
# USENIX Security '25 Artifact Appendix: ALERT: Machine Learning-Enhanced Risk Estimation for Databases Supporting Encrypted Queries

Longxiang Wang[†,*], Lei Xu[‡,†,*], Yufei Chen[†], Ying Zou[‡], Cong Wang[†]

[†]City University of Hong Kong [‡] Nanjing University of Science and Technology

## A Artifact Appendix

### A.1 Abstract

The artifact is used to evaluate ALERT in Dynamic Searchable Symmetric Encryption (DSSE) schemes. It includes source code for reproducing ALERT and three baseline LAAs, along with the datasets and scripts required to run the experiments. A comprehensive README file is also included to facilitate easy reproduction of the results.

### A.2 Description & Requirements

In this section, we detail the accessibility of our artifact, specify the required hardware and software dependencies, and describe the benchmarks used in our experiments.

Specifically, our artifact contains eight folders: baselines, dataset, final_result, log, model, result, scripts, and src. The "baselines" folder contains code for evaluating results from three baseline LAAs. The "dataset" folder stores preprocessed documents for searchable encryption. The "final_result", "log", "model", and "result" folders contain essential results and data (e.g., trained models, extracted co-occurrence matrices) in the risk assessment process. The "scripts" folder provides scripts for launching risk assessments across different SSE schemes and adversary assumptions. The "src" folder contains ALERT's core source code, including data preprocessing utilities, classifier training modules, and query risk assessment implementations.

#### A.2.1 Security, privacy, and ethical concerns

Throughout our study, we have conducted a thorough ethical assessment of our research methodology and potential impacts. Our investigation does not involve human participants, personal information, or any sensitive data. The research relies exclusively on three publicly available datasets for experimental evaluation, neither of which contains any personally identifiable information. Therefore, there are no ethical concerns in this paper.

* The first two authors contributed equally to this work.

#### A.2.2 How to access

Our artifact is available through Zenodo. The artifact can be accessed at https://doi.org/10.5281/zenodo.14726862.

#### A.2.3 Hardware dependencies

We implemented and evaluated ALERT on a server equipped with two Intel Xeon Platinum 8383C CPUs, eight Nvidia RTX A6000 GPUs, 2TB of RAM, and 12TB of disk storage. While the implementation of ALERT is GPU-based, its main functionality can also be validated by training the model on alternative CPU/GPU configurations.

#### A.2.4 Software dependencies

Our experiments were primarily conducted on Ubuntu 22.04, though our system is compatible with any operating system that supports Python. For dependency management, we use "Conda" as our environment management system. All required packages and their specific versions are documented in the provided "requirements.txt" file in the published repository.

#### A.2.5 Benchmarks

In the experiment, we use three widely adopted public datasets from different domains: the Enron email dataset, the NYTimes article dataset, and the Wikipedia encyclopedia entries dataset. All datasets used in our experiments are included in the "dataset" directory of our published artifact. Detailed information about the workloads is presented in Section A.4.2.

### A.3 Set-up

This section provides detailed instructions for setting up our experimental environment. The setup utilizes Conda for environment management and includes a verification step to confirm the correct installation of all components.

#### A.3.1 Installation

To ensure full compatibility with ALERT, we recommend creating a new conda environment with Python 3.9.16:

```
conda create -n alert python=3.9.16
conda activate alert
```

Next, install the required dependencies:

```
pip install -r requirements.txt
```

The experiments can be reproduced with the provided scripts.

#### A.3.2 Basic Test

We provide a simple test to verify that all dependencies are properly installed. Execute the following commands:

```
cd scripts
bash test_single_risk_assessment.sh
```

If dependencies are correctly installed, a file should appear in the "final_result/test_single_risk_assessment" directory.

### A.4 Evaluation workflow

#### A.4.1 Major Claims

**(C1):** ALERT demonstrates robust risk assessment capabilities across diverse datasets and adversary settings. This is validated through experiments (**E1**) detailed in Sections 5.2.2 and 5.2.3, with results in Figure 4.

