Probabilistic Fingerprinting Scheme for Correlated Data

Emre Yilmaz[1(✉)] and Erman Ayday[2]

[1] University of Houston-Downtown, Houston, TX 77002, USA
yilmaze@uhd.edu
[2] Case Western Reserve University, Cleveland, OH 44106, USA
exa208@case.edu

Abstract. In order to receive personalized services, individuals share their personal data with a wide range of service providers, hoping that their data will remain confidential. Thus, in case of an unauthorized distribution of their personal data by these service providers, data owners want to identify the source of such data leakage. We show that applying existing fingerprinting schemes to personal data sharing is vulnerable to the attacks utilizing the correlations in the data. To provide liability for unauthorized sharing of personal data, we propose a probabilistic fingerprinting scheme that efficiently generates the fingerprint by considering a fingerprinting probability (to keep the data utility high) and publicly known inherent correlations between data points. To improve the robustness of the proposed scheme against colluding malicious service providers, we also utilize the Boneh-Shaw fingerprinting codes as a part of the proposed scheme. We implement and evaluate the performance of the proposed scheme on real genomic data. Our experimental results show the efficiency and robustness of the proposed scheme.

Keywords: Fingerprinting · Liability · Data sharing

1 Introduction

In today's data-driven world, individuals share vast amount of personal information with several service providers (SPs) to receive personalized services. During such data sharings, data owners usually do not want SPs to share their personal data with other third parties. Such issues are typically addressed via a consent (or data usage agreement) between a data owner and an SP to determine how much the SP gains the ownership of the user's data. However, user's data may often end up in the hands of unauthorized third parties since (i) SPs sometimes share (or sell) users' personal information without their authorization or (ii) databases of SPs are sometimes breached (e.g., due to insufficient or non-existing security measures). When such a leakage occurs, data owners would like to know the source of it to keep the corresponding SP(s) liable due to the leakage.

Published by Springer Nature Switzerland AG 2023
V. Atluri and A. L. Ferrara (Eds.): DBSec 2023, LNCS 13942, pp. 69–90, 2023.
https://doi.org/10.1007/978-3-031-37586-6_5

Digital fingerprinting [15] is a technique to identify the recipient of a digital object by embedding a unique mark (called fingerprint) into the digital object. Fingerprinting techniques have been developed for different types of digital content, such as audio, video, and software. However, such techniques are not directly applicable for our scenario (sharing personal correlated data) because (i) they (especially for multimedia) utilize the high redundancy in the data, (ii) the embedded marks need to be large, which reduces the utility of shared data, and (iii) they do not consider the correlations between data points.

We propose a fingerprinting technique for sequential personal data having correlations between data points, such as genomic data or location data. We consider several different malicious behavior that can be launched by the malicious SPs against the proposed fingerprinting scheme including: (i) flipping data points, (ii) using a subset of the data points, (iii) utilizing the correlations in the data, and (iv) colluding SPs to identify and/or distort the fingerprint. Boneh-Shaw codes [4] and Tardos codes [14] are considered as the state-of-the-art fingerprinting codes to prevent collusion attacks. However, as we show in our experimental evaluation, they do not provide robustness against other attacks against a fingerprinting scheme which use data correlations. The proposed fingerprinting scheme essentially relies on adding controlled noise into particular data points in the original data and keeps the data utility high by controlling the fraction of fingerprinted data points. By building a correlation model, the proposed scheme guarantees that the fingerprint is consistent with the nature of the data. Furthermore, the proposed fingerprinting scheme utilizes Boneh-Shaw codes [4] to improve its collusion resistance while also providing robustness against other types of malicious behavior, such as flipping or utilizing correlations in the data (that are not considered by Boneh-Shaw codes).

We implement the proposed fingerprinting scheme for genomic data sharing using real-life datasets. Via simulations, we show the robustness of the proposed scheme for a wide-variety of attacks against a fingerprinting scheme and also compare it with the state-of-the-art. We show that the proposed scheme is efficient and scalable in terms of its running time.

The rest of the paper is organized as follows. We summarize the related work in Sect. 2. We describe the problem settings in Sect. 3. We propose the probabilistic fingerprinting scheme in Sect. 4 and explain how to utilize Boneh-Shaw codes to resist collusion attacks in Sect. 5. We present our experimental results in Sect. 6. Section 7 concludes the paper.

2 Related Work

Digital watermarking is the act of embedding an owner-specific mark into a digital object to prove the ownership of the object [6]. Watermarking techniques have been proposed for multimedia [3,8], text documents [5], graphs [16], and spatiotemporal datasets [9]. Digital fingerprinting can be seen as a personalized version of watermarking since embedded mark (i.e., fingerprint) is different in each copy of data with the objective to identify the recipient if data is disclosed

to a third party without authorization of data owner. Fingerprinting schemes have been proposed for multimedia [15], relational databases [10], and sequential data [2]. Fingerprinting schemes for multimedia utilize the high redundancy in digital object, which are not applicable to personal data sharing (which typically includes less redundancy). Fingerprinting schemes for relational databases [10] do not consider correlations between attributes. Moreover, since databases consist of numerous tuples, the redundancy is much higher compared to personal data. In [2], authors proposed a fingerprinting algorithm for sequential data. Since this scheme solves an optimization problem at each step of the algorithm, it is not scalable. In addition, they minimize the probability of identifying the whole fingerprint in a collusion attack. However, the attackers can achieve their goal by modifying some of the fingerprinted points.

Since each fingerprinted copy of a data stream is different, malicious recipients can collude and detect fingerprinted points by comparing their copies. Boneh and Shaw proposed a general fingerprinting solution for binary data that is robust against collusion [4] by including overlapping fingerprints in each copy. However, fingerprint length needs to be significantly long to guarantee robustness against collusion, which reduces the utility of the shared data. Tardos also proposed probabilistic fingerprinting codes [14], which requires shorter codes than Boneh-Shaw by assigning a probability to each data point. However, the length of the fingerprint still needs to be high to provide robustness guarantees. The robustness of both schemes are guaranteed if the colluding SPs do not change the value of a data point when they all received the same value (i.e., marking assumption). In practice, colluding parties can behave randomly or use correlations in the data. Therefore, their theoretical analysis may not work in practice. Since we deal with correlations and probabilistic adversarial behavior, we choose to show the robustness statistically, with experimental evaluation in this work. Although there are some works relaxing marking assumption [7] or improving utility of Boneh-Shaw and Tardos codes [11,13], none of these schemes consider other attacks against a fingerprinting scheme, such as using correlations in the data for detecting fingerprinted points. Due to such limitations, directly using these codes for personal data sharing will result in low robustness and utility. In Sect. 6, we show how our proposed scheme provides higher robustness than Boneh-Shaw and Tardos codes against various attacks. As we explain in Sect. 5.2, we utilize Boneh-Shaw codes to improve the robustness of the proposed scheme against collusion attacks. We prefer to integrate Boneh-Shaw codes to improve the robustness of our scheme rather than Tardos codes, because our goal is to explicitly assign overlapping fingerprints as proposed by Boneh-Shaw codes. Therefore, we develop an efficient fingerprinting scheme which is robust against a wide-variety of attacks.

3 Problem Settings

In this section, we present our system model, threat model, and robustness measures. We describe the system and data models in a general way (i.e., without

relying on a specific data type or size). In Sect. 6, we discuss and evaluate the use of the proposed fingerprinting scheme for genomic data.

