

USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Butterfly: Scalable Multi-Party Circuit-PSI via Triplet Zero-Sharing

Ranyang Liu, *the College of Cryptology and Cyber Science, DISSec, Nankai University;* Xiaojie Guo, *Shanghai Qi Zhi Institute;* Tong Li and Zheli Liu, *the College of Cryptology and Cyber Science, DISSec, Nankai University*

https://www.usenix.org/conference/usenixsecurity26/presentation/liu-ranyang

This paper is included in the Proceedings of the 35th USENIX Security Symposium.

August 12–14, 2026 • Baltimore, MD, USA

ISBN 978-1-939133-58-8

Open access to the Proceedings of the 35th USENIX Security Symposium is sponsored by



جامعة الملك عبدالله للعلوم والتقنية

King Abdullah University of Science and Technology

ARTIFACT EVALUATED
USENIX
AVAILABLE

Butterfly: Scalable Multi-Party Circuit-PSI via Triplet Zero-Sharing

Ranyang Liu[†], Xiaojie Guo[‡], Tong Li[†,✉], Zheli Liu[†]
[†]*the College of Cryptology and Cyber Science, DISSec, Nankai University*
liury@mail.nankai.edu.cn, {tongli, liuzheli}@nankai.edu.cn
[‡]*Shanghai Qi Zhi Institute,* guoxiaojie@sqz.ac.cn
[✉]*Corresponding Author*

Abstract

Multi-party circuit private set intersection (MP-CPSI) enables m parties to compute the intersection of their private sets in secret-shared form, facilitating secure downstream computations. However, existing protocols face severe efficiency bottlenecks: they either rely on generic circuit-based approaches (e.g., Sort-Compare-Shuffle) that incur prohibitive non-linear complexity, or they aggregate pairwise two-party protocols using full-scale m-party MPC, where the communication cost scales super-linearly with the number of participants.

In this work, we propose Butterfly, a highly scalable MP-CPSI protocol that achieves the communication cost of each party independent of the number of parties m and linear in the set size n. Our construction reduces the general m-party problem to an honest-majority three-party committee via a novel dual-execution blueprint and triplet zero-sharing, leveraging only efficient key-value stores (KVS) and symmetric-key operations. Furthermore, our protocol decouples the complexity of downstream generic computations from m, allowing the subsequent circuits to be evaluated using efficient generic honest-majority three-party computations among the committee (the "body"), while the other $m-3$ clients (the "wings") achieve only one-shot communication and are dropout-tolerant. The security of our protocol relies on the non-collusion assumption among the committee and is secure against any adversary corrupting arbitrary clients and at most one committee party. We further provide an efficient extension of our protocol to support associated payloads, enabling downstream complex analytics with practical efficiency.

Compared to the state-of-the-art baselines without non-collusion assumptions, our protocol achieves up to 10130× and 3231× reductions in communication and running time, respectively, for $m = 30$. In the special case of $m = 3$, our protocol requires 4× less communication and runs 14× faster than the honest-majority baseline in the same model. We separately evaluate our protocol in large-scale settings, which runs in just 2.66 (resp. 65.5) seconds under a 10Gbps (resp. 200Mbps) network for 120 parties each holding 2^{20} items, incurring only 1405 MB communication costs in total.

1 Introduction

Private Set Intersection (PSI) is a specialized functionality in secure multi-party computation (MPC) that enables mutually distrusting parties to compute the intersection of their private sets without revealing any additional information. PSI has been deployed in two-party settings (2P-PSI), such as for contact discovery, and decades of research have optimized 2P-PSI to achieve practical performance [16, 23, 39, 40]. However, the dedicated primitives developed in these two-party protocols cannot be directly scaled to multi-party settings. Consequently, recent works have focused on developing efficient Multi-Party PSI (MP-PSI) protocols, proposing distinct blueprints from the two-party setting [6, 13, 15, 24, 34, 45].

Despite the superior efficiency of plain PSI, the increasing complexity of data analysis has driven a strong demand for *Circuit-PSI* (CPSI). Unlike PSI, CPSI does not output the plaintext intersection; instead, it distributes secret shares of membership indicators among the parties. This feature allows parties to perform extensive subsequent computations on the intersection and associated payloads using generic MPC techniques. Since Huang *et al.* [20] first proposed circuit-based PSI, a series of works have significantly improved the performance of the two-party variant (2P-CPSI) [7, 19, 36, 39–41].

In this work, we focus on the analogous multi-party extension of CPSI (MP-CPSI). Here, $m \geq 3$ parties $P_1, \ldots, P_m$, with respective sets $X^1, \ldots, X^m$, wish to evaluate subsequent computations over the intersection $X^1 \cap \ldots \cap X^m$. Similar to the 2P-CPSI, MP-CPSI aims to output secret-shared indicators $\{\mathsf{b}_i\}$, where b_i indicates whether the i-th item of a designated leader belongs to the intersection.

While the motivation for MP-CPSI is compelling, extending 2P-CPSI protocols to the multi-party setting efficiently is non-trivial. While the MP-CPSI remains largely unexplored, we observe key points for designing and applying MP-CPSI to achieve practical efficiency:

- **Functionality and Subsequent Computations.** The popular definition of CPSI functionality computes secret-shared membership indicators [36, 40]. This implies that

the construction would involve the use of generic MPC techniques based on secret sharing, *e.g.*, secure equality tests. Furthermore, since MP-CPSI aims to facilitate more general computations over the intersection, it would further involve mixed-mode computations to efficiently support both arithmetic and Boolean operators [12, 32].

- **Efficiency and Security Tradeoff.** The efficiency is dominated by the two parts: the dedicated primitives and generic MPC techniques. The dedicated primitives, for example, are constructed from public-key operations in 2P-CPSI protocols [36, 40]. With respect to the generic MPC techniques, the efficiency varies significantly as the number of parties m and corruption threshold t increase. Ideally, we expect the protocol to tolerate an optimal corruption threshold $t = m-1$ (dishonest-majority); however, the concrete efficiency is still far from practical applications. By assuming an honest majority ($t < m/2$), the MPC from Shamir secret sharing provides much better efficiency, yet is still unsatisfactory for large-scale settings and mixed-mode computations.
- **Asymmetric Roles and Resources.** In many real-world scenarios, the multi-party collaboration campaigns are usually launched and led by a few parties, while the other participants contribute their data, *e.g.*, the leading large companies vs. small clients [5, 15, 42]. Besides the asymmetric roles, the parties usually also have unbalanced computational and communication resources. In previous MP-PSI and MP-CPSI protocols [6, 13, 15, 24, 43, 45], there are also hierarchical architectures (star, tree, or ring) and asymmetric roles, where some specific parties are designated to undertake more computation tasks and communication costs, due to their motivations and possible conflicting interests in practical scenarios.

From the perspectives presented above, we briefly review the existing MP-CPSI protocols, which follow the 2P-CPSI blueprints and adapt them to the multi-party setting [6, 13, 43]. However, these protocols incur efficiency and scalability bottlenecks as outlined below.

MP-CPSI via Generic MPC Circuit. Su *et al.* [43] extended the Sort-Compare-Shuffle (SCS) circuit of Huang *et al.* [20] to the multi-party setting and realized a functionality that outputs the secret shares of shuffled intersection items themselves. In terms of efficiency, they provided a tradeoff by assuming two non-colluding parties that collect the shared inputs of all other parties, similar to the tradeoff in recent efficient MP-PSI protocols [15, 34]. This assumption yields performance gains by enabling the two central parties to use a more efficient generic 2PC [12] than a generic m-party computation. However, the concrete efficiency remains unsatisfactory because it relies entirely on the generic circuit of $O(mn(\log mn)^2)$ complexity and fails to leverage recent dedicated primitives [24, 36, 40] that reduce complexity.

MP-CPSI via Pairwise 2P-CPSI. The protocols of Chandran *et al.* [6] and Dong *et al.* [13] follow the blueprint of state-of-the-art 2P-CPSI protocols based on cuckoo hashing and OPPRF [7, 36, 40] and employ a privileged leader. Their common high-level strategy is *divide-conquer-aggregate*: (1) Divide the membership test, *e.g.*, $x_i^m \in_? \bigcap_{j\in\{1,\ldots,m-1\}} X^j$, into $m-1$ pairwise sub-tests $x_i^m \in_? X^j$; (2) Invoke pairwise two-party primitives between the leader P_m and each client P_j; (3) Aggregate $m-1$ pairwise sub-tests using generic m-party computations to obtain the membership indicators $(x_i^m \in_? X^1) \wedge \ldots \wedge (x_i^m \in_? X^{m-1})$, which is exactly $x_i^m \in_? \bigcap_{j\in\{1,\ldots,m-1\}} X^j$. However, both approaches suffer from significant communication overhead. Firstly, the pairwise OPPRF instances require $O(m)$ costly public-key operations. Secondly, since they only assume honest majority [6] and dishonest majority [13] without specific non-colluding parties, both protocols require all m parties to participate in the MPC circuit evaluations. This leads to both protocols suffering from the prohibitively heavy overhead introduced by the full-scale honest/dishonest-majority MPC techniques. Finally, because the use of cuckoo hashing could reveal the leader's private input distribution in subsequent computations [37], the output shares should be obliviously shuffled, incurring quadratic complexity $O(m^2n)$ [13, 14]. Considering asymmetric scenarios, the protocols incur substantial overhead and interaction rounds for all parties, which are unrealistic for the clients with limited resources [5, 22].

Given the state of affairs, we are motivated to design a concretely efficient MP-CPSI with practical performance in scalable settings, while achieving one-shot clients with lightweight overhead suitable for asymmetric scenarios.

1.1 Our Contribution

New Blueprint for Three-Party Circuit-PSI. We first propose a new blueprint for three-party circuit-PSI in the honest majority setting, based on the novel dual-execution structure and triplet zero-sharing semantics. Compared with the pairwise two-party instances in previous works, our protocol requires all three parties to participate in each execution to generate triplet zero-sharings for intersection items. We provide an efficient semi-honest construction for triplet zero-sharings by utilizing the advanced Key-Value Store (KVS) structures in our setting, which involves only symmetric-key operations without additional shuffle overhead in previous works. By parallelizing the dual executions and aggregating to obtain the triplet zero-sharings, we realize an efficient 3P-CPSI based on efficient generic 3PC techniques.

Efficient Generalization to Multi-Party Circuit-PSI. We extend our blueprint to the general m-party setting by proposing a "distribute-filter-aggregate" strategy. We maintain the dual-execution structure by partitioning the more parties ($m-3$ clients) into two groups (the "wings"), and then reducing the multi-party membership tests to the triplet zero-

sharings among the basic honest-majority three-party committee (the "body"), thus obtaining Butterfly, a concretely efficient MP-CPSI in the semi-honest model, involving only symmetric-key operations and does not require shuffling. Our protocol achieves the communication cost of each party linear in the set size n and independent of the number of parties m, as shown in Table 1. As our protocol decouples the downstream secure computation from full-scale m-party techniques, the committee could evaluate the subsequent circuits using generic 3PC techniques with communication independent of m. Meanwhile, the isolation of $m-3$ clients from generic circuit evaluations also achieves one-shot client-side communication and dropout tolerance. The security relies on the non-collusion assumption of an honest-majority three-party committee without further assumptions, allowing the adversary to corrupt $t < m-1$ parties (including at most one committee party). We further extend our protocol to support parties' payloads, which facilitates more versatile subsequent computations over the intersection with payloads.

