Privacy-preserving and robust watermarking on sequential genome data using belief propagation and local differential privacy

Abdullah Çağlar Öksüz¹, Erman Ayday¹,²*, and Uğur Güdükbay¹,*
¹Department of Computer Engineering, Bilkent University, Ankara, Turkey and ²Department of Electrical Engineering and Computer Science, Case Western Reserve University, Cleveland, OH, USA

*To whom correspondence should be addressed.

Abstract

Motivation: Genome data is a subject of study for both biology and computer science since the start of the Human Genome Project in 1990. Since then, genome sequencing for medical and social purposes becomes more and more available and affordable. Genome data can be shared on public websites or with service providers (SPs). However, this sharing compromises the privacy of donors even under partial sharing conditions. We mainly focus on the liability aspect ensued by the unauthorized sharing of these genome data. One of the techniques to address the liability issues in data sharing is the watermarking mechanism.

Results: To detect malicious correspondents and SPs—whose aim is to share genome data without individuals’ consent and undetected—, we propose a novel watermarking method on sequential genome data using belief propagation algorithm. In our method, we have two criteria to satisfy. (i) Embedding robust watermarks so that the malicious adversaries cannot temper the watermark by modification and are identified with high probability. (ii) Achieving local differential privacy in all data sharings with SPs. For the preservation of system robustness against single SP and collusion attacks, we consider publicly available genomic information like Minor Allele Frequency, Linkage Disequilibrium, Phenotype Information and Familial Information. Our proposed scheme achieves 100% detection rate against the single SP attacks with only 3% watermark length. For the worst case scenario of collusion attacks (50% of SPs are malicious), 80% detection is achieved with 5% watermark length and 90% detection is achieved with 10% watermark length. For all cases, the impact of on precision remained negligible and high privacy is ensured.

Availability and implementation: https://github.com/acoksuz/PPRW_SGD_BPLDP

Contact: exa208@case.edu or gudukbay@cs.bilkent.edu.tr

Supplementary information: Supplementary data are available at Bioinformatics online.

1 Introduction

Digital watermarking is one of the most important technological milestones in digital data hiding. It is used as a technique to hide a message or pattern within the data itself for various reasons like copyright protection or source tracking of digitally shared data. Watermarks may contain information about the legal owner of data, distribution versions and access rights (Lee and Jung, 2001). Although watermarking has a wide range of applications, implementation schemes require different configurations for each use case and data type. For embedding copyright information in data and source tracking, robustness (Barni and Bartolini, 2004) against modifications is the crucial factor to preserve. The factors influencing such configurations alter depending on the characteristics of data, as well. Non-existent redundancy, the existence of correlations and prior knowledge for inference and the utility’s impact on attacker inference in sequential data are such factors that prevent the explicit implementation of digital data watermarking methods on sequential data.

We propose a novel watermarking scheme for sharing sequential genomic data consisting of single-nucleotide polymorphism (SNPs) with three states to be used by medical service providers (SPs). Each SP has access to the uniquely watermarked version of some individuals’ genomic data. The preliminaries expected from this watermarking scheme are robustness against watermark tampering attacks such as modification and removal, imperceptibility for not revealing watermark locations, preserving utility of original data to a degree through a minimum number of changes and satisfy local differential...
privacy (LDP) in watermarks so that the watermarked versions of the data are indistinguishable from actual human genomic data and provide plausible deniability to data owners. By doing so, the watermarked data will not be shared in an unauthorized way and the source(s) of a leak will be easily identified by the data owner. To solve this multi-objective optimization problem, we use the belief propagation (BP) algorithm, which helps us to determine optimal watermarking indices on data that may preserve robustness with the highest probabilities. In BP, we consider the public knowledge about the human genome like Minor Allele Frequencies (MAFs) of SNPs, point-wise correlations between SNPs, called Linkage Disequilibrium (LD) and the prior knowledge of genotype and phenotype information that may potentially leak probabilities about watermarked points. Through conversion of prior information (MAF, LD and so on) into the marginal probability distribution of the three SNP states, we manage to infer the state probabilities of each SNP. Our contributions are as follows:

1. We introduce a novel method for watermarking sequential data concerning the privacy of data and the robustness of watermark at the same time. We present the method’s strengths and weaknesses in various attack scenarios and provide insight into the weaknesses.

2. Our method uses prior information (MAFs, phenotype information and so on) and inherent correlations to infer the state probabilities of SNPs. Using these inferred probabilities, we select SNPs that satisfy the following two criteria in a non-deterministic setup: a low probability of robustness decrease (change resistance) when attacked and a low utility loss (efficient index selection) when changed. By giving priority to these SNP points for watermarking, we guarantee the preservation of robustness and utility in data against various attacks. Besides, the identification probabilities of single SNPs using prior information are decreased with this method.

3. We test the robustness and limitations of our method using collusion (i.e., a comparison using multiple watermarked copies of the same data) and modification attacks and demonstrate how to reach a high probability of detection with various parameters, such as the watermark length, the number of SPs, the number of malicious SPs and the ϵ coefficient of LDP.

4. We introduce randomly distributed non-genome-conflicting noise generated for the data to act naturally as watermarks and create imperceptible watermark patterns from the normal human genome if not attacked with collusion. Hence, rather than creating a fixed number of point-wise changes and tracking these changes for source tracking, we evaluate the whole data and reach a high probability of detection with a minimum number of changes.

5. We introduce watermarking schemes that satisfy ϵ-LDP and plausible deniability in data along with it for data owners who value additional manners of enhanced privacy.

We provide a summary of related background on genomics, specifically Minor Allele Frequency (MAF) and LD, and LDP, in Supplementary Appendix Section 1.

2 Related works
Recent advances in molecular biology and genetics and next-generation sequencing increased the amount of genomics data significantly (Carter, 2019). While achieving a breakthrough in the genomics field, genomics data poses an important privacy risk for individuals by carrying sensitive information, i.e. kinship, disease, disorder or pathological conditions (Grishin et al., 2019). Thus, collecting, sharing and conducting research on genomic data became difficult due to privacy regulations (CMS, 1996). Furthermore, Humbert et al. (2013) show that sharing genomic data also threatens the relatives due to kin relation of genomic data. To this end, several works have been conducted to find emerging ways of privacy-preserving collection and analysis of the genomic and medical data in the last decade. Some of the privacy-preserving techniques used for medical data collection are k-anonymity, l-diversity, de-identification, perturbation, anonymization, or ϵ-closeness (Kargupta et al., 2003; Li and Li, 2006; Machanavajjhala et al., 2007; Samurai and Sweeney, 1998; Wylie and Mineau, 2003).

These methods, however, provide limited privacy protection and are prone to inference attacks. Ayday et al. (2013) proposed obfuscation methods in which the output domain is divided into several sections and one section is reserved for genomic data protection.

Digital watermarking is a technique usually used for copy protection by inserting a pattern to the digital signal such as a song, image, or video (Cox et al., 2008). It is an attack counter-measure for the case of leakage or sharing without consent. Watermarking does not prevent leakage; it is used as a detection technique for malicious parties. Watermarking schemes for sequential data, especially for genomic data, are very rare. Iritkhar et al. (2015) proposed a robust and distortion-free watermarking scheme, GenInfoGuard, for genomic data. Similar to ours, Liss et al. (2012) proposed a permanent watermarking scheme in synthetic genes that embeds binary string messages on open-frame synonymous amino-acid codon regions. Heider and Barneckow (2008) proposed the use of artificial dummy strands to act like watermarks on DNA.