3.1 System and Data Model

We show our system model in Fig. 1. We assume a data owner (Alice) with a sequence of data points $\mathcal{X} = [x_1, \ldots, x_l]$, where l is the length of the data and each x_i can have a value from the set $\mathcal{D} = \{d_1, \ldots, d_m\}$. Alice wants to share her data with multiple service providers (SPs) to receive a service from these SPs related to her data. Thus, we consider the scenario, in which the data owner shares her personal data (as a stream) with several service providers (SPs). We represent these SPs with a set $\mathcal{S}$ and each SP with an index such that $SP_i \in \mathcal{S}$. We assume Alice wants to detect the source of data leakage in the case of an unauthorized data sharing by any of these SPs. In other words, upon observing that her data is included in a leaked/breached dataset, the goal of the data owner is to statistically prove which specific SP leaked her data. Hence, for each $SP_i \in \mathcal{S}$, Alice creates a unique fingerprinted copy ($\mathcal{X}'_i = [x'_{i,1}, \ldots, x'_{i,l}]$) of $\mathcal{X}$ by changing the values of some data points. Note that the proposed fingerprinting scheme detects the source of data leakages; it does not provide guarantees to SPs about the correctness of data provided by Alice. Thus, similar to other fingerprinting schemes, we do not consider data credibility issues.

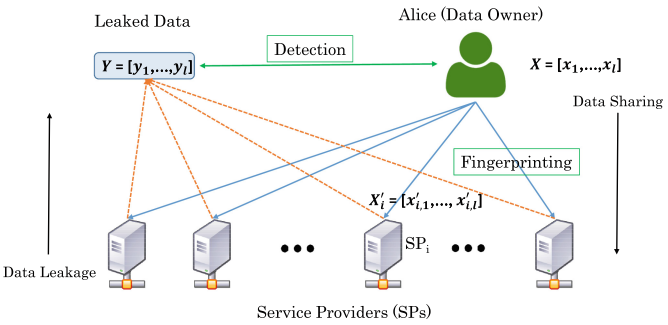


Fig. 1. The system model. Alice shares her data with several SPs after fingerprinting. In case of a data leakage, the goal of Alice is to identify the guilty SP (via the detection algorithm) that is responsible for this leakage.

Changing the states of more data points for fingerprinting increases the chance of Alice to detect the malicious SP(s) who leaks her data. However, fingerprinting naturally degrades the utility of shared data and one of our goals is to minimize this degradation while providing a robust fingerprinting scheme. We provide a general definition for the utility of a shared copy $\mathcal{X}'_i$ as $\mathcal{U}_i = \sum_{j=1}^{l} u_j D_j^i$, where u_j is the utility of data point x_j, $D_j^i = 1$ if $x_j = x'_{i,j}$, and $D_j^i = -1$ if $x_j \neq x'_{i,j}$.

We assume that data points are correlated and we mainly consider pairwise correlations between consecutive data points for clarity of presentation. The proposed mechanism can also be extended to consider more complex correlations, which may result in eliminating more possible values with low correlations. Thus, we let conditional probabilities (e.g., $P(x_{j+1} = d_\beta | x_j = d_\alpha)$ values) representing the correlations between data points be publicly available for any $j \in \{1, \ldots, l-1\}$ and $d_\alpha, d_\beta \in \mathcal{D}$. For different data types, such as location patterns and genomic data, data points may have pairwise correlations between each other. Therefore, a malicious SP may use such correlations to identify and exclude fingerprinted data points (e.g., the ones that are not compliant with the expected correlation model) in its unauthorized data sharing. Note that we do not expect the data owner to know and understand these correlations. We foresee that the system that will deploy the proposed algorithm to compute the correlations using public datasets and to use them in fingerprinting algorithm.

3.2 Threat Model

In this section, we present the attacks which may be performed by malicious SPs (attackers). We consider these attacks when developing the proposed scheme and we evaluate the robustness of the proposed scheme against these attacks in Sect. 6. In all of these attacks, the main goals of the attacker(s) are (i) to avoid being detected by the data owner and (ii) to share as many correct data points as possible. The attacker(s) needs to modify the values of some data points in its copy to distort the fingerprint. These modifications in the data mostly cause utility loss for the attacker(s). Let $\mathcal{Y} = [y_1, \ldots, y_l]$ be the leaked copy of Alice's data $\mathcal{X}$. Similar to the utility of shared data, we provide the general definition for the utility of the attacker(s) as $\mathcal{U}_\mathcal{Y} = \sum_{j=1}^{l} u_j D_j$, where u_j is the utility of x_j, $D_j = 1$ if $x_j = y_j$ and $D_j = -1$ if $x_j \neq y_j$. Flipping, subset, and majority attacks have already been introduced [10], whereas we propose correlation attack in the following and probabilistic majority attack in Sect. 5.1.

Flipping Attack. A malicious SP flips the values of some data points randomly to distort the fingerprint (before it does the unauthorized sharing). The malicious SP flips each data point with probability p_f. If it decides to flip, then it selects one of the remaining $(m-1)$ values (states) of the corresponding data point with equal probability and shares that state. While the probability of being detected by Alice becomes lower for higher p_f, the utility of shared data decreases as well.

Subset Attack. A malicious SP excludes (removes) some randomly chosen data points before leaking data, instead of flipping them. We denote the probability of excluding a data point as p_s. This attack is not as powerful as the flipping attack because flipping data points might create a fingerprint pattern that looks similar to some other SP's fingerprint pattern, and hence Alice may falsely accuse an innocent SP. However, to succeed in the subset attack, the malicious SP needs to exclude almost all of the fingerprinted data points.

Correlation Attack. As discussed, the correlations between consecutive data points are assumed to be publicly known. A malicious SP can use these correlations to degrade the robustness of the fingerprinting mechanism (we will define the robustness of the mechanism later). Assume the malicious SP_i receives $\mathcal{X}'_i = \{x'_{i,1}, \ldots, x'_{i,l}\}$ from Alice and let two data points in the received data be $x'_{i,j+1} = d_\beta$ and $x'_{i,j} = d_\alpha$. If $P(x_{j+1} = d_\beta | x_j = d_\alpha)$ is low, then the attacker infers that either $x'_{i,j}$ or $x'_{i,j+1}$ is fingerprinted with a high probability. After detecting such a pair, the malicious SP may change their values or exclude them before unauthorized sharing.

Since flipping attack can be more powerful (as discussed in Sect. 3.2), we assume the malicious SP combines correlation attack with flipping attack as follows: For two consecutive data points x_j and x_{j+1}, the malicious SP_i checks $P(x_{j+1} = x'_{i,j+1} | x_j = x'_{i,j})$. If this probability is less than a threshold τ_c, the malicious SP changes the value of $x'_{i,j+1}$ to a different value from the set $\mathcal{D}$ that provides the highest conditional probability (correlation) between $x'_{i,j}$ and $x'_{i,j+1}$. Otherwise, the malicious SP flips the value of $x'_{i,j+1}$ with probability p_f (as in the flipping attack). By doing so, the malicious SP (i) distorts the fingerprint with a high probability (by distorting the data points that are not compliant with the inherent correlations in the data) and (ii) adds random noise to the data to further reduce the chance of being detected by Alice. Following a similar strategy, the malicious SP checks all pairs up to $x'_{i,l-1}$ and $x'_{i,l}$. If the malicious SP does not combine correlation attack with flipping attack, Alice can perform a similar correlation attack on $\mathcal{X}'_i$ and obtain a similar result with the malicious SP. Thus, the randomness in the flipping attack makes it difficult for Alice to detect the malicious SP.

Collusion Attack. If multiple malicious SPs collude, by comparing their copies, they may detect and distort the fingerprinted data points. The goal of the colluding SPs is to share a single copy of $\mathcal{X}$ without being detected. One well-known collusion attack against fingerprinting schemes, called majority attack in the literature [10], is when colluding SPs compare all their received data points and choose to share the data value that is observed by the majority of the colluding SPs. However, doing such an attack alone cannot be successful for attackers if there is no randomness in the attack. Otherwise, Alice can simulate the majority attack of the colluding SPs (and identify the malicious SPs easily). Hence, in Sect. 5.1, we explain a more powerful collusion attack which also includes correlation and flipping attacks.

