Bloom Filter-Based MPSI

Weekly Progress Meeting 27 Nov 2025

Andra Alăzăroaie

Supervisor: Lilika Markatou

Daily Supervisor: Tjitske Koster

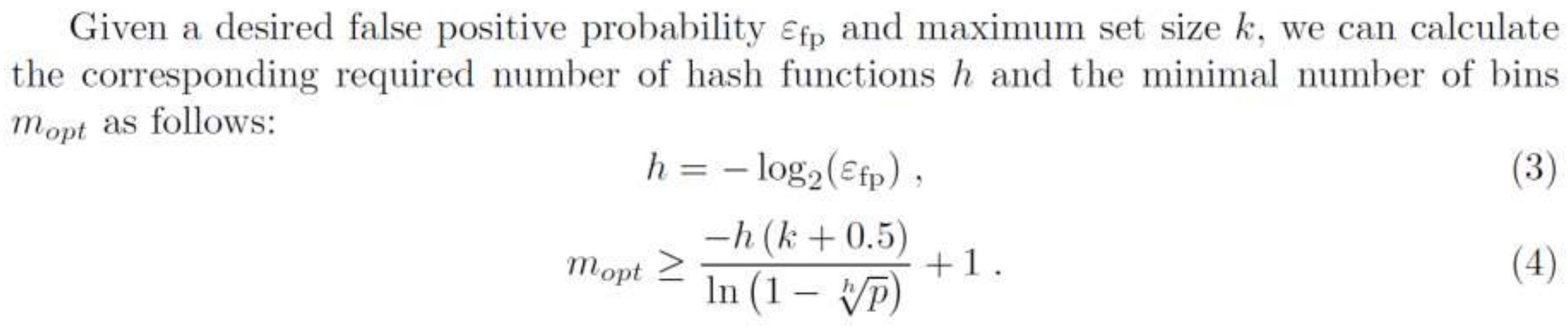
27-11-2025

Progress

- Read [1], [2], [3]
    - [1] fully, but there are still parts that I have to revise
    - [2] and [3] partially
- Starting from Jelle’s Rust code for the attack on Bloom Filter-based PSI
    - Translated the Bloom Filter structure to C++
    - Implemented some basic operations and benchmarking

Progress

- The code is based on [1]. We choose set size K and desired false positive probability e. Based on this, we derive the optimal filter size m and number of hash functions h**.**
- For benchmarking, each of the parties generates a dataset. They then construct their local Bloom filters by hashing these elements and setting the corresponding bits.
- To find the intersection, one of the parties aggregates the filters using bitwise AND operations (simulating a start topology).



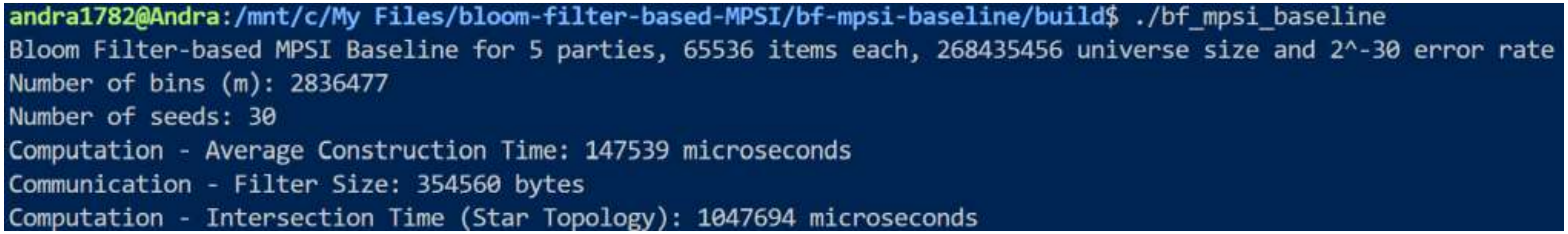
Given a desired false positive probability $\varepsilon_{\mathrm{fp}}$ and maximum set size k, we can calculate the corresponding required number of hash functions h and the minimal number of bins m_{opt} as follows:

$$h = -\log_2(\varepsilon_{\mathrm{fp}}) \, , \tag{3}$$

$$m_{opt} \geq \frac{-h\,(k+0.5)}{\ln\left(1-\sqrt[h]{p}\right)} + 1 \, . \tag{4}$$

Progress

- Computation measured for:
  - Construction of Bloom filters based on parties' sets.
  - Computing the intersection for a start topology, where the 1st party is the leader and everyone sends their set to them.
- Communication measured as filter size in bytes (there is no network communication in this implementation).



```
andra1782@Andra:/mnt/c/My Files/bloom-filter-based-MPSI/bf-mpsi-baseline/build$ ./bf_mpsi_baseline
Bloom Filter-based MPSI Baseline for 5 parties, 65536 items each, 268435456 universe size and 2^-30 error rate
Number of bins (m): 2836477
Number of seeds: 30
Computation - Average Construction Time: 147539 microseconds
Communication - Filter Size: 354560 bytes
Computation - Intersection Time (Star Topology): 1047694 microseconds
```

Challenges

- Remembering C++.
- Learning a bit of Rust, to be able to translate Jelle’s Bloom Filter code to C++.
- Reading through the details of the attack paper, proofs are difficult to understand and it is going slow.

Next Week

- Read [4].
- Understand Jelle’s C++ code for [2].
- Understand [3]’s construction well.
- Start an implementation of an existing protocol.
  - [3]’s idea is based on [2], but uses ElGamal instead of Paillier. I am thinking it would be a good candidate for a better baseline.
- Continue on going though the attack and proofs in [1].

Bibliography

[1] Jelle Vos, Jorrit van Assen, Tjitske Koster, Evangelia Anna Markatou, and Zekeriya Erkin. On the insecurity of bloom filter-based private set intersections. Cryptology ePrint Archive, 2024.

[2] Aslı Bay, Zekeriya Erkin, Jaap-Henk Hoepman, Simona Samardjiska, and Jelle Vos. Practical multi-party private set intersection protocols. IEEE Transactions on Information Forensics and Security, 17:1–15, 2021.

[3] Ou Ruan, Changwang Yan, Jing Zhou, and Chaohao Ai. A practical multiparty private set intersection protocol based on bloom filters for unbalanced scenarios. Applied Sciences, 13(24):13215, 2023.

[4] Ou Ruan and Chaohao Ai. An efficient multi-party private set intersection protocols based on bloom filter. In Second International Conference on Algorithms, Microchips, and Network Applications (AMNA 2023), volume 12635, pages 282–287. SPIE, 2023.

TU Delft

Thank you!

28-11-2025