Multiparty Private Set Intersection and Beyond

Speaker: Ni Trieu

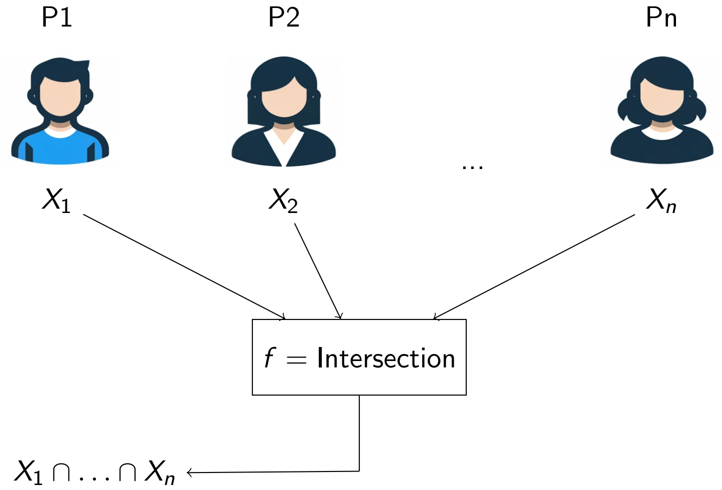
Presented at NIST WPEC, September 2024

Joint work with: Vladimir Kolesnikov, Naor Matania, Benny Pinkas, Mike Rosulek, Ofri Nevo, Avishay Yanai, Jiahui Gao

Arizona State University

Outline

Multi-party Private Set Intersection(MPSI)

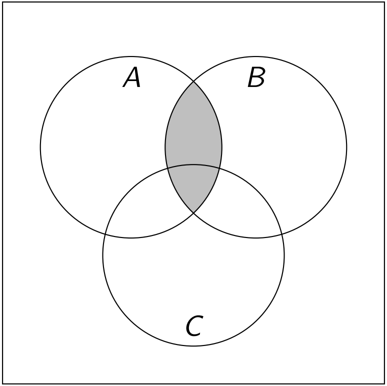


$\Rightarrow$ MPSI does not reveal any items beyond intersection.

$\Rightarrow$ Circuit MPSI: Computing a function on intersection items.

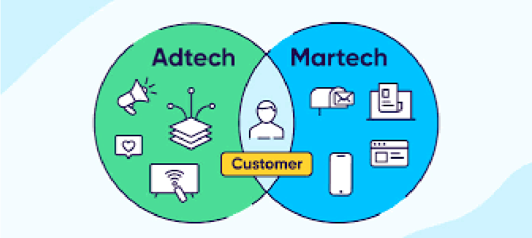
MPSI – Challenges

- The partial intersection must remain hidden from all parties
- Colluding parties should gain no additional information

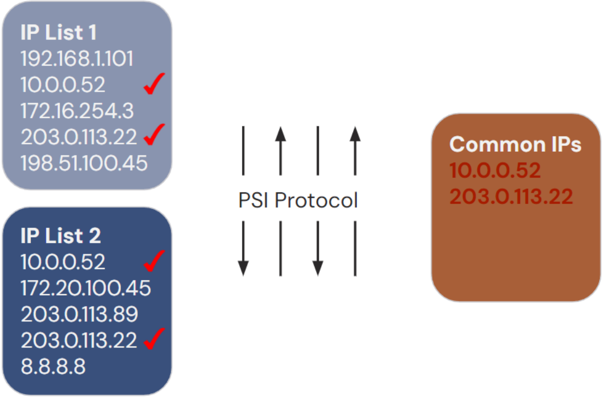


MPSI Applications

- Ad Tech and Marketing: In targeted advertising, multiple advertisers or platforms may wish to compare customer lists to optimize campaigns

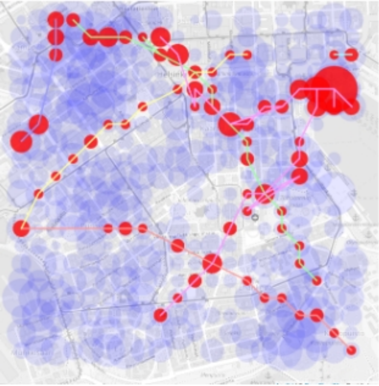


- Collaborative Threat Detection: Different organizations (such as banks or cybersecurity firms) may want to identify common security threats, such as malware signatures or suspicious IP addresses, without revealing their entire dataset to others.

