital video watermarking," *IEEE Trans. Circuits Syst. Video Technol.*, 2018. [Online]. Available: https://doi.org/10.1109/TCSVT.2017.2712162

[17] M. Begum and M. S. Uddin, "Digital image watermarking techniques: A review," *Inf.*, 2020. [Online]. Available: https://doi.org/10.3390/info11020110

[18] H. Ma, C. Jia, S. Li, W. Zheng, and D. Wu, "Xmark: Dynamic software watermarking using collatz conjecture," *IEEE Trans. Inf. Forensics Secur.*, 2019. [Online]. Available: https://doi.org/10.1109/TIFS.2019.2908071

[19] X. Zhou, H. Pang, K. Tan, and D. Mangla, "Wmxml: A system for watermarking XML data," in *International Conference on Very Large Data Bases (VLDB)*. ACM, 2005. [Online]. Available: http://www.vldb.org/conf/2005/papers/p1318-zhou.pdf

[20] R. Agrawal and J. Kiernan, "Watermarking relational databases," in *Proceedings ofInternational Conference on Very Large Data Bases, VLDB*, 2002. [Online]. Available: http://www.vldb.org/conf/2002/S05P03.pdf

[21] R. Agrawal, P. J. Haas, and J. Kiernan, "A system for

watermarking relational databases," in *ACM SIGMOD International Conference*, 2003. [Online]. Available: https://doi.org/10.1145/872757.872865

[22] T. Wang and F. Kerschbaum, "RIGA: covert and robust white-box watermarking of deep neural networks," in *WWW: The Web Conference*, 2021. [Online]. Available: https://doi.org/10.1145/3442381.3450000

[23] S. Rani and R. Halder, "Comparative analysis of relational database watermarking techniques: An empirical study," *IEEE Access*, vol. 10, pp. 27 970–27 989, 2022. [Online]. Available: https://doi.org/10.1109/ACCESS.2022.3157866

[24] N. Agarwal, A. K. Singh, and P. K. Singh, "Survey of robust and imperceptible watermarking," *Multim. Tools Appl.*, 2019. [Online]. Available: https://doi.org/10.1007/s11042-018-7128-5

[25] R. Agrawal, P. J. Haas, and J. Kiernan, "Watermarking relational data: framework, algorithms and analysis," *VLDB J.*, 2003. [Online]. Available: https://doi.org/10.1007/s00778-003-0097-x

[26] T. Ji, E. Yilmaz, E. Ayday, and P. Li, "The curse of correlations for robust fingerprinting of relational databases," in *RAID : International Symposium on Research in Attacks, Intrusions and Defenses*. ACM, 2021. [Online]. Available: https://doi.org/10.1145/3471621.3471853

[27] E. Quiring, D. Arp, and K. Rieck, "Forgotten siblings: Unifying attacks on machine learning and digital watermarking," in *IEEE European Symposium on Security and Privacy, EuroS&P*. IEEE, 2018. [Online]. Available: https://doi.org/10.1109/EuroSP.2018.00041

[28] A. Cohen, J. Holmgren, R. Nishimaki, V. Vaikuntanathan, and D. Wichs, "Watermarking cryptographic capabilities," *SIAM J. Comput.*, 2018. [Online]. Available: https://doi.org/10.1137/18M1164834

[29] X. Tang, Z. Cao, X. Dong, and J. Shen, "Pkmark: A robust zero-distortion blind reversible scheme for watermarking relational databases," in *IEEE International Conference on Big Data Science and Engineering*, 2021. [Online]. Available: https://doi.org/10.1109/BigDataSE53435.2021.00020

[30] W. Li, N. Li, J. Yan, Z. Zhang, P. Yu, and G. Long, "Secure and high-quality watermarking algorithms for relational database based on semantic," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–14, 2022.

[31] M. L. P. Gort, M. Olliaro, A. Cortesi, and C. F. Uribe, "Semantic-driven watermarking of relational textual databases," *Expert Syst. Appl.*, 2021. [Online]. Available: https://doi.org/10.1016/j.eswa.2020.114013

[32] C. Lin, T. Nguyen, and C. Chang, "LRW-CRDB: lossless robust watermarking scheme for categorical relational databases," *Symmetry*, 2021. [Online]. Available: https://doi.org/10.3390/sym13112191

[33] S. Kumar, B. K. Singh, and M. Yadav, "A recent survey on multimedia and database watermarking," *Multim. Tools Appl.*, vol. 79, no. 27-28, pp. 20 149–20 197, 2020. [Online]. Available: https://doi.org/10.1007/s11042-020-08881-y

