Fingerprinting of Relational Databases for Stopping the Data Theft

Eesa Al Solami 1, Muhammad Kamran 1,*, Mohammed Saeed Alkatheiri 1, Fouzia Rafiq 2 and Ahmed S. Alghamdi 1

1 Department of Cybersecurity, College of Computer Science and Engineering, University of Jeddah, Jeddah 21959, Saudi Arabia; eaalsulami@uj.edu.sa (E.A.S.); msalkatheri@uj.edu.sa (M.S.A.); ahmedg@uj.edu.sa (A.S.A.)
2 Department of Computer Science, COMSATS University Islamabad, Wah Campus, Wah Cantt 47040, Pakistan; fozia.ciit17@gmail.com
* Correspondence: mkkamran@uj.edu.sa

Received: 4 June 2020; Accepted: 1 July 2020; Published: 4 July 2020

Abstract: The currently-emerging technology demands sharing of data using various channels via the Internet, disks, etc. Some recipients of this data can also become traitors by leaking the important data. As a result, the data breaches due to data leakage are also increasing. These breaches include unauthorized distribution, duplication, and sale. The identification of a guilty agent responsible for such breaches is important for: (i) punishing the culprit; and (ii) preventing the innocent user from accusation and punishment. Fingerprinting techniques provide a mechanism for classifying the guilty agent from multiple recipients and also help to prevent the innocent user from being accused of the data breach. To those ends, in this paper, a novel fingerprinting framework has been proposed using a biometric feature as a digital mark (signature). The use of machine learning has also been introduced to make this framework intelligent, particularly for preserving the data usability. An attack channel has also been used to evaluate the robustness of the proposed scheme. The experimental study was also conducted to demonstrate that the proposed technique is robust against several malicious attacks, such as subset selection attacks, mix and match attacks, collusion attacks, deletion attacks, insertion attacks, and alteration attacks.

Keywords: fingerprinting; information hiding; information security; machine learning; traitor tracing

1. Introduction

The recent trend in information and communication technology (ICT) has resulted in generation and storing of the digital data on a large scale in the form of audio, video, image, text, xml document, and relational databases. These data are used to extract useful knowledge that can later be used to improve the relevant products and services concerning that data. For this purpose, different techniques, such as data mining, machine learning, and some of the statistics based techniques, are applied on the data. Such applications require data sharing as owners can not work independently and are not generally experts in each domain. This demands sharing of the data among experts of various domains, such as data mining, machine learning, business intelligence, information systems, data analytics, and so on; consequently, the threat of data theft (unauthorized sharing or sale) is relevant. The recent data breaches (https://www.scmagazine.com/home/security-news/data-breach/360000-quebec-teachers-pii-possibly-compromised/, https://9to5google.com/2020/02/24/samsung-find-my-mobile-notification-data-breach/, https://threatpost.com/data-breach-occurs-at-agency-in-charge-of-secure-white-house-communications/153160/) tended to emphasize the importance of tracking the sources of the respective data breaches. In this context, it is necessary to trace the guilty agent who leaked the data, because the data, being in digital form, can easily be distributed.
Digitizing information is quick and efficient for a large amount of data but it is also prone to theft and illegal use for various purposes. Organizations such as hospitals and financial institutes store their data on servers for quick and easy access to provide prompt services to their clients. Breach of such private and personal data causes great harm to the concerned companies and their clients who trusted the companies with their personal information. Data can be shared within the organization, or for yet broader use, can be outsourced, which makes it more vulnerable to data theft and illegal use, causing serious concerns to the data owner and other relevant stakeholders. The impact and severity of data theft depends upon the importance and size of the dataset and its potential users.

Fingerprinting techniques help to identify the source of the data leakage. While watermarking is used to identify the owner of the data, fingerprinting is a variation of watermarking and is used for the identification of data recipients. In a fingerprinting technique, usually, unique hidden signatures are used for each data recipient (or a subset of them) to identify the users of the data. In case of a data breach, this signature can then be extracted from the compromised data to identify the recipient(s) of this copy of the data.

In other words, a fingerprinting technique helps in identification of the recipient for each copy of the data that can in turn help to identify the guilty agent(s) who became the source(s) of data leakage. Therefore, such techniques are used for tracing the guilty agent because they embed hidden information or mark (signature) into the underlying dataset differently for each data recipient or some group of recipients. The embedded signature should be robust enough so that the attacker cannot destroy or corrupt it without knowing the secret parameters or the original data that are known only to the data owner (Alice). Furthermore, the embedding of fingerprint should also not affect the data quality. Fingerprinting techniques are applied to different host contents, such as text, images, videos, and XML documents, but our focus is towards relational databases.

The fingerprinting of relational databases presents a unique challenge as compared to other data formats, such as images, videos, or texts, in the following ways: (i) part (or subset) of the relational databases is also usable as compared to other data formats where usually a subset of the image or video frames is not useful for the attacker; (ii) reordering of database tuples by the attacker to disturb the embedded mark does not affect the data quality a great deal, as opposed to other data formats; and (iii) the attacker can easily change the data by inserting new tuples in the marked database, deleting tuples from the marked databases, or altering the tuples in the marked database. Therefore, the fingerprinting of relational databases is different compared to other data formats and the fingerprinting techniques for multimedia content cannot be applied directly on relational databases.

Keeping in view the above challenges and issues, we consider various scenarios to show the motivation for fingerprinting of databases as follows.

1.1. Scenarios of Data Theft

Data are the center of nearly every application by acting as the raw material for information and creating a knowledge-based economy. In this way, data are very important asset for all stakeholders concerned. As other assets face some challenges, data also face some of them, such as stealing of data, illegal sharing or distribution, and fraudulent claims of database ownership. To deal with this challenge of handling data theft, there can be the following scenarios.

1.1.1. Scenario 1

As mentioned earlier, the individuals and organizations generate and store the digital data at quite a large scale in various data formats. Such datasets are used to extract useful knowledge that can later be used to improve the relevant products and services concerning that data. This demands sharing of the data outside the organization among experts of various domains, such as data mining, machine learning, business intelligence, information systems, data analytic, and so on; consequently, the threat of data theft (illegal sharing or sale) by the outsiders, with whom the data have been shared,
becomes relevant. In this context, it is necessary to trace the guilty agent (an outsider) who leaked the data.

1.1.2. Scenario 2

The second aspect of threat for data theft is when an insider leaks the data. In this respect, again, it is a must to identify the real culprit for two main reasons: (i) the individual(s) who committed the crime should be duly punished; and (ii) the innocent employee who is not involved in the incident should not be accused. For dealing with this scenario, a mechanism is needed to differentiate between numerous users or clients of the same relational database.