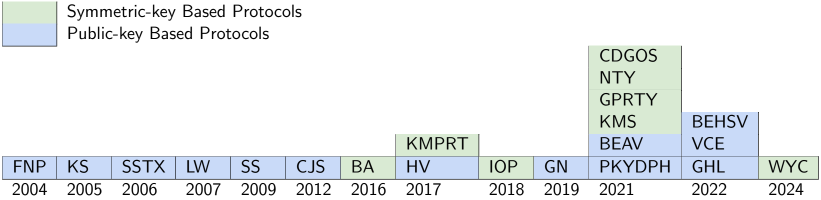


MPSI Applications

- Heatmap Detection: the generation of heatmaps from sensitive data while ensuring that individual data points remain confidential.



MPSI History



MPSI Constructions

Public-key based Construction

- Polynomial roots.
- Polynomial with payloads
- Bit set for small universe
- Bloom filter

Symmetric-key based Construction

- Sorted multisets
- Oblivious key-value store (OKVS), including
  - Garble Bloom filter
  - Programmable OPRF

MPSI Constructions

Public-key based Construction

- Polynomial roots.
- Polynomial with payloads
- Bit set for small universe
- Bloom filter

Symmetric-key based Construction

- Sorted multisets
- Oblivious key-value store (OKVS), including
  - Garble Bloom filter
  - Programmable OPRF

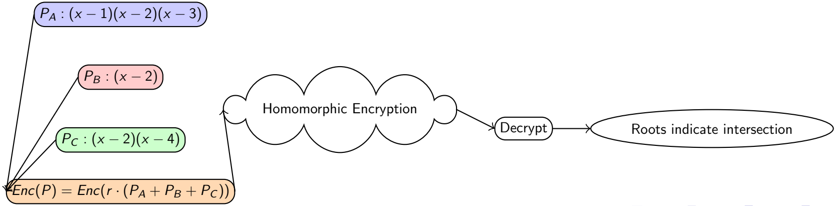
MPSI Constructions

Public-Key Based Constructions

Public-Key Based Construction

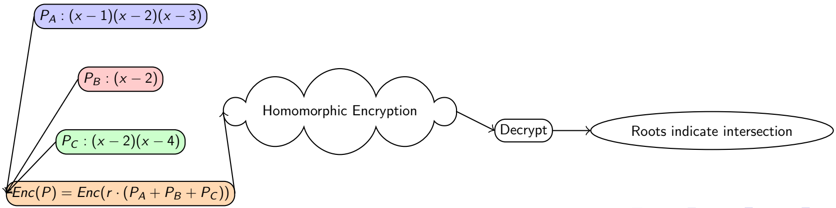
Polynomial-Based MPSI [KS05]

- A common approach for public-key based constructions
  $\Rightarrow$ Computational expensive
- Key ideas:
  - Represent each set as a polynomial whose roots are the items.
  - Use homomorphic encryption to eliminate non-intersection items from the roots of the global polynomial.
  - Decrypt the encrypted polynomial to extract the intersection items from the roots.



Polynomial-Based MPSI [KS05]

- A common approach for public-key based constructions
  $\Rightarrow$ Computational expensive
- Key ideas:
  - Represent each set as a polynomial whose roots are the items.
  - Use homomorphic encryption to eliminate non-intersection items from the roots of the global polynomial.
  - Decrypt the encrypted polynomial to extract the intersection items from the roots.



