USENIX Security '24 Artifact Appendix: MUSES: Efficient Multi-User Searchable Encrypted Database

Tung Le
Virginia Tech

Rouzbeh Behnia
University of South Florida

Jorge Guajardo
Robert Bosch LLC – RTC

Thang Hoang
Virginia Tech

A Artifact Appendix

A.1 Abstract

Our MUSES prototype is a multi-writer encrypted search system that splits trust between multiple servers in order to efficiently hide search, result and volume patterns from a semi-honest adversary that controls all but one of the servers. MUSES system consists of an honest reader who can perform encrypted keyword search over database contributed by multiple independent writers. Also, MUSES offers writer-efficient permission revocation that can revoke search permission of the reader with small overhead on the writer. MUSES is written in `C++` for approximately 2,500 lines of code. Our experiments use AWS EC2 instances.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

Our implementation uses standard cryptographic libraries and other common ones that can be easily installed via package management tools (e.g. `apt` on Ubuntu). It neither contains any threat to the system's integrity or privacy that executes our source code nor raises any ethical concerns.

A.2.2 How to access

Our artifact is publicly available at: `https://github.com/vt-asaplab/MUSES/tree/USENIX_2024`.

A.2.3 Hardware dependencies

Amazon AWS r5n.4xlarge, 128GB RAM and 1TB disk space. Other configurations with at least 8-core CPU and 128GB RAM also satisfy hardware requirements.

A.2.4 Software dependencies

We used standard cryptographic libraries, including `OpenSSL` for IND-CPA encryption and hash functions, `NTL` (`https://libntl.org/`) and `libsecp256k1` (`https://github.com/bitcoin-core/secp256k1`) for implementing public-key encryption in our scheme, and `EMP-Toolkit` (`https://github.com/emp-toolkit`) for IKNP OT protocol. We used `libzeromq` (`https://github.com/zeromq/cppzmq`) for network communication between servers and client.

A.2.5 Benchmarks

We use Enron email dataset to choose system parameters and set up sample documents. To expedite initialization without affecting system performance evaluation, we employ randomized data to accelerate initializing database on cloud.

A.3 Set-up

A.3.1 Installation

We provide specific instructions for installing libraries and building from source code at `https://github.com/vt-asaplab/MUSES`. The file `README.md` contains detailed information, and the bash script file `auto_setup.sh` can automate installation and compilation. Note that the `EMP-Toolkit` library is already enclosed in our source code and does not need to be installed explicitly.

A.3.2 Basic Test

For evaluation, the client acts as both the reader and writers. We describe a simple test with default parameters as follows. The server application is launched with port number 12345, Bloom filter size $m = 1120$ with $N = 1024$ documents of a single writer, and search result size $n_s = 255$ as default:

- **On Server 1**: Go to folder *Server* and execute:
```
$ ./Server 1 12345
```
- **On Server 2**: Similarly, execute:
```
$ ./Server 2 12345
```
- **On Client**: Go to folder *Client* and execute:
```
$ ./Client
```

It outputs processing latency and bandwidth overhead of operations including permission revocation, document update and keyword search, in which keyword search has a preprocessing (offline) phase to prepare computational materials.

A.4 Evaluation workflow

A.4.1 Major Claims

Our MUSES scheme is a multi-writer encrypted search platform that achieves a high-level of security (hide *all* statistical information) while featuring user efficiency (minimal communication and computation overhead) as follows:

(C1): For Bloom filter size $m = 2000$, the number of documents $N = 2^{15}$, the search result size $n_s = 255$, and the number of writers n_w increasing from 25 to 150, the search latency of MUSES is 2.0s–11.1s, with the reader communication overhead is 0.4MB–2.2MB.

(C2): For the numbers of documents $N = 2^{10}$–2^{19} and the corresponding Bloom filter sizes m = 1120–3120, the end-to-end permission revocation latency of MUSES is 2.0s–183.1s, where the writer latency is 27.9ms–76.1ms, and the writer communication overhead is 4.4MB–12.1MB.

