PWS Cup 2019 Sample Program

PWS Cup 2019 Organizing Committee

August 16, 2019 (Last updated on August 29, 2019)

1 Introduction

This paper describes PWS Cup 2019 sample programs for data anonymization, ID disclosure, and trace inference. The notations used in this paper are given in the following document:

- PWS Cup 2019 rule paper submitted to CSS2019 [1].

2 PWSCup2019 Sample Program

2.1 Anonymization Sample Program

2.1.1 A1-none.py

A1-none outputs the original trace as an anonymized trace; i.e., it does nothing (except for shuffling IDs).

2.1.2 A2-MRLH.py

A2-MRLH(μ_x, μ_y, λ) (Merging Regions and Location Hiding; also called Precision Reducing and Location Hiding) [2] generalizes (merges) each region in the original trace by dropping lower μ_x (resp. μ_y) bit(s) for the x-coordinate (resp. y-coordinate) expressed as a binary sequence, and hides (deletes) a region with probability λ. Figure 1 shows an example of A2-MRLH$(1, 1, 0.5)$.

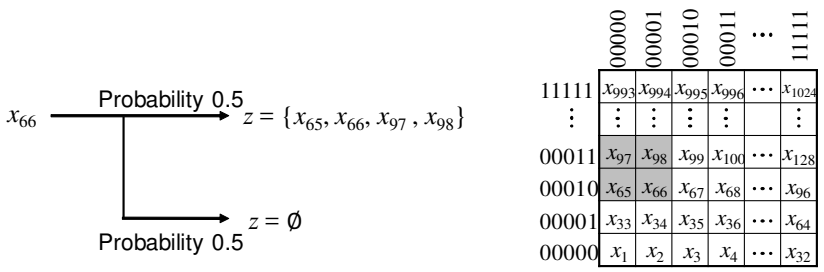

Figure1 Example of A2-MRLH$(1, 1, 0.5)$

2.1.3 A3-kRR.py

A3-kRR(ϵ) (k-ary randomized response) [3] perturbs (adds noise to) each region in the original trace using the k-ary randomized response with parameter (privacy budget) ϵ assigned for each region. Specifically, it outputs the original region with probability $\frac{e^\epsilon}{m-1+e^\epsilon}$, and randomly outputs another region with the remaining probability. It provides ϵ-differential privacy for each region (see [3] for more details).

2.1.4 A4-PL.py

A4-PL(l, r) (Planar Laplace mechanism) [4] perturbs (adds noise to) each region in the original trace according to the planar Laplacian with l-privacy within r km assigned for each region. It provides ϵ-geo-indistiguishability ($\epsilon = l/r$), which gurantees l-differential privacy within the radius of r km, for each region (see [4] for more details).

2.1.5 A5-YA.py

A5-YA(p) (Yamaoka anonymization; also called cheating anonymization or shuffling anonymization) selects the first p ($0 \leq p \leq 1$) of all users (e.g., user ID 1 to 200 if $p = 0.1$) as a subset of users, and randomly shuffles user IDs and replaces the whole traces within the subset. A5-YA(p) with smaller p provides higher utility. A5-YA(p) with $p = 0$ is equivalent to A1-none.

2.2 ID-Disclosure Sample Program

2.2.1 I1-rand.py

I1-rand randomly re-identifies users without replacement; i.e., outputs a random permutation of $1, 2, \cdots, n$ as an estimate of user IDs.

2.2.2 I2-VisitProb.py

I2-VisitProb re-identifies traces based on visit-probability vectors. Specifically, it first trains a visit-probability vector, which is composed of the probability for each region, for each user by using reference traces (for an element with a zero-probability, it assigns a very small positive value ($= 10^{-8}$) to guarantee that the likelihood never becomes zero).

It then re-identifies each trace as follows. It computes the likelihood (the product of the likelihood for each region) for each user. For generalized regions, we average the likelihood over generalized regions. For location hiding (deletion), we don't update the likelihood. After computing the likelihood for each user, it outputs a user ID with the highest likelihood as an identification result. Figure 2 shows an example of I2-VisitProb.

For efficiency, we update the likelihood for each region with probability 10% (i.e., we do not update a likelihood for each region with probability 90%). We also randomly choose 10 regions for a generalized-region with more than 10 regions.