**(C2):** ALERT is able to provide risk assessment maps among keywords and illustrate listed findings: (1) Larger $\gamma$ generally leads to higher overall leakage risk. (2) An increase in leakage ($\gamma$) may introduce noise that obscures distinct patterns, making certain keywords harder to recover. **E2** detailed the experiment steps, corresponding to results in Section 5.2.1 and Figure 3.

**(C3):** ALERT shows robustness across FP/BP-DSSE schemes (with different $\lambda$ values). Experimental validation (**E3**) confirms the findings in Section 5.4, with corresponding results visualized in Figure 6.

**(C4):** Keyword clustering based on volume information is effective. Section 5.3 describes experiment results (**E4**), whose results are plotted in Figure 5.

**(C5):** Keyword dynamic clustering mechanism (DCM) can significantly reduce risk assessment latency while maintaining risk assessment accuracy. **E5** shows the experiment steps, corresponding to Section 5.3 and Table 1.

**(C6):** ALERT is the only method to consistently achieve a high query recovery rate with stable convergence under low-latency scenario among the selected risk analysis methods. **E6** demonstrates the experiment steps, corresponding to Section 5.5.1 and Figure 7 in the paper.

**(C7):** ALERT achieves competitive query recovery performance compared to other LAA methods without time constraints. The validation workflow (**E7**) corresponds to Section 5.5.2 and results in Figure 8 in the paper.

**(C8):** ALERT exhibits superior recovery performance (median query recovery rates) compared with Jigsaw under similar runtime constraints in large keyword universe sizes scenario. Experimental validation through **E8** corresponds to Section 5.5.3 and Figure 9 in the paper.

**(C9):** ALERT shows higher median query recovery rates compared with prior alternatives against three padding countermeasures. **E9** specifies experiment workflows, corresponding to Section 5.5.4 and Figure 10.

#### A.4.2 Experiments

This section provides experimental procedures to validate our major claims. Each experiment description is structured into three main blocks: Preparation, Execution, and Results, ensuring a clear approach to reproducing our findings.

**Preparation:** Before running each script, change the directory to the "scripts" directory and activate the created conda environment. Set the "if_gpu" field in each script to choose between CPU or GPU training. If using GPU mode, specify the desired GPU deviced with "GPU_NUM" (e.g., "GPU_NUM=0:1" to use GPUs 0 and 1).

**(E1):** [Query Recovery Performance on Different Datasets and Adversarial Assumptions] [5 human-minutes + 60 compute-hours + 510GB disk]: The experiment performs risk assessments for Enron, NYTimes, and Wikipedia under two adversarial settings.

**Execution:** To run the experiment:

```
bash main_recovery.sh
```

**Results:** The results are stored in the "final_result/main_recovery" folder. Each file contains a "cumulative_accuracy" field that records recovery rates for different keyword numbers. The filename indicates the experimental settings: dataset names (Enron/NYTimes/Wiki), adversary type (partial/sample), and adversary assumptions (data partial known rate $\gamma$, and data sampling rate $\alpha$).

**(E2):** [An Example of Recovery Map of ALERT ] [5 human-minutes + 4 compute-hours + 10GB disk]: This experiment demonstrates a visual representation of ALERT (Figure 3 in paper). It's important to note that risk assessment results for concrete queries may exhibit variance across different training sessions due to randomness in training data selection. Therefore, we provide specific training results (probability matrices) for generating the heat map shown in the paper.

**Execution:** To generate the heatmap:

```
bash test_heatmap.sh
```

**Results:** The heatmaps generated with different data partial known rates $\gamma$ are available in "final_result/heatmap". When $\gamma$ is small, the random selection of the training data may introduce variability in risk assessment outcomes, resulting in figures that deviate from those reported in the paper. For reference,

we provide the original data and heatmaps for reference in "final_result/heatmap/ref".

**(E3):** [Performance across Forward/Backward Privacy-DSSE] [5 human-minutes + 10 compute-hours + 50GB disk]: This experiment aims to illustrate the effectiveness of ALERT in FP/BP-DSSE schemes.

**Execution:** We vary data deletion rates $\lambda$ and show its impact on query recovery rates. To run the experiment:

```
bash test_fpbp.sh
```

**Results:** The results of the experiment are stored in "final_result/test_fpbp" directory, demonstrating the recovery rates under different data deletion rates $\lambda$.