Ayday et al. (2019) proposed a robust watermarking scheme for sharing sequential data against potential collusion attacks by using non-linear optimization. Our objective model is similar to theirs. Different from their study, we consider prior information in which besides correlations, all sequential genomic data related information like familial genomes, phenotype states can be included by using factor nodes in the BP algorithm. Besides, we designed single SP, collusion and removal attacks that incorporate all information so that the worst-case scenarios are assumed and the attack model becomes more inclusive. Another difference between our method and theirs is the incorporation of ϵ-LDP as an extra measure of privacy without impacting security. Andrés et al. (2012) proposed a method of embedding noise in sequential location data for geo-indistinguishability without violating the differential privacy. Inspired by their study and their new differential privacy criteria, we implemented a local setup that prevents the LDP violations in every data index.

3 Problem definition
We present the data, system and threat models and the objective of our system. Frequently used symbols and notations are presented in Supplementary Appendix Section 2.1, Table 3.

3.1 Data model
Sequential data contain ordered data points \(x_1, x_2, \ldots, x_{d_l}\), where \(d_l\) is the length of the data. The values of \(x_i\) can be in different states from the set \(\{y_1, y_2, \ldots, y_m\}\) depending on the data type. For example, \(x_i\) can be an hour, minute, or second triplets ranging from 0 to 23, 59, 59, respectively, for timestamp data. For our system, we will use 0, 1 and 2 for the SNP states of homozygous major, heterozygous and homozygous minor, respectively. The data length is \(d_l\) and the number of points that will be watermarked at the end of the algorithm is \(w_0\).

3.2 System model
We consider a system between the data owner (Alice) and multiple SPs with whom Alice shares sequential genome data. SPs can be medical researchers, medical institutions, or bio-technical companies. Alice may decide to share the whole data or parts of it to receive different services. Besides, the parts shared with each SP may differ.

For all cases listed above, Alice wants to ensure solely that her data will not be shared unauthorized by SPs. If the data is shared,
she wants to preserve a degree of differential privacy and detect the malicious SP(s) who shared the data. She uses watermarking and shares a watermarked version of the data, which satisfies the degree of privacy she desires. These versions are produced by removing certain parts or modifying the data. Data indices most optimal for the satisfaction of the mentioned criteria and the purposes of SPs who may use the data for including but not limited to research should be calculated beforehand by considering the structure, distribution and vulnerabilities of the data. Therefore, the proposed solution to the watermarking problem should have the element of flexibility to differing requirements of Alice and SPs, and the different pieces of information that might be used for the inference of data. To calculate the complex probability distributions of multi-variable sequential data among many things we use BP. Please see Supplementary Appendix Section 2 for the details of BP and our proposed solution. Other graph inference methods could be used but BP is adapted because of its approximation efficiency in non-loopy graph networks.

Watermarking is mostly done by changing the status of data indices. Adding dummy variables is an example of methods that do not change the actual values but common methods used for watermarking are usually removal or modification. Since a slight addition in sequential data causes a shift in other indices, it impacts the rest of the retrieval and embedding processes like the butterfly effect. We stick with the watermarking method by removal or modification. In a broader sense, non-sharing can be considered as modifying the status of a certain index into ‘non-available’. Normally, the security of a watermarking scheme increases along with the length of the watermark against attacks. However, a robust watermark should be short and as efficient as possible to maximize the detection probability of malicious SPs without reducing utility significantly.

3.3 Threat model

The objective is in our proposed system and watermarking, in general, is identifying the source(s) of leakage when the data is shared in an unauthorized way. In the threat model, contrary to our objective, the only goal of malicious SPs is to share the data undetected. SPs can achieve this goal by decreasing the robustness of the watermark, which prevents the identification of the leakage source(s). Malicious SPs can identify high probability watermark points and tamper with the watermark pattern by removal or modification. For such scenarios, we presume that malicious SPs will not do blind attacks without the prior information of watermarked indices. These types of attacks will decrease the utility of data more than the robustness of the watermarking scheme and render the data useless. In favor of SPs though, we assume that they have all the prior information, such as parental genome data, observable traits for truly reflecting a worst-case scenario. Hence, we introduce three attack models that incorporate the additional one within both based on probabilistic identification that tests the robustness of the watermarks that our proposed method generates.

Single SP attack:

In this attack, a single malicious SP is expected to use the prior information available to infer the actual states of the data and identify the watermarked indices without collaborating with other SPs. Examples of prior information include MAFs, genotype and phenotype information of parents. For each data point, malicious SP finds the posterior probability of each state given the prior information Pr(yi = y | prior information) and compares it with the expected probability of given state xi = y | y ∈ {y1, y2, y3, ..., yN}. If the difference between the posterior and expected probabilities for the given state is high, it may indicate a watermarked index. We assume that the malicious SP knows the watermark length wI. Hence, SPs select the top wI indices with the highest differences in probability as watermarked and implement an attack.

Another vulnerability that malicious SPs can exploit is the inherent correlations and their values in the data to infer the actual states of the watermarked indices. For genomic data, linkage disequilibrium (LD); non-random association of certain alleles is an example of such correlations. LD is a property of certain alleles; not their loci.

The correlation of alleles {A, B} in loci {I1, I2} will not hold if either A or B changes. The asymmetric correlation observed in LD is a valid method of representation for other sequential data types. We treat the correlations in the data as pairwise and asymmetric in the proposed system.

Collusion attack:

In addition to the knowledge obtained via a single SP attack, multiple SPs that receive the same proportion of data can vertically align their data to identify watermarked points. When SPs align their data, there will be indices with different states that can be considered as definitely watermarked. The proportion of data shared with SPs may differ, which will decrease the efficiency of alignment. However, for the construction of a strong model against worst-case scenarios, the system considers the same data is shared among all SPs. Potentially watermarked indices received from collusion attack can be used along with prior information obtained from running a single SP attack. This type of attack detects further more watermarked indices than the single SP attack.

Removal attack:

Using the knowledge from collusion attacks, malicious SPs may remove the watermarked indices or more data points from the received data. In collusion attacks, these indices are changed to other states of SNPs rather than being removed, which reduces the robustness of the detection algorithms. We design this attack to simulate a scenario of partial-sharing and its main focus is to remove the index sets that are longer than wI. Removal or change of indices longer than wI is not a preferred strategy in other attack types due to the increased reduction of utility per unit data, which should be desired less by the malicious SPs.

4 Proposed solution

When Alice wants to share her data with SPs, they employ the following protocol. SP sends a request to Alice providing the indices required from her data, denoted as I. Alice then generates a list of available indices most suitable for watermarking f, that satisfies f ⊆ I, and |f| = wI. The BP-based watermarking algorithm generates f, which we discuss in the sequel. Finally, Alice inserts a watermark into the indices of f. If the data is in binary form, it is as simple as changing 0 to 0 or 1 vice versa. Otherwise, for the given state x, a different state y from the set y1, y2, ..., ym and y ≠ x is chosen to be a part of the watermark pattern. In non-binary selection, if the given index contains correlation with other indices, the selection is determined by the probabilities and statistics of the correlated indices so that the watermark would not be vulnerable to collusion attacks. Otherwise, it is a random selection with uniform distribution.