Implementation and Evaluation. We implement our protocols and provide a comprehensive evaluation under various network environments and parameter settings. The extensive experimental results show that our protocol outperforms state-of-the-art baselines.[1] Specifically, our protocol reduces the total communication by 10130× and 99×, and runs up to 3231× and 122× faster, compared to existing dishonest-majority and honest-majority baselines without the non-collusion assumption, respectively, for 30 parties each with a million items. In the special case $m = 3$, our protocol requires 4× and 14× less communication cost and running time than the honest-majority baseline in the same model with the same corruption threshold $t = 1$. We separately evaluate our protocol in large-scale and bandwidth-restricted settings, which handle 120 parties with each dataset of 2^{20} items in just 2.66 (resp. 65.5) seconds under the 10Gbps (resp. 200Mbps) network and 1405 MB total communication costs. Our protocol is also suitable for these asymmetric scenarios as it achieves lightweight client-side overhead independent of m, requiring fast local computation of less than 0.1s and one-shot communication of 10.9 MB for set size 2^{20}.

1.2 Related Works

Two-Party Circuit-PSI. Huang *et al.* [20] first proposed the SCS circuit for two-party PSI using generic 2PC techniques. Subsequently, a series of works proposed more efficient 2P-CPSI protocols [7, 35–37, 39, 40] that compute the secret-shared membership indicators, leveraging the dedicated cryptographic primitive OPPRF [24], hashing-to-bin techniques [37], and equality test circuits [12]. There are also unbalanced protocols focusing on achieving sublinear complexity in the size of the larger set, employing fully homomorphic encryption (FHE) [41], or private information retrieval (PIR) [19]. The state-of-the-art balanced and unbalanced 2P-CPSI protocols all employ the hashing-to-bin techniques, and the achieved functionality is parameterized by the hashing functions, *i.e.*, the output order is aligned with the cuckoo hashing map. Pinkas *et al.* [38] noted that the cuckoo hashing map could incur information leakage in the subsequent uncertain functions and should be randomly shuffled.

The concurrent work of Liu *et al.* [30] proposes an efficient helper-aided 2P-CPSI protocol based on a formalized distributed KVS scheme, which utilizes the linearity of key-value stores and suffices for a two-party setting aided by an inputless helper. In Butterfly, we aim to achieve the different functionality of MP-CPSI without relying on a helper efficiently, via a novel dual-execution structure and "distribute-filter-aggregate" strategy. While we also use key-value stores in a distributed manner, we do not directly utilize the linearity due to different problem settings and security models. Instead, as we will show in Section 2.2, we propose a different homomorphic property and different distribution methods that allow decoding on multiple representations of the keys (whereas the linearity in DKVS relies on a consistent representation), which allows us to achieve secure MP-CPSI against adversaries corrupting any one of the three committee parties.

General Multi-Party Setting. In Table 1, we present a comprehensive comparison to the existing MP-CPSI protocols [6, 13, 43]. Chandran *et al.* [6] proposed the first MP-CPSI protocol in 2021, where a leader invokes a batch of the pairwise 2P-CPSI instances with other $m-1$ clients using a consistent cuckoo hashing map, and each underlying OPPRF-based 2P-CPSI has $O(n\lambda\kappa)$ communication. The parties then compute the membership-conditional zero-sharing among all the parties by first converting the pairwise indication bits in Boolean shares into arithmetic shares among all the parties, and then computing the sum of the bits minus $m-1$, which equals zero if and only if the corresponding item is in the intersection and can be fed into the subsequent EQZ circuit using honest-majority MPC techniques. In 2024, Su *et al.* [43] extended the SCS circuit of Huang *et al.* [20] to the multi-party setting and proposed an entirely circuit-based MP-CPSI that outputs the secret-shared intersection items. To optimize performance, the protocol employs two non-colluding parties to evaluate the circuit, while all other $m-2$ clients only share their inputs with them, thereby avoiding the expensive full-scale m-party computations. However, since it is entirely based on the circuit, the $O(mn(\log mn)^2)$ complexity is superlinear in both m and n, and the concrete efficiency is not practical. Recently, Dong *et al.* [13] proposed a series of multi-party set operations, including an MP-CPSI protocol, similar to the pairwise blueprint of [6] except for two main differences: (1) instead of executing pairwise 2P-CPSI instances, they proposed more efficient membership-conditional zero-sharings by aggregating the intermediate outputs of pairwise

[1] Dong *et al.* [13] report the online performance and exclude the preprocessing costs for generating correlated randomness, *e.g.*, multiplication triples, while we benchmark and report the overall performance.

Table 1: Protocol Comparisons. **Communication complexity** includes (1) "Generation of Sharings" (membership-conditional zero-sharings of [6], [13], and ours, as well as input secret-sharing of [43]) and (2) "Circuits over Sharings" (equality-to-zero circuits of [6], [13], and ours, as well as sort-compare-shuffle circuit of [43]). **Computation complexity** includes the overall phases. **Building Block** of the OPPRF and dishonest-majority generic 2PC/MPC in [6, 13, 43] are based on OT, while our protocol only relies on fast symmetric-key (SK) operations. n and m are the set size and the number of parties, respectively. λ and κ are the statistical and computational security parameters, respectively, with input item length $O(\lambda)$. **Shuffle HashMap** denotes the communication complexity of the multi-party shuffle [14] required to avoid potential leakages caused by the underlying cuckoo hashing map [37] in subsequent computations of CPSI protocols [6, 13]. **Security** represents the corruption threshold t in the same semi-honest model, and the signal * implies that the protocol makes non-collusion assumptions among specific parties.

Protocol	**Generation of Sharings**		**Circuits over Sharings**		**Computation**		**Building**	**Shuffle**	**Security**
	Leader/Pivot	Client	Leader/Pivot	Client	Leader/Pivot	Client	**Block**	**HashMap**	
[6]	$O(mn\kappa\lambda)$	$O(n\kappa\lambda)$	$O(mn(\log m)^2)$	$O(n(\log m)^2)$	$O(mn(\lambda+\log m))$	$O(n(\lambda+\log m))$	OT	$O(m^2n)$	$t<\frac{m}{2}$
[43]	$O(n\lambda)$	$O(n\lambda)$	$O(mn(\log mn)^2\kappa\lambda)$	0	$O(mn(\log mn)^2\lambda)$	$O(n)$	OT	-	$t<m^*$
[13]	$O(mn\lambda)$	$O(n\lambda)$	$O(mn\kappa\lambda)$	$O(mn\kappa\lambda)$	$O(mn\lambda)$	$O(mn\lambda)$	OT	$O(m^2n)$	$t<m$
Ours	$O(n\lambda)$	$O(n\lambda)$	$O(n\lambda)$	0	$O(n(\lambda+m))$	$O(n)$	SK	-	$t<m-1^*$

OPPRF instances (each has $O(n\lambda)$ communication), based on their observation about membership conjunctions; (2) they instantiated the subsequent generic m-party computations, including EQZ circuits, with the dishonest-majority techniques, achieving optimal corruption threshold ($t<m$) at the cost of heavier communication costs. We also note that the work of Zhang [47] briefly sketched an extension of the proposed MP-PSI protocol to MP-CPSI (cannot support payloads), but without formalizing necessary functionalities or constructions, *e.g.*, OPPRF, to realize MP-CPSI. As it also lacks detailed analysis and implementation, prior works (MP-PSI of [15] and MP-CPSI of [13]) and our work fail to provide formal complexity or experimental comparisons.

Compare to Outsourced and Committee-based Model. We note that our protocol is similar to the outsourced 3PC model, where the clients outsource (secret-share) their private data to three non-colluding servers (yet inputless) such that the computation tasks can be performed using the honest-majority 3PC techniques [1, 32], *e.g.*, privacy-preserving Machine Learning as a Service (MLaaS) [26, 44] and secure database analysis service [11, 18].

More closely, the committee-based MPC also improves the efficiency by employing a small subset of c parties with respective inputs, *i.e.*, a c-party committee. Despite the common idea of our butterfly blueprint and committee-based MPC, we highlight the differences as follows. Specifically, the committee-based MPC aims to support generic circuits by delegating to the committee using c-party computations over the secret-shared inputs of all parties, while the other $m-c$ parties stay idle after secret-sharing their inputs among the committee. For example, the MP-CPSI of Su *et al.* [43] is an instance of two-party committee-based MPC with the delegated sort-compare-shuffle circuit [20]. In contrast, our butterfly blueprint is highly dedicated to MP-CPSI, leveraging non-generic techniques involving all m parties to efficiently generate triplet zero-sharings among a three-party honest-majority committee, and achieves much better efficiency than directly delegating the sort-compare-shuffle circuit of $O(mn\log mn)$ communication complexity to the three-party committee. Furthermore, since circuit delegations by switching to c-party computations are natural, recent committee-based MPC works have focused on designing circuits secure against adaptive corruptions by choosing dynamic ephemeral committees [9, 17]. In contrast, multi-party private set operations have seen dedicated protocols based on the implicit committee ideas [15, 34, 43] recently, and related literature, including ours, mainly focuses on the static corruption model. For example, in recent advancements of the close problem of MP-PSI, the protocols [15, 34] also implicitly assume a static committee of two non-colluding parties, to achieve improved efficiency with a similar security trade-off to the MP-CPSI of ours and Su *et al.* [43].

We refer interested readers to the comprehensive introduction of Dong *et al.* [13] for a broader view of multi-party set operations.

2 Technical Overview

In this work, we focus on scaling MP-CPSI to the multi-party setting efficiently by exploring novel blueprints and primitives distinct from previous works. We begin by exploring a novel blueprint and efficient construction for the basic three-party CPSI (3P-CPSI) setting ($m=3$) with an honest majority among the three-party committee. Based on that, we further extend to general multi-party settings ($m>3$) while decoupling the complexity of generic computation from m, *i.e.*, still using the efficient 3PC among the core three-party committee. We show the *"Butterfly" topology* of our blueprint in Figure 1.

2.1 The "Body": Basic 3P-CPSI

In 3P-CPSI, our goal is to let a leader P_3 and two pivots P_1 and P_2 jointly compute the secret shared indicators $b_i \in \{0,1\}$ such that $b_i=1$ if and only if $x_i^3 \in X^1 \cap X^2$.

Honest-Majority Three-Party Committee. Our starting point is that, in the wide literature of practical MPC and applications, the three-party honest-majority model is particularly promising due to its potential of being extremely efficient and practical. This model has seen many works proposing efficient generic 3PC techniques and mixed-mode computations [1, 26, 32], as well as useful primitives and practical applications [2, 3, 11, 18, 25, 28, 44], where the non-colluding three-party committee can be established, *e.g.*, through structural independence, strict regulations, conflict of interest, and reputational risks. Therefore, we propose our 3P-CPSI within this model assuming an honest majority among (P_1, P_2, P_3), and the real-world applications include Finance (e.g., regulator, exchange, and bank for market surveillance), Health (e.g., administration, pharma, and hospital for medicine analytics), and Tech: Apple, Google, and Mozilla, which are subject to the factors discussed above and have initiated web-related collaborations, can form the three-party committee for broader privacy-preserving collaborations, e.g., threat detection and user insight, under incentives of better platforms and products.

Dual-Execution Structure and Triplet Zero-Sharings. In our blueprint, we propose a *dual-execution* structure and *triplet zero-sharings*. Instead of pairwise checks, all three parties participate in two parallel executions to generate triplet zero-sharings as follows:

- **Execution 1 (X^1 check):** For each leader's item x_i^3, the parties generate a triplet of l-bit shares (a_i^1, a_i^2, a_i^3) such that $a_i^1 \oplus a_i^2 \oplus a_i^3 = 0^l$ if and only if $x_i^3 \in X^1$, and each party P_j obtains corresponding a_i^j.
- **Execution 2 (X^2 check):** For each leader's item x_i^3, the parties generate a triplet of l-bit shares (b_i^1, b_i^2, b_i^3), such that $b_i^1 \oplus b_i^2 \oplus b_i^3 = 0^l$ if and only if $x_i^3 \in X^2$ and each party P_j obtains corresponding b_i^j.

