Bloom Filter-Based MPSI

Weekly Progress Meeting 22 Jan 2026

Andra Alăzăroaie

Supervisor: Lilika Markatou

Daily Supervisor: Tjitske Koster

22-01-2026

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.

[5] Jelle Vos, Mauro Conti, and Zekeriya Erkin. Fast multi-party private set operations in the star topology from secure ands and ors. Cryptology ePrint Archive, 2022.

[6] Vladimir Kolesnikov, Naor Matania, Benny Pinkas, Mike Rosulek, and Ni Trieu. Practical multi-party private set intersection from symmetric-key techniques.In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pages 1257–1272, 2017.

[7] Alireza Kavousi, Javad Mohajeri, and Mahmoud Salmasizadeh. Efficient scalable multi-party private set intersection using oblivious prf. In International Workshop on Security and Trust Management, pages 81–99. Springer, 2021.

Progress

From the plan last week:

- Continue writing:
  - Why the attack in [1] works on my implementation.
  - Parts of introduction and related work.
- Get started on mitigations:
  - Read [6] and [7], learn about oblivious PRFs.

Progress

- Explaining why [3] is vulnerable to the attack in [1]
  - through explaining why it reduces to the idealized behavior π_{BF} of a BF-based MPSI protocol
  - (not shown here) and explaining how [1] distinguishes π_{BF} from an idealized approximate MPSI behavior

Vos et al. classify the protocol by Ruan et al. as an instantiation of the abstract Bloom filter protocol Π_{BF}. While Ruan et al. use cryptographic primitives (threshold ElGamal encryption and Shamir secret sharing), the logic of the intersection operation remains the same as the unencrypted operations in Π_{BF}.

In the abstract functionality Π_{BF}, the output is the subset of the leader's elements that return "True" when queried against the combined Bloom filter of all other parties:

$$\text{Output}_{\Pi_{BF}} = \{x \in S_{\text{server}} \mid \text{contains}(\bigwedge_{i=1}^{t-1} \hat{X}_i, x)\} \tag{2.4}$$

Ruan's protocol implements this logic adding encryptions. The server computes a combined ciphertext c_j for an element x by homomorphically multiplying the responses from all clients:

$$c_j = \left(\prod_{i=1}^{t-1} c_{j,i}\right) \times w_j \tag{2.5}$$

where w_j is a blinded version of the server's element and $c_{j,i}$ is the encrypted value from client i at the hashed indices of x. The protocol considers an element in the intersection if and only if the decryption of c_j equals the blinded element w_j.

In Ruan's construction, the '0' bits in the Bloom filter are replaced by random values before encryption, while '1' bits are encrypted as they are. Due to the multiplicative homomorphic property of ElGamal, the product $\prod c_{j,i}$ will decrypt to 1 if and only if every component $c_{j,i}$ is an encryption of 1. If even one client has a randomized '0' at the queried index, the product becomes an encryption of a random value. Therefore, the cryptographic check in Ruan is equivalent to the boolean AND operation in Π_{BF}:

$$\text{Dec}(c_j) = w_j \iff \forall i : \text{contains}(\hat{X}_i, x) \tag{2.6}$$

Hence, Ruan's protocol outputs the same set of elements as Π_{BF}, including all false positives.

Progress

- Would be even better to simulate [3] using π_{BF}?

Vos et al. classify the protocol by Ruan et al. as an instantiation of the abstract Bloom filter protocol Π_{BF}. While Ruan et al. use cryptographic primitives (threshold ElGamal encryption and Shamir secret sharing), the logic of the intersection operation remains the same as the unencrypted operations in Π_{BF}.

In the abstract functionality Π_{BF}, the output is the subset of the leader's elements that return "True" when queried against the combined Bloom filter of all other parties:

$$\text{Output}_{\Pi_{BF}} = \{x \in S_{\text{server}} \mid \text{contains}(\bigwedge_{i=1}^{t-1} \hat{X}_i, x)\} \tag{2.4}$$

Ruan's protocol implements this logic adding encryptions. The server computes a combined ciphertext c_j for an element x by homomorphically multiplying the responses from all clients:

$$c_j = \left(\prod_{i=1}^{t-1} c_{j,i}\right) \times w_j \tag{2.5}$$

where w_j is a blinded version of the server's element and $c_{j,i}$ is the encrypted value from client i at the hashed indices of x. The protocol considers an element in the intersection if and only if the decryption of c_j equals the blinded element w_j.

