USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Efficient Multi-Party Private Set Union Without Non-Collusion Assumptions

Minglang Dong, *School of Cyber Science and Technology, Shandong University; Quan Cheng Laboratory; Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University;* Cong Zhang, *Institute for Advanced Study, BNRist, Tsinghua University;* Yujie Bai and Yu Chen, *School of Cyber Science and Technology, Shandong University; Quan Cheng Laboratory; Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University*

https://www.usenix.org/conference/usenixsecurity25/presentation/dong-minglang

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 34th USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 34th USENIX Security Symposium.

August 13–15, 2025 • Seattle, WA, USA

978-1-939133-52-6

Open access to the Artifact Appendices to the Proceedings of the 34th USENIX Security Symposium is sponsored by USENIX.

USENIX Security '25 Artifact Appendix: Efficient Multi-Party Private Set Union Without Non-Collusion Assumptions

Minglang Dong[1,2,3], Cong Zhang[4], Yujie Bai[1,2,3], and Yu Chen[1,2,3](✉)

[1]School of Cyber Science and Technology, Shandong University, Qingdao 266237, China
[2]Quan Cheng Laboratory, Jinan 250103, China
[3]Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Qingdao 266237, China
[4]Institute for Advanced Study, BNRist, Tsinghua University, Beijing, China
{minglang_dong,baiyujie}@mail.sdu.edu.cn, zhangcong@mail.tsinghua.edu.cn, yuchen@sdu.edu.cn

A Artifact Appendix

A.1 Abstract

We present an open-source C++ implementation of the two MPSU protocols proposed in our paper. The artifact includes the complete source code and configurations needed to replicate our experiments in Section 7. We also provide detailed setup instructions for installing dependencies, building the project, and running the protocols. Our implementations enable evaluators to run these two protocols with customizable parameters, such as number of parties and set size, to evaluate the online and offline performance under various settings. The artifact is available, functional, and reproducible. We welcome suggestions and performance reports for future reproducibility.

A.2 Description & Requirements

Our artifact is structured as follows:

- SKMPSU/ and PKMPSU/: Complete implementations of our SK-MPSU and PK-MPSU protocols, respectively.
- README.md: Step-by-step guides for dependency installation, project compilation, and execution.
- autoTestshell/: Automated scripts to test both SK-MPSU and PK-MPSU protocols, allowing for quick and efficient evaluation across different configurations.

Our implementations support up to 10 parties with sets of up to 2^{20} items each, on machines with 128GB memory. This has been tested on Ubuntu 22.04 with g++ 11.4.0.

A.2.1 Security, privacy, and ethical concerns

None.

A.2.2 How to access

Our implementations are open-source on GitHub at https://github.com/real-world-cryptography/MPSU. Our artifact is also available at https://doi.org/10.5281/zenodo.14694832.

A.2.3 Hardware dependencies

All experiment results of our paper are obtained by running the artifact on a single server with 32-cores Intel Xeon 2.70GHz CPU and 128GB of RAM.

A.2.4 Software dependencies

Our implementations are built on Vole-PSI (https://github.com/Visa-Research/volepsi) and require additional library dependencies including OpenSSL (https://github.com/openssl/openssl) and OpenMP (https://www.openmp.org) libraries. They have been tested on Ubuntu 22.04 with installations of g++ 11.4.0 and CMake 3.22.1.

A.2.5 Benchmarks

We set the computational security parameter $\lambda = 128$ and the statistical security parameter $\sigma = 40$. We test the balanced scenario by setting all input sets to be of equal size. In our SK-MPSU, the element are 64-bit strings. In our PK-MPSU, the elements are encoded as EC points in compressed form.

A.3 Set-up

A.3.1 Installation

The complete installation guide with all dependencies and build instructions is available in our GitHub repository: https://github.com/real-world-cryptography/MPSU/blob/main/README.md

A.3.2 Basic Test

A.3.3 Test Parameters

We use the following notations for test parameters:

- `nn` $\in \{10,12,14,16,18,20\}$: Logarithm of set size $\{2^{10},2^{12},2^{14},2^{16},2^{18},2^{20}\}$.
- `nt` $\in \{1,2,4,8\}$: Number of threads
- `k`: Number of parties
- `r` $\in \{0,\ldots,$ `k` $-1\}$: Party index

A.3.4 SK-MPSU Test.

To test SK-MPSU for 3 parties with sets of 2^{12} items each in a single thread, execute the following commands in the `SKMPSU/build` directory:

- `./main -genSC -k 3 -nn 12`
- `./main -preGen -k 3 -nn 12 -r 0 & ./main -preGen -k 3 -nn 12 -r 1 & ./main -preGen -k 3 -nn 12 -r 2`
- `./main -psu -k 3 -nn 12 -r 0 & ./main -psu -k 3 -nn 12 -r 1 & ./main -psu -k 3 -nn 12 -r 2`

A.3.5 PK-MPSU Test.

