Appendix A
Artifact Appendix

A. Description & Requirements

1) How to access: Our code can be pulled from the public repository on GitHub. Use the following command: git clone https://github.com/XiaochenLi-w/Stream-release-with-delay-time.git.

2) Hardware dependencies: None.

3) Software dependencies: Python 3.10 and above. 'numpy', 'math', 'random', 'os', 'sys', 'matplotlib', 'scipy' and 'sklearn' libraries are required.

4) Benchmarks: All datasets used in the experiments are stored in the 'data' folder. The source code for other competitors (Adapub: https://dbresearch.uni-salzburg.at/projects/dpbench/ and DPI: https://github.com/ShuyaFeng/DPI) is included in the 'other_competitor' folder, which is also ready for evaluation.

B. Artifact Installation & Configuration

We have tested the source code in Python 3.10.8 and 3.11.4 environments, the code runs without issues in these environments. Additionally, please ensure that the 'numpy', 'math', 'random', 'os', 'sys' and 'matplotlib' libraries are installed. The code does not have high hardware requirements; a single-core CPU is sufficient.

C. Major Claims

- (C1): Allowing a delay of only 10 timestamps can bring about a substantial increase in accuracy for the released streams (Compare CompOrder, Discontin_pp, BucOrder, Contin_reduce, Discontin_reduce with Naive, PeGaSus, PeGaSus_delay).
- (C2): The performance of PeGaSus is not improved compared to the naïve approach and is even lower than the naïve approach.
- (C3): The advantage of the order-based approach (CompOrder, BucOrder) is more pronounced when the delay time is short compared to the group-based approaches (Discontin_pp, Contin_reduce, Discontin_reduce, PeGaSus_delay).
- (C4): The data sensitivity truncation algorithm has a significant improvement on all datasets, especially on streams with large data domains such as Outpatient and Foodmart datasets.

D. Evaluation

In this section, we provide a detailed step-by-step verification of the experimental results included in the paper. (E1) and (E2) involve validating the effectiveness of the approaches proposed in this paper, where we will elaborate on how the experimental results correspond to the four key statements in Major Claims. (E3) and (E4) evaluate the impact of parameters on the proposed approaches, (E5) compares the proposed methods with state-of-the-art methods from other privacy settings, and we will detail how to verify the results presented in the paper based on experimental outcomes.

TABLE I
The Running Time of Experiment (E1)

Delay	COVID	Unemployment	Outpatient	Foodmart
$w = 10$	5.9s	5.7s	1min33s	2min42s
$w = 100$	8.9s	22.5s	3min10s	13min12s

1) Experiment (E1): [The Accuracy Improvement of the Delay-allowed Approaches.][About 21min]

This part of the experiment corresponds to Figures 2 and Figure 3 in the paper and requires testing with the 'est_compall.py' file in the estimator folder. We have clearly specified all the parameters, including the datasets and privacy parameters. The evaluator only needs to modify the 'delay_time' parameter on line 128.

The approximate running times for the four datasets under two different delay_time parameters are shown in Table I. These times were tested on a 12th Gen Intel(R) Core(TM) i5-1240P 1.70 GHz processor. Running times may vary with different processors.

[Preparation] Please ensure that the numpy, matplotlib, random, math, os, and sys libraries are installed.

[Execution] There are two steps for verifying Figure 2 and Figure 3.

Step 1. Please do not modify other parameters. Simply set 'delay_time=10' on line 128 and run 'python est_compall.py'. The terminal will sequentially display the error results of all methods on the four datasets, and comparison line charts for all methods will pop up. Note that you need to manually close the popped-up line charts for the code to continue running the next dataset.

Step 2. Simply set 'delay_time=100' on line 128 and rerun 'python est_compall.py'. Other operations are the same as in Step 1.

[Results] In each step, the results displayed in the terminal and the line charts popped up by the code correspond to Figures 2/Figure 3 (a), (b), (c), and (d) in the paper. The line colors in the charts do not match those in the paper, so please compare the results of each method using the legend. There may be some fluctuations, the trend of each method's error with changes in ϵ, and the relative magnitudes should be consistent with those in the paper. The observed results in this part correspond to C1, C2, and C3 in the Major Claims.

2) Experiment (E2): [The Effectiveness of Data Sensitivity Truncation Algorithm.][About 7min47s]

This part corresponds to Figure 4 in the paper and is used to evaluate the effectiveness of the sensitivity reduction method proposed in this paper. We use 'est_sensitivity.py' to run this evaluation.

[Preparation] Same as in Experiment (E1).