Public-Key Based Construction

Bloom Filter-Based MPSI [VCE22]

- Applicable only for small input domains.
- Key ideas:
  - Insert item sets into a Bloom filter (BF).
  - Obliviously compute the BF for the intersection set.
  - Check each element against the obtained BF to learn the intersection.

Public-Key Based Construction

Bloom Filter-Based MPSI [VCE22]

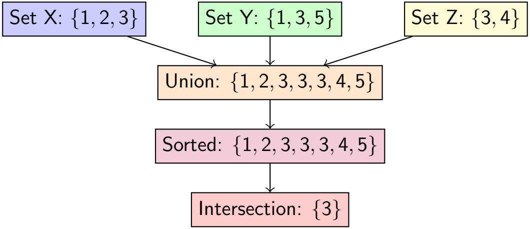
- Applicable only for small input domains.
- Key ideas:
  - Insert item sets into a Bloom filter (BF).
  - Obliviously compute the BF for the intersection set.
  - Check each element against the obtained BF to learn the intersection.

MPSI Constructions

Symmetric-Key Based Constructions

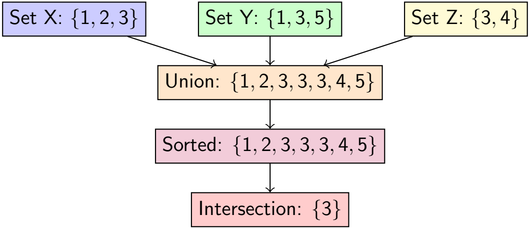
Sorting Based MPSI [BA12]

- Utilizes MPC to implement the framework
  $\Rightarrow$ Communication-expensive
- Key ideas:
  - Obliviously combine and sort the union of sets.
  - Eliminate non-intersection items.



Sorting Based MPSI [BA12]

- Utilizes MPC to implement the framework
  $\Rightarrow$ Communication-expensive
- Key ideas:
  - Obliviously combine and sort the union of sets.
  - Eliminate non-intersection items.



OKVS-Based Construction [KMP+17, GPR+21, NTY21, CDG+21, WYC24]

- Among the fastest protocols available.
- Key Concepts:
  - Use OKVS to encode input sets.
  - Generate zero shares for each item.
  - If all parties have the same item, the corresponding shares remain the share of zero after the OKVS executions.

OKVS-Based Construction [KMP+17, GPR+21, NTY21, CDG+21, WYC24]

- Among the fastest protocols available.
- Key Concepts:
  - Use OKVS to encode input sets.
  - Generate zero shares for each item.
  - If all parties have the same item, the corresponding shares remain the share of zero after the OKVS executions.

Symmetric-Key Based Construction

OKVS-Based Constructions

- Initial/Baseline construction: [KMP+17] (in the semi-honest setting)
- Enhanced constructions:
  - [GPR+21]: Proposes a malicious MPSI
  - [NTY21]: Considers a subset of t corrupted malicious parties
  - [CDG+21]: Extends to circuit MPSI in the semi-honest setting
  - [WYC24]: Proposes an efficient MPSI using O-Ring and K-Star communication in the semihonest setting

Symmetric-Key Based Construction

OKVS-Based Constructions

- Initial/Baseline construction: [KMP+17] (in the semi-honest setting)
- Enhanced constructions:
  - [GPR+21]: Proposes a malicious MPSI
  - [NTY21]: Considers a subset of t corrupted malicious parties
  - [CDG+21]: Extends to circuit MPSI in the semi-honest setting
  - [WYC24]: Proposes an efficient MPSI using O-Ring and K-Star communication in the semihonest setting

Symmetric-Key Based Construction

OKVS-Based Construction

Preliminary: OKVS [GPR+21]

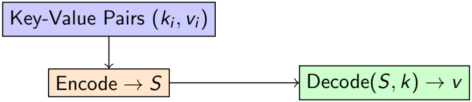
Definition

An Oblivious Key-Value Store (OKVS) consists of two algorithms:

- Encode: Takes a list of key-value pairs (k_i, v_i) and outputs an abstract data structure S.
- Decode: Takes the data structure S and a key k as input, returning an output. If called with k_i (used to generate S), it returns the corresponding value v_i.

