

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

Abdullah Çağlar Öksüz¹, Erman Ayday^{1,2*} and Uğur Güdükbay^{1*}

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

1 Related Background

1.1 Genomics

1.1.1 Minor Allele Frequencies

The frequency of minor alleles prompts an immense amount of selection in heritability, and therefore, recorded as publicly available data for medical institutions (Hernandez *et al.*, 2019). Using these publicly available Minor Allele Frequency (MAF) values, the probability of each genetic state in populations can be inferred using the following equations assuming HW equilibrium:

$$\begin{aligned} AA = 0 : P(\text{Homozygous Major}) &= (1 - \text{MAF})^2 \\ Aa = 1 : P(\text{Heterozygous}) &= 2 \times (\text{MAF}) \times (1 - \text{MAF}) \\ aa = 2 : P(\text{Homozygous Minor}) &= (\text{MAF})^2 \end{aligned}$$

1.1.2 Linkage Disequilibrium

Linkage Disequilibrium (LD) is the non-random heritable associations of alleles in different loci (Slatkin, 2008). Because it is an indication of population genetic forces on the genome formation, it is a widely investigated and exploited research topic in evolution and demographics studies (Stephens, 2001). Factors that have an impact on LD may vary due to genetic reshuffling, mutation rate, allelic drift, and so on. In genomic privacy, LD can be used to infer the state probabilities, and hence, the values of multiple Single Nucleotide Polymorphisms (SNPs) in correlated loci given the state value of a single SNP. Therefore, highly correlated states can be used for the enhancement of beliefs in belief propagation setup on the other SNPs. By exploring all the coexisting pairs of SNPs in the large sample population, linkage disequilibrium loci of high correlation values can be identified.

1.1.3 Mendelian Inheritance Probabilities

Tables 1 and 2 present the Mendelian inheritance probabilities using the Law of Segregation and the Law of Dominance, respectively.

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

Mother SNP \ Father SNP	0	1	2
	0	[1, 0, 0]	[0.5, 0.5, 0]
1	[0.5, 0.5, 0]	[0.25, 0.5, 0.25]	[0, 0.5, 0.5]
2	[0, 1, 0]	[0, 0.5, 0.5]	[0, 0, 1]

Table 2. Mendelian inheritance probabilities using Law of Dominance.

Observed phenotype trait	SNP distribution
Dominant (AA or Aa)	[0.5, 0.5, 0]
Recessive (aa)	[0, 0, 1]

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 $\rho < \epsilon$ 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 D_1 and D_2 with an outcome O .

$$P[A(D_1) = O] \leq e^\epsilon \times P[A(D_2) = O]. \quad (1)$$

Equation 1 is symmetrical and valid for any two neighboring databases D_1 and D_2 . So, this equation can also be written as:

$$e^{-\epsilon} \times P[A(D_2) = O] \leq P[A(D_1) = O] \leq e^\epsilon \times P[A(D_2) = O]. \quad (2)$$

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 $\frac{100}{e^\epsilon + 1}$ % 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 (Cormode *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-DP part, our framework updates the probabilities of watermarking options that violate the local differential privacy condition of Andrés *et al.* and his modified privacy criteria is given as

$$e^{-\epsilon \times r} \times \frac{P(x)}{P(x')} \leq \frac{P(x|S)}{P(x'|S)} \leq e^{\epsilon \times r} \times \frac{P(x)}{P(x')}, \quad (3)$$

$$\forall r > 0, \quad \forall x, x' : d(x, x') \leq r,$$

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 for r , 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 *ae_is*. The second part refers to the unchanging probabilities of states given the prior information and it corresponds to the ratio between *var_is*. We can compare the results for each $x_i = y, y \in \{0, 1, 2\}$, 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) gets 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 tree-like 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 and ϵ -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 3. Frequently used symbols and notations

x_1, \dots, x_{d_l}	Set of data points
y_1, \dots, y_m	Possible values (states) of a data point
d_l	Length of the data
w_l	Length of the watermark
h	Total number of SPs
I_k	Index set of data points that are shared with k^{th} SP
J_k	Index set of data points that are watermarked for k^{th} SP
Z_k	Watermark pattern of k^{th} SP
W_k	Watermarked data shared with k^{th} SP
ϵ	Local differential privacy coefficient
$\sigma_{s,j}$	Correlation coefficient between s^{th} and j^{th} SNPs
ρ	Correlation coefficient threshold
S_i^k	Set of states for index i that are shared with the first k SPs

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.