Subsequently, in either execution, the parties can input their respective shares into the equality-to-zero (EQZ) circuit using generic 3PC and learn secret-shared bit indicators for $x_i^3 \in_? X^1$ and $x_i^3 \in_? X^2$. By securely ANDing two arrays of indicators, the parties can obtain the final results about $x_i^3 \in_? X^1 \cap X^2$. By further utilizing the property of triplet zero-sharing, we can optimize the membership tests by first XORing these shares to obtain the final triplet (c_i^1, c_i^2, c_i^3), where it still holds that the sum $c_i^1 \oplus c_i^2 \oplus c_i^3 = 0^l$ if and only if $x_i^3 \in X^1 \cap X^2$, and thus the parties can evaluate only an EQZ circuit on (c_i^1, c_i^2, c_i^3).

Generation of Triplet Zero-Sharings. We propose efficient generation of our triplet zero-sharing based on symmetric-key operations. Recall that in the basic case of [6, 13] based on pairwise zero-sharing, P_σ ($\sigma \in \{1,2\}$) must program distinct random values r_i^σ for its input X^σ, such that the values s_i^σ obtained by the leader P_3 look random in P_3's view, no matter whether each item $x_i^3 \in X^\sigma$. Otherwise, if P_σ programs the whole set X^σ using the same value r, the leader P_3 could find that several obtained $\{s_i^\sigma\}$ (*e.g.*, $i \in I$) are the same, which indicates that the items $\{x_i^3\}_{i \in I}$ form the intersection. Hence, the parties are required to employ the hash-to-bin to reduce the comparison domain and utilize the heavy OPPRF primitive to generate distinct zero-sharings.

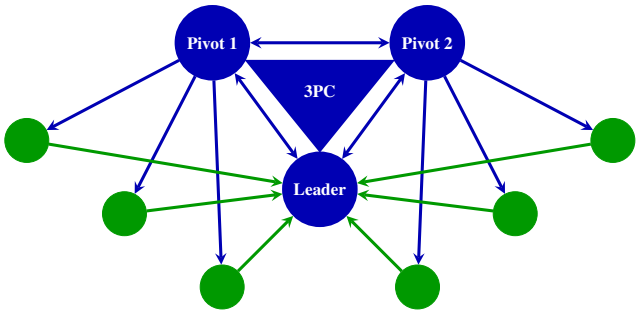

Figure 1: The *"Butterfly" topology* of our MP-CPSI ($m = 9$), with symmetric shape representing our dual-execution structure. The *main "body"* demonstrates the core three-party committee with an honest majority, including two pivots P_1, P_2 and a leader P_3 (Blue), and the *two "wings"* show our extension to support more clients $P_4, ..., P_m$ (Green) divided into two groups. Arrows represent communication flows. The clients only send lightweight one-shot messages to the leader, and the core parties will participate in the subsequent computations using efficient generic honest-majority 3PC.

Our key observation is that, in our semantics of triplet zero-sharing, the programmer P_1 (resp. P_2) can program the whole set X^1 (resp. X^2) using the same value, since either the leader P_3 or the other pivot P_2 (resp. P_1) only receive random shares from its own view and learns nothing about the programmed value, as long as there is at most one corrupted party. Based on this advantage, we propose an efficient construction of triplet zero-sharing, which relies on no public-key operations and hash-to-bin techniques, unlike pairwise OPPRF-based protocols from OT and VOLE [6, 13]. Specifically, our construction of triplet zero-sharing leverages recent efficient structures of Key-Value Store (KVS) [4, 16, 39], which consist of a pair of algorithms (Encode, Decode). On input a set of n key-value pairs $\{(x_i, v_i)\}$, Encode outputs a storage D of linear size with a small constant factor. The correctness ensures that, for any key x^*, the decode output $\mathsf{Decode}(D, x^*)$ is v_i if $x^* = x_i$ which is encoded to generate D. The state-of-the-art candidates of KVS also provide several useful properties, such as the (doubly-)obliviousness KVS (OKVS), which states that D is indistinguishable from a random structure if the encoded values are uniformly random. Another key property of *randomness* ensures that decoding a non-encoded item yields a random value, no matter whether the encoded values are uniformly random. Furthermore, all the KVS structures satisfy the *linearity*, which guarantees that $\mathsf{Decode}(D, x^*) \oplus \mathsf{Decode}(D, x^*) = \mathsf{Decode}(D \oplus D, x^*)$.

Our construction is as follows. In Execution 1, for example, pivot P_1 encodes its set X^1 with a constant value a, generating

structure D_1 that is not doubly-oblivious. To maintain privacy, P_1 secret-shares D_1 among P_2 and P_3. Due to linearity, P_2 and P_3 can locally decode their shares of D_1 using the leader P_3's input x_i^3 and learn a_i^2 and a_i^3, respectively. The correctness ensures that, if $x_i^3 \in X^1$, the reconstructed value $a_i^2 \oplus a_i^3$ is a; otherwise, it is random. This forms the basis of our triplet zero-sharing (a, a_i^2, a_i^3). Note that the above assumes the x_i^3 to be decoded is held by P_3 and P_2 in common. To handle the fact that P_2 can not know X^3, we additionally employ the pseudorandom permutation (PRP) F. In particular, we let P_1 and P_3 jointly sample a key k_1, and then P_3 sends ciphertexts $\{F(k_1, x_i^3)\}$ to P_2, who learns nothing from the encrypted items. Consequently, the corresponding encoding/decoding operations can be performed on encrypted x_i^3. Similarly, in Execution 2, the two pivots switch their roles and compute (b_i^1, b, b_i^3).

In summary, our 3P-CPSI only involves symmetric-key operations and requires no hash-to-bin techniques, thus it is more efficient than previous protocols. Since the order of the zero-sharings and indicators is aligned to the original order of X^3, the outputs of our 3P-CPSI do not need to be shuffled before subsequent computations. The formal description will be shown in Section 4.

2.2 "With Wings": Extending to MP-CPSI

We show how to extend our core 3P-CPSI based on triplet zero-sharing to the general m-party setting for $m > 3$.

A direct extension could be to allocate the parties into different groups of three parties, similar to the pairing strategy of [6, 13], and aggregate the final results jointly. However, the security model is complicated and impractical as it requires an honest majority among each small group, and the aggregation based on generic m-party computation is much less efficient.

Generalized Blueprint with Non-Colluding Committee. Our high-level idea is to maintain the dual-execution structure and the core honest-majority three-party committee without any additional assumptions, *i.e.*, allowing the other $m-3$ clients to collude with at most one of the core three parties (achieving a special corruption threshold $t < m-1$), similar to the recent MP-(C)PSI protocols [15, 34, 43] that also assume a non-colluding small committee among the m parties. Based on that, we allow more parties that may be more vulnerable to corruption (*e.g.*, local banks, regional hospitals, or mid-tier web suppliers in prior discussed scenarios) to participate in MP-CPSI, contribute more datasets, and benefit from the collaborative analytics, while preserving the security of each party's input by only maintaining the honest-majority three-party committee via the suggested factors in Section 2.1.

Our generalized blueprint starts by partitioning the $m-3$ non-committee parties (clients) into *two groups* indexed by $\mathcal{G}_1 = \{4, ..., h\}$ and $\mathcal{G}_2 = \{h+1, ..., m\}$ respectively.[2] Our goal is, in each execution, to test whether the leader's x belongs to the intersection of a single pivot and a single group. Specifically, in the first execution, we require P_1, P_2, and P_3 to obtain the triplet sharing of zero if and only if P_3's item $x_i^3 \in X^1 \cap (\bigcap_{j \in \mathcal{G}_1} X^j)$. Similarly, in the dual second execution, P_1, P_2 and P_3 should obtain the triplet zero-sharing if and only if P_3's item $x_i^3 \in X^2 \cap (\bigcap_{j \in \mathcal{G}_2} X^j)$. The clients only participate in the computations, but they are not designated to obtain the shares. Thus, the leader and pivots can locally XOR the outputs of two executions, which is still a legal zero-sharing if and only if $(x_i^3 \in X^1 \cap (\bigcap_{j \in \mathcal{G}_1} X^j)) \wedge (x_i^3 \in X^2 \cap (\bigcap_{j \in \mathcal{G}_2} X^j))$, i.e., x_i^3 is presented in all parties' set. By evaluating the EQZ circuit as in 3P-CPSI, the three parties can learn the secret-shared indicators and proceed with subsequent computations over the intersection using efficient 3PC techniques.

Reductions to Triplet Zero-Sharings. We propose a *"distribute-filter-aggregate"* strategy to reduce the multi-party membership tests to the triplet zero-sharings among the core three parties, by further exploiting the linearity property of KVS. Here, we describe the first execution for simplicity.

Distribute. Similar to the 3P-CPSI, P_1 and P_3 jointly sample k_1, and P_1 encodes KVS with the same random value a to obtain structure $D_1 = \mathsf{Encode}(\{(F(k_1, x_i^1), a)\})$ that does not meet obliviousness, while P_3 sends $F(k_1, X^3)$ to the other pivot P_2. Since we have more clients and sets to intersect, we let P_1 generate $h+2$ additive shares $\{D_1^2, ..., D_1^h\}$ of D_1 and distribute $\{D_1^j\}_{j \in \mathcal{G}_1}$ to corresponding clients P_j in the first group, and (D_1^2, D_1^3) to the other pivot P_2 and the leader P_3 respectively. Notably, it holds that:

$$\bigoplus_{j \in \mathcal{G}_1 \cup \{2,3\}} \mathsf{Decode}(D_1^j, x') = \mathsf{Decode}(D_1, x') \tag{1}$$

for a common $x' = F(k_1, x)$ by the linearity.

Filter. Then, we require the clients to filter the partial intersection items. Specifically, each client P_j first decodes and obtains $\{c_i^j = \mathsf{Decode}(D_1^j, F(k_1, x_i^j)) \mid x_i^j \in X^j\}$ for $j \in \mathcal{G}_1$, assuming that the PRP key k_1 is also held by the clients. Then, we let each client P_j re-encode $R_1^j = \mathsf{Encode}(\{(F(k_1, x_i^j), c_i^j)\})$ using its own encrypted items and decode values. This helps to filter out the P_1's items that are not in $X^j \cap X^1$, since $\mathsf{Decode}(R_1^j, F(k_1, x^1)) \neq \mathsf{Decode}(D_1^j, F(k_1, x^1))$ for any $x^1 \notin X^j$ by the randomness property. Meanwhile, for the intersection item $x \in X^j \cap X^1$, it still keeps that $\mathsf{Decode}(R_1^j, F(k_1, x)) = \mathsf{Decode}(D_1^j, F(k_1, x))$. Putting all clients' R_1^j together, it holds that for $x' = F(k_1, x)$:

$$\bigoplus_{j \in \mathcal{G}_1} \mathsf{Decode}(R_1^j, x') = \bigoplus_{j \in \mathcal{G}_1} \mathsf{Decode}(D_1^j, x') \tag{2}$$

if and only if $x \in (\bigcap_{j \in \mathcal{G}_1} X^j) \cap X^1$.

Aggregate. By combining equation 1 and 2, we can obtain that for $x' = F(k_1, x)$, $(\bigoplus_{j \in \mathcal{G}_1} \mathsf{Decode}(R_1^j, x')) \oplus$

[2] Note that the partition index h can be set to $3 + \lceil \frac{m-3}{2} \rceil$ for overhead balancing, and setting one group empty is also secure as we will prove.