1.1.3. Scenario 3

Alice (owner) sells her data to different buyers; one or more than one buyer might, in this case, act as a guilty agent and may illegally redistribute or sell the data without the permission of the owner. This buyer (or group of buyers) needs to be identified to be punished accordingly and innocent buyers should not be accused for what they have not done.

A diagrammatical depiction of these scenarios has been presented in Figure 1 where different type of users (listed as U1–U9 for demonstration) have access to the same data.

Figure 1. Scenarios of data sharing and data leaking.

Now when the data breach occurs, it is important to identify the guilty agent responsible for this breach. This can only be possible if we are able to distinguish between the guilty agent(s) and the innocent users. Fingerprinting of digital data provides one such mechanism and has quite widely been applied in different data formats [1–3]. However, fingerprinting of relational data has still lots of room for new research. In Reference [4], the pioneer work on fingerprinting of relational databases was proposed by Li et al. by extended the watermarking technique of Agarwal et al. presented in [5]. The technique in [4] works by embedding the buyer-specific bit string in the relational data such that these bits can be used to distinguish among the copies of the same data. As a result, the specific buyer whose data were leaked can be identified. Some recent works presented in [6–8] use different bits for identifying the different users of the data. However, these techniques do not provide any mechanism to associate the mark bits used for fingerprinting to the relevant user to a biometric feature of the user. The use of a secure but random bit sequence might sometimes result in an error in the identification of the guilty agent. For example, among three recipients with signatures 10010, 10011, and 10101, recipient 2 with signature 10011 is the culprit leaking the data, but if his bits match to user 1 by changing of only one bit after the attack, the recipient 1 will be traced as the culprit. Therefore, it is imperative that we use more unique features to get more accurate results.

The use of a biometric feature has benefits in that it could map to only one user who could be easily identified, and this feature is unique to everyone; consequently, there would be only one match
to each pattern of mark bits for each user. Moreover, the biometric feature (the user’s thumb impression (we use the term thumb impression to refer to the thumb impression of the hand of the user (a unique biometric feature for each user) and fingerprint to refer to the unique signature embedded as the identity of the data user or recipient)) can serve as the unique public key to be used along with a secret private key for mark embedding. Another important requirement for mark embedding is to ensure the data’s usability after fingerprint insertion. To this end, we used a machine learning technique, linear regression [9], to ensure that the usability of the data is not affected as a result of fingerprint insertion. It is important to mention here that we select a numeric feature for fingerprint insertion and usually databases have numeric features. Having said that, we would like to extend this technique for non-numeric features as well to make it generic, but in this paper we only consider the numeric feature for fingerprint embedding.

The major contributions of the work presented in this paper are as follows:

• Utilizing a biometric feature (the thumb impression of the user) for generating his/her digital signature that can also serve as a kind of unique public key;
• Embedding the digital signature in a numeric feature using a new algorithm utilizing linear regression in the data in such a way that it does not disturb the quality of data;
• Proposing a mechanism to identify each recipient of the data uniquely by using new fingerprint decoding algorithm;
• Proving the robustness against certain malicious attacks by launching several attacks and then successfully decoding the embedded fingerprint;
• Using the detected fingerprint to identify the guilty agent in the case of leakage to deal with the data theft.

The rest of this paper has been organized as follows. Section 2 presents the current fingerprinting techniques with their limitations. A glimpse of the approach’s overview is presented in Section 3, and a detailed explanation of the proposed fingerprinting technique has been given in Section 4. The experimental evaluation the proposed technique has been presented in Section 5. Finally, Section 6 concludes the paper with an outlook toward future work.

2. Related Work

The study of fingerprinting and watermarking has mainly been emphasized on multimedia, but the past decade and a half has seen some fingerprinting techniques for relational databases presented in [4,5,10,11], and some research for watermarking the relational databases was presented in [12–14]. There is still plenty of room for improvement in the research in fingerprinting for relational databases, and a recently granted patent [15] indicates that this field is still new.

In order to safeguard the data from breaches, fingerprinting methods in [4] embed a particular fingerprint following three approaches. These three approaches do not use a primary key and can be outlined as: (a) S-Scheme: Using candidate bits of single numeric attributes as primary keys. However, this scheme has some issues: (i) the candidate bits may not be allowed to be unique for each tuple, (ii) the attacker can drop this numeric attribute that was used as the primary key, so the information of all the fingerprints may drop, and (iii) this may result in a deletion problem. (b) E-Scheme: Authors create a primary key for each attribute after examining the value of each numeric attribute in a tuple. The deletion problem that occurs in S-Scheme is solved by this new scheme, but the duplication problem still occurs. (c) M-Scheme: It dynamically chooses positions of bits to compose a primary key, and for solving the duplication problem of E-Scheme, different attributes are selected. In an extension of this work in [16], tuples for fingerprinting are selected by using a hash function. Other such techniques were proposed in [17,18]. According to the technique in [18], when a database is accessed by the user, the fingerprint process is invoked. Fingerprinting tasks are performed by a manager module employing various fingerprinting parameters, but these parameters are vulnerable to attacks that modify the data values and the identification of a guilty agent becomes the problem because authentication is also
verified by the manager module using these parameters. A similar scheme was proposed in [19] and suffers the same problems for robustness.

In another technique presented in [20], Liu et al. used blocks of available bits for embedding fingerprints. The mark was then inserted and the bit positions for marking were computed respectively for these blocks. In the fingerprinting detection phase, this technique sorts the suspicious datasets using the primary key. If the records are less than those of original fingerprinted data, these records are added from the other fingerprinted datasets. The blocks that are created using bits are then compared with embedded blocks. Such dependencies on other versions of the dataset make this technique vulnerable to collision attacks.

The fingerprinting technique proposed in [21] works with two levels of fingerprint insertion. At the first level, the dataset is segregated into $g$ parts and fingerprints are embedded in the dataset. After this, tuples are marked with fingerprints and a level of confidence is assigned to this mark in the second level. This mark is used as a secret key in the second level. For avoiding the conflict between two levels, the tuples that are already fingerprinted are not chosen as the candidates for fingerprint embedding in the second level. For decoding, a bit is extracted from each group or part. This technique is also vulnerable to attacks that result in disturbing the group boundaries because the decoding phase of this scheme is dependent on these groups (or parts) of the data.

A recent fingerprinting technique [6] works with non-numeric features and is based on generation of fake rows, but such techniques are affected by data modification attacks, particularly if they are applied on numeric features, as there is more room for alteration for numeric features; for instance, real or integer value attributes considered in our research. Although this technique needs very minor modifications to be used for numeric features, it still suffers for robustness when the attackers target large portions of database rows.