Our method relies on the BP algorithm that uses prior information and previous shared versions of the data to identify the indices with maximized detection probabilities of malicious SPs, ensuring privacy and minimized utility loss when modified for watermarking. BP is an iterative message-passing algorithm used for the inference of unobserved random variables from the observed ones. We use this algorithm to infer the probability distributions of indices given the multi-variable prior information, attack scenarios and privacy criteria. Normally, the factorization of prior information marginal probabilities could be used for a part of the inference of state probabilities. However, probability calculation gets exponentially complex as the dimensions of the data and the variety of prior information increases. Because BP approximates the actual state probabilities in a finite number of iterations, it is much more efficient than factorized calculation. The main idea is to represent the probability distribution of variable nodes by factorization into products of local functions in factor nodes. Therefore, given new prior information (e.g. complex correlations, phenotype indicators and disease history), factor nodes can be extended or re-formulated.
We describe the general setup and the details of the proposed BP-based algorithm for genomic data. BP consists of factor nodes, variable nodes and messages between them. Connections between variable nodes and factor nodes we use are given in the factor graph (see Fig. 1).

The notations for the messages at the \(v\)th iteration are as follows:

- \(\mu^v_{i\rightarrow k}\): Message from variable node \(i\) to factor and attack-eLDP (attack-eLDP) node \(k\).
- \(\phi^v_{i\rightarrow k}\): Message from familial node \(i\) to variable node \(k\).
- \(\alpha^v_{i\rightarrow k}\): Message from phenotype node \(p\) to variable node \(k\).
- \(\delta^v_{i\rightarrow k}\): Message from correlation attack-eLDP node \(a\) to variable node \(k\).

### 4.1 Nodes and messages

#### 4.1.1 Variable nodes

Variable (decision) nodes represent unknown variables. Each variable node sends and receives messages to/from factor nodes to learn and update its beliefs. Its purpose is to infer the marginal state probabilities of all indices that can be obtained from prior information. For genomic data, this information is the publicly known statistics, such as LD correlations, familial genomic traits and phenotype features.

For each node, we have a marginal probability distribution of states \(y_1, y_2, \ldots, y_m\). Each variable node \(i\), var, represents the marginal probability distributions of the \(i\)th unknown variable in the format of \(P(x_i = y_1), P(x_i = y_2), \ldots, P(x_i = y_m)\) so that each \(P\) corresponds to the probability of one \(y\) and all sums up to 1. The probability distributions in variable nodes are calculated by multiplying the probability distributions coming from the neighboring factor nodes, such as correlation, familial and phenotype nodes. The message \(\mu^v_{i\rightarrow k}\) of variable node \(i\) to factor node \(k\) indicates that \(P(x_i = y)\) at \(v\)th iteration where \(y \in \{0, 1, 2\}\). Equation 1 provides the function for the representation of a message from variable node \(i\) to correlation factor node \(k\):

\[
\mu^v_{i\rightarrow k} P(x_i = y) = \frac{1}{Z} \prod_{s \neq i} \phi^v_{i\rightarrow s} P(x_i = y | \text{fam}_i) \times \phi^v_{i\rightarrow k} P(x_i = y | \text{phe}_i) \times \delta^v_{i\rightarrow k},
\]

where \(Z\) is a normalization constant and \(\sum \mu^v_{i\rightarrow k} P(x_i = y)\) for all \(y\) must be equal to 1.

#### 4.1.2 Factor nodes

Factor nodes represent the functions of factorized joint probability distributions of variable nodes. The messages received or sent by them might be dependent on multiple variable nodes as well as a single variable node. Factor nodes might also be independent and fixed from the start. For genomic data, the correlation between SNPs (LD) can be given as the example of the first case. In such scenario, variable node \(\text{var}\) is connected to a correlation factor node \(\text{cor}\) along with the correlated variable node \(\text{var}\). For the second case of dependency on a single variable, a message passed into the AE-node is determined by the current state of any variable node can be given as an example. For the third case of dependency, family genomic information predetermined from the start can be given as an example.

Let us assume for an SNP \(x\), genomic information obtained from the family (father and mother) of certain individual \(f\) is \(x_{f \rightarrow m} = 0\) (homozygous major) and \(x_{f \rightarrow m} = 1\) (heterozygous). Then, we can safely predict the marginal probability distribution of that individual’s SNP as \(P_{\text{cor}}(x_i | [0.5, 0.5, 0])\) using the Mendelian Law of Segregation. This probability distribution is constant and not dependent on any value that the variable node might get. Therefore, throughout the algorithm, this probability distribution is propagated unchanged for any SNP \(x_i\) and receives no message \(\mu^v_{i\rightarrow k}\) from its corresponding variable node.

**Correlation factor nodes:** We use LD to enhance the robustness of the system against correlation attacks. Hence, malicious SPs will not be able to use the SNPs, which are correlated with other SNPs with high probability, for watermark detection. For every SNP pair, correlation coefficients are calculated before the iteration and the pairs with coefficients \(\sigma_{ij} \) higher than the \(r\) threshold are marked as correlated and sensitive. Correlation coefficients may differ dependent on the states of each data point and their impact on estimating the probability distributions are typically asymmetric. For each sensitive SNP pair, there is one correlation node that keeps track of the correlations inside the data.

The intuition for calculating the message to be sent by correlation node is derived from the definition of \(r\)-squared, or the coefficient of determination, that explains how good the proportion of variance in the dependent variable predicts the proportion of variance in the independent variable (Glantz et al., 2016). Since our system uses and infers marginal probability distributions in BP, we use \(\sigma^2_{ij}\) as a metric of how well we can predict the probability distribution of one state using the probability distributions of other correlated states (Miller, 1994).

The messages from correlation node \(\text{cor}\) to \(\text{var}\) are calculated as:

\[
\overline{\mu}^v_{i\rightarrow k} P(x_i = y) = \sigma^2_{ij} \times \mu^v_{i\rightarrow k} P(x_i = t), y, t \in \{0, 1, 2\}.
\]

In these equations,

\[
\begin{align*}
\{s, t = y\} & \Rightarrow \sigma_{ij} \text{ (Equation 2)}, \\
\{s = t, y\} & \Rightarrow \sqrt{\sigma_{ij}} \text{ (Equation 3)},
\end{align*}
\]

where \(s\) is the neighbor variable node, \(\sigma_{ij}\) denotes the correlation coefficient, and \(\sigma^2_{ij}\) denotes the coefficient of determination. For further insight, please see the example in Supplementary Appendix Section 2.2.1.

**Familial factor nodes:** Familial factor node, \(\text{fam}\), calculates message \(\phi^v_{i\rightarrow k} P(x_i = y | \text{fam}_i, m)\), \(y \in \{0, 1, 2\}\) using the Mendelian Inheritance Law of Segregation and sends it to variable node \(k\). Please see Supplementary Appendix Section 1, Table 1 for Mendelian inheritance probabilities using the Law of Segregation. In the message, \(f_i\) and \(m\) corresponds to the \(i\)th SNP values of father and mother, respectively.

