Query Recovery from Easy to Hard: Jigsaw Attack against SSE

Hao Nie and Wei Wang, *Huazhong University of Science and Technology;* Peng Xu, *Huazhong University of Science and Technology, Hubei Key Laboratory of Distributed System Security, School of Cyber Science and Engineering, JinYinHu Laboratory, and State Key Laboratory of Cryptology;* Xianglong Zhang, *Huazhong University of Science and Technology;* Laurence T. Yang, *Huazhong University of Science and Technology and St. Francis Xavier University;* Kaitai Liang, *Delft University of Technology*

https://www.usenix.org/conference/usenixsecurity24/presentation/nie

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 33rd USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 33rd USENIX Security Symposium.

August 14–16, 2024 • Philadelphia, PA, USA

978-1-939133-44-1

Open access to the Artifact Appendices to the Proceedings of the 33rd USENIX Security Symposium is sponsored by USENIX.

USENIX Security '24 Artifact Appendix: Query Recovery from Easy to Hard: Jigsaw Attack against SSE

Hao Nie[1], Wei Wang[1,✉], Peng Xu[1,2,3,4,✉], Xianglong Zhang[1], Laurence T. Yang[1,5], and Kaitai Liang[6]

[1]*Huazhong University of Science and Technology*
[2]*Hubei Key Laboratory of Distributed System Security, School of Cyber Science and Engineering*
[3]*JinYinHu Laboratory*
[4]*State Key Laboratory of Cryptology*
[5]*St. Francis Xavier University*
[6]*Delft University of Technology*
✉*Corresponding authors: viviawangwei@hust.edu.cn, xupeng@hust.edu.cn*

A Artifact Appendix

A.1 Abstract

Searchable symmetric encryption (SSE) schemes often unintentionally disclose certain sensitive information. Attackers can exploit such leakages and other available knowledge related to the user's database to recover queries. We find that the effectiveness of query recovery attacks depends on the volume/frequency distribution of keywords. Queries containing keywords with high volumes/frequencies are more susceptible to recovery, even when countermeasures are implemented. Attackers can also effectively leverage these "special" queries to recover all others. By exploiting the above finding, we propose a Jigsaw attack that begins by accurately identifying and recovering those distinctive queries. Leveraging the volume, frequency, and co-occurrence information, our attack achieves 90% accuracy in three tested datasets.

The results consist of three aspects: 1) Evaluations of Jigsaw (Section 5 and Appendix B and C). These evaluations show that the idea of recovery from easy to hard in Jigsaw is viable and Jigsaw can successfully recover the queries by utilizing the leakage of SSE. 2) Comparisons with other attacks (Section 6 and Appendix D). These comparisons show that Jigsaw provides $> 90\%$ accuracy surpassing the Graphm and SAP attacks, similar to the RSA and IHOP. Jigsaw also has other superior aspects to IHOP and RSA. 3) Comparisons with other attacks when against countermeasures (Section 7 and Appendix E, F, and G). Jigsaw maintains high accuracy against the padding (in most cases) and obfuscation and takes the lead in accuracy in most cases. We provide all of the scripts to reproduce the experimental results of the paper along with the datasets.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

None. Our experiment does not have risks for evaluators while executing artifacts. The test datasets are publicly available, and the attacks just use the simulated leakage generated by the publicly available datasets.

A.2.2 How to access

Codes are available in GitHub by link: https://github.com/JigsawAttack/JigsawAttack/tree/AEStableVersion2.

A.2.3 Hardware dependencies

The artifacts do not require special hardware. Machines that can run Python code can run the artifacts. However, to run the experiments related to the Wikipedia dataset, large memory like 64GB is suggested. We use Python 3.95 to simulate and run codes in Ubuntu 22.04.1 with 16 cores of an Intel(R) Xeon(R) Gold 5120 CPU (2.20GHz) and 64 GB RAM.

A.2.4 Software dependencies

The artifacts require the following software or packages:

- Python v3.9 or above.
- Packages of Python: numpy, matplotlib, tqdm, and scipy.

To test the Graphm attack, other packages are needed to execute the Graphm attack. We use the PATH algorithm from the GraphM package (https://projects.cbio.mines-paristech.fr/graphm/) to implement the Graphm attack.

A.2.5 Benchmarks

We use the Enron, Lucene, and Wikipedia datasets to test the attacks. The Enron and Lucene datasets are included in the repository and the Wikipedia dataset can be downloaded in *https://drive.google.com/file/d/1ltB3oyiDV0Ef7v0dRBlXV_wwvBUwas05/view?usp=drive_link* and *https://drive.google.com/file/d/18_RG5GiH65IAI64HE5Em7yC7hbIkfBSG/view?usp=drive_link* or in Zenodo with *https://zenodo.org/records/12683869*. To produce the result in Wikipedia when keyword universe size $\leq 3,000$, run the "preprocess.py" first, and when keyword universe size $= 5,000$, run the "preprocess.py" with line 7 changing to "kws_universe_size = 5000". The "preprocess.py" may take hours to finish.

A.3 Set-up

A.3.1 Installation

Installing python and the packages listed above is enough to run all attacks except for the Graphm attack in our paper. To run the Graphm attack, please follow the instructions in https://projects.cbio.mines-paristech.fr/graphm/.

A.3.2 Basic Test

Run the "run_single_attack.py" to simulate a single attack. Using *"python run_single_attack.py attack=Jigsaw dataset=enron kws_universe_size=500"* to simulate a single attack of Jigsaw. Replace "Jigsaw" with "IHOP", "RSA", "SAP", or "Graphm" to test other attacks. Replace "enron" with "lucene" or "wiki" to test other datasets. Also the "kws_universe_size" can be set to other values.

This will output the accuracy of the tested attack and its runtime.