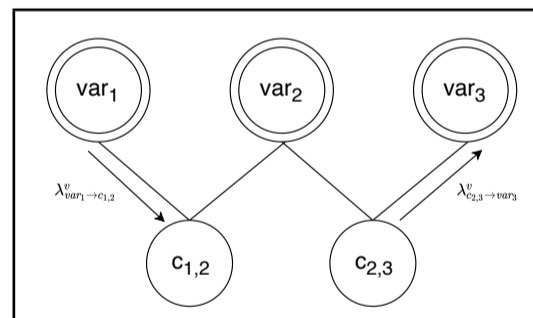


Fig. 1. The relationship between variable nodes and correlation nodes. Both nodes may receive and send messages. For simplicity, one message for each type is shown.

For example, SNP_1 is connected with SNP_2 via $c_{1,2} = i$ for $x_1 = 0$, $x_2 = 2$ with $\sigma_{1,2} = 0.9$ and $\mu_{1 \rightarrow i}^v P(x_1 = y) = [0.3, 0.6, 0.1]$, $y \in \{0, 1, 2\}$ at the v^{th} iteration. Then, we may calculate the message from i to 2^{nd} variable node, $\lambda_{i \rightarrow 2}^v P(x_2 = y)$ as:

$$\lambda_{i \rightarrow 2}^v P(x_2 = y) =$$

$$\left[\frac{1 - (0.9^2 \times 0.3)}{3}, \frac{1 - (0.9^2 \times 0.3)}{3}, \frac{1 - (0.9^2 \times 0.3)}{3} \right]$$

$$+ [0, 0, (0.9^2) \times (0.3)],]$$

$$\lambda_{i \rightarrow 2}^v P(x_2 = y) = [0.2523, 0.2523, 0.4954].$$

2.2.2 Attack-eLDP Node Example

For example, assume for the index i , $var_i = \mu_{i \rightarrow ae_i}^v P(x_i=y) = [0.6, 0.4, 0]$ sends the following message to the *attack-eLDP* node ae_i and $S_i^{k-1} = \{0, 0, 1, 0, 1, 0\}$ where $k = 7$. We may calculate the watermarking probabilities of index i for SP_7 as follows:

$$\begin{aligned} \text{For } x_i^7 = 0, \alpha = 5 &\implies \\ \text{Binomial}(S_i^7 | x_i^7 = 0) &= \binom{7}{5} \times (0.6)^5 \times (0.4)^{7-5} = 0.261, \\ \text{For } x_i^7 = 1, \alpha = 3 &\implies \\ \text{Binomial}(S_i^7 | x_i^7 = 1) &= \binom{7}{3} \times (0.4)^3 \times (0.6)^{7-3} = 0.290, \\ \text{For } x_i^7 = 2, \alpha = 1 &\implies \\ \text{Binomial}(S_i^7 | x_i^7 = 2) &= \binom{7}{1} \times (0)^1 \times (1)^{7-1} = 0. \end{aligned}$$

$A_i^7 = \text{Normalized}([0.261, 0.290, 0]) \approx [0.474, 0.526, 0]$ is the updated watermarking probability distribution of index i for SP_7 .