3.3 Robustness Measures

The attacks described in Sect. 3.2 may cause Alice to accuse an innocent SP due to the data leakage, and hence cause false positives in the fingerprint detection algorithm. A fingerprinting scheme is considered robust if it resists malicious attacks and allows the detection of the guilty SP that leaks the data after performing an attack. In this work, we assume that the detection algorithm always

returns a guilty SP when Alice observes an unauthorized copy of her data. The proposed detection methods compute a score for each SP having a copy of Alice's data and identify the guilty SP based on these scores. In order to quantify the robustness of the proposed scheme, we use the accuracy (a) of the detection algorithm which is defined as the probability of detecting the guilty SP from the leaked data. In [10], misattribution false hit (fh^A) is defined as a robustness measure (defined as the probability of detecting an incorrect fingerprint from the leaked data). Therefore, a is equivalent to $(1 - fh^A)$. Similar to most collusion resistant fingerprinting schemes, such as Boneh-Shaw codes [4], we also define a as the probability of detecting one guilty SP from the leaked data in case of collusion attack.

4 Probabilistic Fingerprinting Scheme for Correlated Data

In this section, we propose our probabilistic fingerprinting scheme which considers correlations in data (considering the attack in Sect. 3.2). In Sect. 5, we will improve this scheme to also consider colluding malicious SPs.

4.1 Proposed Fingerprinting Algorithm

Assume data owner (Alice) has a sequence of data points $\mathcal{X} = [x_1, \ldots, x_l]$ and she wants to share her data with an SP_i as $\mathcal{X}'_i = [x'_{i,1}, \ldots, x'_{i,l}]$ after fingerprinting. Alice determines a fingerprinting probability p, which means on average $p \cdot l$ data points will be fingerprinted (i.e., their value will be changed) when sharing l data points with SP_i. Lower p values increase the utility of shared data by decreasing the robustness.

Under these settings, a naive algorithm fingerprints each data point with the same probability, without considering correlations in the data. Hence, each data point is shared correctly ($x'_{i,j} = x_j$) with probability $1 - p$ and incorrectly/fingerprinted ($x'_{i,j} \neq x_j$) with probability p. For each fingerprinted data point, the shared state is selected among $(m-1)$ states in $\mathcal{D}$ with equal probability. However, if this naive scheme is used, a malicious SP can detect some of the fingerprinted data points using the correlations and distort them via flipping, as discussed in Sect. 3.2.

In order to prevent such an attack, one needs to consider the correlations in the fingerprinting scheme. In our proposed probabilistic fingerprinting scheme, for each data point x_j, considering the correlations in the data, we assign a different probability for sharing each different state of this data point in $\mathcal{D}$. Let P_{x_j,d_k} be the probability of sharing data point x_j as d_k (i.e., $x'_{i,j} = d_k$). The proposed scheme assigns a P_{x_j,d_k} value for all $j \in \{1, \ldots, l\}$ and $k \in \{1, \ldots, m\}$. In the following, for simplicity, we describe the proposed fingerprinting algorithm by assuming pairwise correlations between consecutive data points and sequential order of processing of the data points for fingerprinting. However, the proposed system also works for different correlation models. We propose an

iterative algorithm that starts from the first data point x_1 and assigns probabilities $P_{x_1,d_1}, \ldots, P_{x_1,d_m}$. Since we consider correlations between consecutive data points, for the first data point x_1, similar to the naive approach, Alice shares the correct value with probability $1-p$ and each incorrect value with probability $p/(m-1)$. Based on these probabilities, the algorithm selects a value for $x'_{i,1}$ from the set $\mathcal{D}$. For the subsequent data points, the algorithm computes the fingerprinting probabilities by checking the correlation with the preceding data points.

Let the shared value of x_{j-1} (i.e., $x'_{i,j-1}$) be d_α. The algorithm checks the conditional probabilities $P(x_j = d_k | x_{j-1} = d_\alpha)$ for all $d_k \in \mathcal{D}$. If the algorithm decides to share x_j as d_k, and if $P(x_j = d_k | x_{j-1} = d_\alpha)$ is low, a malicious SP can detect that either $x'_{i,j-1}$ or $x'_{i,j}$ is fingerprinted. To eliminate such correlation attack, the proposed algorithm uses a threshold τ and sets $P_{x_j,d_k} = 0$ if $P(x_j = d_k | x_{j-1} = d_\alpha) < \tau$. Hence, the algorithm never selects d_k as the value of $x'_{i,j}$ if d_k is not consistent with the inherent correlations in the data. Let d_c be the actual value of x_j. If $P(x_j = d_c | x_{j-1} = d_\alpha) \geq \tau$, P_{x_j,d_c} is set to $1-p$. All of the remaining probabilities ($P_{x_j,d_k}, k \neq c$) are assigned directly proportional to the value of $P(x_j = d_k | x_{j-1} = d_\alpha)$. After assigning all probabilities, the algorithm chooses one of the values from $\mathcal{D}$ based on the assigned probabilities and sets the value of $x'_{i,j}$ accordingly.

Since the proposed algorithm considers correlations, total number of fingerprinted points in the original data may significantly deviate from the expected number ($p \cdot l$). Considering correlations may cause fingerprinting significantly more (or fewer) data points than anticipated. To prevent this, we dynamically decrease (or increase) the fingerprinting probability p depending on number of currently fingerprinted data points. To keep the average number of fingerprinted data points as $p \cdot l$, the proposed algorithm divides the data points into blocks consisting of $\lceil 1/p \rceil$ data points. We expect (on the average) one fingerprinted data point in each block. Therefore, the algorithm keeps a count of the number of fingerprinted data points at the end of each block. If the ratio of fingerprinted data points is less than p, the algorithm sets the fingerprinting probability for the next block as $p \cdot (1+\theta)$. Here, θ is a design parameter in the range $[0, 1)$ and we evaluate the selection of θ in Sect. 6. If the ratio of fingerprinted data points is greater than p, the algorithm sets the fingerprinting probability for the next block as $p \cdot (1-\theta)$.

For each SP, Alice executes the same algorithm with a different seed value (i.e., starting point in generating random numbers) and stores the fingerprint pattern of each SP to use it in the detection in case her data is leaked. If the data size is large, Alice can just store the seed value for each SP as the key of the fingerprinting algorithm and seed value can be used to generate the same fingerprint.

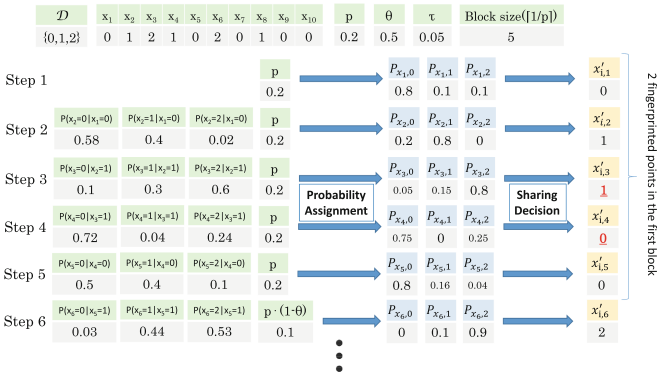


Fig. 2. A toy example showing the execution of the proposed algorithm. Input parameters of the algorithm are shown at the top as the original data $\mathcal{X}$, the fingerprinting probability p, probability adjustment parameter θ, correlation threshold τ, and the block size $\lceil 1/p \rceil$.