A.4 Evaluation workflow

A.4.1 Major Claims

Our major claims are:

(C1.1): The distribution of queries in tested datasets has the distribution of frequency and volume like Figure 1 in Section 4.1 and Figures 11 and 12 in Appendix B.

(C1.2) Module 1 of Jigsaw shows that high-frequency or high-volume queries are easier to recover (Module 1 has higher accuracy). The results are shown in Figures 2 and 3 in Section 5.2. Module 1 and 2 of the Jigsaw attack can have a high attack accuracy when recovering a part of queries. The experiment results are illustrated in Table 2 in Section 5.2.

(C1.3) Jigsaw can have high accuracy with/without frequency information (in Figure 4 Section 5.3). The Jigsaw remains stable when the observation time is long from the auxiliary information (in Table 3, Section 5.3).

(C2.1) Jigsaw has similar accuracy compared to RSA and IHOP and has higher accuracy than SAP and Graphm (in Figure 5, Section 6.2).

(C2.2) Jigsaw has higher accuracy comparisons with IHOP within similar runtime (in Figure 6 Section 6.2).

(C3) Jigsaw takes lead in accuracy in most cases when against countermeasures (Padding in CGPR, Obfuscation in CLRZ, Padding in SEAL, and the Cluster-based padding), especially in the Wikipedia dataset. Results are shown in Figures 7,8,9, and 10 in Sections 7.1 and 7.2 (and in Figures 16, 17, 18, and 19 in Appendix F).

A.4.2 Experiments

Preparation: [2 human-minutes][2 computer-minutes + 0.5GB disk]: Download or clone the repository in https://github.com/JigsawAttack/JigsawAttack/tree/AEStableVersion.

Preparation for tests on Wikipedia: [2 human-minutes][200 computer-minutes + 10GB disk]: Download the dataset of Wikipedia in https://drive.google.com/file/d/1ltB3oyiDV0Ef7v0dRBlXV_wwvBUwas05/view?usp=drive_link and https://drive.google.com/file/d/18_RG5GiH65IAI64HE5Em7yC7hbIkfBSG/view?usp=drive_link and put them in fold "dataset". To produce the result in Wikipedia when keyword universe size $\leq 3,000$, run the "preprocess.py" first, and when keyword universe size $= 5,000$, run the "preprocess.py" with line 7 changing to "kws_universe_size = 5000". The "preprocess.py" may take hours to finish. *If this is not executed, the results on Enron and Lucene can still be produced.*

Preparation for tests of Graphm: [30 human-minutes][10 computer-minutes + 0.1GB disk]: Following the instructions in https://projects.cbio.mines-paristech.fr/graphm/ to install the package of graph match algorithms. *The codes in this link may contain bugs in some machines as many packages it relies on are outdated. We note that the Graphm attack is the only attack that is relevant to this package and the results of the Graphm attack do not affect the major claims.*

(E1.1): [5 human-minutes][10 computer-minutes]: The experiment shows the distribution of quires and whether they are recovered by the simple attack.

Preparation: Create an empty fold named "results" under the main fold.

Execution: Run "show_distribution.py" with python.

Results: It will output six pictures saved at "./results" as Figure 1 in Section 4.1 and Figures 11 and 12 in Appendix B. It will also output the recovery accuracy in each quadrant.

(E1.2): [20 human-minutes + 2 compute-hours]: The experiments test Modules 1 and 2 of Jigsaw and their accuracy with different parameters.
Execution: Run "test_alpha.py" and then run "generate_test_alpha_pics.py" with python. Run "test_base_conf_rec.py".
Results: The middle results will be saved at "./results/test_alpha" and the pictures (Figure 2 and 3 in Section 5.2) will be saved as "./results/test_alpha_pic_<dataset name>". At the end of running "test_base_conf_rec.py", it will output the table of Table 2 in Section 5.2 in latex form.

(E1.3): [20 human-minutes + 100 compute-hour]: The experiments test the whole Jigsaw in different situations with different parameters.
Execution: Run "test_beta.py" and then run "generate_test_beta_pics.py" with python. Run "test_durability.py".
Results: The middle results and the generated pictures (Figure 4 Section 5.3) will be saved at "results/test_beta/". For "test_durability.py", the accuracy will be directly outputted.

(E2.1): [20 human-minutes + 300 compute-hour]: The experiments test all the considered attacks. The experiments need a long time to finish as large keyword space (such as 2000 in Enron and Lucene and 5000 in Wikipedia) is tested and the Graphm attack takes a long time to finish.
Execution: Run "test_comparison.py" and then run "generate_test_comparison_pics.py".
Results: The middle results will be saved at "results/test_comparison/<attack name>" and the generated pictures (Figure 5 Section 6.2) will be saved at "results/test_comparison".

(E2.2): [20 human-minutes + 100 compute-hour]: The experiment tests Jigsaw and IHOP with similar runtime limitations.
Execution: Run "test_compare_with_IHOP_with_limited_runtime.py" and then run "generate_compare_with_IHOP_with_limited_runtime.py".
Results: The results and the generated picture (Figure 6 Section 6.2) will be saved at "/results/test_comparison_with_IHOP_with_limited_time/".

(E3): [60 human-minutes + 300 compute-hour]: The experiments test Jigsaw, RSA, and IHOP when against different countermeasures. These experiments take a long time to finish as we test four countermeasures and use a large dataset, i.e. Wikipedia, with large keyword universe sizes, i.e. 3000 and 5000.
Execution: Run "test_against_countermeasure.py" and then run "generate_test_against_countermeasure.py".
Results: The results and pictures will be saved at "./results/test_against_countermeasures/" (Figures 7,8,9, and 10 in Sections 7.1 and 7.2 and Figures 16, 17, 18, and 19 in Appendix F).

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/usenixsec2024/.