In Ruan's construction, the '0' bits in the Bloom filter are replaced by random values before encryption, while '1' bits are encrypted as they are. Due to the multiplicative homomorphic property of ElGamal, the product $\prod c_{j,i}$ will decrypt to 1 if and only if every component $c_{j,i}$ is an encryption of 1. If even one client has a randomized '0' at the queried index, the product becomes an encryption of a random value. Therefore, the cryptographic check in Ruan is equivalent to the boolean AND operation in Π_{BF}:

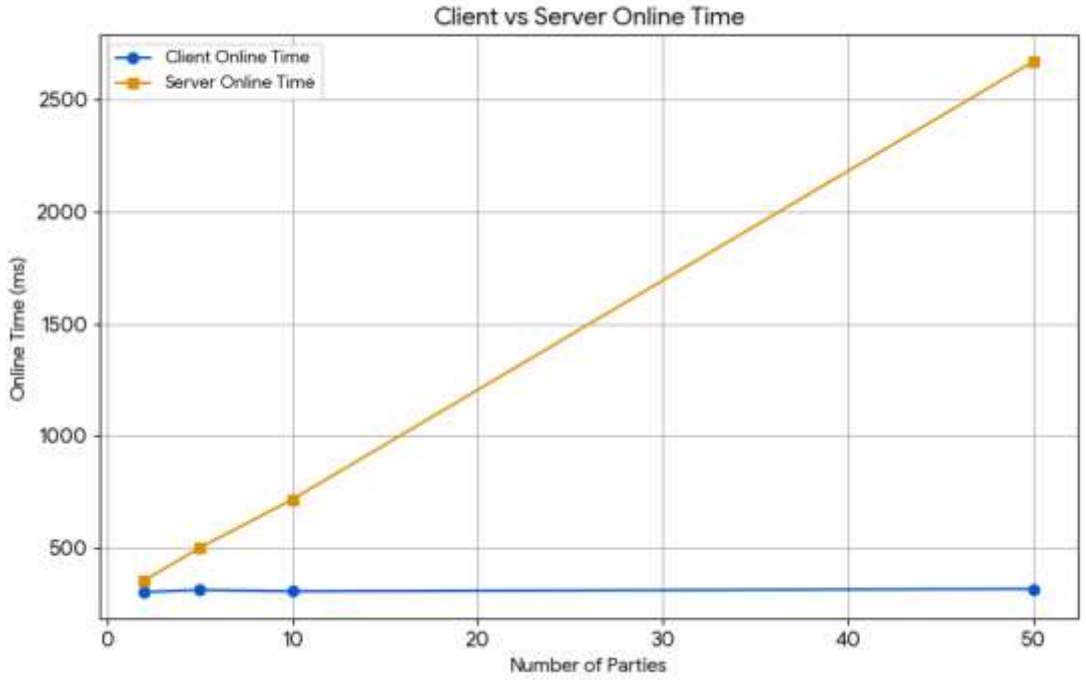
$$\text{Dec}(c_j) = w_j \iff \forall i : \text{contains}(\hat{X}_i, x) \tag{2.6}$$

Hence, Ruan's protocol outputs the same set of elements as Π_{BF}, including all false positives.

Progress

Computation on the online stage of [3]:

- Client's online time does not increase with the number of parties
- Matches the results from the paper



Challenges

- When tweaking the false positive probability (which determines the number of bins in the Bloom filter and the number of hash functions) in order to see false positives appearing in the final intersection, one issue was setting the universe size for the experiments in order to have a non-empty intersection as the number of parties grew.
- When the server's set is larger than the clients', is it correct to calculate the number of bins m using the server's set size? The formula used in the paper is $-\text{size} * \log_2 e * \log_2 P_{err}$
  - This influences the number of FPs given by the experiments: for client's sets of 2^8 elements and server set of 2^{10} elements, setting the FP probability to 2^{-7} gives false positives when m is calculated using the clients' set size, but not when using the server's set size. The domain size might be relevant here too.
- In [2]'s experiments, the FP probability is set to only 2^{-7}, while in [3]'s, it is 2^{-30}. [3]'s experiments include a comparison in computation costs with [2], showing that it performs better (since ElGamal is faster than Paillier). How do we show the mitigation if the experiments don't reveal any false positives when using 2^{-30}?

Next Week

- Continue writing and experimenting.
- Mitigation OPRF:
  - Read [6] and [7], learn about oblivious PRFs.

Thank you!

22-01-2026