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

$$\text{For } x_i^7 = 0, \frac{P(x|S)}{P(x'|S)} = \frac{0.474}{0.526} = 0.901, \text{ and } \frac{P(x)}{P(x')} = \frac{0.6}{0.4} = 1.500.$$

This means $e^{-\epsilon} \times 1.500 \leq 0.901 \leq e^{\epsilon} \times 1.500$ must be satisfied for not violating the ϵ -local differential privacy. Since $\frac{0.901}{1.500} \leq 1$ and $\ln(0.601) = -0.509$, if $\epsilon \leq 0.509$ the state probability of $P(x_i^7 = 0)$ must be updated for not violating the privacy by averaging the attack node probabilities with variable node probabilities. After $P(x)$ is set, the distribution is continuously normalized to converge into the probability that satisfies the condition.

$$\text{For } x_i^7 = 1, \frac{P(x|S)}{P(x'|S)} = \frac{0.526}{0.474} = 1.110, \text{ and } \frac{P(x)}{P(x')} = \frac{0.4}{0.6} = 0.667.$$

This means $e^{-\epsilon} \times 0.667 \leq 1.110 \leq e^{\epsilon} \times 0.667$ must be satisfied for not violating the ϵ -local differential privacy. Since $\frac{1.110}{0.667} = 1.664$ and $\ln(1.664) = 0.223$, if $\epsilon \leq 0.223$ the state probability of $P(x_i^7 = 1)$ must be updated likewise.

$$\text{For } x_i^7 = 2, \frac{P(x|S)}{P(x'|S)} = \frac{0}{1} = 0, \text{ and } \frac{P(x)}{P(x')} = \frac{0}{1} = 0.$$

This means $e^{-\epsilon} \times 0 \leq 0 \leq e^{\epsilon} \times 0$ and it never violates the local differential privacy for any ϵ just like the case of $x_i^7 = 0$. However, we know that $P(x_i = 2) = 0$ and it is an impossible watermarking case regardless of the violation.

If Alice set her privacy criteria to $\epsilon < 0.509$, ae_i 's final marginal probability distribution will be equal to the distribution enforced by the eDP 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 the watermarking algorithm. The input of the algorithm includes $data$, $atks$, $wScore$ and w_l . The only output is $newdata$. In the algorithm, $data$ represents the original data of the data owner with length d_l . $atks$ represents the probability distributions of SNPs generated by the belief propagation algorithm considering all the prior information, sharings with SPs, and ϵ -LDP requirements as a 3-by- d_l 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_l 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_l .