[34] M. H. Jony, F. T. Johora, and J. F. Katha, "A robust and efficient numeric approach for relational database watermarking," in *IEEE International Conference on Sustainable Technologies for Industry 4.0 (STI)*, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9732582

[35] M. Shehab, E. Bertino, and A. Ghafoor, "Watermarking relational databases using optimization-based techniques," *IEEE Trans. Knowl. Data Eng.*, 2008. [Online]. Available: https://doi.org/10.1109/TKDE.2007.190668

[36] D. Ibosiola, B. A. Steer, Á. García-Recuero, G. Stringhini, S. Uhlig, and G. Tyson, "Movie pirates of the caribbean: Exploring illegal streaming cyberlockers," in *Proceedings of the Twelfth International Conference on Web and Social Media, ICWSM*. AAAI Press, 2018. [Online]. Available: https://aaai.org/ocs/index.php/ICWSM/ICWSM18/paper/view/17835

[37] W. Zhou, J. Hu, and S. Wang, "Enhanced locality-sensitive hashing for fingerprint forensics over large multi-sensor databases," *IEEE Trans. Big Data*, 2021. [Online]. Available: https://doi.org/10.1109/TBDATA.2017.2736547

[38] Y. Lei, Q. Huang, M. S. Kankanhalli, and A. K. H. Tung, "Locality-sensitive hashing scheme based on longest circular co-substring," in *Proceedings of the 2020 International Conference on Management of Data, SIGMOD*. ACM, 2020. [Online]. Available: https://doi.org/10.1145/3318464.3389778

[39] D. Chang, M. Ghosh, S. K. Sanadhya, M. Singh, and D. R. White, "Fbhash: A new similarity hashing scheme for digital forensics," *Digit. Investig.*, 2019. [Online]. Available: https://doi.org/10.1016/j.diin.2019.04.006

[40] C. N. K. Osiakwan and S. G. Akl, "The maximum weight perfect matching problem for complete weighted graphs is in pc*," *Parallel Algorithms Appl.*, 1995. [Online]. Available: https://doi.org/10.1080/10637199508915506

[41] Z. Galil, "Efficient algorithms for finding maximum matching in graphs," in *ACM CSUR*, 1986.

[42] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2009.

[43] E. Ayday, E. Yilmaz, and A. Yilmaz, "Robust optimization-based watermarking scheme for sequential data," in *International Symposium on Research in Attacks, Intrusions and Defenses, RAID*, 2019. [Online]. Available: https://www.usenix.org/conference/raid2019/presentation/ayday

[44] T. Ji, E. Ayday, E. Yilmaz, and P. Li, "Robust fingerprinting of genomic databases," *CoRR*, vol. abs/2204.01801, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2204.01801

[45] M. Kamran and M. Farooq, "A comprehensive survey of watermarking relational databases research," in *arXiv preprint arXiv:1801.08271*, 2018.

[46] A. S. Panah, R. G. van Schyndel, T. K. Sellis,

and E. Bertino, "On the properties of non-media digital watermarking: A review of state of the art techniques," *IEEE Access*, 2016. [Online]. Available: https://doi.org/10.1109/ACCESS.2016.2570812

[47] M. E. Farfoura, S. Horng, J. Lai, R. Run, R. Chen, and M. K. Khan, "A blind reversible method for watermarking relational databases based on a time-stamping protocol," *Expert Syst. Appl.*, 2012. [Online]. Available: https://doi.org/10.1016/j.eswa.2011.09.005

[48] Y. Li and R. H. Deng, "Publicly verifiable ownership protection for relational databases," in *Proceedings of th ACM Symposium on Information, Computer and Communications Security, ASIACCS*. ACM, 2006. [Online]. Available: https://doi.org/10.1145/1128817.1128832

[49] D. Hu, D. Zhao, and S. Zheng, "A new robust approach for reversible database watermarking with distortion control," *IEEE Trans. Knowl. Data Eng.*, 2019. [Online]. Available: https://doi.org/10.1109/TKDE.2018.2851517

[50] H. M. El-Bakry and M. Hamada, "A novel watermark technique for relational databases," in *Artificial Intelligence and Computational Intelligence - International Conference, AICI 2010, Sanya, China, October 23-24, 2010, Proceedings, Part II*, ser. Lecture Notes in Computer Science. Springer, 2010. [Online]. Available: https://doi.org/10.1007/978-3-642-16527-6_29

[51] S. M. Darwish, H. A. Selim, and M. M. El-Sherbiny, "Distortion free database watermarking system based on intelligent mechanism for content integrity and ownership control," *J. Comput.*, 2018. [Online]. Available: https://doi.org/10.17706/jcp.13.9.1053-1066

[52] Y. Zhang, B. Yang, and X.-M. Niu, "Reversible watermarking for relational database authentication," 2008.

