A Bridge Between Private Set Intersection Algorithms

Transforming MPSI into OT-MPSI

Author: Maciej Kozik (M.Kozik@student.tudelft.nl)
Responsible professor: Dr. Zeki Erkin
Examiner: Dr. Merve Gürel

Affiliation
EEMCS, Delft University of Technology, The Netherlands

Introduction

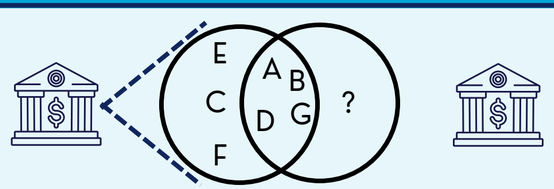
Multiparty Private Set Intersection (MPSI) allows parties to compute the intersection of confidential datasets without revealing the sets themselves. It's a promising direction in financial crime detection as it allows parties to privately compare sets of suspected clients.

Standard MPSI can be too strict - there's good reason to suspect a client that is flagged by all parties except one, yet MPSI will not include them.

Threshold Multiparty Private Set Intersection (T-MPSI) - it's a relaxation that includes an item in the intersection only if it's in at least T private sets. It mitigates the issue highlighted above.

Both protocols are implemented from scratch by the current algorithms, with no clear bridge between them.

Since MPSI algorithms offer better runtime complexities in general, an efficient and secure transformation from MPSI to T-MPSI would be desirable.



The naive solution [1] works by running MPSI between all subsets of parties of size T. It is highly inefficient, and reveals both the item counts and their owners in the result. We introduce a transformation from an MPSI by Bay et al. [2] that hides the owners by utilising dummy Bloom Filters and Mental Poker [3]. We still reveal item counts, hence to differentiate it from strict T-MPSI, we call our protocol Over-Threshold-MPSI (OT-MPSI).

Research Question

The objective of this project is to answer the following question:

How to securely transform an MPSI protocol to an OT-MPSI protocol with an arbitrary threshold in a semi-honest setting?

Preliminaries

Bloom Filter [4]

A Bloom Filter is a space efficient set representation. It is an array with all entries initially equal to 0. When one wants to add an item to the set, they hash it by a number of hash functions to get the indices of the array that are then all set to 1. Checking if an item is in the set requires checking if those indices are all set to 1.

Homomorphic Encryption

An encryption scheme is homomorphic, if it supports operations on ciphertexts that correspond to operations on plaintext. E.g. an Additively Homomorphic Scheme support addition of encrypted plaintexts and multiplication by a scalar.

MPSI by Bay et al. [2]

We differentiate two roles in this protocol:

- server: a single party that coordinates the protocol; only they get the output, and with only the items that belong to their set
- clients: all other parties

The protocol works as follows:

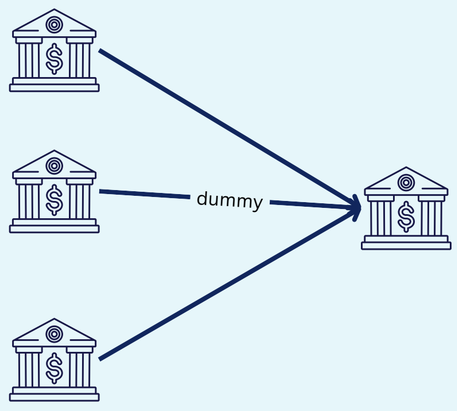
1. Clients compute the Bloom Filters corresponding to their sets, flip all bits in them, encrypt them bitwise by an Additively Homomorphic Scheme and send them to the server.
2. The server, for each of the items in its set, homomorphically adds all corresponding indices from all encrypted Bloom Filters.
3. All parties collaborate to decrypt the added values by the server in such a way that they are 0 only if they were encryptions of 0, otherwise they are random.
4. The server, based on the decrypted values, adds the items to the intersection.

Masking the Participants

The security property of T-MPSI prohibits the parties from knowing which other parties contributed an item to the intersection. We present an approach to securely run the MPSI algorithm by Bay et al. [2] multiple times between different subsets of parties to achieve OT-MPSI. We operate in the "individual" setting, meaning only one party (server) gets the result. We denote the number of parties as t, and the threshold as T.

We use dummy Bloom Filters, meaning Bloom Filters that have all entries equal to 1. If a party is not in the current subset, they send a dummy Bloom Filter instead of their actual one.

Note that, as long as the server does not know the current subset, by the security guarantees of the protocol by Bay et al., they have no way of figuring out which parties sent an actual Bloom Filter and which ones a dummy.