For example, if the father has SNP \(f_i = 1\) and the mother has SNP \(m_i = 2\) for the \(i\)th SNP, the message from familial node \(\text{fam}\) is as follows:

\[
\phi^v_{i\rightarrow k} P(x_i = y | f_i = 1, m_i = 2) = [0, 0.5, 0.5], y \in \{0, 1, 2\}.
\]

**Phenotype factor nodes:** Phenotype factor node, \(\text{phe}\), is designed for representing probabilistic observable traits (e.g., certain hereditary diseases, tongue-curling, or hair color) that can be used for the inference of SNP states. In our experiments, we simulated this node with Mendelian phenotype traits only mainly due to the challenges of finding an exact causative gene and determining its exact
predication value in non-Mendelian inheritance (van Heyningen and Yeyati, 2004). However, other observable traits are also viable with re-formulation. pdf computes message $o_{pdf}^i P(x_i = y/p_i) = \{0, 1, 2\}, p_i \in \{\text{dominant}, \text{recessive}\}$ using the Mendelian Inheritance Law of Dominance and sends it to variable node $k$. Please see Supplementary Appendix Section 1, Table 2, for Mendelian inheritance probabilities using the Law of Dominance. In the message, $p_i$ corresponds to the dominance trait of the observed phenotype in $i^{th}$ SNP. $p_i$ can be either dominant or recessive. For example, if the data owner is known to have blue eyes (recessive gene), which we encode in the $i^{th}$ SNP, the message from phenotype node pdf is as follows:

$$o_{pdf}^i P(x_i = y/p_i = \text{recessive}) = \{0, 0.1, 1\}, y \in \{0, 1, 2\}.$$ 

**Attack-eLDP nodes:** This bold definition is not used like a title, but as a definition similar to Correlation, Familial and Phenotype Factor Nodes. The attack and e-LDP (attack-eLDP) node is designed to simulate the inference power of the attackers on data and calculates the inverse probabilities that will keep the attacker uncertainty at maximum against single SP and collusion attacks while keeping the LDP criteria intact by updating the watermarked state options which violate e-LDP. This node receives a message from the variable node. Although acting as another factor node, it does not send the message to the variable node. Instead, the attack-eLDP node sends its message along with a variable node message to the watermarking algorithm as parameters.

Inside the attack-eLDP node, the attack part re-calculates the watermarking probabilities of all indices based on the variable node probability distributions and previously shared versions of states to simulate single SP and collusion attacks. In every $S^i$ watermarking, a set of previous sharings $S_i^{k-1}$ for each index $i$ or set of indices $i$ are used as a prior condition. Then the probabilities of the potential next states are calculated using binomial distribution given $S$. Finally, updated probability distributions are sent to the watermarking algorithm as the watermarking probability of each state. The calculation procedure followed by the node’s attack part is described in the sequel:

$$x = |(x_i = y)| \in S^y; \quad x \text{ is the number of states equal to } y \text{ in set } S^y.$$ 

$$\text{Binomial}(S^y|x^i = y) = \binom{k}{x} \times P(x_i = y)^x \times P(x_i = y)^{k-x}.$$ 

$P(x_i = y)$ and $P(x_i = y')$ are calculated from the variable node’s message.

$$a_0(x^k = 0|S^k|x^i = 0) \approx P(S^k|x^i = 0) = \text{Binomial}(S^k|x^k = 0),$$

$$a_1(x^k = 1|S^k|x^i = 1) \approx P(S^k|x^i = 1) = \text{Binomial}(S^k|x^k = 1),$$

$$a_2(x^k = 2|S^k|x^i = 2) \approx P(S^k|x^i = 2) = \text{Binomial}(S^k|x^k = 2).$$

$A^k$ is Normalized($[a_0, a_1, a_2]$), where $A$ is the updated marginal watermarking probability distribution of $i^{th}$ index for the $S^i$.

The eLDP part checks whether any states violate the LDP (Kairouz et al., 2014) condition for Alice who wants to have a plausible deniability factor for the versions of data she shares. The condition is satisfied if no state violates Equation 3 in Supplementary Appendix Section 1.2. Probability distributions of the violating states converge by continuous averaging with variable nodes until non-violation. This incorporation creates watermarks for all the SNPs of the data owner and acts as a lower bound of privacy ensured along with lower and upper bounds on confidence degree by its very definition. For further insight, please see the example in Supplementary Appendix Section 2.2.2.

### 4.2 Watermarking algorithm

SNP state inferences are assumed to be conducted by malicious SPs as well, given their prior information on the data for SP and Correlation Attacks (cf. Section 3.3). We consider attacker inference strength and privacy criteria at the same time during watermarking in which modifying the actual state of the data is mandatory. Modifying more indices than necessary results in losing utility on data. These modifications increase the detection probabilities of changed indices by malicious SPs and decrease efficiency. These modifications must be interpreted as actual data, not to give further means to malicious SPs for detecting watermarked indices. For example, watermarking a SNP, with MAE = 0 is meaningless. Because no state of the SNP, other than homozygous major is observed, any change will be artificial and interpreted as watermarked.

Our watermarking scheme uses a probabilistic watermarking pattern rather than a deterministic one. To this end, we use a different set of indices and states to be watermarked for each SP. If we use fixed watermarked indices, it presents a risk of compromising watermark robustness against modifications and removals in single SP attacks and collusion attacks. If we use fixed watermarked states for each index, the data do not reflect the population distribution, and using the probabilistic inference, attackers can identify the indices that show discrepancies with the population.

Given these criteria, we calculate a watermark score that helps us to list indices better to watermark in descending order. This score is calculated by comparing the attack-eLDP marginal probability distributions with the original states of data. Firstly, the probability of the actual state in attack-eLDP distribution is subtracted from one. This gives us the probability of that index being watermarked. Then these indices are sorted in descending order to give priority to indices most likely to be watermarked. For further insights, see the watermarking algorithm given in Supplementary Appendix Section 3.

### 5 Evaluation

We evaluated the proposed scheme in various aspects like security against detection (robustness), the length of the watermark, utility loss and privacy guarantees. These aspects and their correspondence to the dependent variables are also provided. We give the details of the data model, the experimental setup and the results of the experiments in the sequel.

#### 5.1 Data model and experimental setup

For the evaluation, we used the SNP data of 1000 Genomes Project (IGSR, 2013). The data set contains the 7690 SNP-long data of 99 individuals in the form of 0s, 1s and 2s, which is represented as a 99 × 7690 matrix. This data set is used for learning the linkage-disequilibrium and MAF statistics along with parental data generation based on the method proposed in Deznabi et al. (2018). While the dataset and the parental data generated from it are processed, the HW equilibrium is assumed. These statistics are then employed in the BP algorithm for probabilistic state inference. The threshold of pairwise correlations used for the results is specified as $\rho = 0.9$, since the changes in results down to $\rho > 0.5$ are insignificant. The results for other $\rho$ values can be examined in Supplementary Appendix Section 4.6. Throughout the experiments, the length of data $d_1$ is fixed to 1000, the number of SPs $(b)$ is fixed to 20, and $w_1$ values vary between 10 and 100. In exceptional cases, watermarks with $w_1 > 100$ are also tested, too.