[53] W. Wang, C. Liu, Z. Wang, and T. Liang, "FBIPT: A new robust reversible database watermarking technique based on position tuples," in *International Conference on Data Intelligence and Security, ICDIS*. IEEE, 2022, pp. 67–74. [Online]. Available: https://doi.org/10.1109/ICDIS55630.2022.00018

[54] G. Gupta and J. Pieprzyk, "Reversible and blind database watermarking using difference expansion," *Int. J. Digit. Crime Forensics*, 2009. [Online]. Available: https://doi.org/10.4018/jdcf.2009040104

[55] K. Jawad and A. Khan, "Genetic algorithm and difference expansion based reversible watermarking for relational databases," *J. Syst. Softw.*, 2013. [Online]. Available: https://doi.org/10.1016/j.jss.2013.06.023

[56] M. B. Imamoglu, M. Ulutas, and G. Ulutas, "A new reversible database watermarking approach with firefly optimization algorithm," *Mathematical Problems in Engineering*, 2017. [Online]. Available: https://doi.org/10.1155/2017/1387375

[57] C. Chang, T. Nguyen, and C. Lin, "A reversible database watermark scheme for textual and numerical datasets," in *IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, SNPD*. IEEE, 2021. [Online]. Available: https://doi.org/10.1109/SNPD51163.2021.9704991

[58] C. Iordanou, N. Kourtellis, J. M. Carrascosa, C. Soriente, R. Cuevas, and N. Laoutaris, "Beyond content analysis: detecting targeted ads via distributed counting," in *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies, CoNEXT*. ACM, 2019. [Online]. Available: https://doi.org/10.1145/3359989.3365428

[59] G. Cormode, S. Maddock, and C. Maple, "Frequency estimation under local differential privacy," *Proc. VLDB Endow.*, 2021. [Online]. Available: http://www.vldb.org/pvldb/vol14/p2046-cormode.pdf

[60] J. Katz and Y. Lindell, *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014. [Online]. Available: https://www.crcpress.com/Introduction-to-Modern-Cryptography-Second-Edition/Katz-Lindell/p/book/9781466570269

[61] D. İşler, E. Cabana, A. Garcia-Recuero, G. Koutrika, and N. Laoutaris, "Freqywm: Frequency watermarking for the new data economy," IMDEA Networks Technical Report, Tech. Rep., 2022.

[62] "Chicago Data Portal," 2022, https://data.cityofchicago.org/Transportation/Taxi-Trips/wrvz-psew.

[63] "Adult Dataset," 1996, https://archive.ics.uci.edu/ml/datasets/Adult.

[64] A. Clauset, C. R. Shalizi, and M. E. J. Newman, "Power-law distributions in empirical data," *SIAM Rev.*, 2009. [Online]. Available: https://doi.org/10.1137/070710111

[65] D. Goldberg and K. Sastry, *Genetic algorithms: the design of innovation*. Springer, 2007.

[66] A. Kerckhoffs, "A. kerckhoffs, la cryptographie militaire, journal des sciences militaires ix, 38 (1883)," in *Journal des sciences militaires*, 1883.

[67] A. Adelsbach, S. Katzenbeisser, and H. Veith, "Watermarking schemes provably secure against copy and ambiguity attacks," in *ACM workshop on Digital rights management*, 2003. [Online]. Available: https://doi.org/10.1145/947380.947395

[68] S. Behnezhad, "Dynamic algorithms for maximum matching size," in *ACM-SIAM Symposium on Discrete Algorithms, SODA*. SIAM, 2023. [Online]. Available: https://doi.org/10.1137/1.9781611977554.ch6

[69] S. Solomon, "Fully dynamic maximal matching in constant update time," in *IEEE Annual Symposium on Foundations of Computer Science, FOCS*. IEEE Computer Society, 2016. [Online]. Available: https://doi.org/10.1109/FOCS.2016.43

[70] T. Ji, E. Ayday, E. Yilmaz, and P. Li, "Differentially-private fingerprinting of relational databases," *CoRR*, vol. abs/2109.02768, 2021. [Online]. Available: https://arxiv.org/abs/2109.02768