In the last three years (2018–2020), a patent [22] one for fingerprinting of audio datasets, Ref. [23] one for correlated data, and some for watermarking techniques, such as [24–27], have been applied for. They cannot be applied directly for fingerprinting of relational databases and need significant modifications for the problem addressed by us in this paper. For this, we also studied some recent survey papers, such as [28,29], for identifying the challenges and requirements related to substantial enhancements in the literature to solve real world problems of data breaches.

To conclude form the literature survey, all the above-mentioned techniques for database fingerprinting use a secure but a random bit sequence for fingerprinting. The use of a secure but random bit sequence sometimes may cause an error in the identification of the data user who is guilty of data breach. For example, among three recipients with signatures 10010, 10011, and 10101, suppose recipient 10011 is the culprit leaking the data; if his bits match to user 1 by changing the least significant bit only after the attack, the recipient 1 will be traced as the culprit. Therefore, it is imperative that we use unique features to get more accurate results. In this context, we propose a fingerprinting technique that uses a biometric feature—the thumb impression of the user—to construct the fingerprint bits and also make sure that the embedded fingerprint is robust enough to deal with the malicious attacks and to save the innocent user by providing a mechanism to significantly differ the hidden signatures for every one. One of the major requirements of fingerprinting techniques is their robustness against collusion attacks. In this context, collusion resistant codes, such as Boneh–Shaw [1], facilitate the use of the same fingerprint for different users, but using such schemes does not help in a great deal to pinpoint the exact culprit (the user guilty of data breach). For this reason, we use a relatively better scheme for collusion-resistant fingerprinting proposed in [30] that helps in identifying the culprit even if the embedded fingerprint has been damaged due to malicious attacks. We prefer to use the collusion secure coding [30] over orthogonal modulations, such as [31], because orthogonal codes put a limit on number of users or the data recipients.
3. Approach Overview

Fingerprinting techniques use hidden signatures in relational databases and these signatures can be random or meaningful. As mentioned earlier, a flaw in the use of a secure but random bit sequence sometimes causes an error in the identification of a data thief. Consider, for the example mentioned above, where among three recipients with signatures 10010, 10011, and 10101 were given the access to the data. The recipient 10011 leaked the data, but if his bits match to recipient 1 after the attack, recipient 1 will be detected as the culprit. Therefore, it is imperative that we use unique features to get more accurate results. Thumb impressions are unique and versatile identification marks that work even after the death of a person. They are a person’s unique identity that matches no one else in the whole world and are used widely in various information and communication technology (ICT) fields. This motivated us to use them for our work in the proposed technique in this paper. It should be noted that we use the term “thumb impression” to refer to the biometric feature of human hands and “fingerprint” to refer to the hidden signature embedded in the database.

The two main phases of the proposed fingerprinting technique are: (i) insertion phase of the digital mark (signature); (ii) detection phase of the digital mark (signature). In our fingerprinting technique, the relational database is marked with a different signature for each recipient so that each data recipient can be uniquely identified, and in the case of a data leakage, the embedded signature can then be used to identify the data recipient whose data were leaked. A buyer-specific digital signature (fingerprint) is generated for each user of a relational database and embedded in the original relational database $D$ using an embedding algorithm that uses a secret key $\kappa$, as shown in Figure 2.

The fingerprint is embedded directly into the database and the original database of Alice becomes $D_f$ after fingerprint embedding. For the ease of the readers, Table 1 lists the symbols used in the paper.

| Symbol | Description |
|--------|-------------|
| $D$    | Original database |
| $P$    | Primary key attribute |
| $\kappa$ | Secret key |
| $r$    | Number of tuples |
| $L$    | Length of fingerprint |
| $n$    | Number of attributes |
| $p$    | Number of partition |
| $\top$ | Threshold for controlling data distortions |

For extraction, we have also proposed a novel decoding algorithm that is resilient against malicious attacks that try to damage the embedded mark.

![Figure 2. Overview of the proposed scheme.](image-url)
4. Proposed Methodology

This section provides the details about the proposed methodology that consists of several phases, as discussed below.

4.1. Data Partitioning

Consider the original relational database table schema $D = (P, A_0, A_1, ..., A_{n-1})$, where $P$ refers to the primary key attributes while $A_0, A_1, ..., A_{n-1}$ are $n$ other attributes of the dataset. This dataset is then divided into $g$ number of partitions $(p_0, ..., p_{g-1})$ that are non-overlapping such that for two partitions $i$ and $j$, $p_i \cap p_j = \emptyset$. All the partition sets are non-empty and unite to construct the original database $D$ represented as $D = p_0 \cup p_1 \cup ... \cup p_{g-1}$.

To make the partitioning assignment secure, we use the standard application of machine authentication code (MAC) with 256 bits in which a volatile size input $i$ is given to a hash function $H$ that converts it to fixed size $h$.

$$ H : i \rightarrow h $$

Another advantage of using MAC is that it can help in uniformly distributing the database tuples in $g$ partitions. We calculate MAC for each database tuple $r_d \in D$ and then use it to assign this tuple to a partition using the relation:

$$ Par(r_d) = H(K || H(r_d \times P || K)) \mod g $$

where $r_d \times P$ refers to primary key of current tuple under consideration in the original database; $H()$ is hash function; $||$ is the operator that is used for concatenation. It is important to mention here that the database is logically partitioned and is not physically separated. Moreover, the fingerprint insertion (details follow shortly) across each partition is performed in such a way that the attacker, through collusion attacks, can not guess the marked tuples. In order to make the fingerprint insertion more robust, the purpose of performing data partitioning is to distribute the fingerprint across the whole database rather than inserting it in a particular part of the database. This is particularly useful in subset selection, insertion, and deletion attacks. Having said all that, however, the probability of successful collusion attack will increase if a smaller number of tuples are selected for fingerprint insertion.