#### 5.2 Evaluation metrics

We evaluate the proposed scheme via precision, which corresponds to the percentage of attackers correctly identified as malicious, utility lost using entropy and kinship coefficients, and e-LDP achieved for various attack types and parameter configurations. In collusion attacks, two SP collusion scenario contains all 190 pairs of 20 SPs since, $b$ is fixed to 20 and $C(20, 2) = 190$. This number increases...
rapidly as the number of collusion SPs increases. To keep the computational cost low, we took the number of malicious SP scenarios as 190 unique random sets for each case. Besides, we kept the number of malicious SPs up to \( k = 10 \) since we assume that we know the parameter \( k \) and \( k > 10 \) increases our detection results back. We find the malicious set of SPs by checking the watermark patterns. For the details of the detection algorithms, please see Section 4.1 in Supplementary Appendix.

5.3 Results of attacks

We evaluate the proposed scheme for the attack model described in Section 3.3. The robustness of the watermark is evaluated against collusion and removal attacks, in which the knowledge of single SP and correlation attacks are incorporated to reflect the worst-case scenario. For the details and results of single SP attacks that have the highest precision results, please see Supplementary Appendix Section 4.2. In these experiments, we assume worst-case scenarios to create lower bounds. The assumptions that give maximum malicious SP information are as follows.

- Malicious SPs know the exact value of watermark length \( (w_l) \).
- For every SP, \( I_k \) is identical \( k \in \{1, 2, \ldots, b\} \). It means all SPs have the same set of indices of data.
- Malicious SPs have all the population information e.g., correlations, MAFs, frequency of states.
- Malicious SPs know the SNPs of the data owner’s father and mother.
- Malicious SPs know all the observable (phenotypical) features of the data owner and correspondent SNP states.

5.3.1 Collusion attack

In collusion attacks, multiple SPs collude and bring their data together to detect and modify the watermarked indices. Firstly, the states different for the same SNP are identified as watermarked because the watermark pattern is unique for each SP. The maximum number of indices that can be identified as watermarked by malicious SPs is \( w_l \times k \), where \( k \) is the number of malicious SPs. Secondly, when the malicious SPs find fewer points in collusion attack than \( w_l \times k \), they target additional indices as watermarked using the prior information on the data, e.g., MAFs and LD correlations, similar to the single SP attack. These indices are usually the least likely states when prior information is considered. At the end of the collusion attacks, malicious SPs change the states of data in two ways. The states that are not the same across all malicious SPs are modified to the most frequent ones. Then, the states that are the same across all malicious SPs but having the least likelihoods are modified to the most likely states possible. Our initial detection method, modification setups and assumptions on the collusion attack are similar to those of a single SP attack. Since collusion attacks contain much more information about the probability of a state than the single SP attack, we expect the precision results of collusion attacks to be lower than single SP attacks. We expect a decrease in precision with the increasing number of malicious SPs.

Figure 2 shows the impact of watermark length on precision for \( \epsilon = 0 \). Supplementary Appendix Section 4.3 shows the impact of detection methods and \( \epsilon \) on precision. Similar to single SP attacks, \( \epsilon \) has no significant impact on precision. Among 20 SPs, \( k = 10 \) gives the worst results. Precision decreases as the number of malicious SPs increases, but after \( w_l \geq 50 \), almost all \( k \) malicious SP scenarios are detected with precision rates higher than 80%. With \( w_l = 100 \), precision increases up to 90% even for the worst case of 10 malicious SPs. We conducted experiments for \( w_l > 100 \) and in some cases achieved precision up to 98% for \( k = 10 \).

5.3.2 Removal attack

In the removal attack, index sets longer than \( w_l \) are targeted and the multiplier factor of \( w_l \) for removal is denoted by \( \epsilon \). For instance, results for \( \epsilon = 1.6 \) on \( w_l = 100 \) means that \( 1.6 \times 100 = 160 \) indices are removed. Detection method, modification and identification of the indices to be removed are same as the collusion attack.

Figure 3 shows the impact of watermark length and the multiplier factor \( \epsilon \) for 10 colluding malicious SPs. Apart from a few inconsistencies, the precision decreases as \( \epsilon \) increases, but for \( \epsilon < 1.4 \), the results are almost identical. Figure 3 also demonstrates that our proposed system is robust against removal attacks with 85% precision when malicious SPs remove up to 120% more data than \( w_l \). For \( \epsilon > 2.4 \), the entire watermark is distorted and the precision drops to the level of randomness. It is important to note that these removals are performed in the conditions most optimal to the attackers with the combined knowledge of colluding SPs and all the prior information available used in the watermarking algorithm. Therefore, results reflect the worst-case scenario. Supplementary Appendix Section 4.4 shows the results of the removal attacks of 2 and 6 malicious SPs.

5.4 Utility evaluation

Besides the number of indices changed, we evaluate the utility using two methods. The first method calculates the utility loss using entropy. We calculate the total entropy of SNPs using their MAF values and HW equilibrium. The watermark-embedded SNPs lose their utility. In this method, percentage-wise changes in the total entropy between embedded and non-embedded versions are reflected as the utility loss. The second method calculates the kinship coefficients for the embedded and non-embedded versions using the equations from (Kale et al., 2017) and tables of interpretation from (Manichaikul et al., 2010). The details of the second method can be found in Supplementary Appendix Section 4.5.
5.4.1 Entropy-based utility

In this method, for $w_l = 100$ which corresponds to 10% of $d_o$, utility loss is linear with respect to $w_l$, and it is around 12%, as can be seen in Figure 4. This value may seem high, but robustness against collusion attacks dictates that SNPs with high variance (entropy) should be favored for robust watermarking with low $w_l$. For the malicious SP-generated data used for calculating precision against attacks, utility loss is even higher. As the number of malicious SPs increases, the uncertainty of SPs about the states of SNPs decreases. Hence, they produce data with similar utility loss to the watermarked versions.

6 Conclusion

We propose a novel watermarking scheme for sequential genome data employing BP with ensured e-LDP. This system is designed for use between the data owner and SPs some of whom are assumed to be malicious. We implemented the algorithm against the worst-case scenario of malicious SPs. We assume that SPs know almost all the statistics of the data. Therefore, we tested the robustness of watermarks against single SP attacks, collusion attacks and removal attacks. The BP algorithm greatly mitigates the risk of unauthorized sharing. The algorithm secure high precision rates even against the worst-case scenarios when all potential prior information that malicious SPs can use are considered. We keep the changes on data minimum for preserving the utility of data by using a short watermark length $w_l$. Our experiments show that when $w_l$ is kept higher than 50, even for a high number of malicious SPs, robustness is preserved more than 80% and utility is preserved up to 95%. We observe that $e$ does not significantly affect precision. Privacy is preserved without disturbing precision, which addresses a potential liability issue from data being known and offers a privacy measure of plausible deniability needed, especially for rare SNPs.

Acknowledgements

Research reported in this publication was supported by the National Library of Medicine of the National Institutes of Health under Award Number 1R01LM013429.

Conflict of Interest: none declared.

References


dl_22 pages.