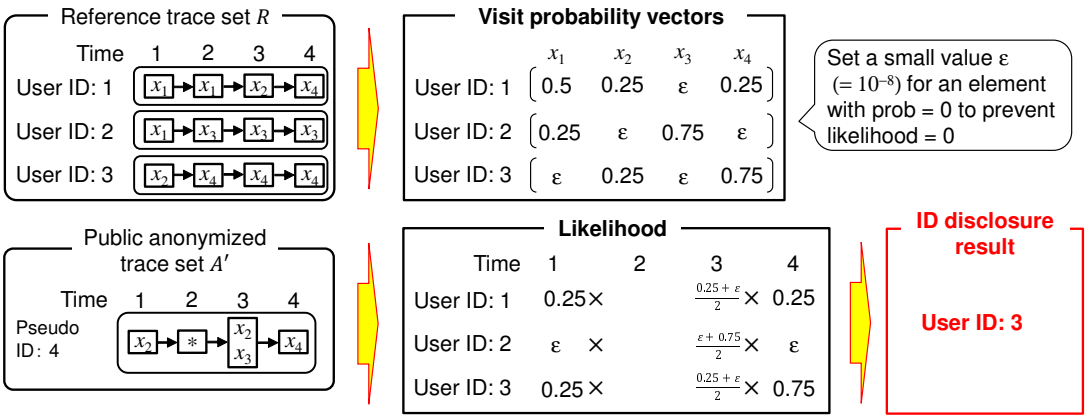

Figure2 Example of ID-disclosure using I2-VisitProb

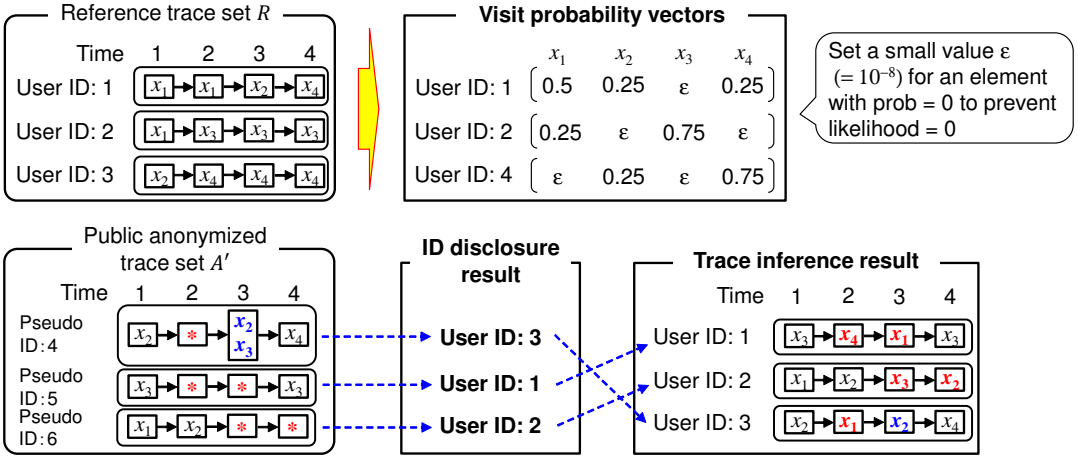

Figure3 Example of trace inference using T2-VisitProb

2.2.3 I3-HomeProb.py

I3-HomeProb re-identifies traces based on the fact that a user tends to be her home region between 8:00 and 8:59 (it is a modification of I2-VisitProb to use only regions between 8:00 and 8:59). That is, it trains a visit-probability vector by using regions between 8:00 and 8:59 in the reference traces. Then it computes the likelihood for each user by using regions between 8:00 and 8:59 in the anonymized traces, and outputs a user ID with the highest likelihood as an identification result.

2.3 Trace Inference Sample Program

2.3.1 T1-rand.py

T1-rand randomly infer the original regions; i.e., outputs a random number between 1 and m as an estimate of each region ID.

2.3.2 T2-VisitProb.py

T2-VisitProb re-identifies traces based on the visit-probability vectors, and then infers the original regions. Specifically, it re-identifies each anonymized trace by using I1-VisitProb.py (it does not choose an already re-identified user to avoid duplication of user IDs). Then it infers the original regions for each re-identified trace. For noise, we output the noisy location as is. For generalization, we randomly choose a region from generalized regions. For location hiding (deletion), we randomly choose a region from all regions.

2.3.3 T3-HomeProb.py

T3-HomeProb is a modification of T2-VisitProb so that it uses only regions between 8:00 and 8:59 (i.e., uses I3-HomeProb) in re-identification.

3 Evaluation Experiment of Sample Programs

3.1 Experiment Condition

Osaka metropolitan area (latitude from 34.64 to 34.74 and longitude from 135.44 to 135.56) in the SNS-based people flow data [5] is divide equally into 32×32 regions and used as learning data to formulate a Markov model-based trace generator. The trace generator is then used to create reference trace sets $R^{(i,j)}$ and original trace sets $O^{(i,j)}$ for two teams (i.e., $1 \leq i \leq 2$, $1 \leq j \leq 2$). We conduct the following experiments with $R^{(1,1)}$ and $O^{(1,1)}$.