In the algorithm, the while loop embeds a watermark to the indices with the highest $wScore$, denoted as $wScore(j, 2)$ and j goes from 1 to d_l until the number of embedded indices reach to w_l . Basically, $wScore(j, 2)$ 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 $atk = [0.32, 0.03, 0.55]$. 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 $wmFlag$ are changed into 2, and the watermark tracker 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_l)

```

1:  $j \leftarrow 1$ 
2:  $k \leftarrow 0$ 
3:  $newdata \leftarrow data$ 
4: while  $k < w_l$  do
5:    $i \leftarrow wScore(j, 2)$  {index of the SNP}
6:    $temp \leftarrow data(i)$  {actual state of SNP}
7:    $wmflag \leftarrow -1$ 
8:    $r \leftarrow \text{random}(0, 1)$ 
9:   if  $atks(i, 1) \geq r$  then
10:    if  $temp \neq 0$  and  $wmflag = -1$  then
11:       $newdata(i) \leftarrow 0$ 
12:       $wmflag(i) \leftarrow 0$ 
13:       $k++$ 
14:    end if
15:    else if  $atks(i, 1) + atks(i, 2) \geq r$  then
16:      if  $temp \neq 1$  and  $wmflag = -1$  then
17:         $newdata(i) \leftarrow 1$ 
18:         $wmflag(i) \leftarrow 1$ 
19:         $k++$ 
20:      end if
21:    else if  $atks(i, 1) + atks(i, 2) < r$  then
22:      if  $temp \neq 2$  and  $wmflag = -1$  then
23:         $newdata(i) \leftarrow 2$ 
24:         $wmflag(i) \leftarrow 2$ 
25:         $k++$ 
26:      end if
27:    end if