Appendix: Privacy-Preserving and Robust Watermarking on Sequential Genome Data using Belief Propagation and Local Differential Privacy

Abdullah Çağlar Öksüz¹, Erman Ayday¹,²∗ and Uğur Güdükbay¹∗

¹Department of Computer Engineering, Bilkent University, Ankara, Turkey
²Department of Electrical Engineering and Computer Science, Case Western Reserve University, Cleveland, OH, USA

To whom correspondence should be addressed.
Contact: exa208@case.edu or gudukbay@cs.bilkent.edu.tr

Table 1. Mendelian inheritance probabilities using the Law of Segregation.

<table>
<thead>
<tr>
<th>SNP</th>
<th>Mother SNP</th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>[0, 0.0]</td>
<td>[0.5, 0.5, 0]</td>
<td>[0.1, 0]</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>[0.5, 0.5, 0]</td>
<td>[0.25, 0.5, 0.25]</td>
<td>[0.05, 0.5]</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>[0, 1.0]</td>
<td>[0.0, 0.5]</td>
<td>[0, 0, 1]</td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Mendelian inheritance probabilities using Law of Dominance.

<table>
<thead>
<tr>
<th>SNP</th>
<th>Observed phenotype trait</th>
<th>SNP distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA</td>
<td>0.5, 0.5, 0</td>
<td>[0, 0, 1]</td>
</tr>
<tr>
<td>Aa</td>
<td>0.5, 0.5, 0</td>
<td>[0, 0, 1]</td>
</tr>
<tr>
<td>aa</td>
<td>0.5, 0.5, 0</td>
<td>[0, 0, 1]</td>
</tr>
</tbody>
</table>

1.2 Local Differential Privacy

Differential Privacy is a system of public data sharing that uses the patterns of groups in the dataset and while doing so without compromising the privacy of individuals in the dataset (Dwork, 2006). The main intuition is that an algorithm is differentially private if the use of any particular individual’s data cannot be inferred from the computations. If any inference probability is limited to the upper bound of ρ < ϵ in the dataset, the algorithm is ϵ-differentially private. ϵ-differential privacy is derived for a process A if Equation 1 is satisfied in any two neighboring databases D₁ and D₂ with an outcome O₁.

\[ P(A(D₁) = O₁) ≤ e^\epsilon × P(A(D₂) = O₁). \] (1)

Equation 1 is symmetrical and valid for any two neighboring databases D₁ and D₂. So, this equation can also be written as:

\[ e^{-\epsilon}×P(A(D₂) = O₁) ≤ P(A(D₁) = O₁) ≤ e^\epsilon × P(A(D₂) = O₁). \] (2)

Local Differential Privacy (LDP) is the localized version of differential privacy that targets not the datasets or databases but the data indices. In LDP, the data is intentionally perplexed by the data owners so that plausible deniability is ensured without a “trusted party.” The privacy assured by data owners is expressed as ϵ-local differential privacy. This ϵ value can be thought to provide \( e^{\epsilon} \) % plausible deniability. As ϵ gets smaller, although the outcomes become less likely to be different from one another, a high level of privacy is ensured. In our use case, LDP is the local implementation and satisfying Equation 2 on every single data point or sequential data. We use LDP as an additional privacy preservation measure. It is a technology adopted by major technology firms like Google, Apple, and Microsoft for collecting mass anonymized data like web browsing behaviors, typing behaviors, and telemetry data (Comodi et al., 2018).

A watermarking algorithm with ϵ-local differentially privacy must normally satisfy Equation 2, for all sharings of all SNPs. This condition is used for limiting the amount of information gained by the exclusion of each shared data from the total set of sharings.

In Equation 2, the uncertainty increases with increasing ϵ at the expense of privacy. As ϵ decreases, more privacy is ensured. However, Equation 2 does not cover a localized setup but counts on databases or datasets as a...
whole. Therefore, we use a variant of differential privacy adapted from the geo-indistinguishability study of Andrés et al. (2012) that is both localized in SNP level and well suited to our sequential data. In e-LDP part, our framework updates the probabilities of watermarking options that violate the local differential privacy condition of Andres et al. and his modified privacy criteria is given as

\[
\forall r > 0, \forall x, x' : d(x, x') \leq r, e^{-\epsilon x r} \times \frac{P(x)}{P(x')} \leq \frac{P(x|S)}{P(x'|S)} \leq e^{\epsilon x r} \times \frac{P(x)}{P(x')},
\]

where \( S \) represents the set of previous sharings of data and \( r \) represents the distance between the states. In location data, \( r \) is calculated as the maximum Euclidean distance between states. Since our data is sequential genomic data and the states have no priority over one another, we used Hamming distance, which is equal to one all the time. It is important to note that the first part of Equation 3 refers to the newly updated probabilities obtained from modifications such as adding and removing noise and it corresponds to the ratio between \( \epsilon \). The second part refers to the unchanging probabilities of states given the prior information and it corresponds to the ratio between \( \epsilon \). We can compare the results for each \( x_1 = y, y \in \{0, 1, 2\} \), and decide whether the states should be updated or not for preventing violations against privacy condition.

### 2 Belief Propagation Algorithm

**Belief Propagation (BP),** also known as sum-product message passing, is a message-passing algorithm used for the inference of networks and graphs like Bayesian Networks and Markov Networks (Braunstein et al., 2005). BP calculates marginal probability distributions of unknown variables in factor graphs in an iterative manner using the information from previous states. In a factor graph, two types of nodes are used: (i) Factor Nodes, and (ii) Variable Nodes. BP is a widely used technique in graphs because the marginal probability computation of variables that have a dependency on multivariate data (factors) gen exponentially complex as the number of factors increases. Besides, BP provides the flexibility of including any desired factor, provided that the marginal probability conversion equations are available. Moreover, marginal probabilities of factors must be re-computed given the new distribution of variables. With a finite number of iterations, BP approximates the actual distribution with a low computational cost. The steps of the proposed BP-based algorithm are as follows:

- The algorithm starts in a variable node with an initial probability distribution.
- The algorithm collects messages from the factor nodes for updating the probability distributions of the targeted unknown variable nodes. In loopy bilateral graphs, this process is handled in iterations until convergence. However, this approach is changed to a top-to-bottom approach with one or two iterations for treelike graphs like ours for efficient approximation.
- Variable nodes generate the factor node messages by multiplying all incoming messages from the neighbors except the receiver neighbor.
- Factor nodes generate the messages by using local functions and send them to corresponding variable nodes.
- At the end of each iteration, the marginal probability distribution of each variable node is updated by multiplying all incoming messages from neighbors.
- The algorithm approximately calculates the beliefs of the variable nodes and passes it to the “Attack- \( r \)-local differential privacy” (Attack-eLDP) node.

- The **Attack-eLDP** node acts as a secondary factor node and calculates a new message that considers both attack scenarios and local differential privacy criteria.
- Finally, the **Attack-eLDP** node passes its message together with variable node messages as parameters into the watermarking algorithm.