Figure 2 shows a toy example to illustrate the execution of the proposed algorithm. Each step shows one iteration for deciding the value of one data point. In the first step, probabilities are assigned just by using p. In the next steps, correlations are also used to determine the sharing probabilities. When the correlation is less than $\tau = 0.05$, the algorithm assigns 0 for the probability of selecting the corresponding value. At the end of the first block (a block includes 5 data points in the example), since the number of fingerprinted data points (2) is greater than the expected (1), the fingerprinting probability is adjusted as $p \cdot (1 - \theta) = 0.1$ for the second block.

4.2 Detecting the Source of Data Leakage

Let the leaked copy of Alice's data be $\mathcal{Y} = [y_1, \ldots, y_l]$. The goal of Alice is to detect the SP that leaks her data. Here, Alice can apply a probabilistic detection algorithm which was proposed in [12]. With some independence assumptions, their algorithm computes the probability of being guilty for each SP and returns the SP with the highest probability. We also propose to use the similarity between leaked data and fingerprinted copies in the following and compare the performance of probabilistic detection and similarity-based detection in terms of their accuracy to identify the guilty SP in Sect. 6.

Similarity-Based Detection. For an SP_i, Alice compares the leaked data $\mathcal{Y}$ with the copy $\mathcal{X}'_i$ and counts the matching data points in the fingerprint pattern. In other words, Alice checks the size of the following set: $\mathcal{M}_i = \{x_j \mid x_j \in \mathcal{X},\ x_j \neq x'_{i,j},\ x'_{i,j} = y_j\}$. Alice also counts the fingerprinted data points $\mathcal{F}_i = \{x_j \mid x_j \in \mathcal{X},\ x_j \neq x'_{i,j}\}$ of SP_i. Eventually, the SP with the maximum $sim_i = |\mathcal{M}_i|/|\mathcal{F}_i|$ is identified as guilty.

5 Considering Colluding Service Providers

Here, we first present a strong attack against the proposed fingerprinting scheme (also against the existing fingerprinting schemes in general) by integrating colluding SPs, correlations in the data, and the flipping attack. Then, we propose utilizing Boneh-Shaw codes [4] to improve robustness against such a strong collusion attack.

5.1 Probabilistic Majority Attack

As discussed, the goal of the colluding SPs is to share (leak) a copy of the data without being detected by Alice. In a standard collusion attack, the colluding SPs compare their received values for each data point and select the most observed value to share. We propose an advanced collusion attack called "probabilistic majority attack" (to distinguish it from the standard majority attack), in which the colluding SPs decide the value of each leaked data point by considering (i) all observed values for that data point, (ii) correlation of that data point with the others, and (iii) the probability of adding a fingerprint to a data point (p). If the colluding SPs do not know the fingerprinting probability p, we assume they use an estimated probability p_e in their attack ($p_e = p$ if p is publicly known).

Let the set of colluding SPs be $\mathcal{C}$ and $|\mathcal{C}| = n$. The goal of the colluding SPs is to create a copy $\mathcal{Y} = [y_1, \ldots, y_l]$ to share and avoid being detected by Alice. In this attack, the colluding SPs decide the value of each data point y_j by computing a probability P_{y_j,d_k} for each possible state $d_k \in \mathcal{D}$ of y_j. Let c_{j,d_k} be the number of observations of d_k for data point j by n colluding SPs (in $\mathcal{C}$). In the standard majority attack, the colluding SPs choose d_k with the maximum c_{j,d_k} value as the value of y_j (assuming it is the original value of the data point) to avoid detection by Alice. However, it is possible (with lower probability) that other values with lower c_{j,d_k} may also be the original value of y_j.

As discussed in Sect. 3.2, correlations can be used to detect and distort fingerprinted data points by the attackers. Therefore, the attackers tend to select each leaked value that have high correlation with the previous shared (leaked) values. In order to integrate the correlations with the collusion attack, the conditional probabilities (due to correlations) are used as weights to determine the sharing probabilities of colluding SPs (P_{y_j,d_k}). The colluding SPs first compute the weighted probability values (referred as t_{j,d_k}) and then compute P_{y_j,d_k} by normalizing the weighted probability values. The weighted probability for d_k value of a data point j is computed as: $t_{j,d_k} = (1-p_e)^{c_{j,d_k}} \cdot (\frac{p_e}{m-1})^{n-c_{j,d_k}} \cdot P(x_j = d_k | x_{j-1} = y_{j-1})$. Here, $(1-p_e)^{c_{j,d_k}} \cdot (\frac{p_e}{m-1})^{n-c_{j,d_k}}$ is the probability of d_k to be the original value of data point j by assuming each data point is fingerprinted with probability p_e. The conditional probability is used as a weight. Then, t_{j,d_k} values are normalized as $P_{y_j,d_k} = t_{j,d_k} / (\sum_{k=1}^{n} t_{j,d_k})$. The colluding SPs decide on the value of each shared point y_j proportional to these probabilities. Thus, they do not necessarily select the value observed by the majority. By doing so, we allow the malicious SPs to further distort the fingerprint compared to the standard majority attack.

Furthermore, existing techniques to prevent collusion attacks (e.g., Boneh-Shaw codes) assign common fingerprints to multiple SPs, which allows data owner to detect colluding SPs with a high chance. However, to avoid such detection, the attackers may flip some random data points before they leak the data. Thus, in the probabilistic majority attack, we also let colluding SPs flip each y_j with probability p_f (as discussed in Sect. 3.2). A toy example to illustrate this attack is given in Appendix A.

5.2 Integrating Boneh-Shaw Codes

In order to provide robustness against collusion attacks, Boneh and Shaw proposed fingerprinting codes (B-S codes) for detecting one of the colluding SPs [4]. The effectiveness of their codes depends on the "marking assumption", which states that when the colluding SPs have the same value for the same data point j, they choose this value as y_j as the leaked data point. Hence, it is assumed that colluding SPs cannot detect the fingerprint if all of them have the same fingerprint. However, B-S codes do not consider correlation and flipping attacks, and hence their detection method is not successful when colluding SPs also utilize the correlations and the flipping attack.

In Boneh-Shaw (c,r)-codes, ith codeword consists of $(i-1)\cdot r$ zeros and $(c-i)\cdot r$ ones. In order to fingerprint data, some data points are selected from the original data and XOR'ed with the permuted codeword in order to prevent colluding SPs from detecting and distorting the fingerprint. The data points that are XOR'ed with ones in the codeword becomes fingerprinted. Therefore, the ones in the binary code represent fingerprints. Thus, increasing r decreases the error in detection, but it also decreases the utility by increasing the fingerprint length. We show how to utilize these codes in the proposed scheme in the following.

Using Boneh-Shaw Codes in the Proposed Scheme. The marking assumption of [4] does not consider the flipping attack and correlation attack which are included in the probabilistic majority attack. If the colluding SPs flip some of the bits randomly or based on correlations, the data owner may accuse an SP who is not involved in the collusion. Therefore, using B-S codes and their detection algorithm in our scenario directly results in low robustness against the correlation and flipping attacks. Instead, we utilize B-S codes to assign overlapping fingerprints between different SPs.

The fingerprints of two SPs may be the same for some random data points when Alice creates fingerprinted copy of each SP independently as described in Sect. 4. Here, we explicitly assign overlapping fingerprints using B-S codes to improve robustness against collusion attacks. When a data point is fingerprinted using B-S codes, the same value is also used as a fingerprint in the copies of other SPs if their codewords also include one for the same data point.