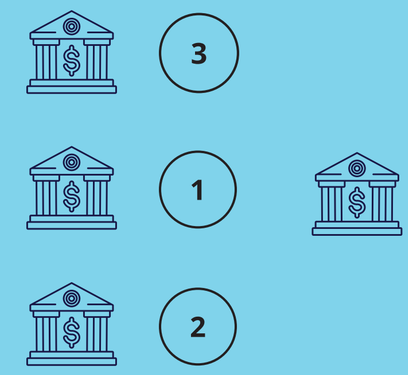


Achieving Collusion Resistance

We assume a setting, where, although parties stick to the protocol, they can communicate between each other (collude) to gain information they should not obtain. To make sure that no coalition of colluding parties gains an advantage, we leverage Mental Poker [3]. It solves the problem of shuffling and dealing the cards over the network.

In our solution, the clients play Mental Poker to deal unique identifiers between them. When the server broadcasts the current subset of parties to the clients, they use their confidential identifiers to judge if they are in the subset. Note that, despite the server communicating the subsets, this approach successfully makes sure that the server does not know the participants of each intermediate MPSI, as they do not know the random permutation of the identifiers.

This approach achieves security against collusion since the Mental Poker protocol ensures no client knows the identifier of another client.



Results

T=t-1

For this threshold, we see in Figure 1 that our solution and the naive one perform similarly. Since the underlying MPSI's complexity scales linearly with the number of parties, the overhead is negligible as our solution runs it for each intermediate MPSI for one more party than the naive one.

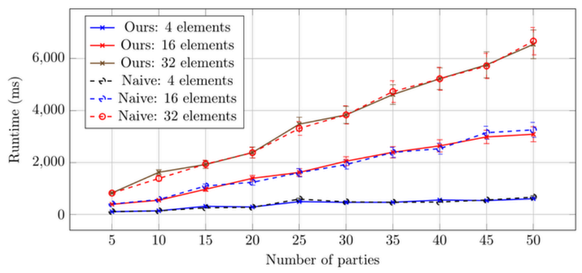


Figure 1: Runtime comparison between our OT-MPSI protocol and the naive protocol at T = t − 1, for different numbers of parties and set sizes.

T=t/2

For this threshold, we see in Figure 2 that the two solutions diverge in terms of runtime. This is explained by the fact that we run the intermediate MPSIs for all parties, whereas the naive runs it for only half of them.
This threshold is also the worst case in terms of complexity for our protocol. Figure 2 showcases the exponential runtime.

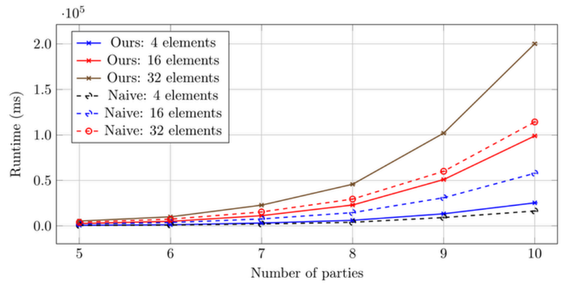


Figure 2: Runtime comparison between our OT-MPSI protocol and the naive protocol at T = t/2, for different numbers of parties and set sizes.

Conclusions and Future Work

Since our transformation's computational complexity scales exponentially with the number of parties in the worst case, it is not a viable solution for real-world applications.

However, it successfully hides the intermediate participants, hence masking the owners of the items in the final intersection. It is an improvement over the naive solution in terms of security. Therefore, we believe it is a step towards finding a practical transformation from MPSI to T-MPSI.

Future work:

- Before the efficiency problem is solved, this transformation will remain impractical. Future research should be conducted to improve the complexity.
- Our solution discloses the count for each item in the final intersection. This may be treated as a security concern. Therefore, a transformation that hides this data would also be desirable.
- We use a specific MPSI in our transformation. Future research should focus on making it work for a generic MPSI.

References

[1] C. Guan, "A comparative study of threshold multiparty private set intersection protocols," Master's thesis, Delft University of Technology, Delft, The Netherlands, 2024. [Online]. Available:https://resolver.tudelft.nl/uuid:425aff9c-a92c-4272-8674-33630fdf062a
[2] A. Bay, Z. Erkin, J. Hoepman, S. Samardjiska, and J. Vos, "Practical multi-party private set intersection protocols," IEEE Trans. Inf. Forensics Secur., vol. 17, pp. 1–15, 2022. [Online]. Available: https://doi.org/10.1109/TIFS.2021.3118879
[3] A. Shamir, R. L. Rivest, and L. M. Adleman, "Mental poker," in The Mathematical Gardner, D. A. Klarner, Ed. Boston, MA: Springer, 1981, pp. 37–43. [Online]. Available: https://doi.org/10.1007/978-1-4684-6686-7_5
[4] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," Commun. ACM, vol. 13, no. 7, pp. 422–426, 1970. [Online]. Available: https://doi.org/10.1145/362686.362692