#### 2.1 Data Model Notations

Table 3 shows the frequently used symbols and notations in the data model.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1, \ldots, x_d )</td>
<td>Set of data points</td>
</tr>
<tr>
<td>( w_i )</td>
<td>Possible values (states) of a data point</td>
</tr>
<tr>
<td>( d_i )</td>
<td>Length of the data</td>
</tr>
<tr>
<td>( w_c )</td>
<td>Length of the watermark</td>
</tr>
<tr>
<td>( h )</td>
<td>Total number of SPs</td>
</tr>
<tr>
<td>( I_k )</td>
<td>Index set of data points that are shared with ( k^{th} ) SP</td>
</tr>
<tr>
<td>( J_k )</td>
<td>Index set of data points that are watermarked for the ( k^{th} ) SP</td>
</tr>
<tr>
<td>( Z_k )</td>
<td>Watermark pattern of ( k^{th} ) SP</td>
</tr>
<tr>
<td>( W_k )</td>
<td>Watermarked data shared with ( k^{th} ) SP</td>
</tr>
<tr>
<td>( r )</td>
<td>Local differential privacy coefficient</td>
</tr>
<tr>
<td>( \rho )</td>
<td>Correlation coefficient threshold</td>
</tr>
<tr>
<td>( S_k )</td>
<td>Set of states for index ( k ) shared with the first ( k ) SPs</td>
</tr>
</tbody>
</table>

#### 2.2 Nodes in Belief Propagation Algorithm

We provide examples of correlation nodes and attack-eLDP nodes.

#### 2.2.1 Correlation Node Example

Figure 1 shows how correlation nodes are connected with variable nodes and how they send messages to one another.

![Diagram of correlation nodes and variable nodes](attachment:image.png)

Fig. 1. The relationship between variable nodes and correlation nodes. Both nodes may receive and send messages. For simplicity, one message for each type is shown.

For example, SNP1 is connected with SNPs via \( c_{1, 2} = i \) for \( x_1 = 0 \), \( x_2 = 2 \) with \( c_{1, 2} = 0.9 \) and \( \lambda_{1, 2}^{\text{SNP}}(x_1 = y) = [0.3, 0.6, 0.1] \), \( y \in \{0, 1, 2\} \) at the \( v^{th} \) iteration. Then, we may calculate the message from \( i \) to \( 2^{nd} \) variable node, \( \lambda_{1, 2}^{\text{Var}}(x_2 = y) \) as:

\[
\lambda_{1, 2}^{\text{Var}}(x_2 = y) = \left[ \begin{array}{c} 1 - (0.9^3 \times 0.3) \\ 0.9^3 \times 0.3 \\ 0 \end{array} \right] \
+ [0, 0, (0.9^2) \times (0.3)] \
\lambda_{1, 2}^{\text{SNP}}(x_2 = y) = [0.2523, 0.2523, 0.4954].
\]
Appendix: Privacy-Preserving and Robust Watermarking on Sequential Genome Data

2.2.2 Attack-eLDP Node Example

For example, assume for the index \( i \), \( \gamma_{i,1} = \mu_{i-1} \times P(x_i = 0) = \left[ 0.6, 0.4, 0 \right] \) sends the following message to the attack-eLDP node \( \sigma_{A} \), and \( S^{-1} = [0.0, 0, 1, 0, 1, 0] \) where \( k = 7 \). We may calculate the watermarking probabilities of index \( i \) for SP\(_7\) as follows:

For \( x_i^2 = 0, \alpha = 5 \to \)

\[
\text{Binomial}(S_i^2|x_i^2 = 0) = \left( \frac{7}{1} \right) \times (0.6)^5 \times (0.4)^{-5} = 0.261,
\]

For \( x_i^1 = 1, \alpha = 3 \to \)

\[
\text{Binomial}(S_i^1|x_i^1 = 1) = \left( \frac{7}{1} \right) \times (0.4)^7 \times (0.6)^{ -7} = 0.290,
\]

For \( x_i^2 = 2, \alpha = 1 \to \)

\[
\text{Binomial}(S_i^2|x_i^2 = 2) = \left( \frac{7}{1} \right) \times (0.0)^7 \times (1)^{-7} = 0.
\]

\[
A_i^2 = \text{Normalized}(0.261, 0.290, 0) \approx [0.474, 0.526, 0] \text{ is the updated watermarking probability distribution of index } i \text{ for } SP_i.
\]

Continuing from the above example, we can calculate the privacy conditions as follows:

For \( x_i^2 = 0, \quad P(x_i|s) = \frac{0.474}{0.526} = 0.901, \quad \text{and } P(x_i^1|x_i^2 = 0) = 0.6, \quad P(x_i^1|x_i^2 = 1) = 0.4. \}

This means \( e^{-1} \times 1500 \approx 0.901 \leq e^i \times 1.500 \) must be satisfied for not violating the \( i \)-local differential privacy. Since \( 0.901 \leq 1 \) and \( \ln(0.901) = -0.509 \), if \( e < 0.509 \) the state probability of \( P(x_i^2 = 0) \) must be updated for not violating the privacy by averaging the attack node probabilities with variable node probabilities. After \( P(x_i) \) is set, the distribution is continuously normalized to converge into the probability that satisfies the condition.

For \( x_i^2 = 1, \quad P(x_i|s) = \frac{0.526}{0.474} = 1.101, \quad \text{and } P(x_i^1|x_i^2 = 1) = 0.4, \quad P(x_i^1|x_i^2 = 0) = 0.6. \}

This means \( e^{-1} \times 0.667 \leq 1.101 \leq e^i \times 0.667 \) must be satisfied for not violating the \( i \)-local differential privacy. Since \( 1.101 \leq 1 \) and \( \ln(1.101) = 0.253 \), if \( e < 0.253 \) the state probability of \( P(x_i^1 = 1) \) must be updated likewise.

For \( x_i^2 = 2, \quad P(x_i|s) = \frac{0}{0.474} = 0, \quad \text{and } P(x_i^2|x_i^2 = 1) = 0, \quad P(x_i^2|x_i^2 = 0) = 1. \}

This means \( e^{-1} \times 0 \leq 0 \leq e^i \times 0 \) it never violates the local differential privacy for any \( i \) just like the case of \( x_i^1 = 0 \). However, we know that \( P(x_i|s) = 0 \) and it is an impossible watermarking case regardless of the violation.

If Alice set her privacy criteria to \( e < 0.509 \), \( \sigma_{A} \)'s final marginal probability distribution will be equal to the distribution enforced by the eLDP part. Otherwise, the distribution remains the same as attack part determined, which is equal to \([0.474, 0.526, 0]\).

3 Watermarking Algorithm

Algorithm 1 gives the pseudo-code of the watermarking algorithm. The input of the algorithm includes data, atks, wScore and \( w_i \). The only output is newdata. In the algorithm, data represents the original data of the data owner with length \( d \). atks represents the probability distributions of SNPs generated by the belief propagation algorithm considering all the prior information, sharings with SPs, and eLDP requirements as a 3-by-\( d \) array. wScore represents the watermarking probability of SNPs by subtracting the atks probability of the actual state from 1. wScore values are organized in descending order to give priority on SNPs with the highest watermarking probability. \( w_i \) represents the length of the watermark. Finally, the output. newdata, represents an altered version of the original data with a watermark pattern of length \( w_i \).