Key Property

The fundamental property of an OKVS is that the structure S hides the keys k_i when the values v_i are chosen randomly.



v $= v_i$ if $k = k_i$,
otherwise v is random

Preliminary: OKVS [GPR+21]

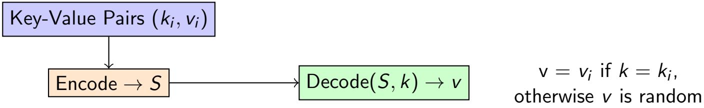
Definition

An Oblivious Key-Value Store (OKVS) consists of two algorithms:

- Encode: Takes a list of key-value pairs (k_i, v_i) and outputs an abstract data structure S.
- Decode: Takes the data structure S and a key k as input, returning an output. If called with k_i (used to generate S), it returns the corresponding value v_i.

Key Property

The fundamental property of an OKVS is that the structure S hides the keys k_i when the values v_i are chosen randomly.



Preliminary: Zero Sharing

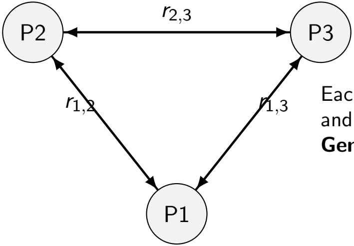
Functionality

- For n parties, generate random values $s_1, \ldots, s_n$ such that:

$$\bigoplus_{i=1}^{n} s_i = 0$$

- Output the share s_i to party P_i.

Construction



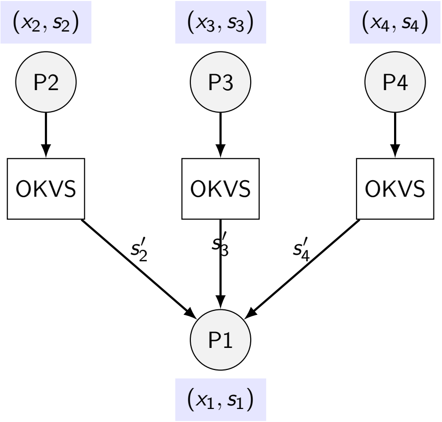
Each party P_i picks random seeds $r_{i,j}$ for $j = i+1, \ldots, n$ and sends seed $r_{i,j}$ to P_j.
General Formula:

$$S_i(ind) = \left(\bigoplus_{j=1}^{i-1} F(r_{j,i}, ind) \right) \oplus \left(\bigoplus_{j=i+1}^{n} F(r_{i,j}, ind) \right)$$

[KMP+17, GPR+21]'s Construction

[KMP+17, GPR+21]'s Construction

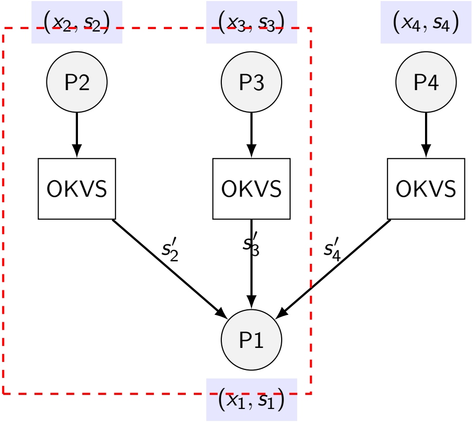
- Considering a simple case where each party has only 1 item
-
-



- s_i is the zero share i.e., $s_1 \oplus ... \oplus s_4 = 0$
- P_1 and P_i execute OKVS[a]:
  - $P_{i \in [2,4]}$ inputs (x_i, s_i)
  - P_1 inputs x_1
  - P_1 receives the s'_i
- P_1 checks whether $s_1 \oplus s'_2 \oplus s'_3 \oplus s'_4 = 0$