We integrate the B-S codes into our scheme as follows: Alice creates the first fingerprinted copy of her data $\mathcal{X}'_1$ as described in Sect. 4. Approximately $p\cdot l$ data points are fingerprinted in $\mathcal{X}'_1$. Let f be the number of fingerprinted data points

for SP_1. We want to use some portion of these f data points as B-S codes. Then, Alice decides the value of c and r such that $(c-1) \cdot r \leq f$ to apply B-S codes for the next sharings. As mentioned, c and r are design parameters of B-S codes determining the length of codes and error in detection. We represent $(c-1) \cdot r$ as f_1, which is the length of B-S codeword. Here, c is the number of B-S codewords that Alice can create and r is the block size. If Alice wants to share her data with more than c SPs, she assigns the same B-S codewords in a similar order. SP_{c+1} receives the same codeword as SP_1, SP_{c+2} receives the same codeword as SP_2, and so on. In this way, although same B-S codewords are assigned to some SPs, since other parts of their fingerprints will be different, the proposed detection algorithm can still identify the guilty SP using the entire fingerprint. For higher f_1, the robustness against collusion attacks increases, however, the robustness against the attacks performed by single SP decreases. Hence, we set f_1 approximately equal to $f/2$ to detect the guilty SP regardless of whether the attack is performed by single SP or multiple SPs.

Alice randomly selects f_1 of f fingerprinted points in $\mathcal{X}'_1$. These f_1 fingerprinted data points are considered as the first codeword in B-S codes. For her next sharing with SP_2, Alice randomly selects $f_2 = f_1 - r$ of f_1 points to assign the same fingerprints to SP_2 (i.e., $x'_{2,j} = x'_{1,j}$ for these f_2 data points). Moreover, Alice assigns the original value for the remaining r points (i.e., $x'_{2,j} = x_j$ for these r data points). In other words, the B-S codeword of SP_2 consists of r zeros and $f_1 - r$ ones. In order to assign approximately $p \cdot l$ fingerprinted points to SP_2, the fingerprinting probability of SP_2 is selected as $\frac{p \cdot l - f_2}{l - f_1}$ since f_2 fingerprints are already assigned before running the probabilistic algorithm. Alice runs the proposed algorithm to sequentially add the remaining fingerprints. Also, since f_2 fingerprints and r original values are already assigned (as the B-S codeword of SP_2) before the algorithm, the algorithm will skip these points while adding fingerprints. Furthermore, when the algorithm is determining the probabilities for each possible value of a data point (i.e., inner loop of the algorithm), it also considers the correlations of the data points with the already assigned B-S codeword. The updated algorithm is shown in Algorithm C.1 in Appendix C , which includes these new conditions. For each SP_i, Alice repeats the same process by first adding f_i fingerprints and $f_1 - f_i$ original values (i.e., B-S codeword). Then, Alice runs Algorithm C.1 to determine the values of remaining points in $\mathcal{X}'_i$. We illustrate this process with a toy example in Appendix B.

5.3 Detection Algorithm

Here, we propose a detection algorithm for the proposed collusion-resilient fingerprinting scheme. In practice, when Alice realizes that a copy $\mathcal{Y}$ of her data is leaked without her consent, she cannot know whether $\mathcal{Y}$ is leaked by single SP (by performing flipping or correlation attack) or multiple SPs (by performing collusion attack). Thus, Alice cannot use the detection techniques in Sect. 4.2 or the detection technique of B-S codes directly. We propose a detection algorithm that utilizes both techniques. We describe the detection algorithm using

similarity-based detection since it performs slightly better than probabilistic detection as we show in Sect. 6.

To detect a guilty SP, Alice initially computes a similarity score (sim_i) in the range $[0, 1]$ for each $SP_i \in \mathcal{S}$ as explained in Sect. 4.2. If there is a collusion of two SPs and the colluding SPs observe different values for a data point, they select either of these values with equal probabilities. Thus, we expect that they damage approximately half of the fingerprinted data points (this will be more if the collusion includes more than 2 SPs). Hence, we assume that there is an attack by single SP if the similarity score of an SP is greater than 0.5. In such a case, Alice identifies the SP with the highest similarity score as guilty. Otherwise, there is a collusion attack with high probability, and hence Alice identifies the suspects according to their similarity scores and returns one of them utilizing the detection technique of B-S codes.

Let the index of SP with maximum similarity score be max. Alice generates a suspect list by including $\lfloor 1/sim_{max} \rfloor$ SPs having highest similarity scores. Hence, if sim_{max} is greater than 0.5, there will be just one SP in the suspect list and the algorithm will return SP_{max} as guilty. If sim_{max} is less than or equal to 0.5, there will be more than one suspects, which means that a collusion attack is performed with high probability. In this case, the algorithm returns one of the suspects using the detection method of B-S codes [4] as follows: In B-S codes, it is expected that the colluding SPs create a copy consisting of several random values followed by all ones and the starting point of ones (a block with all ones) gives us one of the colluding SPs. Let B_R represents a block (consists of r data points) having at least one zero value and B_1 represents a block having all ones. Assuming the leaked copy created by colluding SPs is $[B_R, ..., B_R, B_1, ..., B_1]$, SP_i is identified as guilty if the first observed B_1 block is the ith block. However, as a result of probabilistic majority attack described in Sect. 5.1, some ones may turn into zeros and some zeros may turn into ones. Therefore, the detection algorithm may fail against such an attack. To avoid this, we define $\hat{B}_1$ as a block having majority of points as one and $\hat{B}_R$ as a block having majority of points as zero. Then, the algorithm checks all suspects in the suspect list starting from SP_{max}. If (max)th block is $\hat{B}_1$ and $(max - 1)$th block is $\hat{B}_R$, the algorithm returns SP_{max} as guilty. Otherwise, the algorithm continues with the other SPs in the suspect list in the order of decreasing similarity scores. For each SP_i in the suspect list, the algorithm returns it as guilty if the (i)th block is $\hat{B}_1$ and the $(i - 1)$th block is $\hat{B}_R$. When such an SP is found, the algorithm stops and returns the SP as guilty. If there is no such an SP, the algorithm returns SP_{max} as guilty. The steps of the proposed detection algorithm are also shown with an example in Appendix D.

6 Evaluation

To evaluate its robustness and utility, we implemented the proposed fingerprinting algorithm (in Sect. 5.2) and the detection algorithm (in Sect. 5.3). We implemented Tardos codes [14] as a state-of-the-art fingerprinting scheme for comparison. We also implemented B-S codes as a standalone fingerprinting scheme

and observed that its robustness against the attacks is similar to Tardos codes. Hence, we here present the results of Tardos codes to show the superiority of our scheme to deal with correlation and flipping attacks.

6.1 Data Model and Settings

Nowadays, individuals can obtain their genome sequences easily and share their genomic data with medical institutions and direct-to-consumer service providers for various genetic tests or research purposes. Since genomic data contains sensitive personal information, such as the risk of developing particular diseases, sharing genomic data without the authorization of the data owner causes privacy violations. Hence, fingerprinting genomic data can be a solution or disincentive to prevent its unauthorized sharing. Furthermore, genomic data contains inherent pairwise correlations between point mutations (single nucleotide polymorphisms - SNPs), which makes genomic data an ideal usecase to evaluate the proposed scheme. SNP is the variation of a single nucleotide (from the set $\{A, T, C, G\}$) in the population. For each SNP position, only two different nucleotides can be observed: (i) major allele, which is observed in the majority of the population and (ii) minor allele, which is observed rarely. Each SNP consists of two nucleotides, one is inherited from the mother and the other from the father. Since each SNP is represented by the number of its minor alleles, $\mathcal{D} = \{0, 1, 2\}$ for genomic data.