4.2. Dataset Selection for Fingerprint Embedding

In this phase, the abovementioned steps are implemented for tuple selection for fingerprinting. For each attribute, threshold $T$ is computed in this phase. If the attribute value of selected tuple is higher as compared to this threshold, then this tuple will be selected for fingerprinting. For calculating threshold for an attribute we use the following equation:

$$ T = q \times \mu - \sigma $$

where $q$ refers to is a secret parameter with a value between 0 and 1, $\mu$ is mean, and $\sigma$ is standard deviation of the values of selected attribute. The parameter $q$ is private so that the attacker fails to guess which tuple is selected for fingerprint insertion. Only the tuples whose value is greater than threshold $T$ are selected for fingerprint insertion. It is important to mention here that the calculated threshold is different as compared to the techniques presented in [16,32] where $q \times \sigma$ is used instead of $q \times \mu$. The main weakness of their method is that an attacker can relatively easily guess that tuples selected for mark embedding had a value greater than $\mu$ because adding any positive value in $\mu$ will result in a value greater than $\mu$. But in the proposed fingerprinting technique, an attacker cannot guess the marked tuples, because in this approach any tuple having any value can be selected for fingerprinting using the secret parameter $q$. Mathematically, a database table can be presented as the union of all these
tuples in the form of a matrix, as shown below:

\[
M = \begin{bmatrix}
    d_{v00} & d_{v01} & \ldots & d_{v0(n-1)} \\
    d_{v10} & d_{v11} & \ldots & d_{v1(n-1)} \\
    \vdots & \vdots & \ddots & \vdots \\
    d_{v(r-1)0} & d_{v(r-1)1} & \ldots & d_{v(r-1)(n-1)}
\end{bmatrix}
\]

(4)

where \( r \) represents the number of tuples in the table and \( n \) shows the number of columns in the table.

The tuples with greater value as compared to threshold will make a subset original relation database \( D \) as \( D_T' \), that is:

\[
D_T' \leftarrow D_{R>T}
\]

(5)

Note that the dataset \( D_T' \) is not physically separated from the original dataset and it only shows the tuples selected for fingerprint embedding.

This phase has two goals: (i) enhance security by hiding tuples from the attacker; and (ii) reduce the number of tuples to be marked in a bid to reduce the data distortion. These distortions are usually application dependent and are controlled by the usability constraints defined by the data owner.

4.3. Fingerprint Insertion

Without the loss of generality, assume that database has \( p \) partitions and it has only one attribute in it \( A_t \in p_i \). In the fingerprint insertion algorithm, each bit of the fingerprint \( b_0, b_1, \ldots, b_{t-1} \) is inserted in each partition \( p_i \). This step is taken to ensure the robustness of the inserted fingerprint because the fingerprint bits can be decoded from the rest of the partitions in case the attacker is able to corrupt the fingerprint in a particular position \( p_i \). While marking a partition, the bits generated from the biometric feature (user’s thumb impression) are taken one at a time, and if this bit is 1, the value of the selected tuple is updated by moving it towards an allowed maximum, subject to usability constraints. Similarly, when this bit is 0, the value of selected tuple is updated by moving it towards an allowed minimum, subject to usability constraints. If it is necessary to use less fingerprint bits for a sensitive table, Alice can chose not to perform any modifications in the data for some bits of the fingerprint. Again, this can be done while defining the usability constraints because these constraints help to control modifications that result due to fingerprint insertion in the original data. Furthermore, these usability constraints also help to define a threshold that decides the amount of alterations that can be tolerated during the fingerprint embedding stage while keeping the usability of the data intact. We determine this threshold using linear regression and this threshold is different from the data selection threshold. Linear regression is a supervised machine learning technique and is mostly used to predict a value using some independent variables. In our case, we use the tuple values as the independent variable and predict the threshold using linear regression while constraining this problem with the usability constraints. In this way, we provide a novel mechanism to compute the threshold to ensure data usability while fingerprinting the database, and to the best of our knowledge, no other technique provides any such mechanism. Mathematically, we use Equation (6) to compute the root mean square error (RMSE) between the predicted value and the actual value of the tuple with an objective of minimizing RMSE. The actual value in this equation is the original value of the tuple, while the predicted value is the value of this tuple after fingerprint insertion.

\[
D_f = \theta_1 + \theta_2 \times D_T'
\]

(6)

In the above equation, \( \theta_1 \) and \( \theta_2 \) represent the slope and coefficient of the independent value (actual value of the tuple). The advantage of using linear regression for threshold computation is that it employees machine learning to make sure that the data usability remains intact during and
after fingerprint insertion. We denote this threshold by a parameter $\top$ that is also required during fingerprint extraction stage.

Algorithm 1 lists the steps of fingerprint insertion phase.

**Algorithm 1 Digital_Mark_Insertion.**

**Require:** Dataset $D$, Secret Key $K$

**Ensure:** $D_f, \top$

Set $D_f = D$ and $D_f' \leftarrow D_{R>T}$

Compute $\top$

for each $p$ in $D_f'$ do

if fingerprint_bit $b_i = 1$ then

Compute $\top$ for ensuring usability constraints

maximize the value of this tuple subject to usability constraints

else

Compute $\top$ for ensuring usability constraints

minimize the value of this tuple subject to usability constraints

end if

update $D_f$

end for

return $D_f, \top$

It is worth mentioning here that the fingerprint embedding does not affect the size of the original database, as it neither adds new rows in the database nor deletes the rows of the original database. Accordingly, the attacker would not be able to detect such a database. Additionally, the usability constraints defined by the data owner, Alice, are enforced such that there is no difference in size of the query result on multiple copies of the database running the same query for most common queries. We make this claim because being the data owner, Alice will have the knowledge of most of the queries running on her dataset as she (or the person(s) she hires for it) will know about applications of the data and major queries running on them. In this context, while fingerprinting, Alice will have to keep those queries in mind so that their results for original and fingerprinted database should be the same. Still, some specific queries involving differing values of same tuples in the different copies of the fingerprinted database can generate results of different sizes; as a result, the attackers can successfully launch collision attacks. We will discuss this scenario when evaluating the robustness of the proposed technique against collision attacks.

4.3.1. Ensuring Robustness against the Collusion Attack

In a collusion attack, the attacker Mallory compares two (or more) versions of the dataset with the aim of identifying the marked tuples; however, he does not have the access to the original data. As a consequence, he has to compare the differing values in different versions of the same dataset and might think that these differing values are the locations of hidden signatures (fingerprints). These different versions of the same dataset contain different fingerprints. This is the job of the data owner (Alice), during the fingerprint embedding stage, to make this embedding resilient to collusion attacks. In this context, the proposed scheme utilizes the idea of minimum distortions to the dataset such that the data distribution of original and the fingerprinted dataset remain the same; as a result, Mallory would not be able to locate the exact places where the marks are hidden and will have no option other than to guess the position of a hidden mark (or fingerprint bits).

Now let us consider the case of more than one version of the same dataset but marked with different fingerprints. This can also be seen as more than one attacker collaborating to corrupt or delete the embedded fingerprint. For brevity, let us consider that there are two versions of the same dataset
with two different fingerprints. Now, in this case, some marked tuples can have different values in both the datasets while some other marked tuples might have same values in both the datasets. We discuss both of these cases below.