(C3): For $n_w = 25$ writers with $N = 2^{17}$ documents per writer, and Bloom filter size $m = 2520$, when increasing search result size n_s from 511 to 32,767, the end-to-end keyword search delay of MUSES increases from 3.6s to 4.9s, while the corresponding reader bandwidth overhead increases from 408.9KB to 3558.9KB.

(C4): For $n_w = 100$ writers with $N = 2^{16}$ documents, Bloom filter size $m = 2240$ and search result size $n_s = 511$, when increasing the number of servers L from 2 to 6, in which the inter-server network latency < 1ms, the search latency of MUSES is 7.4s–8.6s, while the end-to-end permission revocation delay is 16.5s–23.8s.

The memory required on each server depends on the configuration of the experiments, including the Bloom filter size m, the number of documents N, the number of writers n_w and the number of servers L, which is approximate:

$$\frac{(2m \cdot N + 1073m + 8N) \cdot n_w + 24N + 2N^2}{2^{30}} \text{ (GB)}.$$

This is the estimated memory size needed to execute both the preprocessing and online phases.

A.4.2 Experiments

(E1): Estimated time is 1 hour. Execute the following commands and replace the parameter value followed by `-w` with the corresponding number of writers $n_w \in \{25, 50, 75, 100, 125, 150\}$. For example:

- **On Server 1**:

```
$ ./Server 1 12345 -b 2000 -d 32768 -w 25
```

- **On Server 2**:

```
$ ./Server 2 12345 -b 2000 -d 32768 -w 25
```

- **On Client**:

```
$ ./Client -b 2000 -d 32768 -w 25
```

These experiments show the reader bandwidth cost and the end-to-end keyword search latency of MUSES w.r.t. varying numbers of writers n_w.

(E2): Estimated time is 0.5 hour. Execute the following commands with the parameter value followed by `-b` in $\{1120, 1440, 1800, 2240, 2800, 3120\}$ to configure Bloom filter size m and the parameter value followed by `-d` in $\{2^{10}, 2^{12}, 2^{14}, 2^{16}, 2^{18}, 2^{19}\}$ to configure the corresponding number of documents N. For example with the 1st pair of parameters $m = 1120$ and $N = 2^{10}$:

- **On Server 1**:

```
$ ./Server 1 12345 -b 1120 -d 1024
```

- **On Server 2**:

```
$ ./Server 2 12345 -b 1120 -d 1024
```

- **On Client**:

```
$ ./Client -b 1120 -d 1024
```

These experiments show the bandwidth cost and the processing latency of the writer, as well as the end-to-end permission revocation delay of MUSES w.r.t. different database sizes.

(E3): Estimated time is 1.5 hour. Execute the following commands and replace the parameter value followed by `-ns` with the search result size $(n_s + 1)$ in $\{512, 2048, 8192, 32768\}$. For instance with $n_s = 511$:

- **On Server 1**:

```
$ ./Server 1 12345 -b 2520 -d 131072 -w
25 -ns 512
```

- **On Server 2**:

```
$ ./Server 2 12345 -b 2520 -d 131072 -w
25 -ns 512
```

- **On Client**:

```
$ ./Client -b 2520 -d 131072 -w 25 -ns
512
```

These experiments show the end-to-end latency of keyword search and the reader bandwidth cost w.r.t. varying search result sizes.

(E4): Estimated time is 2 hour. We run experiments with Bloom filter size $m = 2240$, the number of documents $N = 2^{16}$ for $n_w = 100$ writers, with $2 \leq L \leq 6$ servers. Modify the configuration to the corresponding number of servers and recompile both server and client applications, then execute the commands as above with `-b 2240 -d 65536 -w 100 -ns 512`. These experiments show the effect of increasing the number of servers L to the end-to-end keyword search and permission revocation latency.

A.5 Notes on Reusability

We implement MUSES protocols including keyword search, document update and permission revocation as described in the body of the paper. We support keyword search with a small, configurable false positives rate. This is an academic proof-of-concept prototype which has not received careful code review and not ready for production use.

A.6 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/`.