$\mathsf{Decode}(D_1^2, x') \oplus \mathsf{Decode}(D_1^3, x') = \mathsf{Decode}(D_1, x')$ if and only if the aggregated condition $x \in (\bigcap_{j\in\mathcal{G}_1} X^j) \cap X^1$ holds. Based on that, we can generate triplet zero-sharings as follows. Let the clients send their R_1^j to the leader P_3. Then, P_3 and P_2 can use the common encrypted items $F(k_1, X^3)$ of P_3 as input to decode. Specifically, for P_3's each $x_i^3 \in X^3$, P_3 can compute $a_i^3 = \bigoplus_{j\in\mathcal{G}_1} \mathsf{Decode}(R_1^j, F(k_1, x_i^3))) \oplus \mathsf{Decode}(D_1^3, F(k_1, x_i^3))$, and P_2 can computes $a_i^2 = \mathsf{Decode}(D_1^2, F(k_1, x_i^3))$. As illustrated above, if $x_i^3 \in (\bigcap_{j\in\mathcal{G}_1} X^j) \cap X^1$, it has $a_i^2 \oplus a_i^3 = \mathsf{Decode}(D_1, x_i^3) = a$ and (a, a_i^2, a_i^3) forms a legal triplet zero-sharing. Otherwise, the randomness property and re-encoding ensure that $a_i^3 \oplus a_i^2$ is a random value.

The correctness can be easily derived that we can test whether each $x_i^3 \in (\bigcap_{j\in\mathcal{G}_1} X^j) \cap X^1$ by evaluating the EQZ circuit over (a, a_i^2, a_i^3), and the false positive probability of a random value equals a can be bounded to be negligible.

Corrupt Any One out of Core Parties. The construction is secure when the adversary corrupts either P_1 or P_3, and any clients. However, it is not secure when the pivot P_2 is corrupted, since the pivot P_2 receives the encrypted input $F(k_1, X^3)$ of the leader P_3 while the clients know the decryption key k_1.

To resist the adversary that corrupts any one of the core three parties, we further explore a specific homomorphic property of KVS in our setting. Our key insight is that, given input pairs $\{(x_i, v_i)\}$ and two different keyed PRP functions, say $F(k, \cdot)$ and $F(k', \cdot)$, we can encode two OKVS structures D and D' such that for a member query $x \in \{x_i\}$, the decoded values over the different ciphertexts of x are still the linear secret shares of the encoded value, *i.e.*, $\mathsf{Decode}(D, F(k, x_i)) \oplus \mathsf{Decode}(D', F(k', x_i)) = v_i$. We realize this by secret-sharing the values y_i into random shares vs_i and vs_i', such that $vs_i \oplus vs_i' = v_i$. Then we can encode $D = \mathsf{Encode}(\{(F(k, x_i), vs_i)\})$ and $D' = \mathsf{Encode}(\{(F(k', x_i), vs_i')\})$, each separately satisfying obliviousness given the random secret shares.

Based on that, we can let the leader P_3 send $F(k_1', X^3)$ to P_2 (protected by k_1), while the clients $\{P_j\}_{j\in\mathcal{G}_1}$ can only learn the different PRP $F(k_1, \cdot)$, where both k_1 and k_1' are jointly sampled by P_1 and P_3. The protocol of P_1 is similar except that it sends D' to P_2 and secret-shares D among P_3 and the clients. Hence, we can ensure that P_2 colluding with the clients $\{P_j\}_{j\in\mathcal{G}_1}$ can learn nothing from the leader's ciphertexts $F(k_1', X^3)$, due to the inconsistent PRP keys.

The fully-fledged MP-CPSI construction can be obtained via the dual executions, as we will show in Section 5.

2.3 Supporting Associated Payloads

In many real-world applications, the input items are usually the identifiers for unique matching, while each participant has their own payloads associated with the items, such as the gender, salary, and other attributes stored in the database. These payloads make it possible to conduct more diversified and flexible computations over the set intersection, such as the combination of selections and aggregations [8, 21, 33]. However, it is non-trivial to associate all other parties' payloads with the items held by a single designated party, especially for the multi-party setting, since the output shares are aligned with this party (*e.g.*, the leader P_3 in our MP-CPSI). The works of 2P-CPSI [36, 40] explored associating the payloads using OPPRF in the simpler two-party setting, which was naturally inherited into the pairwise OPPRF-based MP-CPSI protocols [6, 13]. In Section 5.2, we further show how to extend our MP-CPSI to support associated payloads.

3 Preliminary

3.1 Notation

For a positive integer n, $[n]$ denotes the index set $\{1, 2, \ldots, n\}$. Let κ and λ denote the computational and statistical security parameters. For a set S, $x \xleftarrow{\$} S$ denotes sampling x uniformly at random from S. We use $x := y$ and $x \leftarrow \mathsf{Algo}(y)$ to denote assigning y and the output of algorithm Algo on y to x, respectively. We write $\bigoplus_{i\in[n]} x_i$ to denote XOR-sum $x_1 \oplus x_2 \oplus \ldots \oplus x_n$. Similarly, $\bigcap_{i\in[n]} X^i$ denotes the intersection $X^1 \cap X^2 \cap \ldots \cap X^n$. We also use $\wedge$ to denote the logical conjunction (AND) operator.

3.2 Security Model

In this work, we follow the standard semi-honest security model in secure m-party computation [29], and assume an honest-majority three-party committee (P_1, P_2, P_3) of which at most one is corrupted, similar to the non-collusion assumptions in the MP-(C)PSI works [15, 34, 43]. In the semi-honest model, the adversary will honestly follow the protocol but may try to learn more information beyond the protocol output. We defer the formal definition to Appendix A.

3.3 Secret Sharing Framework

Among the MPC protocols, the 3PC of Araki *et al.* [1] based on replicated secret sharing exhibits superior efficiency for basic gates and mixed-mode computations [32] in the three-party honest-majority setting. Informally, a secret $x \in \mathbb{Z}_l$ is secret shared as $x_1 + x_2 + x_3 = x$, and the replicated distribution means that each party P_j holds two shares (x_j, x_{j+1}) with wrapping around $3 + 1 = 1$. The addition/XOR gate is free, and the multiplication/AND gate is efficient: P_i locally computes $z_i \leftarrow x_i y_i + x_i y_{i+1} + x_{i+1} y_i$, masks z_i with random zero shares, and sends to party P_{i-1}. The random zero shares can be non-interactively generated by the underlying correlated pseudorandom generators in RSS. We give the functionality of generic computation $\mathcal{F}_{\mathsf{3pc}}$ in Figure 2, where the non-interactive correlated randomness sampling between two parties is also explicitly described by $\mathcal{F}_{\mathsf{2rand}}$.

[**Functionality** $\mathcal{F}_{\mathsf{3pc}}$]: There are three parties (P_1,P_2,P_3). The functionality is parameterized by a circuit $\mathcal{C}:\{0,1\}^{\mathsf{in}_1+\mathsf{in}_2+\mathsf{in}_3}\to\{0,1\}^{\mathsf{out}_1+\mathsf{out}_2+\mathsf{out}_3}$. Upon receiving input $I_1\in\{0,1\}^{\mathsf{in}_1}$ from P_1, $I_2\in\{0,1\}^{\mathsf{in}_2}$ from P_2, and $I_3\in\{0,1\}^{\mathsf{in}_3}$ from P_3, the functionality computes $(O_1,O_2,O_3):=\mathcal{C}(I_1,I_2,I_3)$ and outputs $O_1\in\{0,1\}^{\mathsf{out}_1}$ to P_1, $O_2\in\{0,1\}^{\mathsf{out}_2}$ to P_2, and $O_3\in\{0,1\}^{\mathsf{out}_3}$ to P_3.
[**Functionality** $\mathcal{F}_{\mathsf{2rand}}$]: There are two parties (P_i,P_j) with $i,j\in\{1,2,3\}$. The functionality receives no input, samples a uniformly random $r\xleftarrow{\$}\{0,1\}^\kappa$, and outputs r to P_i and P_j.

Figure 2: Ideal functionalities $\mathcal{F}_{\mathsf{3pc}}$ and $\mathcal{F}_{\mathsf{2rand}}$

Parameters: There are three input parties P_1,P_2 and P_3 with respective input set X^1, X^2 and X^3 of size n.
Functionality:

1. Wait for input $X^j=\{x_1^j,...,x_n^j\}$ from P_j for $j\in[3]$;
2. For $i\in[n]$, sample $(\mathsf{b}_i^1,\mathsf{b}_i^2,\mathsf{b}_i^3)\in(\{0,1\})^3$ uniformly at random, such that $\mathsf{b}_i^1\oplus\mathsf{b}_i^2\oplus\mathsf{b}_i^3=1$ if $x_i^3\in X^1\cap X^2$, and $\mathsf{b}_i^1\oplus\mathsf{b}_i^2\oplus\mathsf{b}_i^3=0$ otherwise;
3. Output $\{(\mathsf{b}_i^j,\mathsf{b}_i^{j+1})\}_{i\in[n]}$ to P_j for $j\in[3]$.

Figure 3: Ideal functionality $\mathcal{F}_{\mathsf{3p\text{-}cpsi}}$

3.4 Key-Value Store and Properties

A KVS is a data structure that encodes a set of keys and the corresponding values [4, 16, 39, 40].

Definition 1. [16] *A key-value store is parameterized by the key space $\mathcal{K}$, value space $\mathcal{V}$, and a set of functions H, and consists of two algorithms:*

- Encode_H *takes as input a set of (k_i,v_i) key-value pairs and outputs an object D (or, with statistically small probability, an error symbol $\perp$).*
- Decode_H *takes as input an object D, a key k, and outputs a value v.*

In the remainder of the paper, the underlying parameter H is omitted, provided the context remains unambiguous. The correctness of KVS ensures that if for all $A\in\mathcal{K}\times\mathcal{V}$ with distinct keys, $(k,v)\in A$ and $\perp\neq D\leftarrow\mathsf{Encode}_H(A)$, it has $\mathsf{Decode}_H(D,k)=v$.
Complexity. The state-of-the-art KVS schemes [4, 39] achieve a small expanded size β of a KVS object $D\in\mathcal{V}^\beta$ that encodes n key-value pairs, *i.e.*, $\beta=(1+\varepsilon)n$ with expansion factor $0<\varepsilon<0.3$, as well as efficient encoding and decoding algorithms with $O(n)$ computation complexity.
Properties. Recent KVS schemes provide additional properties [4, 16, 39, 40] as follows. A KVS scheme is *doubly-oblivious* (OKVS) if, for all distinct $\{k_1,...,k_n\}$ and n values $\{v_1,...,v_n\}$ each drawn uniformly at random from $\mathcal{V}$, Encode does not output $\perp$ for $\{k_1,...,k_n\}$, then $D\leftarrow\mathsf{Encode}(\{(k_i,v_i)\}_{i\in[n]})$ is statistically indistinguishable from an uniformly random element in $\mathcal{V}^\beta$. Another property of *randomness* ensures that for all distinct $A=\{k_1,...,k_n\}$ and n values $\{v_1,...,v_n\}$, if Encode does not output $D=\perp$ for $\{k_1,\ldots,k_n\}$, the decoded value $\mathsf{Decode}(D,k^*)$ on $k^*\notin A$ is statistically indistinguishable from an uniformly random element in $\mathcal{V}$. Note that this property is implied by the correctness and does not require the values to be uniformly at random. Furthermore, a KVS scheme is *linear* if, for all $k\in\mathcal{K}$ and object $D=D_1\oplus D_2$, it has $\mathsf{Decode}(D_1,k)\oplus\mathsf{Decode}(D_2,k)=\mathsf{Decode}(D,k)$. We note that the state-of-the-art KVS schemes all satisfy the above properties [4, 16, 39].

4 Basic Three-Party Construction

In this section, we introduce our novel blueprint for 3P-CPSI based on triplet zero-sharing and dual-execution structure, which will serve as the core base of our multi-party construction. The functionality is shown in Figure 3.