Some Marked Tuples Can Have Different Values in both the Datasets

In this case, Mallory can not be sure about the marked and unmarked values because he has no access to the original data. To launch a successful collision attack on such datasets, Mallory has to trust his instincts of attacking the differing tuples, in multiple copies of the dataset, to affect the identity of data recipient. The probability of corrupting the embedded fingerprint from these tuples is quite high because in this case, Mallory has the liberty of guessing the marked tuples. However, since we have embedded the each bit of the fingerprint in every selected tuple of the database and Mallory does not have the knowledge of original data, the embedded mark can still be recovered from other tuples in case Mallory is successful in corrupting the embedded mark from some tuples. Our claim is further strengthen by the fact that Mallory cannot afford to launch attacks that would affect the data to such an extent that its usability is compromised. Mathematically,

\[ P_{\text{Attack}}(\text{success}) = (0.5)^{\gamma} \]  

(7)

This probability of successful malicious attacks is quite high, but the embedding of each bit of multi-bit fingerprint helps in making the fingerprint robust by decoding it from other tuples in case it has been compromised from some tuples after the attack. Moreover, as is evident from Equation (7), selection of higher percentage of tuples for fingerprint embedding can also help to improve fingerprint robustness, but in this case more data distortions will be introduced. To tackle the latter case, we make sure that the data distortions are within the bounds of data usability constraints defined by the data owner to control data distortions while marking the tuples. Moreover, to make our technique more robust to such attacks, we propose to use the idea of collusion-secure fingerprinting codes, Tardos code [30], which helps in identifying the culprit even if some bits of the embedded fingerprint in the suspicious data have been changed due to malicious attacks.

Some Marked Tuples Can Have the Same Values in both the Datasets

In this particular case, Mallory cannot know whether these tuples have been marked or not. Therefore, he has to randomly select tuples in a bid to affect the identity (that is, fingerprint) of data recipient. Suppose we are given R number of tuples in a relational dataset and out of these R tuples a fingerprint was inserted in a subset r where \( r \subset R \). The probability that the guilty agent can successfully attack these tuples and remove (or corrupt) the inserted digital signature is:

\[ P_{\text{Attack}}(\text{success}) = (0.5)^{r} \times R \]  

(8)

This probability is very low and is made even smaller by employing the idea of [30].

4.4. Fingerprint Detection

This section provides intuition to our fingerprint detection module. The fingerprint decoding algorithm detects the embedded fingerprint mark (signature) using secret key \( K \), number of partitions \( p \), secret parameter \( q \), and the threshold \( (\top) \). The bits are detected in reverse order because it is easier to evaluate the last embedded first. To start, the data partitions \((p_0, p_1, \ldots, p_{g-1})\) are generated using \( D_f \). In the next step, \( D'T \) (the dataset after attacks) and \( D'T_2 \) (the subset of data which is supposed to have marked bits after the attack) are generated following the same procedure of data selection as in the fingerprint insertion phase.

For detecting the inserted fingerprint, a decoding threshold \( D_t \) is computed using \( (\top) \) for decision of bit if it is equal to 1 or 0. If this \( D_t \geq 0 \), the decoded bit is 1; otherwise it will be decoded as 0.
After decoding a fingerprint bit from all the tuples, a majority voting scheme is used to finally decide the marked bit. The steps of this phase have been listed in Algorithm 2.

**Algorithm 2 Decode_One_Bit.**

**Require:** Fingerprinted Dataset $D_f$, $g$, $\top$, Secret Key $K$

**Ensure:** Detected Fingerprint $F_d$

1. $ones = 0$
2. $zeros = 0$
3. Compute $D_{T2}'$ using the same procedure as in fingerprint insertion phase
4. while there is any unprocessed tuple in $D_{T2}'$ do
   1. $D_t = value \times \top$
   2. if $D_t \geq 0$ then
      1. $ones = ones + 1$
   3. else
      1. $zeros = zeros + 1$
   4. end if
5. end while
6. if $ones > zeros$ then
   1. $F_d = 1$
2. else
   3. $F_d = 0$
3. end if
4. return $F_d$

5. Experiments and Results

We performed our experiments on a relatively small rail system ticket pricing related dataset available through Kaggle (https://www.kaggle.com/thegurusteam/spanish-high-speed-rail-system-ticket-pricing) and converted it into a database table. Without the loss of generality, for the purpose of experiments only, we assume that there is only one table in the database and take its numeric feature for fingerprint embedding and detection. The implementation of our algorithms and experiments was performed partly using Micorsoft.net 4.0 platform running on Microsoft Windows 7 and partly using Python 3.7 running on Windows 10. The experiments were designed keeping in mind the two objectives: (i) the embedded fingerprint should be robust to certain malicious attacks; and (ii) the data usability should remain intact after the fingerprint insertion. The robustness of the proposed technique was evaluated by launching various malicious attacks, such as collusion attacks, insertion attacks, alteration attacks, and deletion attacks. The usability of the data after fingerprint embedding was evaluated by measuring the data distortions brought about as a result of fingerprint insertion. The details of these experiments and their results are given below.

5.1. Robustness Analysis

One of the most important objectives of any fingerprinting technique is its robustness against malicious attacks. The attacker, Mallory, is assumed to have an open access to encoding and decoding method but no access to the original data or the secret parameters; therefore, he tries to affect the decoding process by launching several malicious attacks in a bid to corrupt the embedded mark. For this, Mallory is supposed to have infinite resources to launch his attacks, but he is also facing the dilemma of not knowing anything about the original data, secret parameters, and data usability constraints. It is the job of the data owner, Alice, to ensure that the embedded mark (or fingerprint) is robust enough that it can be decoded correctly even in the presence of some malicious attacks.
There are two general methods to evaluate the robustness of fingerprinting techniques: (i) A formal method using mathematics and probabilities in the presence of malicious attacks; and (ii) An empirical study to measure decoding accuracy in the presence of malicious attacks. In our work, we used both of these methods—with more focus on empirical study. We did not use the formal method for analyzing the security of proposed fingerprinting technique against attacks involving data manipulations using tuple insertion attack, tuple deletion attack, and tuple alteration attack because, for these attacks, the attacker (Mallory) has numerous choices to launch these attacks and generalizing these choices will hide some parameters related to the attacks. For instance, for insertion of new tuples in fingerprinted tuples, Mallory can either insert the duplicate tuples of the tuples that are already present in the fingerprint database of Alice (the data owner) or the tuples with some slight modifications as compared to the tuples of the tuples that are already present in the fingerprint database of Alice. These modifications can range from very small to a bit large alterations in the tuple values and are generally dependent on the nature of the data and its possible application. This dependency requires lots of assumptions regarding Mallory’s ability and choices to launch attack on the fingerprint database. However, as mentioned above, Mallory can use infinite options to launch his attacks while facing the dilemma of not knowing anything about the original data, secret parameters, and data usability constraints. Accordingly, generalizing the security analysis using formal methods is not accurate for every database or its various applications as it requires various assumptions on part of Mallory’s ability to attack the database involving data manipulation.