**(E4):** [Keyword Clustering Results with Fixed Parameters] [5 human-minutes + 1 compute-hour]: This experiment validates the effectiveness of keyword clustering based on volume information (as shown in Figure 5).

**Execution:** The experiment requires first running the "main_recovery" script to generate the co-occurrence log. For independent execution, we provide a separate script to generate these co-occurrence logs.

```
bash test_dcm_fixed_param.sh
```

Then, keyword clustering results can be generated by executing the provided ipynb file:

```
generate_result_dcm_fixed_param.ipynb
```

**Results:** The notebook contains three sections, each corresponding to a different subfigure in Figure 5, demonstrating the effects of different keyword numbers ($\vartheta$), sampling rates ($\alpha$), and clustering thresholds ($\theta$).

**(E5):** [Performance with/without Dynamic Keyword Clustering Mechanism] [5 human-minutes + 45 compute-hours + 50GB disk]: This experiment evaluates the impact of keyword dynamic clustering mechanism (DCM) on query recovery rates and risk assessment latencies.

**Execution:** We compare the query recovery rates and program runtime with and without the mechanism. Here is the script how to run the experiment:

```
bash test_dcm.sh
```

**Results:** The results are stored in "final_result/test_dcm" directory. Each output file contains query recovery rates and program runtime.

**(E6):** [Comparisons with Prior Alternatives under Low-latency Scenario] [5 human-minutes + 45 compute-hours + 10GB disk]: This experiment aims to compare the effectiveness of ALERT with prior alternatives in the risk assessment scenario.

**PS**: Although ALERT performs a quick risk assessment, the training process takes about 1-2 hours. For quick validation of the main claims, the parameter "RUN_TIME" can be set to 5 in the scripts of later experiments (**E6, E7, E8, E9**) to ensure minimum iterations required for box plot generation. Note that this simplification may lead to increased variance in the results.

**Execution:** In this section, we adjust the convergence speed parameters for each method: "RefSpeed" for Jigsaw and RSA, "n_iters" for IHOP, and $\tilde{\beta}$ for ALERT. These adjustments control the program runtime and risk assessment time. To run the experiment:

```
bash test_attacks_low_latency.sh
```

**Results:** The results are reserved in "final_result/test_attacks_low_latency" folder, where each filename indicates the corresponding dataset, attack method, and desired risk assessment latency. Each file contains data for box plot generation, including minimum value, first quartile (Q1, 25th percentile), median, third quartile (Q3, 75th percentile), maximum value, and the average runtime in seconds.

**(E7):** [Comparisons with Prior Alternatives without Time Constraints] [5 human-minutes + 50 compute-hours + 10GB disk]: This experiment compares the effectiveness of different methods without runtime constraints.

**Execution:** We provide the script for the experiment:

```
bash test_attacks_default.sh
```

**Results:** Correspondingly, the results are stored in "final_result/test_attacks_default". Note that we set $\tilde{\beta} = 1$ to maximize the risk assessment recovery accuracy when operating without time constraints. For additional validation with $\tilde{\beta} = 0.4$ under default settings, run:

```
bash test_attacks_default_extend.sh
```

**(E8):** [Comparisons in Larger Keyword Universe Sizes under Similar Time Constraints] [5 human-minutes + 90 compute-hours + 30GB disk]: This experiment compares ALERT and Jigsaw in the Wikipedia dataset to evaluate their effectiveness facing larger keyword universe sizes while maintaining comparable execution times.

**Execution:** To run the experiment:

```
bash test_large_keyword_wiki.sh
```

**Results:** The results are stored in "final_result/test_large_keyword_wiki", where each filename indicates the dataset and corresponding keyword universe size used in the experiment.

**(E9):** [Comparisons with Prior Alternatives Against Countermeasures] [5 human-minutes + 90 compute-hours + 30GB disk]: This experiment evaluates ALERT and baseline LAAs against three padding countermeasures.

**Execution:** To facilitate reproduction, we provide a script that executes the complete evaluation process:

```
bash test_against_countermeasures.sh
```

**Results:** The experimental results are stored in the "final_result/test_against_countermeasures" directory. Each filename contains the corresponding dataset name, risk analysis method, and applied countermeasure.

## A.5 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2025/.