4.1 Blueprint Details

Distribution of Triplet Zero-Sharing. We define the triplet zero-sharing as a tuple of l-bit shares (c^1,c^2,c^3) with the XOR-sum $c^1\oplus c^2\oplus c^3$ equal to 0^l. In the context of the secure membership test for 3P-CPSI, our goal is to generate a *conditional* triplet (c_i^1,c_i^2,c_i^3) for the leader P_3's each x_i^3, such that (c_i^1,c_i^2,c_i^3) is a triplet zero-sharing if and only if x_i^3 belongs to the intersection $X^1\cap X^2$. However, we note that we *do not* formalize the definition and impose additional constraints on the distribution of triplet zero-sharings, unlike some related protocols that require their zero-sharing semantics to be uniformly random by introducing additional randomization techniques [13, 15]. This is because, in the context of CPSI, either the triplet zero-sharing of our MP-CPSI or the intermediate results in 2P-CPSI [19, 36, 40, 41] will be securely input to the subsequent equality test circuit, which is evaluated without revealing anything. Hence, it suffices for us to ensure the correctness (sum to 0^l) and security (the parties can not learn anything from their view), though the joint distributions are not uniformly random, as we will demonstrate in the formal proof. As we will see, in our protocol, the a_i^1 and b_i^2 are the constant random value a and b across multiple zero-sharings, like (a,a_i^2,a_i^3) and (b_i^1,b,b_i^3).
Membership Conjunction via Sum of Sharings. We propose the dual-execution structure to split the generation of triplet zero-sharing (c_i^1,c_i^2,c_i^3), corresponding to condition $x_i^3\in X^1\cap X^2$, into two parallel generation tasks of (a_i^1,a_i^2,a_i^3) and (b_i^1,b_i^2,b_i^3) that correspond to $x_i^3\in X^1$ and $x_i^3\in X^2$, respectively. We further leverage the observation that the XOR-sum $(c_i^1=a_i^1\oplus b_i^1,c_i^2=a_i^2\oplus b_i^2,c_i^3=a_i^3\oplus b_i^3)$ is still a legal triplet zero-sharing for the AND of two conditions, *i.e.*, $x_i^3\in X^1\cap X^2$. Based on that, parties can first locally aggregate the two arrays of triplet zero-sharing and then evaluate a single EQZ circuit. We defer the detailed construction of the EQZ circuit,

Parameters: There are three input parties P_1, P_2, and P_3 with respective input sets of size n. A PRP $F : \{0,1\}^\kappa \times \{0,1\}^{\mathsf{in}} \to \{0,1\}^{\mathsf{in}}$. A KVS scheme $(\mathsf{Encode}, \mathsf{Decode})$ that satisfies linearity and randomness with key space $\mathcal{K} \leftarrow \{0,1\}^{\mathsf{in}}$, value space $\mathcal{V} \leftarrow \{0,1\}^l$, and output length β for n key-value pairs. Ideal functionality $\mathcal{F}_{\mathsf{3pc}}$.
Input: P_1 inputs set $X^1 = \{x_1^1, ..., x_n^1\}$, P_2 inputs set $X^2 = \{x_1^2, ..., x_n^2\}$, and P_3 inputs set $X^3 = \{x_1^3, ..., x_n^3\}$.
Protocol:

1. P_1 and P_3 invoke $\mathcal{F}_{\mathsf{2rand}}$ to sample a PRP key $k_1 \xleftarrow{\$} \{0,1\}^\kappa$.
   P_2 and P_3 invoke $\mathcal{F}_{\mathsf{2rand}}$ to sample a PRP key $k_2 \xleftarrow{\$} \{0,1\}^\kappa$;
2. Using k_1, P_1 computes $\{F(k_1, x_i^1)\}_{i\in[n]}$, and P_3 computes $\{F(k_1, x_i^3)\}_{i\in[n]}$ and sends $\{F(k_1, x_i^3)\}_{i\in[n]}$ to P_2.
   Using k_2, P_2 computes $\{F(k_2, x_i^2)\}_{i\in[n]}$, and P_3 computes $\{F(k_2, x_i^3)\}_{i\in[n]}$ and sends $\{F(k_2, x_i^3)\}_{i\in[n]}$ to P_1;
3. P_1 samples a random $a \xleftarrow{\$} \mathcal{V}$ and encodes a KVS object $D_1 \leftarrow \mathsf{Encode}(\{(F(k_1, x_i^1), a)\}_{i\in[n]})$ with $D_1 \in \mathcal{V}^\beta$.
   P_2 samples a random $b \xleftarrow{\$} \mathcal{V}$ and encodes a KVS object $D_2 \leftarrow \mathsf{Encode}(\{(F(k_2, x_i^2), b)\}_{i\in[n]})$ with $D_2 \in \mathcal{V}^\beta$;
4. P_1 randomly samples $D_1^2 \xleftarrow{\$} \mathcal{V}^\beta$, computes $D_1^3 := D_1 \oplus D_1^2$, and sends D_1^j to P_j for $j \in \{2,3\}$.
   P_2 randomly samples $D_2^1 \xleftarrow{\$} \mathcal{V}^\beta$, computes $D_2^3 := D_2 \oplus D_2^1$, and sends D_2^j to P_j for $j \in \{1,3\}$;
5. P_2 and P_3 compute $\{a_i^2 \leftarrow \mathsf{Decode}(D_1^2, F(k_1, x_i^3))\}_{i\in[n]}$ and $\{a_i^3 \leftarrow \mathsf{Decode}(D_1^3, F(k_1, x_i^3))\}_{i\in[n]}$, respectively.
   P_1 and P_3 compute $\{b_i^1 \leftarrow \mathsf{Decode}(D_2^1, F(k_2, x_i^3))\}_{i\in[n]}$ and $\{b_i^3 \leftarrow \mathsf{Decode}(D_2^3, F(k_2, x_i^3))\}_{i\in[n]}$, respectively;
6. P_1 computes $\{c_i^1 := a \oplus b_i^1\}_{i\in[n]}$, P_2 computes $\{c_i^2 := a_i^2 \oplus b\}_{i\in[n]}$, and P_3 computes $\{c_i^3 := a_i^3 \oplus b_i^3\}_{i\in[n]}$;
7. P_1, P_2, and P_3 jointly invoke $\mathcal{F}_{\mathsf{3PC}}$:
   - P_1 inputs $\{c_i^1\}_{i\in[n]}$, P_2 inputs $\{c_i^2\}_{i\in[n]}$, and P_3 inputs $\{c_i^3\}_{i\in[n]}$;
   - For $i \in [n]$, $\mathcal{F}_{\mathsf{3PC}}$ computes $c_i := c_i^1 \oplus c_i^2 \oplus c_i^3$ with each $c_i = c_{i,1}...c_{i,l}$, and computes $\mathsf{b}_i \leftarrow \mathsf{AND}_{j\in[l]} \overline{c_{i,j}}$ where $\overline{c_{i,j}} = \mathsf{NOT}(c_{i,j})$;
   - $\mathcal{F}_{\mathsf{3PC}}$ samples random shares $\mathsf{b}_i^1 \oplus \mathsf{b}_i^2 \oplus \mathsf{b}_i^3 = \mathsf{b}_i$ for $i \in [n]$ and outputs $\{(\mathsf{b}_i^j, \mathsf{b}_i^{j+1})\}_{i\in[n]}$ to P_j for $j \in \{1,2,3\}$.

Figure 4: Our 3P-CPSI Protocol $\Pi_{\mathsf{3p\text{-}cpsi}}$

including the number of gates and depth, to Appendix B.1.

4.2 Our 3P-CPSI protocol

In the two parallel executions, we let all three parties participate in generating corresponding triplet zero-sharing and obtain their own share. This facilitates our efficient 3P-CPSI construction based on linear KVS and generic 3PC in the honest-majority setting. Our protocol is shown in Figure 4.

Parameters: There are m input parties $\{P_j\}_{j\in[m]}$ with input sets $\{X^j\}_{j\in[m]}$ of size n.
Functionality:

1. Wait for input $X^j = \{x_1^j, ..., x_n^j\}$ from P_j for $j \in [m]$;
2. For $i \in [n]$, sample $(\mathsf{b}_i^1, \mathsf{b}_i^2, \mathsf{b}_i^3) \in (\{0,1\})^3$ uniformly at random, such that $\mathsf{b}_i^1 \oplus \mathsf{b}_i^2 \oplus \mathsf{b}_i^3 = 1$ if $x_i^3 \in \bigcap_{j\in[m]\setminus\{3\}} X^j$, and $\mathsf{b}_i^1 \oplus \mathsf{b}_i^2 \oplus \mathsf{b}_i^3 = 0$ otherwise;
3. Output $\{(\mathsf{b}_i^j, \mathsf{b}_i^{j+1})\}_{i\in[n]}$ to P_j for $j \in [3]$.

Figure 5: MP-CPSI Ideal functionality $\mathcal{F}_{\mathsf{mp\text{-}cpsi}}$

Theorem 2. *Protocol $\Pi_{\mathsf{3p\text{-}cpsi}}$ of Figure 4 securely realizes the ideal functionality $\mathcal{F}_{\mathsf{3p\text{-}cpsi}}$ of Figure 3 given that at most one party is corrupted in the semi-honest model.*

The proof can be similarly derived from the proof for general MP-CPSI, and we omit it due to the space limit.

5 Multi-Party Circuit-PSI

To allow more clients to join the MP-CPSI while maintaining lightweight overhead and achieving the corruption threshold $t < m-1$, we propose the "distribute-filter-aggregate" strategy and leverage the homomorphic property under distinct keys. The technical overview is introduced in Section 2.

5.1 Our MP-CPSI Protocol with Optimization

Our formal MP-CPSI protocol is shown in Figure 6, where we introduce the following further optimizations to reduce the communication costs.

Optimization via Reverse Generation. We can reduce the communication costs of the KVS secret-sharing. Recall that, in our protocol, each pivot (let P_1) has to send D' to the other pivot P_2, and secret shares D among the leader P_3 and the corresponding group of clients. Here, the structures D and D' are generated by first secret-sharing the secret value v into vs_i and vs_i', and then encode D and D' using $\{vs_i\}$ and vs_i' respectively. By the doubly-obliviousness of KVS, either D or D' separately satisfies the OKVS requirement.

Our key observation is that, since D is an OKVS that can not be distinguished from a uniformly random structure, we can *reverse the generation* by first sampling a random D and then encoding D' accordingly. This facilitates our further communication optimization, where the D is secret shared among P_3 and clients based on the idea of pseudorandom secret sharing via pseudorandom generators (PRGs) (over pre-distributed seeds).

Specifically, instead of generating and sending the shares of D of size $O(n)$, we let the pivot sample random small seeds (κ bits) for P_3 and each client (Step 3 in Figure 6). Each seed can be extended to a random KVS structure via PRG, and the pivot

Parameters: There are m input parties $\{P_1,...,P_m\}$ with respective input sets of size n. Parties $\{P_4,...,P_m\}$ are split by two groups of indices $\mathcal{G}_1$ and $\mathcal{G}_2$ with $\mathcal{G}_1 \cup \mathcal{G}_2 = \{4,...,m\}$. A PRP $F : \{0,1\}^\kappa \times \{0,1\}^{\mathsf{in}} \to \{0,1\}^{\mathsf{in}}$. A KVS scheme $(\mathsf{Encode}, \mathsf{Decode})$ that satisfies linearity, doubly-obliviousness, and randomness with key space $\mathcal{K} := \{0,1\}^{\mathsf{in}}$, value space $\mathcal{V} := \{0,1\}^l$, and output length β for Encode over n key-value pairs. Ideal functionality of generic 3PC $\mathcal{F}_{\mathsf{3pc}}$ and $\mathcal{F}_{\mathsf{2rand}}$. A PRG $G : \{0,1\}^\kappa \to \mathcal{V}^\beta$.

Input: P_j inputs set $X^j = \{x_1^j,...,x_n^j\}$.