The 1000 Genomes Phase 3 dataset [1] includes partial genomic records of 2504 individuals from 26 populations. Among these, we extracted the 5000 SNPs belonging to 99 people from the Utah residents with northern and western European ancestry (CEU) population of the HapMap project. Using this dataset, we computed the correlations between SNPs to build our correlation model. Unless stated otherwise, we set the data size $l = 1{,}000$, fingerprinting probability $p = 0.1$, and the correlation threshold of the algorithm $\tau = 0.05$. The threshold τ is used in the algorithm to prevent adding fingerprints causing low correlation. Note that the attacker has its own correlation threshold τ_c in its correlation attack. We choose the data size as 1,000 to show the robustness of the proposed scheme for a relatively small data. As we show via experiments later (in Sect. 6.4), robustness increases with the increasing data size because as data size increases, we obtain more fingerprinted data points for the same fingerprinting probability p. Hence, for larger data sizes, p can be selected much lower than 0.1 to provide the same level of fingerprint robustness. As expected, utility of the data owner (as introduced in Sect. 3.1) decreases linearly with increasing p. We observed that the average utility is 0.8 when p is 0.1 and u_j (utility of each data point) is $1/l$. To compare the robustness of the proposed scheme with Tardos codes, we allocated $2p$ of the data points for Tardos codes. Hence, when $p = 0.1$ is used in the proposed scheme, 20% of the data points are allocated for Tardos codes. Since approximately half of them are fingerprinted in probabilistic Tardos codes, 10% of the data points are fingerprinted in each copy, which provides approximately the same utility with the proposed scheme.

We expect to change the value of approximately $p \cdot l = 100$ data points as fingerprint for each SP, and (as discussed in Sect. 5.2) we used approximately half

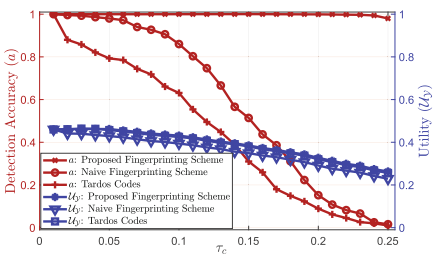
of the fingerprinted data points for B-S codes. We set the number of B-S codes (c) as 10 and the block size (r) as 5. Hence, $(c-1)\cdot r = 45$ fingerprinted data points of first SP were used for B-S codes. Another design parameter in the algorithm is θ can have any value in the range $[0, 1)$. It is used to dynamically adjust fingerprinting probability p to keep the number of fingerprinted data points close to $p \cdot l$. We set $\theta = 0.5$ in our experiments. We repeated all experiments 10 times for each individual in the dataset (totally 99 individuals), and hence all results are given as the average of 990 executions. We also provide a discussion about the complexity and practicality of the proposed scheme in Appendix E.2, which shows that the running times of both fingerprinting algorithm and detection algorithm grow linearly with the design parameters.

6.2 Flipping and Subset Attacks

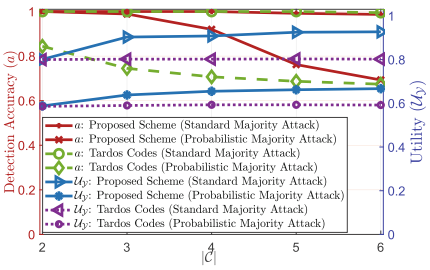
We implemented flipping and subset attacks to compare these attacks as well as to compare similarity-based and probabilistic detection techniques. We present our results in Appendix E.1. From these results, we can conclude that (i) similarity-based detection provides slightly better accuracy than the probabilistic detection, (ii) flipping attack is more powerful than subset attack, (iii) the attacker needs to flip at least half of the data points in the proposed scheme to avoid being detected, and (iv) the proposed scheme is more robust than Tardos codes against flipping and subset attacks. For the rest of the experiments, we use the similarity-based detection algorithm described in Sect. 5.3.

6.3 Correlation Attack

We implemented the correlation attack described in Sect. 3.2 to evaluate the robustness of the proposed scheme. We set the total number of SPs to 1,000. As before, we set the correlation threshold of the algorithm (decided by data owner) as $\tau = 0.05$. Therefore, the fingerprinted copies did not include consecutive pairs of data points whose correlation is less than 0.05. Note however that the correlation threshold of the attack τ_c is determined by the attacker. We also implemented the naive fingerprinting scheme described in Sect. 4, in which each data point is fingerprinted with probability p. In correlation attack, data points whose correlation with the previous data point is less than τ_c is flipped and the remaining data points are flipped with probability p_f. Figure 3a shows the comparison of the proposed scheme with the naive approach and Tardos codes for different values of τ_c when $p_f = 0.2$. The proposed scheme provides 100% detection accuracy up to $\tau_c = 0.2$ and accuracy decreases to 98% when $\tau_c = 0.25$. However, as also shown in Fig. 3a, the utility of the attacker ($\mathcal{U}_\mathcal{Y}$) reduces to 0.263 when $\tau_c = 0.25$. We also observed that both the naive approach and Tardos codes are not robust against correlation attacks and the attacker can easily prevent detection by utilizing the correlations in the data. This clearly shows the importance of considering correlations in the data within the fingerprinting algorithm.



(a) Fingerprint robustness of the proposed algorithm, Tardos codes, and the naive algorithm against correlation attack for different values of correlation attack threshold.



(b) Fingerprint robustness of the proposed scheme and Tardos codes against standard majority attack and probabilistic majority attack for different values of n.

Fig. 3. Fingerprint robustness against different attacks.

6.4 Collusion Attack

We first compare the utility and robustness under standard and probabilistic majority attacks. We set $c = 10$ and $r = 5$. Hence, we can create 10 different B-S codewords with block size of 5. Note that c is the number of codewords and Alice can share her data more than c SPs in the proposed scheme by repeating codewords as we discussed before. We set the total number of SPs ($|\mathcal{S}|$) to 20, $\tau_c = 0.1$, and $p_f = 0.1$. We quantified both utility and fingerprint robustness for different values of number of colluding SPs (n). As shown in Fig. 3b, using probabilistic majority attack decreases the colluding SPs' probability of being detected by reducing the data utility. Since the probabilistic majority attack is more powerful than the standard one for the colluding SPs, we perform the probabilistic majority attack for the rest of the experiments.

As mentioned, both Tardos codes and B-S codes assume the colluding SPs decide the value of a data point randomly if they observe more than one value in their copies (i.e., marking assumption). Therefore, these codes do not provide guarantees against the probabilistic majority attack (including the flipping attack). In Fig. 3b, we showed the robustness of the Tardos codes against collusion attacks (we observed similar results for B-S codes). Although Tardos codes provide high robustness against standard majority attack, robustness of these codes against probabilistic majority attack is lower than our proposed scheme since Tardos codes do not consider correlations and random flipping in probabilistic majority attack. Moreover, to provide the robustness guarantees for Tardos codes and B-S codes, the number of fingerprinted points needs to be high. For instance, to provide 90% robustness guarantee against 3 colluding SPs performing standard majority attack, 2,700 fingerprinted data points are needed in Tardos codes regardless of the data size. Similarly, to create 10 B-S codes with the same guarantee, we need more than 10,000 fingerprinted data points. Fingerprinting such a high number of data points also decreases the utility of

the data owner significantly. Thus, Tardos codes and B-S codes provide guarantees for only standard majority attack with high number of fingerprinted data points (we only use approximately $p \cdot l = 100$ data points for fingerprinting in our experiments and, as we show later, our scheme provides high robustness with smaller p when data size increases). Therefore, we conclude that Tardos codes and B-S codes do not provide robustness against the attacks utilizing correlations and random flipping. While our proposed scheme utilizes the B-S codes to increase its robustness against collusion attacks, Algorithm C.1 generates unique fingerprints to also provide robustness against correlation and flipping attacks.

