Security Analysis of MongoDB Queryable Encryption

Zichen Gui, Kenneth G. Paterson, and Tianxin Tang, *ETH Zurich*

https://www.usenix.org/conference/usenixsecurity23/presentation/gui

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 32nd USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 32nd USENIX Security Symposium.

August 9–11, 2023 • Anaheim, CA, USA

978-1-939133-37-3

Open access to the Artifact Appendices to the Proceedings of the 32nd USENIX Security Symposium is sponsored by USENIX.

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

USENIX'23 Artifact Appendix: Security Analysis of MongoDB Queryable Encryption

Zichen Gui, Kenneth G. Paterson, and Tianxin Tang
Department of Computer Science, ETH Zurich, Zurich, Switzerland

August 9, 2023

A Artifact Appendix

A.1 Abstract

This artifact appendix is provided alongside our paper. In this appendix, we describe the two phases of our attack on MongoDB QE. The first phase is *leakage extraction* (Section 4), followed by the second phase *inference attacks* (Section 7). Due to the complications with QE, leakage extraction is not feasible on a large-scale database, and we had to work with simulated leakage in our experiments (see Appendix B). Therefore, we provide two procedures for leakage extraction, one for real leakage (E1), and the other for simulated leakage (E2). The procedures for inference attacks exploiting leakage from `queryLog` and `opLog` can be found in E3 and E4, respectively.

A.2 Description & Requirements

Requirements. Leakage simulation and inference attacks require resources 10 GB RAM, 16 GB disk, and ~ 3.3 GHz CPU on a single core (is recommend). For small-scale leakage extraction (e.g., 3K - 10K records), the same specification is adequate. However, full-scale leakage extraction involving building a database containing 3M records, requires 50 GB RAM and 600 GB disk. The instructions in this artifact only work for Linux/Unix systems. Ubuntu 22.04 is used for evaluation.

A.2.1 Security, privacy, and ethical concerns

In our experiments, we use the anonymized American Community Survey (ACS) micro data on the person level from 2012 and 2013 and the corresponding codebook, publicly available from https://www.census.gov/programs-surveys/acs/microdata.html. Our inference attacks do not in any way attempt to deanonymize this data.

A.2.2 How to access

URL. https://gitlab.com/mongodbqe/mongo/-/commit/4e9fc09377f26e1760fb510a0b998f777fd9e0f4

README.md and FAQ. We provide a `README.md` in our code repo for more comprehensive instructions. The `README.md` also contains a FAQ section to address common issues that you may encounter during evaluation.

A.2.3 Hardware dependencies

None.

A.2.4 Software dependencies

Our artifact is evaluated on MongoDB 6.0.7.

General. The links provided above direct you to the packages of the latest version only. To reproduce our results with the specific versions of the packages we used in our artifact, please refer to the following example. The instruction works specifically for Ubuntu 22.04. We provide further instruction for other operating systems later.

- `mongod` from `Archive` and `crypt_shared` (6.0.7): https://www.mongodb.com/download-center/enterprise/releases
- `libmongocrypt` (1.7.4): https://www.mongodb.com/docs/manual/core/csfle/reference/libmongocrypt
- `mongoexport` (100.7.3): https://www.mongodb.com/docs/database-tools/installation/installation/
- `mongosh` (1.10.1): https://www.mongodb.com/try/download/shell
- Python 3.10 and the python package dependencies listed in Section A.3.1.

Example for Ubuntu 22.04.

- `mongod Archive` (6.0.7): https://downloads.mongodb.com/linux/mongodb-linux-x86_64-enterprise-ubuntu2204-6.0.7.tgz
- `crypt_shared` (6.0.7): https://downloads.mongodb.com/linux/mongo_crypt_shared_v1-linux-x86_64-enterprise-ubuntu2204-6.0.7.tgz
- `libmongocrypt` (1.7.4): Please refer to `README.md` in the code repository for the commands.

- `mongoexport` (100.7.3): https://fastdl.mongodb.org/tools/db/mongodb-database-tools-ubuntu2204-x86_64-100.7.3.tgz
- `mongosh` (1.10.1): https://downloads.mongodb.com/compass/mongodb-mongosh_1.10.1_amd64.deb
- Python 3.10 and the python package dependencies listed in Section A.3.1.

Similarly, for other operating systems/architectures, you can obtain the packages of specific versions by modifying the operating system/architecture of the links above. See the download links in "General" for examples.

A.2.5 Benchmarks

We use ACS 2012 as *auxiliary data* and ACS 2013 as *recovery target* in our experiments.