Protocol:

Execution 1

1. Parties P_1 and P_3 invoke $\mathcal{F}_{\mathsf{2rand}}$ to jointly sample two random PRP key $k_1 \xleftarrow{\$} \{0,1\}^\kappa$ and $k_1' \xleftarrow{\$} \{0,1\}^\kappa$;
2. P_1 computes $\{F(k_1,x_i^1)\}_{i\in[n]}$ and $\{F(k_1',x_i^1)\}_{i\in[n]}$. P_3 computes $\{F(k_1,x_i^3)\}_{i\in[n]}$ and $\{F(k_1',x_i^3)\}_{i\in[n]}$, and sends $\{F(k_1',x_i^3)\}_{i\in[n]}$ to P_2;
3. P_1 samples random seeds $\{s_1^j\}_{j\in\mathcal{G}_1\cup\{3\}}$, expands each seed $G(s_1^j)$ as pseudorandom secret-share $D_1^j \in \mathcal{V}^\beta$ of $D_1 := \bigoplus_{j\in\mathcal{G}_1\cup\{3\}} D_1^j$, and sends (s_1^j, k_1) to P_j for $j \in \mathcal{G}_1$ and s_1^3 to P_3;
4. For $j \in \mathcal{G}_1 \cup \{3\}$, P_j expands its seed s_1^j into the pseudorandom secret-share D_1^j of D_1;
5. P_1 samples a value $a \xleftarrow{\$} \mathcal{V}$ uniformly at random, encodes $D_1' \leftarrow \mathsf{Encode}(\{(F(k_1',x_i^1), a \oplus \mathsf{Decode}(D_1, F(k_1,x_i^1)))\})$, and sends D_1' to P_2;
6. For $j \in \mathcal{G}_1$, client P_j computes $\{F(k_1,x_i^j)\}_{i\in[n]}$, encodes $R_1^j \leftarrow \mathsf{Encode}(\{(F(k_1,x_i^j), \mathsf{Decode}(D_1^j, F(k_1,x_i^j)))\}_{i\in[n]})$ and sends R_1^j to P_3;
7. P_2 decodes $\{a_i^2 \leftarrow \mathsf{Decode}(D_1', F(k_1',x_i^3))\}_{i\in[n]}$.
   P_3 aggregates KVS object $D_1^* := D_1^3 \oplus (\bigoplus_{j\in\mathcal{G}_1} R_1^j)$ and decodes $\{a_i^3 \leftarrow \mathsf{Decode}(D_1^*, F(k_1,x_i^3))\}_{i\in[n]}$;

Execution 2

1. Parties P_2 and P_3 invoke $\mathcal{F}_{\mathsf{2rand}}$ to jointly sample two random PRP key $k_2 \xleftarrow{\$} \{0,1\}^\kappa$ and $k_2' \xleftarrow{\$} \{0,1\}^\kappa$;
2. P_2 computes $\{F(k_2,x_i^2)\}_{i\in[n]}$ and $\{F(k_2',x_i^2)\}_{i\in[n]}$. P_3 computes $\{F(k_2,x_i^3)\}_{i\in[n]}$ and $\{F(k_2',x_i^3)\}_{i\in[n]}$, and sends $\{F(k_2',x_i^3)\}_{i\in[n]}$ to P_1;
3. P_2 samples random seeds $\{s_2^j\}_{j\in\mathcal{G}_2\cup\{3\}}$, expands each seed $G(s_2^j)$ as pseudorandom secret-share $D_2^j \in \mathcal{V}^\beta$ of $D_2 := \bigoplus_{j\in\mathcal{G}_2\cup\{3\}} D_2^j$, and sends (s_2^j, k_2) to P_j for $j \in \mathcal{G}_2$ and s_2^3 to P_3;
4. For $j \in \mathcal{G}_2 \cup \{3\}$, P_j expands its seed s_2^j into the pseudorandom secret-share D_2^j of D_2;
5. P_2 samples a value $b \xleftarrow{\$} \mathcal{V}$ uniformly at random, encodes $D_2' \leftarrow \mathsf{Encode}(\{(F(k_2',x_i^2), b \oplus \mathsf{Decode}(D_2, F(k_2,x_i^2)))\})$, and sends D_2' to P_1;
6. For $j \in \mathcal{G}_2$, client P_j computes $\{F(k_2,x_i^j)\}_{i\in[n]}$, encodes $R_2^j \leftarrow \mathsf{Encode}(\{(F(k_2,x_i^j), \mathsf{Decode}(D_2^j, F(k_2,x_i^j)))\}_{i\in[n]})$ and sends R_2^j to P_3;
7. P_1 decodes $\{b_i^1 \leftarrow \mathsf{Decode}(D_2', F(k_2',x_i^3))\}_{i\in[n]}$.
   P_3 aggregates KVS objects $D_2^* := D_2^3 \oplus (\bigoplus_{j\in\mathcal{G}_2} R_2^j)$ and decodes $\{b_i^3 \leftarrow \mathsf{Decode}(D_2^*, F(k_2,x_i^3))\}_{i\in[n]}$;

8. P_1 computes $\{c_i^1 := a \oplus b_i^1\}_{i\in[n]}$, P_2 computes $\{c_i^2 := a_i^2 \oplus b\}_{i\in[n]}$, and P_3 computes $\{c_i^3 := a_i^3 \oplus b_i^3\}_{i\in[n]}$ as respective shares of triplet zero-sharings;
9. For $j \in \{1,2,3\}$, P_j inputs $\{c_i^j\}_{i\in[n]}$ into $\mathcal{F}_{\mathsf{3pc}}$ that computes the EQZ circuit, and obtains $\{(\mathsf{b}_i^j, \mathsf{b}_i^{j+1})\}_{i\in[n]}$ with $\mathsf{b}_i^1 \oplus \mathsf{b}_i^2 \oplus \mathsf{b}_i^3 = \mathsf{b}_i$, where the bit $\mathsf{b}_i = 1$ if and only if $c_i^1 \oplus c_i^2 \oplus c_i^3 = 0^l$ (same as the final step in Figure 4).

Figure 6: MP-CPSI Protocol $\Pi_{\mathsf{mp\text{-}cpsi}}$ via *Distribute* (step 3-5), *Filter* (step 6), and *Aggregate* (step 7-8) with dual executions.

reversely defines D as the sum of all random structures (Step 4). Then, the pivot has to generate D' accordingly: first decode random D using $F(k_1,x_i^1)$ to reversely obtain the random masks, and then encode D' over the masked values (Step 5).

In total, this optimization reduces $O(mn)$ communication costs of distributing KVS shares, achieving each pivot's communication independent of m (omit the one-time negligible cost of distributing small seeds to the clients, which can be further amortized in multiple MP-CPSI invocations).

Security and Complexity. Here, we give the security proof and detailed complexity analysis of our protocol.

Theorem 3. *Protocol $\Pi_{\mathsf{mp\text{-}cpsi}}$ of Figure 6 securely realizes the ideal functionality $\mathcal{F}_{\mathsf{mp\text{-}cpsi}}$ of Figure 5 against any semi-honest adversary that corrupts any clients and at most one party of the pivots P_1, P_2, and the leader P_3.*

Proof. We prove the correctness and security of our protocol as follows.

Correctness. The failure of KVS encoding happens with negligible probability $2^{-\lambda}$. By the correctness of PRP, there is no collision between the ciphertexts. In the case $x_i^3 \notin I$ and no collision, it has that $c_i = c_i^1 \oplus c_i^2 \oplus c_i^3$ is a random element in $\mathcal{V}$ and possibly collides with the random value $a \oplus b$, by the randomness property of KVS. We set $l = \lambda + \log n$ to guarantee a negligible probability $2^{-\lambda}$ of such a collision for all $x_i^3 \in X^3$.

Security. We divide the proof into four cases, based on which of the core parties is corrupted.

Case 1: P_1 is corrupted. In this case, $1 \in \mathcal{C} = \{j_1,...,j_t\}$ and let $\mathcal{H} = [m] \backslash \mathcal{C}$ where $\{2,3\} \subseteq \mathcal{H}$. Sim is given $(X^{j_1},...,X^{j_t})$ and $\{(\mathsf{b}_i^1, \mathsf{b}_i^2)\}_{i\in[n]}$. Sim samples random $k_1, k_1' \xleftarrow{\$} \{0,1\}^\kappa$ and invokes the simulator of $\mathcal{F}_{\mathsf{2rand}}$ to simulate the execution of Π_{2rand}. Sim samples random $\{z_i^3 \xleftarrow{\$} \{0,1\}^{\mathsf{in}}\}_{i\in[n]}$ to simulate the PRP ciphertexts from P_2. Sim samples random seeds $\{s_2^j\}_{j\in\mathcal{C}\cap\mathcal{G}_2}$ and key k_2 to simulate the seeds and key from P_2. Sim samples a random D_2' to simulate the OKVS object from P_2. Sim follows the protocol to generate $\{c_i^1\}_{i\in[n]}$. Sim invokes the simulator of $\mathcal{F}_{\mathsf{3pc}}$ with input $\{c_i^1\}_{i\in[n]}$ and output $\{(\mathsf{b}_i^1, \mathsf{b}_i^2)\}_{i\in[n]}$ to simulate the execution of Π_{3pc}.

We argue that the view output by Sim is indistinguishable from the real one.

- Hyb_0 : This hybrid is the view of $\mathcal{C}$ in real execution.
- Hyb_1 : This hybrid is the same as Hyb_0, except that the

PRP keys k_1 and k_1' are chosen uniformly at random by Sim, and Sim runs the simulator of $\mathcal{F}_{\mathsf{2rand}}$ to obtain the simulated view of P_1. The security of Π_{2rand} guarantees that this hybrid is indistinguishable from Hyb_0.

- Hyb_2 : This hybrid is the same as Hyb_1, except that the PRP ciphertexts $\{F(k_2', x_i^3)\}_{i\in[n]}$ are replaced by $\{z_i^3\}_{i\in[n]}$ chosen uniformly at random by Sim. Since k_2' is jointly sampled by honest P_2 and P_3, the pseudorandomness of PRP guarantees that $\{F(k_2', x_i^3)\}_{i\in[n]}$ are indistinguishable from uniformly random elements from the view of corrupted parties without the random key k_2', and thus this hybrid is indistinguishable from Hyb_1.

- Hyb_3 : This hybrid is the same as Hyb_2, except that the seeds $\{s_2^j\}_{j\in\mathcal{C}\cap\mathcal{G}_2}$ and key k_2 are chosen uniformly at random by Sim. Since the seeds and k_2 are also sampled uniformly by honest P_2 in the real execution, this hybrid is indistinguishable from Hyb_2.

- Hyb_4 : This hybrid is the same as Hyb_3, except that the D_2' is chosen uniformly at random by Sim. In the real execution, the encoded values of D_2' depend on $D_2 = \bigoplus_{j\in\mathcal{G}_2\cup\{3\}} G(s_2^j)$, which in turn depends on all the seeds $\{s_2^j\}_{j\in\mathcal{G}_2\cup\{3\}}$. Since $\{s_2^j\}_{j\in\mathcal{H}}$ are held by honest parties and $\mathcal{H}\cap(\mathcal{G}_2\cup\{3\})\neq\emptyset$ (at least P_3 is honest), the encoded values are uniformly random from the view of corrupted parties. The doubly-obliviousness of OKVS ensures that this hybrid is indistinguishable from Hyb_3.

- Hyb_5 : This hybrid is the same as Hyb_4, except that the Sim runs the simulator of $\mathcal{F}_{\mathsf{3pc}}$ with generated $\{c_1^1,...,c_n^1\}$ and output $\{(\mathsf{b}_i^1,\mathsf{b}_i^2)\}_{i\in[n]}$ to obtain the simulated view of P_1. The security of $\mathcal{F}_{\mathsf{3pc}}$ guarantees that this hybrid is indistinguishable from Hyb_4. This hybrid is the output view of Sim.