28:    if  $j == \text{length}(data)$  then
29:       $j \leftarrow 1$ 
30:    else
31:       $j++$ 
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) 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_l 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 (ϵ). Appendix § 4.2 provides the impact of detection methods and ϵ values on the precision for a single SP attack. In all cases, $w_l \geq 40$ 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_l may not suffice for comparison among 50 SPs. We observe that ϵ 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* outperforms the *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 ϵ to 0.5 or 1 does not impact the precision results at all.

Figure 6 depicts the impact of privacy criteria (ϵ) on precision. The case for $k = 2$ provides limited information because all w_l values, except $w_l = 10$, 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 ϵ 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 $i > 2.4$, 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 $i < 2.4$, $i = 1.6$ 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, $\phi = \frac{1}{2} n^{n+1}$, where n is the degree of affinity and $n=0, 1, 2$ 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_l (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 ϕ results. It should be noted that ϕ 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 (ρ)

Figure 9 shows that ρ 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.

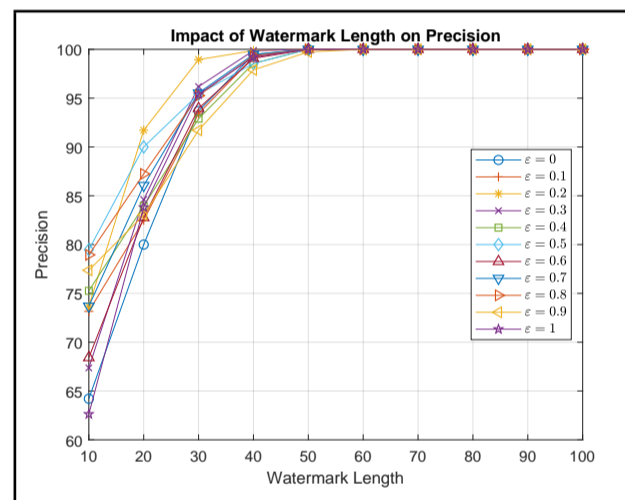


Fig. 2: The impact of watermark length and ϵ on precision for a single SP attack.

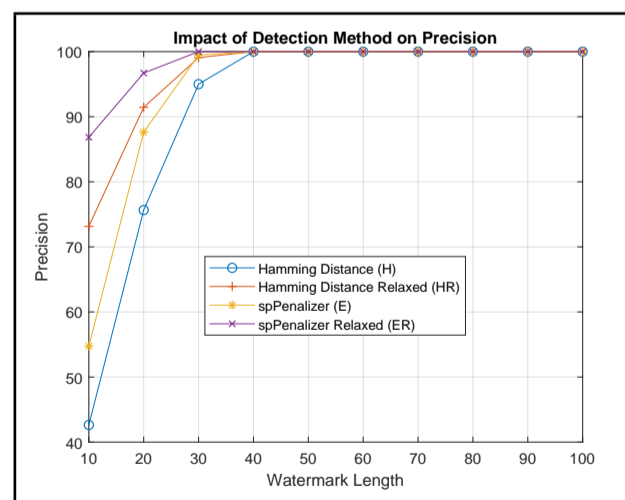


Fig. 3: The impact of watermark length and detection method on precision for a single SP attack ($\epsilon=0$).

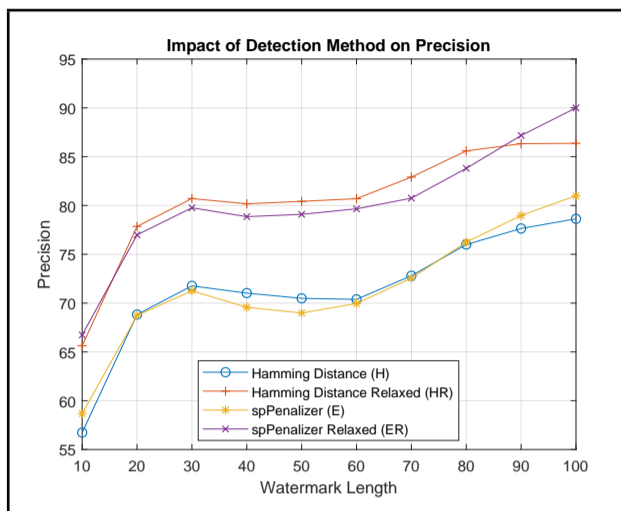
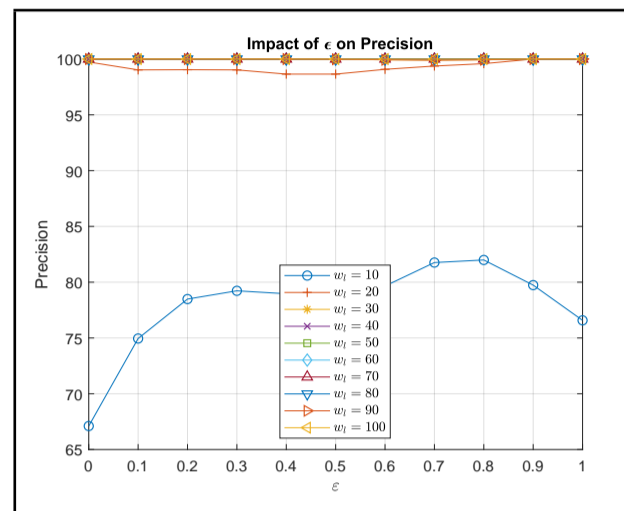
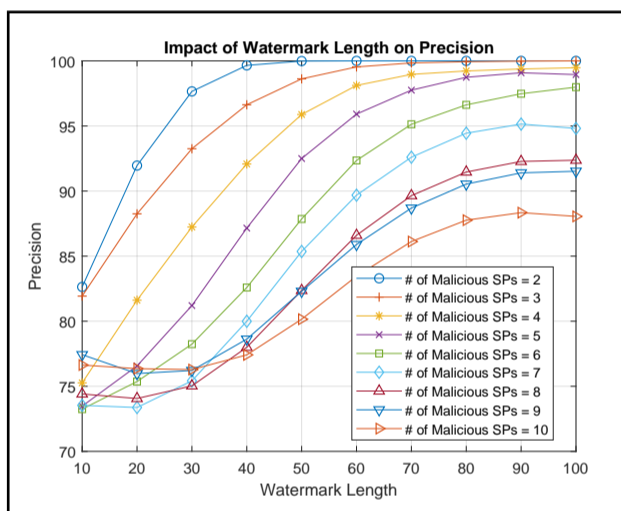


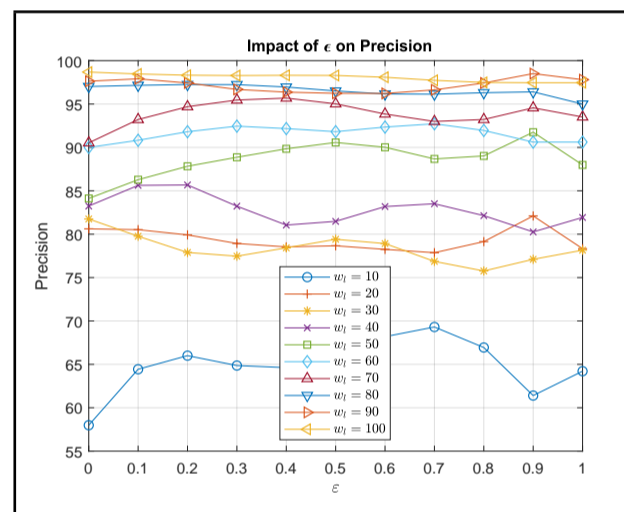
Fig. 4: The impact of watermark length and detection method on precision for a collusion attack ($\epsilon=1$).



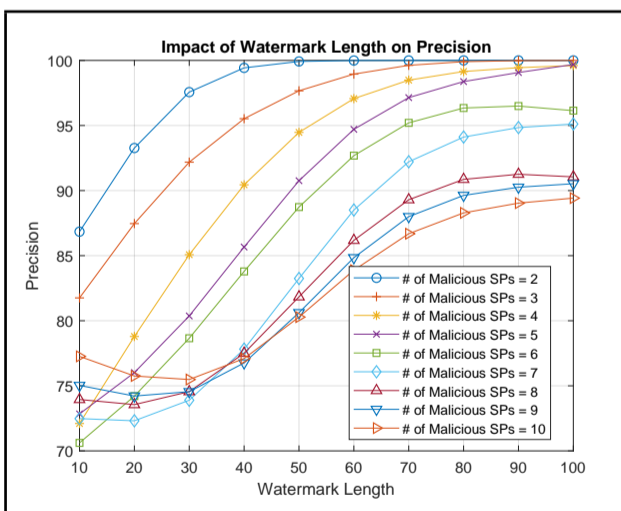
$k = 2$



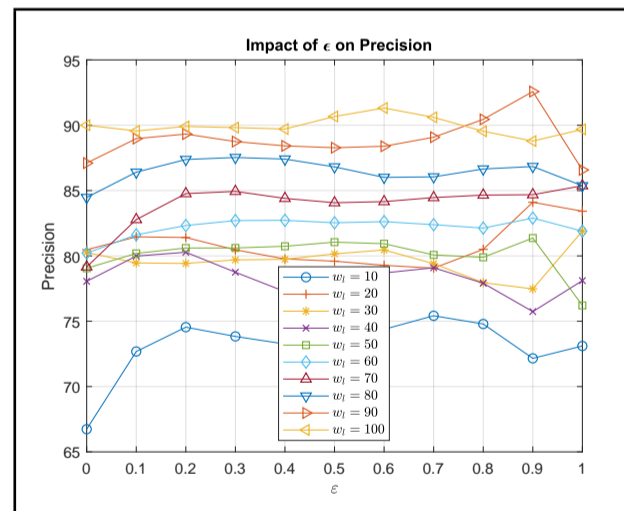
$\epsilon = 0.5$



$k = 6$



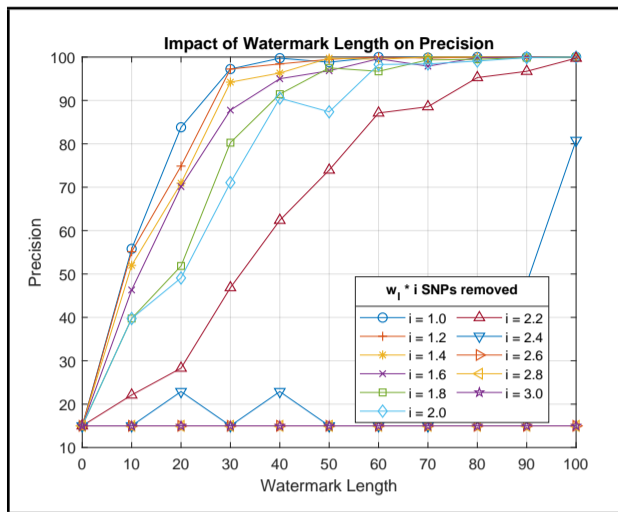
$\epsilon = 1$



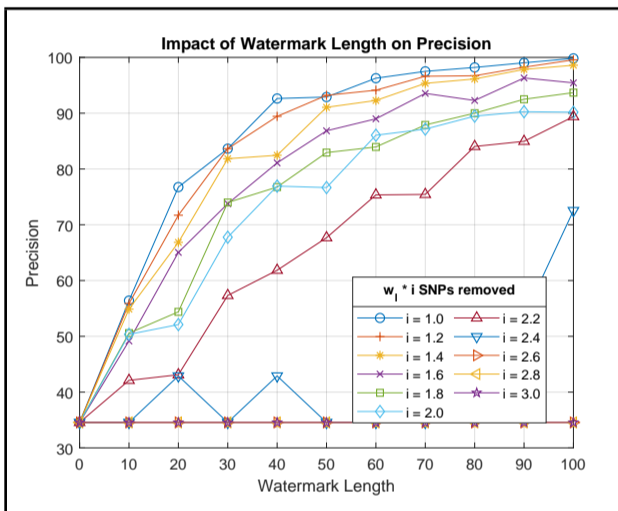
$k = 10$

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

Fig. 6: The impact of ϵ on precision for collusion attacks with different number of malicious SPs and ϵ values.



$SP = 2$



$SP = 6$

Fig. 7: The impact of watermark length on precision for a removal attack with different number of malicious SPs and i values.

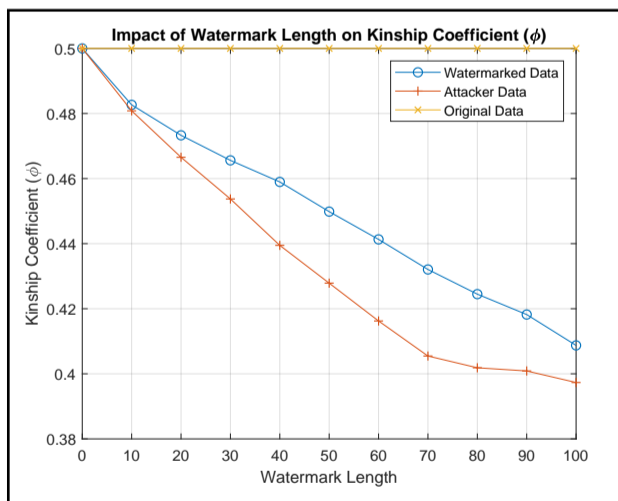


Fig. 8: The impact of watermark length on kinship coefficient.

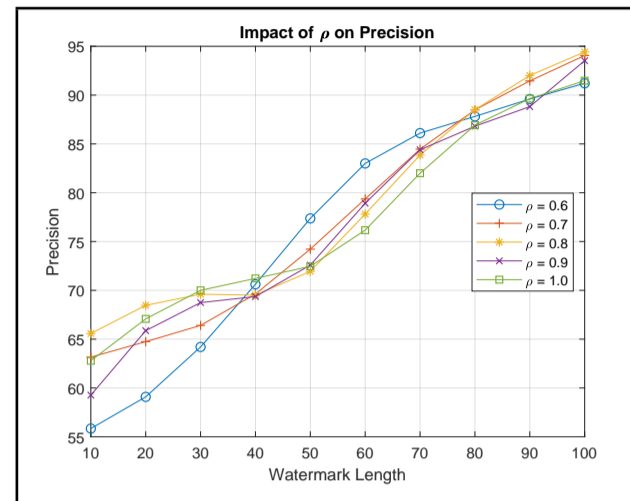


Fig. 9: The impact of watermark length on precision for different ρ values.