In the algorithm, the while loop embeds a watermark to the indices with the highest wScore, denoted as wScore([2]) and \( j \) goes from 1 to \( d \) until the number of embedded indices reach to \( w_i \). Basically, wScore([2]) is the index in data that corresponds to the \( j \)-th index that is stored in wScore’s second column. It is also important to note that the rows of wScore matrix are sorted in descending order by the scores stored in the first column which are collectively denoted as wScore([1]). We determine the embedding via random number generation, denoted as \( r \) and between 0 and 1, matching the probability distribution space. Assume that \( r = 0.42 \) and the distribution of \( \text{atk} = [0.32, 0.03, 0.55] \). Inside the while loop, the first two conditions are not satisfied because \( r < 0.32 \) and \( r > 0.32 + 0.03 \). For the final condition, if the given index is not already equal to 2 and it is not previously embedded a watermark, the new version of the data and wScore are changed into 2, and the watermark index \( k \) is incremented by one. A special condition not included in the pseudo-code algorithm also checks whether a certain index is watermarked in the same way throughout the previous SPs. If an index is embedded with a watermark that is the same across all SPs, it is useless and just a utility decreasing factor.

Algorithm 1 Watermarking Algorithm

Watermark(data, atks, wScore, w_i)

1: \( j \leftarrow 1 \)
2: \( k \leftarrow 0 \)
3: newdata \leftarrow data
4: while \( k < w_i \) do
5: \( i \leftarrow wScore([2]) \) (index of the SNP)
6: temp \leftarrow data([i]) \) (actual state of SNP)
7: if \( temp \neq 0 \) and wScore([i]) = \( \text{atk} \) then
8: wScore([i]) \leftarrow 2
9: end if
10: else if \( wScore([i]) < r \) then
11: newdata([i]) \leftarrow 0
12: end if
13: end while
14: end if
15: end if
16: if \( wScore([i]) = 0 \) then
17: newdata([i]) \leftarrow 0
18: end if
19: end if
20: end if
21: end if
22: if \( wScore([i]) < r \) then
23: newdata([i]) \leftarrow 2
24: end if
25: end if
26: end if
27: end if
28: if \( j = \text{length(data)} \) then
29: \( j \leftarrow 1 \)
30: else
31: \( j \leftarrow j + 1 \)
32: end if
33: end while
34: return newdata

4 Evaluation

4.1 Detection Methods

For detection, we compare the attacked data produced by malicious SPs with each of our previously shared data and watermark patterns. Assuming,
we know the number of malicious SPs, we use two detection methods Hamming Distance (H) and custom spPenalizer (E) and their relaxed versions Hamming Distance Relaxed (HR) and spPenalizer Relaxed (ER). H method as the name suggests, calculates the Hamming distances between attacked data and every shared SP data by checking every index. After the distances are calculated, all SPs are sorted from least different than the attack data (least distance) to the most different (most distance) and top \( k \) of malicious SPs are flagged as malicious. E method, firstly identifies the indices that has not been the same with the original data at least once (conflicting indices) for all SPs. Then, the variance of each conflicting index is calculated as a penalizing factor. Depending on the match of indices between shared SP data and attack data, penalizing factors are added into the matching SPs. Finally, top \( k \) SPs who are penalized the most are flagged as malicious SPs. In the relaxed versions of both methods, top \( k+1 \) are used in flagging process.

4.2 Single SP Attack

In single SP attacks, a single SP uses all the knowledge available to itself for inferring the marginal state probabilities of SNPs. This process is similar to the calculations in the belief propagation part of our watermarking scheme. Later on, malicious SP identifies the top \( w \) SNPs with least probabilities \( P(x_i = y, y \in \{0, 1, 2\}) \) as watermarked, and modifies them to their most likely states for the prior information available to itself.

Figure 2 shows the impact of watermark length on precision for various values of LDP coefficients \( \epsilon \). Appendix § 4.2 provides the impact of detection methods and \( \epsilon \) values on the precision for a single SP attack. In all cases, \( w \geq 40 \) seems to be the breaking point where the precision reaches almost 100%. In our data, we believe that this corresponds to the minimum amount of change needed for distinguishing a version of shared data from another effectively. For example, this \( w \) may not suffice for comparison among 50 SPs. We observe that \( \epsilon \) has no significant impact on precision.

Figure 3 shows a comparison of the detection methods for a single SP attack. We observe that the relaxed versions HR and ER outperform the methods H and E. However, the methods E and ER outperform the methods H and HR.

4.3 Collusion Attack

Figure 4 shows a comparison of the detection methods for a collusion attack. Similar to single SP attack, we observe that the relaxed versions HR and ER outperform H and E methods. Differing from the single SP attack though, the results of H and E and HR and ER are almost identical.

Figure 5 shows that the relaxation of the privacy criteria \( \epsilon \) to 0.5 or 1 does not impact the precision results at all.

Figure 6 depicts the impact of privacy criteria \( \epsilon \) on precision. The case for \( k = 2 \) provides limited information because all \( w \) values, except \( w = 10 \), give 100% precision. For both \( k = 6 \) and \( k = 10 \) though, the precision remains stable with naturally lower precision results for \( k = 10 \). Sometimes, arbitrary fluctuations occur throughout different watermark lengths, but overall, the observed impact of \( \epsilon \) is negligible.

4.4 Removal Attack

Figure 7 shows the impact of watermark length and multiplier factor of indices removed on precision for 2 and 6 colluding malicious SPs. For \( \epsilon > 2.4 \), almost all of the watermarked indices are removed from all malicious SP sets - a trend that is observed in the following figures, too - and they cannot be identified. For \( \epsilon < 2.4 \), \( \epsilon = 1.6 \) seems like a safe-spot where precision results do not deviate. Due to the increasing number of collaborators, overall precision dropped for all watermark lengths from 2 SPs to 6 SPs, similar to the observations in collusion attacks.

4.5 Utility

4.5.1 Kinship Coefficient-based Utility

In this method, \( \phi = \frac{1}{\sqrt{\epsilon + \epsilon}} \), where \( \epsilon \) is the degree of affinity and \( \epsilon = 0 \), 1, and 2 correspond to the monozygotic twin, parent-offspring (or full sibling), and second degree, respectively. We compare the kinship relationship between the original data and embedded-data. As we can see from Figure 8, even for high values of \( w \) (such as 100), even though the kinship coefficient changes (compared to its original value), interpretation of the kinship relationships remain intact. As in the utility loss method, data generated by malicious SPs are more divergent than the original version, which can be observed from \( \phi \) results. It should be noted that \( \phi \) is not a variable with a linear change like utility loss. Therefore, the generation of data with less utility than the utility of the watermarked versions is a penalizing factor for malicious SPs.

4.6 Correlation Threshold (\( \rho \))

Figure 9 shows that \( \rho \) has a minor impact on precision for varying watermark lengths. Keeping \( \rho > 0.5 \) in experiments is to keep correlation coefficients, their corresponding coefficients of determination and by doing so the inference strength of the correlations relatively high.
Appendix: Privacy-Preserving and Robust Watermarking on Sequential Genome Data

Fig. 4: The impact of watermark length and detection method on precision for a collusion attack ($\epsilon = 1$).

Fig. 5: The impact of watermark length on precision for collusion attacks with different number of malicious SPs and $\epsilon$ values.

Fig. 6: The impact of $\epsilon$ on precision for collusion attacks with different number of malicious SPs and $\epsilon$ values.
Fig. 7: The impact of watermark length on precision for a removal attack with different number of malicious SPs and $i$ values.

Fig. 8: The impact of watermark length on kinship coefficient.

Fig. 9: The impact of watermark length on precision for different $\rho$ values.

References


Öksüz et al.