To test PK-MPSU for 3 parties with sets of 2^{12} items each in a single thread, execute the following commands in the `PKMPSU/build` directory:

- `./main -preGen -k 3 -nn 12 -r 0 & ./main -preGen -k 3 -nn 12 -r 1 & ./main -preGen -k 3 -nn 12 -r 2`
- `./main -psu -k 3 -nn 12 -r 0 & ./main -psu -k 3 -nn 12 -r 1 & ./main -psu -k 3 -nn 12 -r 2`

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): In our paper, we claimed that we fully re-implement the state-of-the-art MPSU protocols in C++. This claim can be verified by running the experiments with different configurations in Section A.4.2.

(C2): Before executing the MPSU program, the required offline files must be generated correctly. Therefore, it is crucial to run the commands in the specified sequence.

A.4.2 Experiments

(E1): *[Config network settings] Config network settings using tc.*
How to: Open a terminal, and execute the following command: `tc qdisc add dev lo root netem delay 80ms rate 400Mbit`. Then, the local network is configured as 400Mbit bandwidth with 80ms latency. Evaluators can try other network settings with other parameters, e.g., 50Mbit/80ms, 1Gbit/40ms, 10Gbit/0.02ms.
Results: Execute "sudo tc qdisc show dev lo" to see if the network is configured correctly.

(E2): *[Run SK-MPSU protocol] : Benchmark the running time and communication cost of SK-MPSU protocol across diverse scenarios.*
How to: To test SK-MPSU for 3 parties with sets of 2^{12} items each in a single thread, execute the following commands in the `SKMPSU/build` directory:

- `./main -genSC -k 3 -nn 12`
- `./main -preGen -k 3 -nn 12 -r 0 &`
  `./main -preGen -k 3 -nn 12 -r 1 &`
  `./main -preGen -k 3 -nn 12 -r 2`
- `./main -psu -k 3 -nn 12 -r 0 & ./main -psu -k 3 -nn 12 -r 1 & ./main -psu -k 3 -nn 12 -r 2`

Results: The output messages during execution are as follows:

- `generate sc done`. This indicates the Share Correlation files have been generated successfully.
- `P1 offline preGen time cost = 13.142 s`. This indicates the running time of generating offline files (including Vole Triples, GMW Triples and ROT) is 13.142 s.
- `P1 offline preGen communication cost = 1.829 MB`. This indicates the communication cost of generating offline files (including Vole Triples, GMW Triples and ROT) is 1.829 MB.
- `P1 communication cost = 1.690 MB`. This indicates the online communication cost of the leader in SK-MPSU protocol is 1.690 MB.
- `end  3157.4  3157.364 **********` . This indicates the online running time of the leader in SK-MPSU protocol is 3157.4 ms.
- `Success! union size: 4098`. This indicates the SK-MPSU protocol successfully completed.

(E3): *[Run PK-MPSU protocol] : Benchmark the running time and communication cost of PK-MPSU protocol across diverse scenarios.*
How to: To test PK-MPSU for 3 parties with sets of 2^{12} items each in a single thread, execute the following commands in the `PKMPSU/build` directory:

- `./main -preGen -k 3 -nn 12 -r 0 &`
  `./main -preGen -k 3 -nn 12 -r 1 &`
  `./main -preGen -k 3 -nn 12 -r 2`
- `./main -psu -k 3 -nn 12 -r 0 & ./main -psu -k 3 -nn 12 -r 1 & ./main -psu -k 3 -nn 12 -r 2`

Results: The output messages during execution are as follows:

- `P1 offline pre time cost = 0.984 s.` This indicates the running time of generating offline files (including GMW Triples) is 0.984 s.
- `P1 offline pre communication cost = 0.334 MB`. This indicates the communication cost of generating offline files (including GMW Triples) is 0.334 MB.
- `P1 communication cost = 4.956 MB`. This indicates the online communication cost of the leader in PK-MPSU protocol is 4.956 MB.
- `end 7984.2 7983.944 **********`. This indicates the online running time of the leader in PK-MPSU protocol is 7984.2 ms.
- `Success! union size: 4098`. This indicates the PK-MPSU protocol successfully completed.

A.5 Notes on Experimental Results

Our latest local tests demonstrate a minor improvement in the online performance (computation and communication) of our implementations compared to the results reported in the paper. This difference stems from an update to Vole-PSI.

We retained the original reported data in the paper primarily because we became aware of this update after the camera-ready submission deadline. Additionally, the performance improvement is minor, and future updates to Vole-PSI or other library dependencies may continue to introduce slight variations in results compared to those in the paper.

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/usenixsec2025/.