[a] Indeed, [KMP+17] uses Oblivious Programmable PRF (OPPRF) instead of OKVS, both functionalities are "somewhat" similar

- Considering a simple case where each party has only 1 item
- When P_1, P_2, P_3 collude,
-

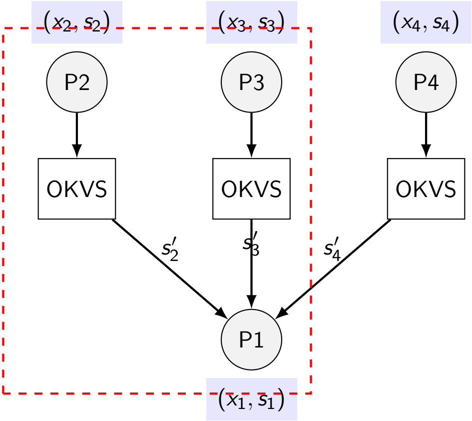


- s_i is the zero share i.e., $s_1 \oplus \ldots \oplus s_4 = 0$
- P_1 and P_i execute OKVS[a]:
  - $P_{i \in [2,4]}$ inputs (x_i, s_i)
  - P_1 inputs x_1
  - P_1 receives the s'_i
- P_1 checks whether $s_1 \oplus s'_2 \oplus s'_3 \oplus s'_4 = 0$

[a] Indeed, [KMP+17] uses Oblivious Programmable PRF (OPPRF) instead of OKVS, both functionalities are "somewhat" similar

[KMP+17, GPR+21]'s Construction

- Considering a simple case where each party has only 1 item
- When P_1, P_2, P_3 collude, they can locally compute s_4
  $\Rightarrow$ learn whether P_4 has x_1 or not
-

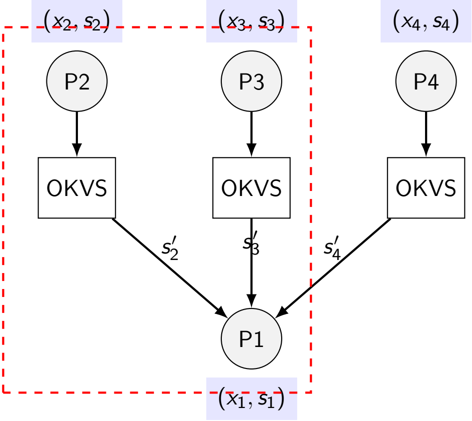


- s_i is the zero share i.e., $s_1 \oplus \ldots \oplus s_4 = 0$
- P_1 and P_i execute OKVS[a]:
  - $P_{i \in [2,4]}$ inputs (x_i, s_i)
  - P_1 inputs x_1
  - P_1 receives the s'_i
- P_1 checks whether $s_1 \oplus s'_2 \oplus s'_3 \oplus s'_4 = 0$

[a] Indeed, [KMP+17] uses Oblivious Programmable PRF (OPPRF) instead of OKVS, both functionalities are "somewhat" similar

[KMP+17, GPR+21]'s Construction

- Considering a simple case where each party has only 1 item
- When P_1, P_2, P_3 collude, they can locally compute s_4
  $\Rightarrow$ learn whether P_4 has x_1 or not
- This leakage is acceptable in a slightly weaker variant of security (aka. augmented semi-honest model)



- s_i is the zero share i.e., $s_1 \oplus \ldots \oplus s_4 = 0$
- P_1 and P_i execute OKVS[a]:
  - $P_{i\in[2,4]}$ inputs (x_i, s_i)
  - P_1 inputs x_1
  - P_1 receives the s'_i
- P_1 checks whether $s_1 \oplus s'_2 \oplus s'_3 \oplus s'_4 = 0$

[a] Indeed, [KMP+17] uses Oblivious Programmable PRF (OPPRF) instead of OKVS, both functionalities are "somewhat" similar

[KMP+17, GPR+21]'s Construction

- Augmented semi-honest model: the corrupt parties are assumed to run the protocol honestly, but the simulator in the ideal world is allowed to change the inputs of corrupt parties.
- [GPR+21] shows that the augmented semi-honest protocol is secure against malicious adversaries despite not being secure in the semi-honest model.
- To achieve semi-honest protocol, we introduce Conditional Zero Sharing: If all parties hold the same value x, they obtain the "correct" zero shares.