5.1.1. Analysis of Collusion Attack

As mentioned earlier, in the collusion attack, the attacker Mallory compares two (or more) versions of the data with an aim of identifying the marked tuples; however, he does not have the access to the original data. As a consequence, he has to compare the differing values in different versions of the same dataset and might think that these differing values are the locations of hidden signature (fingerprint). These different versions of the same dataset contain different fingerprints. We consider the two different scenarios of this attack as mentioned in Section 4.3.1 and examine them in the following.

Some Marked Tuples Can Have Different Values in both the Datasets

Now, for the first case for collision attack, mentioned in Section 4.3.1, consider multiple attackers collaborating with each other to corrupt or delete the embedded fingerprint. For brevity, let us consider that there are two versions of the same dataset with two different fingerprints. To understand it with the help of an example, consider a dataset having 100,000 tuples out of which 10,000 (that is 10%) were selected for inserting digital mark (signature). By comparing different versions of same datasets, Mallory is able to locate these marked tuples. But since they have different values in both versions of the dataset, he cannot guess the original value of these tuples before fingerprint embedding. Accordingly, he will have to try to launch collusion attack on these tuples in a bid to alter the identity of the data recipient so that either innocent user can be blamed for the data breach or the guilty agent cannot be identified. Since 10% of the tuples were selected for fingerprint embedding, the probability of successfully launching such attacks is:

\[
P_{\text{Attack (success)}} = (0.5)^{10} = 0.0009765625 \approx 9.7 \times 10^{-4}
\]

This probability is quite high but the embedding of every bit of multi-bit fingerprint in every selected tuple helps in making the fingerprint robust by decoding it from other tuples in case it has been compromised from some tuples after the attack. This is because of the fact that different versions of the data will have some tuples with same values. Moreover, selection of more tuples for fingerprint robustness also helps and our technique provides this flexibility by making sure that the data distortions are minimized by preserving the usability constraints set by the data owner to control data distortions while marking the tuples. Moreover, use of Tardos code in [30] also helps in
identifying the culprit even if some bits of the embedded fingerprint in the suspicious data has been changed due to malicious attacks.

Some Marked Tuples Can Have the Same Values in both the Datasets

In this case of collusion attack, Mallory cannot know if these tuples have been marked or not. Therefore, he has to randomly select tuples in a bid to affect the identity (that is, fingerprint) of data recipient. Consider, for an example, a dataset having 100,000 tuples out of which 1000 were selected for inserting digital mark (signature). Since Mallory does not have access to the original dataset, he is unable to decide if the tuples having the same value in both the datasets have been marked by Alice or not. In other words, Mallory is unable to locate these marked tuples. Accordingly, he will have to try to launch collusion attack on some randomly selected tuples from these tuples in a bid to alter the identity of the data recipient so that either innocent user can be blamed for the data breach or the guilty agent cannot be identified. In this particular case, the probability of successfully launching collision attack for targeting the identity of the data recipient is:

\[
P_{\text{Attack}}(\text{success}) = (0.5)^{1000} \times 100000 = 9.33 \times 10^{-297} \approx 0
\]

5.1.2. Analysis of Modification Attacks

For data modifications, without inserting new data, Mallory has two choices: (i) Tuple deletion attack in which Mallory deletes some selected tuples from the fingerprinted data; and (ii) tuple alteration attack in which Mallory alters the value of fingerprinted tuples. A detailed analysis of robustness of proposed scheme against these attacks is given in the following.

Analysis of Deletion Attack

In deletion attack, attacker’s intention is to destroy the inserted digital signature from fingerprinted dataset. He may delete \( \aleph \) number of tuples to disturb the embedded fingerprint. It was observed that our proposed fingerprinting technique was able to detect the fingerprint correctly with an accuracy of 100% even when the attacker deleted up to 50% of the tuples from the fingerprinted data.

The performance of proposed scheme has been shown in Figure 3 in terms of watermark decoding accuracy after deletion of some selected tuples.

\[\text{Figure 3. Robustness against tuple deletion attacks.}\]
Analysis of Alteration Attack

Alteration attack is the one in which Mallory changes the values of the tuples in the fingerprinted data in order to disturb embedded the fingerprint bits. In this attack, he may change the values of ℵ number of tuples but our technique is strong enough to overcome this problem. We also examined the robustness of our technique if the attacker alters up to 50% tuples for destroying inserted digital mark (signature) and found out that it shows resilience against these attacks and achieves 100% accuracy when up to 40% tuples are attacked and a decoding accuracy of 95% when up to 50% tuples are attacked as shown in Figure 4.

![Figure 4. Robustness against tuple alteration attacks.](image)

5.1.3. Analysis of Insertion Attack

In this attack, Mallory wants to corrupt the embedded fingerprint in the dataset of Alice to affect the identity of the data recipient. There are two options for Mallory while inserting new records: (i) insert similar records in the fingerprinted dataset by replicating the tuples in the dataset (we show this by 0% alteration in Figure 5); (ii) insert new tuples but with some slight different values than the tuples in the fingerprinted dataset (we show this by 0.5% to 2.5% alteration in Figure 5). Again, our scheme also showed robustness against these attacks as depicted by decoding accuracy in Figure 5 because inserting of new tuples does not affect the fingerprinted tuples that are still present in the dataset. Accordingly, Alice can extract the inserted fingerprint from these fingerprinted tuples.

![Figure 5. Robustness against tuple insertion attacks.](image)
5.2. Comparison of Proposed Technique with a State-of-Art Techniques for Robustness

This section presents the comparison of proposed technique with the techniques of Guo et al. [21] and Ahmad et al. [6] for resilience against different types of attacks. It is important to mention here that we chose this technique of year 2006 because, to the best of our knowledge, there is no significant work done for the fingerprinting of numeric data after this technique; although, some recent works have been proposed in [7,33,34] but they are not prestigious. There is another prestigious technique proposed in [35] but it considers that all the purchasers (or data recipients) share the same constraints which is not practical as different applications of the same data have different constraints. To compare our technique with the most recent relevant technique [6], we tested its slightly modified version to evaluate its robustness against malicious attacks while working with numeric features.