One important parameter for the fingerprinting scheme is data size. In our experiments we used a data with 1,000 SNPs ($l = 1,000$). Therefore, we just changed the state of approximately 100 data points as a fingerprint when fingerprinting probability p was selected as 0.1. By keeping the same fingerprinting probability, increasing data size allows to change more data points as fingerprint. With more fingerprinted data points, the data owner can detect the colluding SPs with higher accuracy. To show the effect of data size (i.e., l) on robustness, we conducted experiments by increasing l (we kept $c = 10$ and increased r proportional to l). As shown in Table 1, Alice detects one of 3 colluding SPs among 100 SPs with 99.7% accuracy when $l = 5,000$ and $p_f = 0.1$. Our results show that the robustness of the proposed scheme significantly improves with increasing data size. Thus, when data size is larger, Alice can select a much lower fingerprint probability (p) to obtain the same robustness guarantees. For instance, when data size (l) is 5,000, decreasing p from 0.1 to 0.05 increases the utility of fingerprinted data from 0.8 to 0.9 while still providing 96.3% accuracy for 3 colluding SPs performing a probabilistic majority attack.

Table 1. Fingerprint robustness (a) of the proposed scheme for different l (data size) values. The number of SPs (that received data owner's data) is 100 and the number of colluding SPs (n) is 3. Flipping probability in the attack is 0.1.

l	1,000	2,000	3,000	4,000	5,000
a	0.759	0.914	0.973	0.993	0.997

7 Conclusion

We have proposed a probabilistic fingerprinting scheme that also considers the correlations in the data during fingerprint insertion. First, we have shown how to assign probabilities for the sharing decision of each data point that are consistent with the inherent correlations in the data. Then, we have described the integration of Boneh-Shaw codes into the proposed algorithm to improve fingerprint robustness against collusion attacks. Our experimental results on genomic data show that the proposed fingerprinting scheme is robust against a wide range of attacks. We plan to evaluate the proposed scheme on trajectory data in the future.

Acknowledgements. The work was partly supported by the National Library of Medicine of the National Institutes of Health under Award Number R01LM013429 and by the National Science Foundation (NSF) under grant numbers 2141622, 2050410, 2200255, and OAC-2112606.

A Toy Example for the Probabilistic Majority Attack

To illustrate the probabilistic majority attack with a toy example, let $n = 4$ and $\mathcal{D} = \{0, 1, 2\}$. Assume 3 of the colluding malicious SPs have received value 0 for the first data point (x_1) and the other malicious SP has received value 1. Let the estimated fingerprinting probability $p_e = 0.1$. Colluding SPs compute $t_{1,0} = (0.9)^3 \cdot (0.1)^1 \cdot 1$, $t_{1,1} = (0.9)^1 \cdot (0.1)^3 \cdot 1$, and $t_{1,2} = (0.9)^0 \cdot (0.1)^4 \cdot 1$. Since x_1 is the first data point and we consider pairwise correlations between consecutive data points, here, conditional probabilities are all considered as 1. Then, colluding SPs choose a value (to share) from $\mathcal{D}$ with the following probabilities: $P_{y_1,0} = 0.0729/0.0739 = 0.987$, $P_{y_1,1} = 0.0009/0.0739 = 0.012$, and $P_{y_1,2} = 0.0001/0.0739 = 0.001$. Finally, the chosen value is flipped with probability p_f.

B Toy Example for Using Boneh-Shaw Codes in the Proposed Algorithm

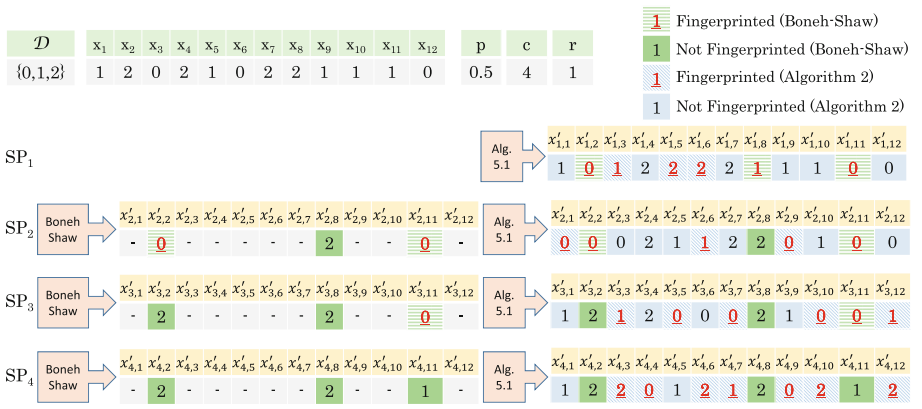


Fig. 4. An example execution of the proposed algorithm by integrating Boneh-Shaw codes.

Here, we describe the algorithm explained in Sect. 5.2 on a toy example, which is also illustrated in Fig. 4. Let the original data of Alice be $\mathcal{X} = [1, 2, 0, 2, 1, 0, 2, 2, 1, 1, 1, 0]$, where $l = 12$ and $\mathcal{D} = \{0, 1, 2\}$. Let p be 0.5 and Alice shares $\mathcal{X}'_1 = [1, \underline{0}, \underline{1}, 2, \underline{2}, \underline{2}, 2, \underline{1}, 1,\ 1, \underline{0}, 0]$ with SP_1 after running Algorithm C.1, where underlined points represent fingerprinted data points.

C The Fingerprinting Algorithm

In Algorithm C.1, we provide the algorithm discussed in Sect. 5.2.

D Toy Example for the Proposed Detection Algorithm

The steps of the proposed detection algorithm (in Sect. 5.3) are also shown with an example in Fig. 5. After checking the similarity of leaked data with the fingerprinted data points of each SP, the algorithm adds $\lfloor 1/sim_{max} \rfloor = 2$ SPs into the suspect list. When the algorithm checks the 1st and the 2nd blocks of leaked data for SP_2, it does not return SP_2 as guilty since both blocks are $\hat{B}_R$. Then, it checks the 3rd and the 4th blocks of leaked data for SP_4 and returns SP_4 as guilty since the 4th block is $\hat{B}_1$ and the 3rd block is $\hat{B}_R$.

Algorithm C.1: Probabilistic fingerprinting scheme after assigning the Boneh-Shaw codeword to provide robustness against the collusion attack. Blue parts represent the difference with algorithm described in Section 4.

input : Original data $\mathcal{X} = [x_1, x_2, \ldots, x_l]$, f_1 already assigned points in $\mathcal{X}'_i$ (f_i of them are fingerprinted), fingerprinting probability $\frac{p \cdot l - f_i}{l - f_1}$, probability adjustment parameter θ, block size $\lceil 1/p \rceil$, correlation threshold τ, pairwise correlations between data points.

output: Fingerprinted copy $\mathcal{X}'_i = [x'_{i,1}, x'_{i,2}, \ldots, x'_{i,l}]$