[KMP+17, GPR+21]'s Construction

- Augmented semi-honest model: the corrupt parties are assumed to run the protocol honestly, but the simulator in the ideal world is allowed to change the inputs of corrupt parties.
- [GPR+21] shows that the augmented semi-honest protocol is secure against malicious adversaries despite not being secure in the semi-honest model.
- To achieve semi-honest protocol, we introduce Conditional Zero Sharing: If all parties hold the same value x, they obtain the "correct" zero shares.

[KMP$^+$17, GPR$^+$21]'s Construction

- Augmented semi-honest model: the corrupt parties are assumed to run the protocol honestly, but the simulator in the ideal world is allowed to change the inputs of corrupt parties.
- [GPR$^+$21] shows that the augmented semi-honest protocol is secure against malicious adversaries despite not being secure in the semi-honest model.
- To achieve semi-honest protocol, we introduce Conditional Zero Sharing: If all parties hold the same value x, they obtain the "correct" zero shares.

[KMP$^+$17]'s Construction – Conditional Zero Sharing

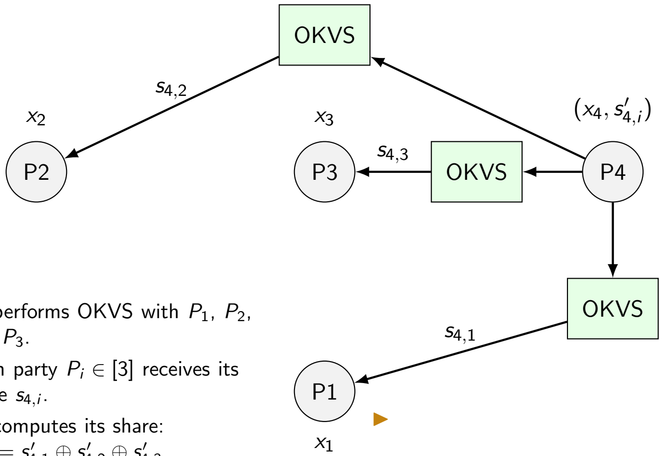
Functionality

- Each of the n parties provides an input x_i.
- Generates random values $s_1, \ldots, s_n$ such that if all x_i are equal, the following holds:

$$\bigoplus_{i=1}^{n} s_i = 0$$

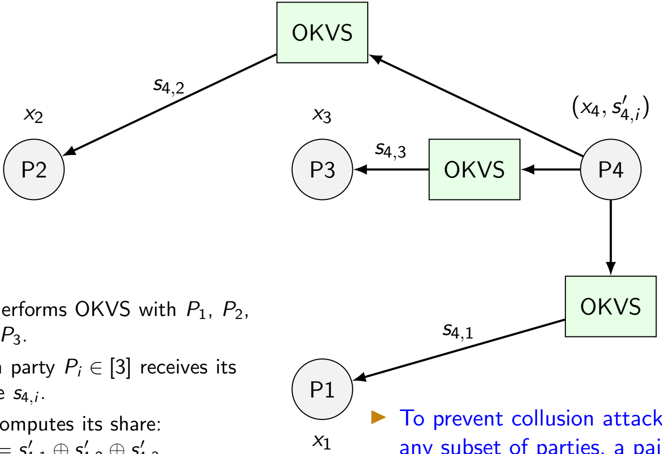
- Gives P_i its corresponding share s_i.

[KMP+17]'s Construction – Conditional Zero Sharing



- P_4 performs OKVS with P_1, P_2, and P_3.
- Each party $P_i \in [3]$ receives its share $s_{4,i}$.
- P_4 computes its share: $s_{4,4} = s'_{4,1} \oplus s'_{4,2} \oplus s'_{4,3}$.
- If all values of x_i are the same, then $s_{4,4} \oplus s_{4,1} \oplus s_{4,2} \oplus s_{4,3} = 0$.