For comparison, let us consider the subset selection attack first. In such attacks, the attacker, Mallory selects subset of the marked dataset to make a new smaller dataset. In other words, he virtually deletes the remaining dataset with the aim to disturb the inserted fingerprints from the fingerprinted dataset, by selecting some records. The technique of [21] provides 100% decoding accuracy up till 10 to 70% of deleted records and their accuracy decreases to 50% as compared to our accuracy of 80% after deletion of 80% of tuples from the marked dataset. Similarly, our technique outperformed the technique of [21] when up to 90% of the tuples were deleted from the marked dataset. We believe that this is due to the fact that we did not use any particular bit for fingerprint insertion in a particular data partition because such techniques become dependent on marked bits during decoding process; consequently, changing of only one bit can disturb the embedded fingerprint. Similarly, the proposed technique also outperformed the technique of Ahmad et al. that uses fake tuples for fingerprint embedding while the proposed technique does not have any such dependency on fake tuples. The results of these experiments have been shown in Figure 6.

In another set of experiments, we compared the performance of our proposed scheme with [21] and [6] for tuple alteration attacks. It is evident from Figure 7 that the proposed technique outperforms the other two techniques of Guo et al. when more than 40% of the records were slightly altered. We believe that this is because of the reason that the proposed technique embed each bit in the selected tuple; as a result, the embedded bit can be detected from remaining unaffected tuples in case some of the fingerprint bits are compromised due to malicious attack.
Figure 7. Comparison of techniques for tuple alteration attacks.

In the last set of experiments for the comparison of resilience against malicious attacks, the performance of proposed technique was compared with [6,21] for Addition (mix and match attacks) in which Mallory makes a new dataset by mixing similar data from more than one sources such that the size of the new dataset after the attacks remains the same as that of fingerprinted dataset. One can easily note from the results in Figure 8 that if more than 40% are taken from a dataset other than the fingerprinted dataset, the fingerprint embedded using [6,21] gets disturbed; as a result, the bit string of multi-bit fingerprint changes. This situation can be quite alarming because this corrupted fingerprint can match to fingerprint embedded in some other copy of the same dataset that was under use by some innocent user who actually was not involved in the data breach. In this particular scenario, the effectiveness of our scheme can be utilized that proposes to use a biometric feature (such as thumb impression of the hand) of the user because each human has a unique thumb impression. This solution helps in preventing innocent user from being accused or punished. Moreover, our proposed scheme also showed better performance than the other two counterparts when the new dataset (formed after mix and match attacks) contains up to 10% of the fingerprinted data.

Figure 8. Comparison of techniques for addition attacks.
5.3. Data Distortion

The data distortions that result due to fingerprint embedding are controlled by usability constraints. These usually constraints are usually defined by the data owner based on sensitivity of the data and its intended use. Ideally, these distortions should be minimum and our proposed technique makes sure to minimize these data distortions by preserving the usability constraints. Moreover, the use of machine learning technique, linear regression, for computing the threshold for data modifications during fingerprint embedding also helped to keep the data distortions to a level where usability constraints are not violated. The histograms shown in Figures 9 and 10 show that the data distribution in the original and the fingerprinted data is almost same.

![Data distribution before fingerprint insertion](image1)

**Figure 9.** Histogram showing the distribution of original data.

![Data distribution after fingerprint insertion](image2)

**Figure 10.** Histogram showing the distribution of fingerprinted data.

6. Conclusions

As we are witnessing tremendous advancements in the ICT domain, more and more digital data are being generated. These data are then used for various applications and at times need to be shared and accessed by several users. In this context, the threat of data breaches is also increasing. In particular, we address the problem of identifying the culprit in a case of data theft in a scenario when more than one user has access to the data. To this end, we proposed a fingerprinting technique using a biometric feature that provides a mechanism for minimum distortions in the data with a high
degree of robustness. The use of a biometric feature (thumb impression in our case) also helps in preventing the innocent users from being accused of data breaches because it is unique for every person in the world. We evaluated the performance of our technique using the attacker channel through experiments and computing the probability of successful attack for generality to account for datasets other than the one used in our experimental study. It was observed that after attacks, the inserted digital mark is extracted from the dataset with a high degree of accuracy, resulting in identification of the culprit. Hence, this approach helps the data owner (Alice) to safeguard her data from data theft or illegal usage; she can now easily distribute her data among multiple recipients, thereby encouraging collaborative work. The experimental results of the proposed fingerprinting technique on a real world dataset proved our claims with more than 90% accuracy, while up to 50% tuples were targeted by the attacker. The proposed technique also showed its better performance in terms of fingerprint robustness compared to its counterparts. In the future, we would like to extend our technique by applying it for version control as well because we believe that our technique can also be used to version control of the same database for different time spans, and also to blockchains wherein more than one user has the access to same database and is continuously adding the blocks of data to the database. One extension of this work can be for non-numeric features. Similarly, another future direction can be a detailed study of collision attacks on fingerprinting schemes because the availability of different versions of the same dataset to the attacker and restrictions on data distortions in order to meet the usability constraints are quite challenging.

Author Contributions: Conceptualization, M.K. and A.S.A.; methodology, M.K. and F.R.; software, M.K. and M.S.A.; validation, M.K. and M.S.A.; formal analysis, M.K.; investigation, A.S.A.; resources, E.A.S. and M.S.A.; data curation, E.A.S. and M.S.A.; writing—original draft preparation, M.K. and F.R.; writing—review and editing, M.K. and A.S.A.; visualization, M.S.A.; supervision, E.A.S.; project administration, E.A.S.; funding acquisition, E.A.S. All authors have read and agreed to the published version of the manuscript.

Funding: The authors would like to thank University of Jeddah for providing funding for this research under the grant number (UJ-02-014-ICGR).

Acknowledgments: The authors are thankful to Muddassar Farooq from Air University Islamabad, Pakistan for providing valuable feedback for improving this paper.

Conflicts of Interest: The authors do not have any conflict of interest for this work.