First, we anonymize the original trace set $O^{(1,1)}$ using the anonymization sample programs. The following algorithms are used:

- A1-none.
- A2-MRLH(μ_x, μ_y, λ), where (μ_x, μ_y, λ) = (0,0,0.1), (0,0,0.2), (0,0,0.5), (0,0,0.8), (1,1,0), (1,1,0.1), (1,1,0.2), (1,1,0.5), or (1,1,0.8).
- A3-kRR(ϵ), where ϵ = 0.1, 1, 2, 4, 6, 8, 10, 12, or 14.
- A4-PL(l,r), where (l, r) = (1,1), (2,1), (3,1), (4,1), (5,1), (6,1), or (7,1).
- A5-YA(p), where p = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, or 1.

We calculate the utility score s_U for each public anonymized trace set.

Then, we use three ID-disclosure sample programs (I1-rand.py, I2-VisitProb.py, I3-HomeProb.py) and three trace inference sample programs (T1-rand.py, T2-VisitProb.py, T3-HomeProb.py) to get the minimum privacy score against ID disclosure ($s_{I,min}$) and trace inference ($s_{T,min}$).

3.2 Experiment Result

Figure 4 shows the result of the experiment. Figure 4(i) summarizes the result when the requirement value is set as $s_{req} \geq 0$ (i.e., all public anonymized trace sets are considered as valid). Figure 4(ii) summarizes the result when the requirement valus is set as $s_{req} \geq 0.7$.

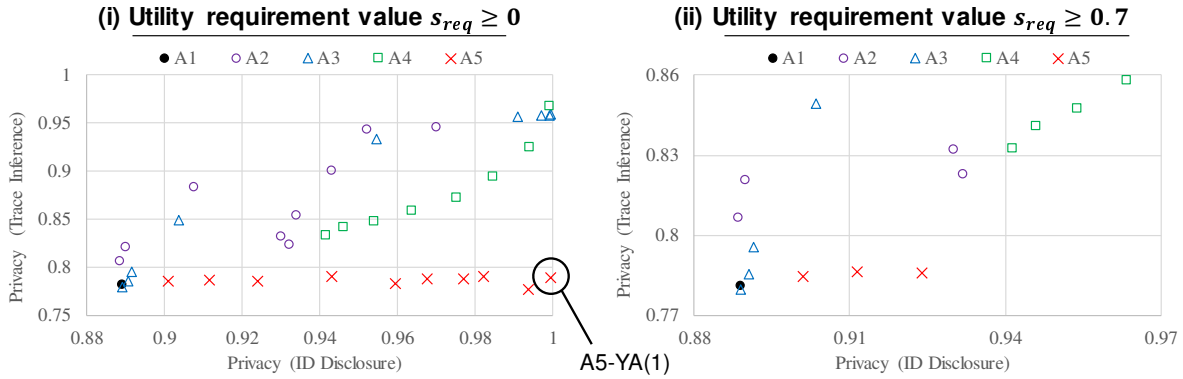

Figure4 Result of evaluation experiment result with sample programs (x-axis: privacy score against ID disclosure $s_{I,min}$, y-axis: privacy score against trace inference $s_{T,min}$)

From the figure, we can see there is some correlation between the privacy score against ID disclosure and the privacy score against trace inference. We can also see that the data anonymization that is strong against ID disclosure attack is not necessarily strong against trace inference attack. This is especially true for A5-YA(1) in Fig. 4(i). This anonymization randomly shuffles traces among all users. The anonymization is very strong against ID disclosure attack, but it is vulnerable to trace inference attack; its privacy score against trace inference attack is almost the same as that of A1-none (that does nothing). This result implies that attackers can successfully infer traces of A5-YA(1) without disclosing user ID (i.e., without successfully inferring original user ID in the anonymized traces).

References

[1] Takao Murakami, Hiromi Arai, Makoto Iguchi, Hidenobu Oguri, Hiroaki Kikuchi, Atsushi Kuromasa, Hiroshi Nakagawa, Yuichi Nakamura, Kenshiro Nishiyama, Ryo Nojima, Takuma Hatano, Koki Hamada, Yuji Yamaoka, Takayasu Yamaguchi, Akira Yamada, Chiemi Watanabe, "PWS Cup 2019: Location Data AnonymizationCompetition," CSS2019.

[2] R. Shokri *et al.*, "Quantifying location privacy," Proc. IEEE S&P'11, pp.247–262, 2011.

[3] P. Kairouz *et al.*, "Discrete Distribution Estimation under Local Privacy," Proc. ICML, pp.2436–2444, 2016.

[4] M. E. Andrés *et al.*, "Geo-Indistinguishability: Differential Privacy for Location-based Systems," Proc. CCS'13, pp.901–914, 2013.

[5] Nightley and Center for Spatial Information Science (CSIS), SNS-based People Flow Data, [online] Available: `http://nightley.jp/archives/1954`.