[KMP+17]'s Construction – Conditional Zero Sharing

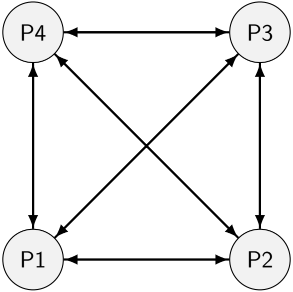


- P_4 performs OKVS with P_1, P_2, and P_3.
- Each party $P_i \in [3]$ receives its share $s_{4,i}$.
- P_4 computes its share: $s_{4,4} = s'_{4,1} \oplus s'_{4,2} \oplus s'_{4,3}$.
- If all values of x_i are the same, then $s_{4,4} \oplus s_{4,1} \oplus s_{4,2} \oplus s_{4,3} = 0$.

- To prevent collusion attacks for any subset of parties, a pairwise OKVS must be implemented between each pair of parties.

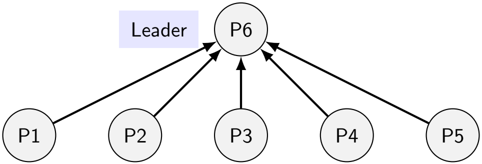
[KMP+17]'s Construction

- MPSI with multiple items can be managed within OKVS (or cuckoo-simple hashing).
- Secure in a semi-honest setting with a dishonest majority
- Execute OKVS between each pair of parties.



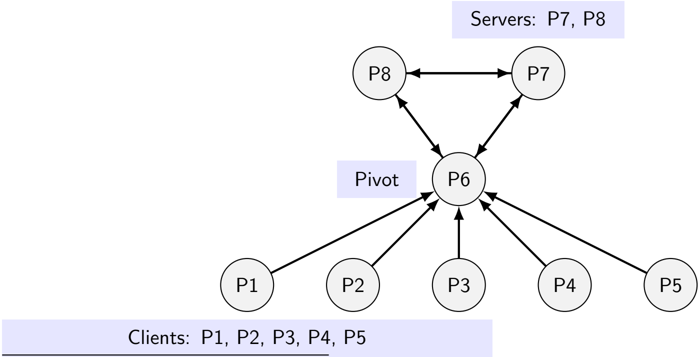
[GPR+21]'s Construction

- Secure in a malicious setting with a dishonest majority
  - The augmented semi-honest protocol is secure against malicious adversaries
- Follow the star network communication structure.



[NTY21]'s Construction

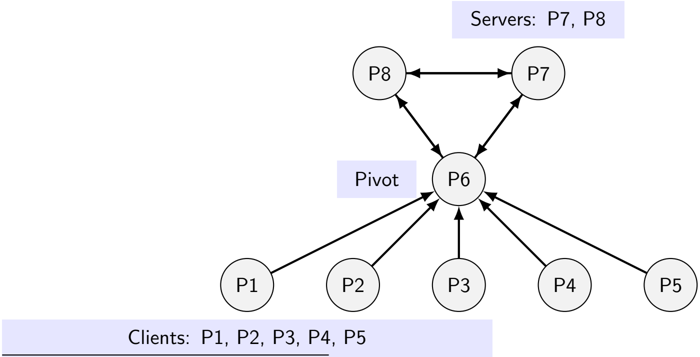
- Secure against up to t colluding parties in a malicious setting[1]
- Distribute n parties into $n - t$ clients, 1 pivot, $t - 1$ servers.
  - Achieve efficient costs on the client's side
  - Distribute the computational burden of a single leader server from the previous protocol across multiple servers.



[1] Recently, [WYC24] showed an attack on this protocol; the fix is simple: replace the direct transmission of PRF keys from clients to servers with OPRF.