ACS 2012 https://www2.census.gov/programs-surveys/acs/data/pums/2012/1-Year/csv_pus.zip

ACS 2013 https://www2.census.gov/programs-surveys/acs/data/pums/2013/1-Year/csv_pus.zip

A.3 Set-up

A.3.1 Installation

1. Download or `git clone` the repo `mongo` from the provided URL in Section A.2.2.
2. Download `csv_pus.zip` for ACS 2012, ACS 2013, respectively listed in Section A.2.5. Unzip the files and get `ss12pusa.csv`, `ss12pusb.csv`, `ss13pusa.csv`, and `ss13pusb.csv`.
3. Place `ss12pusa.csv` and `ss12pusb.csv` in `mongo/acs_data/2012_person_records`; place `ss13pusa.csv` and `ss13pusb.csv` in `mongo/acs_data/2013_person_records`.
4. Work in `mongo/src/` and create a python virtual environment and install the required packages:
   - `python3 -m pip install --user virtualenv`
   - `python3 -m venv env`
   - `source env/bin/activate`
   - `pip3 install -r requirements.txt`
5. Install `mongod`, `crypt_shared`, `libmongocrypt`, `mongoexport`, `mongosh`; urls are listed in Section A.2.4.
6. Configure the library dependencies listed in 5 in `mongo/src/parameters.py`, from line 53 to line 56.
7. Configure the default MongoDB database path at line 57 in `mongo/src/parameters.py`. Please specify a directory that does not require root access.

A.3.2 Basic Test

Working in `mongo/src`, run `python3 check_config.py` to check whether the installation and configuration listed in Section A.3.1 are complete.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): Real leakage about the data encrypted by QE can be extracted from `opLog` alone or from `queryLog` and encrypted document collection (Appendix B).

(C2): Leakage simulations for `opLog` and `queryLog` pass the correctness check, respectively, matching the real leakage (Appendix B).

(C3): The inference attack exploiting simulated query leakage (under uniform and Zipf distributions, with 100, 300, 500 queries per field) from `queryLog` achieve reasonable recovery rates (Section 7.2).

(C4): The inference attack exploiting simulated compaction leakage from `opLog` achieve reasonable recovery rates (Section 7.2).

A.4.2 Experiments

We have repeated our experiments for statistical reasons. The number of experiments can be adjusted based on time and resource constraints. Please refer to `README.md` for details. **Note that E3 and E4 can be run concurrently to reduce the waiting time.**

Sample output. Sample output for E1-E4 is provided in `mongo` repo. E.g., `E1_sample_output.txt`.

(E1): Leakage extraction (at a small scale). 5 human-minutes + 10 compute-minutes + 16 GB disk. The default number of records for this artifact is 3K. You can adjust the number of records using `--limit` argument. A full-scale leakage extraction with 3M records may take 2-4 days.
Preparation: Working in `mongo/src`, run `python3 export_acs_data_sample.py` to generate auxiliary information.
Execution: `python3 main.py` to collect `queryLog` and `opLog`, extract leakage, check its correctness for the sample dataset.
Results: Real leakage about the data can be extracted from logs successfully.

(E2): Leakage simulation. 5 human-minutes + 40 compute-minutes (depending on the number of experiments) + 16 GB disk:
Preparation: Make sure the setup stage in Section A.3 is complete. Work in `mongo/src`.
Execution: `python3 export_acs_data_simulated.py --start=0 --end=1` generates one instance of simulated leakage.
Results: Leakage is simulated for `opLog` and `queryLog` correctly.

(E3): Inference attack with simulated query leakage from `queryLog`. 5 human-minutes + 3 compute-hours (or 2 minutes if using `--fast`) + 16 GB disk:
Preparation: Work in `mongo/src_attack`. (Optional) Core attack parameters such as the number of it-

erations can be set from line 12 to line 29 of `mongo/src_attack/attack.py`.

Execution: If using simulated leakage from `queryLog` generated in E2, then run: `python3 attack.py --uniform n` for `n` queries from uniform distribution, `n in {100, 300, 500}`. Using `--zipf n` for Zipf distribution.

Results: The inference attack using the simulated query leakage achieves a reasonable recovery rate (see C3).

(E4): Inference attack with simulated compaction leakage from `opLog`. 5 human-minutes + 14 compute-hours (or 20 minutes if using `--fast`) + 16 GB disk:

Preparation: Same as in E3.

Execution: `python3 attack.py --oplog`

Results: The inference attack using simulated compaction leakage from `opLog` achieves a reasonable recovery rate (see C4).

A.5 Acknowledgement

We are grateful to the anonymous reviewers for their contributions in the artifact evaluation process. By incorporating their suggestions and refining the artifact, we have significantly enhanced its quality!

A.6 Version

Based on the LaTeX template for Artifact Evaluation V20220926. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at `https://secartifacts.github.io/usenixsec2023/`.