References

- Andrés, M. E., Bordenabe, N. E., Chatzikokolakis, K., and Palamidessi, C. (2012). Geo-indistinguishability: Differential privacy for location-based systems. *CoRR*, abs/1212.1984.
- Braunstein, A., Mézard, M., and Zecchina, R. (2005). Survey propagation: An algorithm for satisfiability. *Random Structures and Algorithms*, 27(2), 201–226.
- Cormode, G., Jha, S., Kulkarni, T., Li, N., Srivastava, D., and Wang, T. (2018). Privacy at scale: Local differential privacy in practice. In *Proc. Int. Conf. Management of Data, SIGMOD '18*, pages 1655–1658, New York, NY, USA. ACM.
- Dwork, C. (2006). Differential privacy. In M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, editors, *Proc. 33rd International Colloquium on Automata, Languages and Programming (ICALP '06), Part II*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12, Venice, Italy. Springer.
- Hernandez, R. D., Uricchio, L. H., Hartman, K., Ye, C., Dahl, A., and Zaitlen, N. (2019). Ultrarare variants drive substantial cis heritability of human gene expression. *Nature Genetics*, 51(9), 1349–1355.
- Slatkin, M. (2008). Linkage disequilibrium — understanding the evolutionary past and mapping the medical future. *Nature Reviews Genetics*, 9(6), 477–485.
- Stephens, J. C. (2001). Haplotype variation and linkage disequilibrium in 313 human genes. *Science*, 293(5529), 489–493.