Case 2: P_2 is corrupted. In this case, $2\in\mathcal{C}=\{j_1,...,j_t\}$ and let $\mathcal{H}=[m]\backslash\mathcal{C}$ where $\{1,3\}\subseteq\mathcal{H}$. Sim is given $(X^{j_1},...,X^{j_t})$ and $\{(\mathsf{b}_i^2,\mathsf{b}_i^3)\}_{i\in[n]}$. The simulation is the same as case 1 since P_1 and P_2 play dual roles in the dual executions.

Case 3: P_3 is corrupted. In this case, $3\in\mathcal{C}=\{j_1,...,j_t\}$ and let $\mathcal{H}=[m]\backslash\mathcal{C}$ where $\{1,2\}\subseteq\mathcal{H}$. Sim is given $(X^{j_1},...,X^{j_t})$ and $\{(\mathsf{b}_i^3,\mathsf{b}_i^1)\}_{i\in[n]}$. Sim samples random $k_1,k_1',k_2,k_2'\xleftarrow{\$}\{0,1\}^\kappa$ and invokes the simulator of $\mathcal{F}_{\mathsf{2rand}}$ to simulate the execution of Π_{2rand}. Sim samples random seeds $\{s_1^j\}_{j\in(\mathcal{C}\cap\mathcal{G}_1)\cup\{3\}}$ and $\{s_2^j\}_{j\in(\mathcal{C}\cap\mathcal{G}_2)\cup\{3\}}$ to simulate the seeds from P_1 and P_2. Sim samples random $\{R_1^j\}_{j\in\mathcal{G}_1\cap\mathcal{H}}$ and $\{R_2^j\}_{j\in\mathcal{G}_2\cap\mathcal{H}}$ to simulate the KVS objects from honest clients. Sim follows the protocol to generate $\{c_i^3\}_{i\in[n]}$. Sim invokes the simulator of $\mathcal{F}_{\mathsf{3pc}}$ with input $\{c_i^3\}_{i\in[n]}$ and output $\{(\mathsf{b}_i^3,\mathsf{b}_i^1)\}_{i\in[n]}$ to simulate the execution of Π_{3pc}.

We argue that the view output by Sim is indistinguishable from the real one.

- Hyb_0 : This hybrid is the view of $\mathcal{C}$ in real execution.

- Hyb_1 : This hybrid is the same as Hyb_0, except that the PRP keys k_1, k_1', k_2, and k_2' are chosen uniformly at random by Sim, and Sim runs the simulator of $\mathcal{F}_{\mathsf{2rand}}$ to obtain the simulated view of P_3. The security of Π_{2rand} guarantees that this hybrid is indistinguishable from Hyb_0.

- Hyb_2 : This hybrid is the same as Hyb_1, except that the seeds $\{s_1^j\}_{j\in(\mathcal{C}\cap\mathcal{G}_1)\cup\{3\}}$ and $\{s_2^j\}_{j\in(\mathcal{C}\cap\mathcal{G}_2)\cup\{3\}}$ are chosen uniformly at random by Sim. Since in real execution all the seeds $\{s_1^j\}_{j\in\mathcal{G}_1\cup\{3\}}$ and $\{s_2^j\}_{j\in\mathcal{G}_2\cup\{3\}}$ are sampled by honest P_2 and P_3, respectively, this hybrid is indistinguishable from Hyb_1.

- Hyb_3 : This hybrid is the same as Hyb_2, except that the $\{R_1^j\}_{j\in\mathcal{G}_1\cap\mathcal{H}}$ and $\{R_2^j\}_{j\in\mathcal{G}_2\cap\mathcal{H}}$ are chosen uniformly at random by Sim. In real execution, the encoded values of each object in $\{R_1^j\}_{j\in\mathcal{G}_1\cap\mathcal{H}}$ and $\{R_2^j\}_{j\in\mathcal{G}_2\cap\mathcal{H}}$ are the decoded values on P_j's private set using own object expanded from the seeds $\{s_1^j\}_{j\in\mathcal{G}_1\cap\mathcal{H}}$ and $\{s_2^j\}_{j\in\mathcal{G}_2\cap\mathcal{H}}$, respectively. Since the seeds are sampled by honest P_1 and P_2 and held by the honest clients, the encoded values of each object in $\{R_1^j\}_{j\in\mathcal{G}_1\cap\mathcal{H}}$ and $\{R_2^j\}_{j\in\mathcal{G}_2\cap\mathcal{H}}$ are indistinguishable from random values. The doubly-obliviousness ensures that $\{R_1^j\}_{j\in\mathcal{G}_1\cap\mathcal{H}}$ and $\{R_2^j\}_{j\in\mathcal{G}_2\cap\mathcal{H}}$ are indistinguishable from random objects, and thus this hybrid is indistinguishable from Hyb_2.

- Hyb_4 : This hybrid is the same as Hyb_3, except that the Sim runs the simulator of $\mathcal{F}_{\mathsf{3pc}}$ with generated $\{c_1^3,...,c_n^3\}$ and output $\{(\mathsf{b}_i^3,\mathsf{b}_i^1)\}_{i\in[n]}$ to obtain the simulated view of P_3. The security of $\mathcal{F}_{\mathsf{3pc}}$ guarantees that this hybrid is indistinguishable from Hyb_3. This hybrid is the output view of Sim.

Case 4: P_1, P_2, and P_3 are not corrupted. In this case, $\mathcal{C}\subseteq\mathcal{G}_1\cup\mathcal{G}_2$ and let $\mathcal{H}=[m]\backslash\mathcal{C}$ where $\{1,2,3\}\subseteq\mathcal{H}$. Sim is given $(X^{j_1},...,X^{j_t})$ and no output. Sim samples random seeds $\{s_1^j\}_{j\in\mathcal{C}\cap\mathcal{G}_1}$ and key k_1 to simulate the seeds from P_1, and random seeds $\{s_2^j\}_{j\in\mathcal{C}\cap\mathcal{G}_2}$ and key k_2 to simulate the seeds and key from P_2. We argue that the view output by Sim is indistinguishable from the real one, because in the real execution, the seeds and keys are also sampled by the honest parties uniformly at random.

□

Complexity Analysis. Here, we give a more detailed complexity analysis of our MP-CPSI. We first recall the complexity of the KVS and EQZ circuit. The state-of-the-art KVS schemes [4, 39] achieves linear encoding/decoding time with linear structure length $\beta=(1+\varepsilon)n$ with small expansion factor $0<\varepsilon<0.3$. The non-free gates of EQZ circuits consist

of $2n+2$ input gates for l-bit Boolean sharing in a single round, and nl AND gates in $\log_2 l$ rounds, where an input gate requires communicating $2l$ bits (using lazy sharing [27]) and the AND gate requires communicating 3 bits in total.

For each client, the computation involves encoding and decoding over n items and has $O(n)$ complexity, and the communication cost consists of sending a KVS object of $(1+\varepsilon)nl = O(n\lambda)$ bits in one shot. Each pivot has $O(n(\lambda+m))$ computation complexity, including $2n$ PRP evaluations, $m/2$ PRG expansions, aggregations of $m/2$ objects in $\mathcal{V}^\beta$, encoding/decoding over n keys, and computations for $nl = O(n\lambda)$ AND gates. The communication cost consists of sending a KVS D' of size $\beta \cdot nl$ bits, and $3nl$ bits for nl AND gates in the EQZ circuit, resulting in $(4+\varepsilon)nl = O(n\lambda)$ bits in total.[3] Similarly, the leader computes $4n$ PRP values, aggregates $m-1$ objects in $\mathcal{V}^\beta$, decodes over $2n$ keys, and computes nl AND gates incurring $O(n(m+\lambda))$ computation complexity. The communication cost consists of sending $2n$ values each of $\mathsf{in} = O(\lambda)$ bits and sending $3nl$ bits in the EQZ circuit, and the total is $2n \cdot \mathsf{in} + 3nl = O(n\lambda)$ bits.

5.2 Supporting Associated Payloads

In many real-world applications, identifying the intersection is a precursor to computing over data associated with the intersection items (*e.g.*, salaries, medical records, or feature vectors). We show how to extend our MP-CPSI framework to support associated payloads, allowing parties to secret-share their payloads corresponding to the intersection items. The challenge is to align the payloads of each party correctly with the membership indicators, which are in the order of the leader's input set X^3, without leaking any more information.

We abstract the goal as enabling the parties to distribute their payloads among the core parties, aligned with the leader P_3's items. For example, in execution 1, each client P_j ($j \in \mathcal{G}_1$) inputs items X^j and payloads Y^j each of ρ bits. We expect that for the leader P_3's item x_i^3, if exists P_j's i'th item $x_{i'}^j = x_i^3$, then the associated payload $y_{i'}^j$ is secret-shared between the non-colluding P_2 and P_3, aligned with ith indicators.

Extension via KVS Reinterpretation. Our key idea is that, since P_1 knows k_1 and k_1' with P_3, we can employ P_1 to help the clients in $\mathcal{G}_1$ to reinterpret the KVS objects. Specifically, P_j encodes a KVS object of $\{(F(k_1, X^j), Y^j)\}$ and secret-shares it to P_1 and P_3. Then, P_3 first decodes on its k_1-pseudonyms $\{F(k_1, X^1)\}$ to learn the payload shares, and then re-encodes a KVS object using the different k_1'-pseudonyms $\{F(k_1', X^1)\}$. Such re-encoding helps reinterpret the clients' KVS objects on k_1-pseudonyms and payloads into new KVS objects on k_1'-pseudonyms and filtered payloads. This enables the P_2 and P_3 with k_1'-pseudonyms $\{F(k_1', X^3)\}$ to jointly decode and obtain the shared payloads. P_1 then secret-shares the re-encoded KVS object among P_2 and P_3, who decode using the common $\{F(k_1', X^3)\}$, and P_3 additionally decodes the shared KVS object from client P_j using $\{F(k_1, X^3)\}$.

In Appendix C.1, we provide the formal ideal functionality (Figure 7) and protocol (Figure 8) that output the Boolean secret shares of the payloads for ease of unified format, aligned with the leader's indices. For categorical attributes (*e.g.*, gender and geographical region), the values in Boolean sharing can be directly used in subsequent Boolean circuits (*e.g.*, selection queries). For numerical attributes, the values can be represented over the ring $\mathbb{Z}_2^\rho$, which is more naturally supported in modern CPUs [10, 32]. Given efficient mixed-mode computations in the honest-majority three-party model [32], we can efficiently convert payloads between the Boolean and arithmetic shares using dedicated conversion protocols, enabling versatile and efficient computations over the payloads.

Security and Complexity. The correctness and security proof can be derived from the basic MP-CPSI construction. Informally, in execution 1, P_1 additionally secret-shares its input-payload KVS to P_2 and P_3, which is secure as P_2 and P_3 are non-colluding. For each client's input-payload KVS E_j, the client first secret-shares it among the P_3 and P_1. Then, P_1 decodes its share and re-encodes a KVS Q_j, which is then secret-shared again among the P_2 and P_3. The secret payloads can be viewed as shared into three shares, such that P_3 can not learn anything from the two shares E_j^3 and Q_j^3, and neither can P_2 with only Q_j^2. This also holds for execution 2 with dual roles. The final 3PC phase reveals nothing to the parties.

Our extension adds about $4m$ KVS objects of $\beta n\rho$ bits and $mn\rho$ AND gates in the generic circuit, incurring additional $O(mn\rho)$ communication complexity. While prior protocols do not provide the payload extension [6, 43] or additional complexity [13], we roughly measure the additional communication costs incurred by the mn payloads, at least $O(m^2 n\rho)$ for [6], $O(mn(\log mn)^2\kappa\rho)$ for [43], and $O(m^3 n\kappa\rho)$ for [13].