$prob \longleftarrow \frac{p \cdot l - f_i}{l - f_1}$;
forall $j \in \{1, 2, \ldots, l\}$ **do**
  if $x'_{i,j}$ *is not assigned* **then**
    forall $k \in \{1, 2, \ldots, m\}$ **do**
      if $j = 1$ & $x_j = d_k$ **then**
        $P_{x_j, d_k} \longleftarrow 1 - prob$;
      else if $j = 1$ & $x_j \neq d_k$ **then**
        $P_{x_j, d_k} \longleftarrow prob/(m-1)$;
      else if $P(x_j = d_k | x_{j-1} = x'_{i,j-1}) < \tau$ **then**
        $P_{x_j, d_k} \longleftarrow 0$;
      else if $x'_{i,j+1}$ *is assigned* & $P(x_{j+1} = x'_{i,j+1} | x_j = d_k) < \tau$ **then**
        $P_{x_j, d_k} \longleftarrow 0$;
      else if $x_j = d_k$ **then**
        $P_{x_j, d_k} \longleftarrow 1 - prob$;
    end
    distribute the remaining probability (1 - (sum of assigned probabilities)) by assigning P_{x_j, d_k} directly proportional to the value of $P(x_j = d_k | x_{j-1} = x'_{i,j-1})$ if P_{x_j, d_k} is not assigned in the previous step;
    $x'_{i,j} \longleftarrow$ random value from $\mathcal{D}$ using probability distribution $P_{x_j, d_1}, \ldots, P_{x_j, d_m}$;
  end
  if j *is multiple of* $\lceil 1/p \rceil$ **then**
    $c \longleftarrow$ total number of fingerprinted data points;
    if $c > p \cdot j$ **then**
      $prob \longleftarrow p \cdot (1 - \theta)$;
    else if $c < p \cdot j$ **then**
      $prob \longleftarrow p \cdot (1 + \theta)$;
    else
      $prob \longleftarrow p$;
    end
  end
end

E Experimental Results

E.1 Flipping and Subset Attacks

In this experiment, we compared the flipping and subset attacks in terms of their effect on the fingerprint robustness. Also, to compare the similarity-based and probabilistic detection techniques (which are the basic building blocks of the proposed detection algorithm in Sect. 5.3), we implemented them for this experiment. We set the total number of SPs to 1,000. Figure 6 shows the accuracy (a) of the both detection techniques for different values of p_f (probability of flipping a data point in flipping attack) and p_s (probability of removing a data point in subset attack).

E.2 Complexity and Practicality

In the proposed fingerprinting algorithm (Algorithm C.1), each data point sequentially decides on a probability for each possible value in set $\mathcal{D}$ and inserts

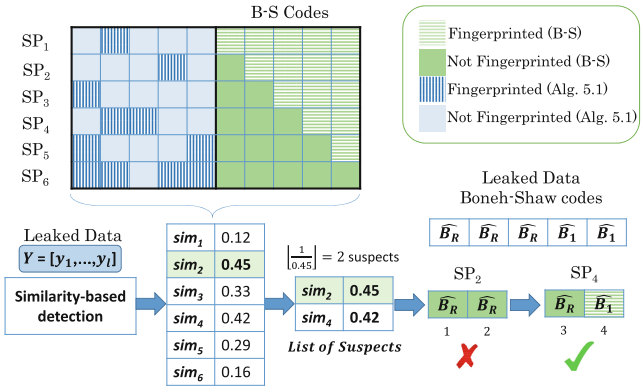


Fig. 5. An example execution of the detection algorithm.

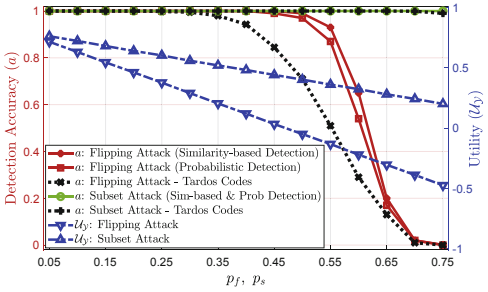


Fig. 6. Fingerprint robustness of the proposed scheme and Tardos codes against flipping and subset attacks for different values of p_f and p_s. Right y-axis is used to show the utility of the attacker.

the fingerprints accordingly. Hence, the complexity of fingerprinting algorithm is $\Theta(l \cdot m)$. To detect the guilty SP in case of data leakage, the data owner needs to compare all fingerprint patterns (given to all SPs) with the leaked data. Since the expected value of fingerprinted data points is $p \cdot l$ in each fingerprinted copy, the complexity of the detection algorithm is $\Theta(|\mathcal{S}| \cdot p \cdot l)$, where $|\mathcal{S}|$ is the number of SPs that received a fingerprinted copy. Note that this is also the storage complexity for Alice if she stores all the fingerprint patterns. As mentioned before, if Alice does not want to store all fingerprint patterns, she can just store the seed value for each SP, which slightly increases the complexity of detection algorithm since it requires Alice to run the fingerprinting algorithm along with the detection algorithm. Thus, we conclude that the running times of both fingerprinting algorithm and detection algorithm grow linearly with the design parameters.

Based on our implementation with Java using a computer with 1.8 GHz Dual-Core Intel Core i5 processor and 8 GB memory, we measured the average running time of fingerprinting algorithm to create one fingerprinted copy as 0.15 ms. and the average running time of detection algorithm as 3.11 ms. when $|\mathcal{S}| = 1000$, $l = 1000$, $p = 0.1$, and $m = 3$. These results also show the efficiency and practicality of the proposed scheme.

References

1. (2023). https://mathgen.stats.ox.ac.uk/impute/1000GP_Phase3.html. Accessed 10-January-2023
2. Ayday, E., Yilmaz, E., Yilmaz, A.: Robust optimization-based watermarking scheme for sequential data. In: 22nd International Symposium on Research in Attacks, Intrusions and Defenses ({RAID} 2019), pp. 323–336 (2019)
3. Bassia, P., Pitas, I., Nikolaidis, N.: Robust audio watermarking in the time domain. IEEE Trans. Multimed. **3**(2), 232–241 (2001)
4. Boneh, D., Shaw, J.: Collusion-secure fingerprinting for digital data. IEEE Trans. Inf. Theory **44**(5), 1897–1905 (1998)
5. Brassil, J., Low, S., Maxemchuk, N.: Copyright protection for the electronic distribution of text documents. Proc. IEEE **87**(7), 1181–1196 (1999)
6. Cox, I.J., Miller, M.L., Bloom, J.A., Honsinger, C.: Digital watermarking. Springer (2002)
7. Fodor, G., Schelkens, P., Dooms, A.: Fingerprinting codes under the weak marking assumption. IEEE Trans. Inf. Forensics Secur. **13**(6), 1495–1508 (2017)
8. Hartung, F., Girod, B.: Watermarking of uncompressed and compressed video. Signal Process. **66**(3), 283–301 (1998)
9. Jin, X., Zhang, Z., Wang, J., Li, D.: Watermarking spatial trajectory database. In: Zhou, L., Ooi, B.C., Meng, X. (eds.) DASFAA 2005. LNCS, vol. 3453, pp. 56–67. Springer, Heidelberg (2005). https://doi.org/10.1007/11408079_8
10. Li, Y., Swarup, V., Jajodia, S.: Fingerprinting relational databases: Schemes and specialties. IEEE Trans. Dependable Secure Comput. **2**(1), 34–45 (2005)
11. Nuida, K., et al.: An improvement of discrete tardos fingerprinting codes. Des. Codes Crypt. **52**(3), 339–362 (2009)
12. Papadimitriou, P., Garcia-Molina, H.: Data leakage detection. IEEE Trans. Knowl. Data Eng. **23**(1), 51–63 (2010)

13. Škorić, B., Katzenbeisser, S., Celik, M.U.: Symmetric tardos fingerprinting codes for arbitrary alphabet sizes. Des. Codes Crypt. **46**(2), 137–166 (2008)
14. Tardos, G.: Optimal probabilistic fingerprint codes. J. ACM (JACM) **55**(2), 1–24 (2008)
15. Wu, M., Trappe, W., Wang, Z.J., Liu, K.R.: Collusion-resistant fingerprinting for multimedia. IEEE Signal Process. Mag. **21**(2), 15–27 (2004)
16. Zhao, X., Liu, Q., Zheng, H., Zhao, B.Y.: Towards graph watermarks. In: Proceedings of the 2015 ACM on Conference on Online Social Networks, pp. 101–112. ACM (2015)