[NTY21]'s Construction

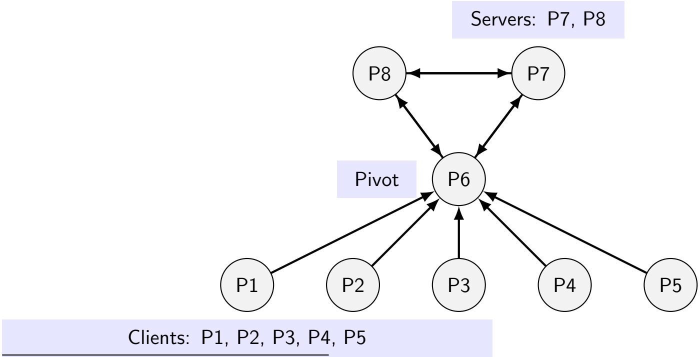
- Secure against up to t colluding parties in a malicious setting[1]
- Distribute n parties into $n-t$ clients, 1 pivot, $t-1$ servers.
  - Achieve efficient costs on the client's side
  - Distribute the computational burden of a single leader server from the previous protocol across multiple servers.



[1] Recently, [WYC24] showed an attack on this protocol; the fix is simple: replace the direct transmission of PRF keys from clients to servers with OPRF.

[NTY21]'s Construction

- Secure against up to t colluding parties in a malicious setting[1]
- Distribute n parties into $n - t$ clients, 1 pivot, $t - 1$ servers.
  - Achieve efficient costs on the client's side
  - Distribute the computational burden of a single leader server from the previous protocol across multiple servers.



[1] Recently, [WYC24] showed an attack on this protocol; the fix is simple: replace the direct transmission of PRF keys from clients to servers with OPRF.

[WYC24]'s Construction

- Secure against up to any t colluding parties in a semi-honest setting.
- Follow the O-Ring and K-Star communication structure.

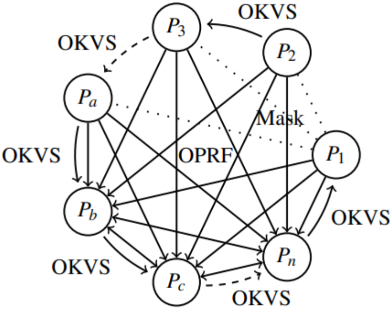


Figure: The workflow of O-Ring [WYC24]

MPSI Extension

- Compute arbitrary functions over intersecting elements (Circuit PSI)
- Output only the number of intersecting items (PSI-Cardinality)

Circuit MPSI [CDG+21]

- Output secret shares of bits indicated the intersecting items (rather than the items themselves).
- Key concept:
  - Introduce an efficient private set membership protocol.
  - Converting between Boolean secret shares and arithmetic shares for efficient computation

MPSI-Cardinality (PSI-CA) [GTY24]

- Secure in the semi-honest setting with an honest majority.
- Key concept:
  - Propose an efficient server-aided Oblivious Programmable PRF (OPPRF).
  - Reduce the problem to a 2-party PSI-CA.

Open Problems

- Improve the performance of MPSI and MPSI extensions.
- Implement MPSI-CA with a dishonest majority and/or ensure security against malicious servers.
- Tailor existing MPSI protocols for different applications.

Thank You

Nishanth Chandran, Nishka Dasgupta, Divya Gupta, Sai Lakshmi Bhavana Obbattu, Sruthi Sekar, and Akash Shah.
Efficient linear multiparty PSI and extensions to circuit/quorum PSI.
In Giovanni Vigna and Elaine Shi, editors, ACM CCS 2021, pages 1182–1204. ACM Press, November 2021.

Gayathri Garimella, Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai.
Oblivious key-value stores and amplification for private set intersection.
In Tal Malkin and Chris Peikert, editors, CRYPTO 2021, Part II, volume 12826 of LNCS, pages 395–425, Virtual Event, August 2021. Springer, Heidelberg.

Jiahui Gao, Ni Trieu, and Avishay Yanai.
Multiparty private set intersection cardinality and its applications.