References

1. Boneh, D.; Shaw, J. Collusion-secure fingerprinting for digital data. *IEEE Trans. Inf. Theory* **1998**, *44*, 1897–1905. [CrossRef]
2. Trappe, W.; Wu, M.; Wang, Z.J.; Liu, K.R. Anti-collision fingerprinting for multimedia. *IEEE Trans. Signal Process.* **2003**, *51*, 1069–1087. [CrossRef]
3. Barg, A.; Blakley, G.R.; Kabatiansky, G.A. Digital fingerprinting codes: Problem statements, constructions, identification of traitors. *IEEE Trans. Inf. Theory* **2003**, *49*, 852–865. [CrossRef]
4. Li, Y.; Swarup, V.; Jajodia, S. Constructing a virtual primary key for fingerprinting relational data. In Proceedings of the 3rd ACM workshop on Digital rights management, Washington, DC, USA, 27 October 2003; pp. 133–141.
5. Agrawal, R.; Kiernan, J. Watermarking relational databases. In Proceedings of the 28th international conference on Very Large Data Bases, Hong Kong, China, 20–23 August 2002; pp. 155–166.
6. Ahmad, M.; Shahid, A.; Qadri, M.Y.; Hussain, K.; Qadri, N.N. Fingerprinting non-numeric datasets using row association and pattern generation. In Proceedings of the 2017 International Conference on Communication Technologies (ComTech), Rawalpindi, Pakistan, 19–21 April 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 149–155.
7. Mohanpurkar, A.; Joshi, M. A Fingerprinting Technique for Numeric Relational Databases with Distortion Minimization. In Proceedings of the 2015 International Conference on Computing Communication Control and Automation, Pune, India, 26–27 February 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 655–660.
8. Mohanpurkar, A.; Joshi, M. Fingerprinting Numeric Databases with Information Preservation and Collusion Avoidance. *Int. J. Comput. Appl.* **2015**, *130*, 13–18. [CrossRef]
9. Seber, G.A.; Lee, A.J. *Linear Regression Analysis*; John Wiley & Sons: Hoboken, NJ, USA, 2012; Volume 329.
10. Li, Y.; Swarup, V.; Jajodia, S. Fingerprint Relational Databases; Technical Report; Center for Secure Information Systems: Fairfax, VA, USA, 2003.

11. Šarcević, T.; Mayer, R. An Evaluation on Robustness and Utility of Fingerprinting Schemes. In International Cross-Domain Conference for Machine Learning and Knowledge Extraction; Springer: Cham, Switzerland, 2019; pp. 209–228.

12. Sion, R.; Atallah, M.; Prabhakar, S. On watermarking numeric sets. In International Workshop on Digital Watermarking; Springer: Berlin/Heidelberg, Germany, 2002; pp. 130–146.

13. Sion, R.; Atallah, M.; Prabhakar, S. Rights protection for relational data. IEEE Trans. Knowl. Data Eng. 2004, 16, 1509–1525. [CrossRef]

14. Kamran, M.; Suhail, S.; Farooq, M. A robust, distortion minimizing technique for watermarking relational databases using once-for-all usability constraints. IEEE Trans. Knowl. Data Eng. 2013, 25, 2694–2707. [CrossRef]

15. Estehghari, S.; Guerin, N. Relational Database Fingerprinting Method and System. U.S. Patent 9,646,161, 9 May 2017.

16. Li, Y.; Swarup, V.; Jajodia, S. Fingerprinting relational databases: Schemes and specialties. IEEE Trans. Dependable Secur. Comput. 2005, 2, 34–45.

17. Zhou, X.; Huang, M.; Peng, Z. An additive-attack-proof watermarking mechanism for databases’ copyrights protection using image. In Proceedings of the 2007 ACM Symposium on Applied Computing, Seoul, Korea, 11–15 March 2007; ACM: New York, NY, USA, 2007; pp. 254–258.

18. Zhou, M.; Wang, J.; Wang, C.; Li, D. A novel fingerprinting architecture for relational data. In Proceedings of the Digital EcoSystems and Technologies Conference, DEST 2007, Inaugural IEEE-IES, Cairns, Australia, 21–23 February 2007; IEEE: Piscataway, NJ, USA, 2007; pp. 477–480.

19. Guo, F.; Wang, J.; Zhang, Z.; Ye, X.; Li, D. An improved algorithm to watermark numeric relational data. In International Workshop on Information Security Applications; Springer: Berlin/Heidelberg, Germany, 2005; pp. 138–149.

20. Liu, S.; Wang, S.; Deng, R.H.; Shao, W. A block oriented fingerprinting scheme in relational database. In International Conference on Information Security and Cryptology; Springer: Berlin/Heidelberg, Germany, 2004; pp. 455–466.

21. Guo, F.; Wang, J.; Li, D. Fingerprinting relational databases. In Proceedings of the 2006 ACM Symposium on Applied Computing, Dijon, France, 23–27 April 2006; ACM: New York, NY, USA, 2006; pp. 487–492.

22. Vlack, K.; Wyss, F.I. System and Method for Fingerprinting Datasets. U.S. Patent 10,552,457, 4 February 2020.

23. Yilmaz, E.; Ayday, E. Collusion-Resilient Probabilistic Fingerprinting Scheme for Correlated Data. arXiv 2020, arXiv:2001.09555. [CrossRef]

24. Gort, M.L.P.; Feregrino-Uribe, C.; Cortesi, A.; Fernández-Peña, F. HQR-scheme: A high quality and resilient virtual primary key generation approach for watermarking relational data. Expert Syst. Appl. 2019, 138, 112770. [CrossRef]

25. Tufail, H.; Zafar, K.; Baig, A.R. Relational database security using digital watermarking and evolutionary techniques. Comput. Intell. 2019, 35, 693–716. [CrossRef]

26. Khanduja, V.; Chakraverty, S. A generic watermarking model for object relational databases. Multimed. Tools Appl. 2019, 78, 28111–28135. [CrossRef]

27. Hu, D.; Zhao, D.; Zheng, S. A new robust approach for reversible database watermarking with distortion control. IEEE Trans. Knowl. Data Eng. 2018, 31, 1024–1037. [CrossRef]

28. Kumar, S.; Singh, B.K.; Yadav, M. A Recent Survey on Multimedia and Database Watermarking. Multimed. Tools Appl. 2020, 1–49. [CrossRef]

29. Kamran, M.; Farooq, M. A comprehensive survey of watermarking relational databases research. arXiv 2018, arXiv:1801.08271.

30. Tardos, G. Optimal probabilistic fingerprint codes. J. ACM 2008, 55, 1–24. [CrossRef]

31. Kiyavash, N.; Moulin, P. Performance of orthogonal fingerprinting codes under worst-case noise. IEEE Trans. Inf. Forensics Secur. 2009, 4, 293–301. [CrossRef]

32. Zhang, YQ.; Emmanuel, S. A novel framework for multiple creatorship protection of digital movies. In Digital Rights Management. Technologies, Issues, Challenges and Systems; Springer: Berlin/Heidelberg, Germany, 2006; pp. 1–12.

33. Lohegaon, D. A robust, distortion minimization fingerprinting technique for relational database. Int. J. Recent Innov. Trends Comput. Commun. 2014, 2, 1737–1741.
34. Mohanpurkar, A.A.; Joshi, M.S. A traitor identification technique for numeric relational databases with distortion minimization and collusion avoidance. *Int. J. Ambient Comput. Intell.* 2016, 7, 114–137. [CrossRef]

35. Lafaye, J.; Gross-Amblard, D.; Constantin, C.; Guerrouani, M. Watermill: An optimized fingerprinting system for databases under constraints. *IEEE Trans. Knowl. Data Eng.* 2008, 20, 532–546. [CrossRef]