[Execution] We have already set all the parameters. Please run 'python est_sensitivity.py' directly.

[Results] The terminal will sequentially display the results of the Naive, Contin, Discontin, CompOrder, and BucOrder methods on the four datasets, both without

sensitivity reduction and with sensitivity reduction. The data under "No Sensitivity Reduce" correspond to the red bars in Figure 4, while the results under "Sensitivity Reduce" correspond to the blue bars in Figure 4. Although there may be fluctuations due to randomness, the magnitude of the data should be consistent with those in Figure 4. For all datasets, except for the BucOrder method, the data under "Sensitivity Reduce" should show a significant decrease compared to "No Sensitivity Reduce". The observed results in this part correspond to C4 in the Major Claims.

3) Experiment (E3): [Comparison of Group-based Approaches and Order-based Approaches.][About 58s]

This part corresponds to Figure 5 in the paper, illustrating the impact of delay time length on group-based and order-based methods respectively. We use 'est_delaylength.py' to run this evaluation.

[Preparation] Same as in Experiment (E1).

[Execution] We have already set all the parameters. Please run 'python est_delaylength.py' directly.

[Results] In the terminal, the results for the discontinuous, CompOrder, and BucOrder methods on two datasets will be sequentially displayed, showing the error in published results as the parameter w varies from 10 to 90. The results for each dataset will be between "Results" and "Results End". All data should correspond to the bars in Figure 5.

We state the following: a. The performance of the discontinuous approach improves as the length of the delay time increases. b. The order-based (CompOrder and BucOrder) approaches do not exhibit a clear pattern of accuracy variation with changes in the delay time. c. When the delay time is short, order-based (CompOrder and BucOrder) approaches demonstrate an advantage over group-based (discontinuous) approaches.

4) Experiment (E4): [Impact of the key parameters.][About 18s and 12s]

This part of the experiment corresponds to Figures 6, 7, 8, 9, and 10, primarily evaluating the impact of parameter variations on the performance of the proposed algorithms. We use 'est_order.py' and 'est_group.py' to run this evaluation.

[Preparation] Same as in Experiment (E1).

[Execution] There are two steps in this part. We have already set all the parameters.

Step 1. Please run 'python est_order.py' directly. You'll need to manually close the popped-up line charts to proceed with running the next dataset.

Step 2. Please run 'python est_group.py' directly.

[Results] For Step 1, you'll see the error results of the BucOrder method on two datasets sequentially displayed in the terminal. Additionally, line charts similar to those in Figure 6 from the paper will pop up. Due to randomness, there may be fluctuations in the lines, but the overall trends and magnitudes of the data should be similar to those in the paper. We claim that too-small bucket size m can lead to a decrease in the accuracy of the released results.

For Step 2, you'll sequentially observe the accuracy of the publication results for four group-based methods across different values of τ on two datasets. On each dataset, the results in the terminal will correspond sequentially to Figures 7, 9, 8, and 10. Based on our multiple experiments, we observed that the impact of θ on group-based methods is not monotonical. In our paper, we state that the accuracy of both continuous and discontinuous approaches does not monotonically change with the threshold θ. The magnitudes of the error results in the terminal should correspond to the respective figures in the paper.

5) Experiment (E5): [Compare with existing works come from other privacy settings.][DPI: 31s, Adapub: 4s] This part of the experiment corresponds to Table III in the paper. The source code for DPI and Adapub comes from their corresponding open-source libraries. We only made some minor modifications to make them meet the event-level privacy and set the parameters for comparison in our experiment. We use 'est_other.py' and 'DPI_DEMO_script.py' to run this evaluation.

[Preparation] scipy and sklearn libraries are required by DPI.

[Execution] There are two steps in this part. We have already set all the parameters.

Step 1. Please run 'python est_other.py' directly. You can directly obtain the results of Adapub required in Table III on two datasets.

Step 2. Please run 'python DPI_DEMO_script.py' directly. Annotate lines 170-181 and uncomment lines 184-195 to run the next dataset. You can obtain the results of DPI required in Table III on two datasets.

[Results] In Step 1 of E1, we have obtained the error results of $\epsilon \in [0.1, 1]$ for Naïve, Discontin, CompOrder, and BucOrder on the Unemployment and Outpatient datasets. By dividing the error results (MAE) of all methods by the error results of naïve, the correctness of Table III can be verified. We claim that even with just 10 timestamps delay, the proposed algorithm shows accuracy advantages over state-of-the-art solutions. Directly comparing the error results of these methods can also validate this claim.