6 Implementation and Evaluation

In this section, we present a comprehensive evaluation of our proposed MP-CPSI protocol. We compare its performance against state-of-the-art baselines across various dimensions, including communication overhead, runtime efficiency under constrained bandwidth, and scalability to large-scale settings (up to 120 parties each with 4 million items).

6.1 Experimental Setup

We conducted all experiments on a single Intel(R) Xeon(R) Platinum 8375C CPU @ 2.90GHz. To emulate network environments on a single machine, we use the Linux `tc` command on the loopback interface:

- **LAN Setting:** 10 Gbps bandwidth with 0.02 ms round-trip latency (RTT).

[3] We omit the negligible cost of each pivot distributing $\frac{m-1}{2}$ PRG seeds (less than $m \times 10^{-5}$ MB) that can be further amortized in multiple executions.

Table 2: Comparison of total running time (in seconds) between our protocol and the baselines without non-collusion assumptions under different networks. "N/A" denotes the execution failure. Special Case $m = 3$: $\mathsf{PairZS}_{\mathsf{hm}}$ and ours have apples-to-apples comparisons ($t = 1$), and no client role exists in our protocol (client cost = 0).

Network	Set Size (n)	Party (m)	$\mathsf{PairZS}_{\mathsf{dm}}$	$\mathsf{PairZS}_{\mathsf{hm}}$	Ours	
			Total (s)	Total (s)	Client (s)	Total (s)
LAN	2^{16}	3	5.84	0.87	0	0.10
		5	8.85	1.18	0.01	0.09
		10	25.99	1.92	0.01	0.10
		30	231.23	7.40	0.02	0.10
	2^{18}	3	22.27	3.24	0	0.30
		5	33.67	4.41	0.02	0.27
		10	102.59	7.45	0.04	0.33
		30	913.87	28.18	0.10	0.34
	2^{20}	3	87.08	13.88	0	0.99
		5	133.24	18.31	0.08	1.01
		10	414.59	30.83	0.14	1.12
		30	3683.40	139.03	0.42	1.14
WAN	2^{16}	3	127.59	3.93	0	2.37
		5	288.42	6.16	0.35	2.38
		10	1154.01	15.11	0.37	2.39
		30	N/A	147.97	0.84	2.81
	2^{18}	3	493.98	9.64	0	4.14
		5	1148.22	16.65	0.82	4.34
		10	4596.16	50.78	1.33	4.81
		30	N/A	547.01	3.41	6.76
	2^{20}	3	1957.70	35.22	0	10.17
		5	4654.78	68.78	3.07	10.69
		10	18461.22	209.93	4.69	12.95
		30	N/A	2203.77	13.84	22.94

- **WAN Setting:** 200 Mbps bandwidth with 96 ms RTT.
- **Restricted Setting:** 100 Mbps bandwidth with 96 ms RTT (only used for scalability evaluation of our protocol).

Note that since the constrained network resources are shared among m parties, the larger m leads to smaller bandwidth per party and longer communication time, which affects the running time measurements, and we discuss it in Appendix B.3.

Baselines and Implementations. We implemented our protocol in C++ using standard libraries. We defer the implementation details to Appendix B.2. We set security parameter $\kappa = 128$, $\lambda = 40$, the input and PRP output length $\mathsf{in} = 128$, and the value length $l = 64$. For comparison, we selected two baselines corresponding to the two major security models:

1. $\mathsf{PairZS}_{\mathsf{dm}}$ (Dishonest Majority): The state-of-the-art MP-CPSI baseline of Dong *et al.* [13] based on the pairwise zero-sharing generation and dishonest-majority MPC.
2. $\mathsf{PairZS}_{\mathsf{hm}}$ (Honest Majority): An honest-majority extension of Dong *et al.* [13] using their more efficient zero-sharing generation than [6] and honest-majority MPC.

Note that since the implementation of Dong *et al.* [13] does not provide a full-fledged MP-CPSI, we employ the high-performance MPC library `MP-SPDZ` to complement the suggested baselines; see Appendix B.2 for more details. We exclude the additional shuffling circuit required to hide the sensitive cuckoo hashing map in the baselines for secure subsequent computations [37]; including its prohibitive $O(m^2 n)$ cost [14] would make our advantage even larger. Also note that the entire circuit-based protocol of Su *et al.* [43] does not provide an open-source code. Looking ahead, they reported 160.39s runtime and 11457.65 MB communication costs for small-scale $m = 3$ parties and $n = 2^{16}$ 32-bit items ([43]'s Table 1), while the $\mathsf{PairZS}_{\mathsf{dm}}$ and $\mathsf{PairZS}_{\mathsf{hm}}$ only require 1876.3 MB and 33.9 MB, respectively, in the same settings with 64-bit items. Hence, we do not compare to it in experiments.

6.2 Runtime Comparison

The results in Table 2 highlight the advantage of our protocol that allows clients to perform local fast computations with one-shot communication and can directly go offline, while the core three parties execute the subsequent EQZ and more circuits. Under the LAN setting, our protocol requires only 0.1s for sets of size $n = 2^{16}$ and completes the large-scale $n = 2^{20}$ task in just 1.14s. Compared to the baselines without non-collusion assumptions, our protocol achieves up to $3231\times$ and $122\times$ improvement over $\mathsf{PairZS}_{\mathsf{dm}}$ and over $\mathsf{PairZS}_{\mathsf{hm}}$ for $m = 30$ and $n = 2^{20}$, respectively. In the WAN setting, the $\mathsf{PairZS}_{\mathsf{dm}}$ baseline fails in large setting of $m = 30$ due to the prohibitive communication (denoted as "N/A"). For $n = 2^{20}$, our protocol runs up to $1425\times$ and $96\times$ faster than the baseline $\mathsf{PairZS}_{\mathsf{dm}}$ (available $m = 10$) and $\mathsf{PairZS}_{\mathsf{hm}}$ ($m = 30$), respectively.

We note that our protocol has apples-to-apples comparisons to the $\mathsf{PairZS}_{\mathsf{hm}}$ in the special case of $m = 3$ and $t = 1$, where our protocol runs about $14\times$ and $3.5\times$ faster than $\mathsf{PairZS}_{\mathsf{hm}}$ for $n = 2^{20}$ under LAN and WAN, respectively.

6.3 Communication Cost Comparison

As shown in Table 3, our protocol significantly reduces the communication cost by orders of magnitude by leveraging the novel structure and lightweight techniques. In the same setting of $n = 2^{20}$ and $m = 30$, our protocol requires only 425.5 MB of total communication costs, about $10130\times$ and $98.7\times$ less than $\mathsf{PairZS}_{\mathsf{dm}}$ and $\mathsf{PairZS}_{\mathsf{hm}}$ without non-collusion assumptions, respectively. In the special case of $m = 3$, our protocol achieves about $4.1\times$ less communication cost than $\mathsf{PairZS}_{\mathsf{hm}}$ in the same model with $t = 1$. Furthermore, our protocol achieves communication cost independent of m per party in each role, and one-shot lightweight client-side communication of about 10.9 MB for large $n = 2^{20}$. The overall communication cost is linearly increased by the small client costs as more clients join the MP-CPSI.

Table 3: Comparison of communication costs (in MB) between our protocol and the baselines without non-collusion assumptions. "Total" indicates the sum of all the parties in offline preprocessing and the online phase.

Set Size (n)	Party (m)	PairZS$_{dm}$			PairZS$_{hm}$			Ours			
		Client	Leader	Total	Client	Leader	Total	Client	Pivot	Leader	Total
2^{16}	3	623.5	629.3	1876.3	9.8	14.3	33.9	0	2.3	3.6	8.2
	5	1241.2	1258.6	6223.3	15.2	28.5	89.3	0.7	2.3	3.6	9.6
	10	2785.4	2831.8	27900.4	31.3	66.9	321.7	0.7	2.3	3.6	12.9
	30	8931.8	9094.1	268117.4	85.0	209.6	2593.1	0.7	2.3	3.6	26.4
2^{18}	3	2489.9	2513.2	7493.0	39.4	57.3	136.2	0	9.2	14.5	32.9
	5	4956.6	5026.3	24852.8	61.1	114.6	358.8	2.7	9.2	14.5	38.4
	10	11123.3	11309.2	111419.3	125.9	268.6	1293.4	2.7	9.2	14.5	51.9
	30	35846.9	36497.6	1076057.5	341.9	841.4	10433.7	2.7	9.2	14.5	106.2
2^{20}	3	10023.8	10117.1	30164.8	158.5	230.1	547.1	0	36.9	58.0	131.8
	5	19954.3	20234.2	100051.6	245.5	460.1	1442.1	10.9	36.9	58.0	153.5
	10	44780.7	45527.0	448553.6	506.4	1078.8	5201.9	10.9	36.9	58.0	207.9
	30	143596.8	146208.9	4310516.3	1376.2	3379.5	41985.6	10.9	36.9	58.0	425.5

In Appendix B.4, we also provide the comparison between ours (assuming an honest majority among three parties) and online-only PairZS$_{dm}$ (assuming a Trusted Third-Party Dealer that preprocesses for m parties).

6.4 Scalability Evaluation

We further separately evaluate the scalability of our protocol in large-scale settings, extending the number of parties m from 60 up to 120, and set sizes up to $n = 2^{22}$ (approx. 4 million items), under a more restricted network (100Mbps). Table 4 reports the breakdown details of metrics by role (client, pivot, and leader). The communication costs of our protocol are linear in the number of parties, as the cost per role remains the same given a fixed set size n and is independent of the number of parties m, which aligns with the $O(n)$ complexity per party in our analysis. For $m = 120$ parties, each holding 4 million items ($n = 2^{22}$), our protocol remains practical, requiring 11.7s in the LAN and approximately 8 minutes (487s) in the restricted network with shared 100Mbps bandwidth.

7 Conclusion

This paper presents Butterfly, an efficient and scalable MP-CPSI protocol. We identify the efficiency bottlenecks of existing works stemming from full-scale m-party computations, and propose a novel blueprint that reduces MP-CPSI to core three-party computation, achieving linear generic MPC complexity independent of m. Our MP-CPSI firstly achieves practical efficiency and facilitates efficient subsequent computations, as well as one-shot lightweight clients.

Limitations. The main limitation lies in the security model, where we assume a non-colluding three-party committee among the m parties. Although related MP-(C)PSI works also make non-collusion assumptions to improve efficiency, employing non-colluding parties remains a widely acknowledged challenge in real-world scenarios, such as the recent privacy-preserving machine learning and collaborative database analytics advances relying on three servers with an honest majority. A minor limitation is the unbalanced burden across different roles, which also occurs in MPC protocols with hierarchical architectures. It could be natural for some asymmetric scenarios and facilitate one-shot edge clients, yet it may require interest-based bargaining in other scenarios.

Acknowledgment

We thank the anonymous reviewers and shepherd for their insightful suggestions and comments that substantially improved the manuscript. This work was supported by the National Cryptologic Science Fund of China under Grant 2025NCSF01010, and the Key Program of the National Natural Science Foundation of China under Grants 62272251, 62432012, and U25B2028.

Ethical Considerations

Stakeholder Analysis. We consider stakeholders in real-world deployments, the research process, and the community. *Real-World Deployments.* Protocol participants can benefit from the useful insights and analytics, reduced deployment

Table 4: Scalability evaluation of $\Pi_{\text{mp-cpsi}}$ with large party sizes ($m \in \{60, 80, 100, 120\}$) and large set sizes ($n \in \{2^{16}, 2^{18}, 2^{20}, 2^{22}\}$). We report Run Time (seconds) across three network settings and Communication Cost (MB) per role.

Set Size (n)		2^{16}				2^{18}				2^{20}				2^{22}			
Party (m)		60	80	100	120	60	80	100	120	60	80	100	120	60	80	100	120
LAN	Client	0.05	0.06	0.08	0.09	0.22	0.28	0.47	0.51	0.88	1.10	1.35	1.63	3.89	4.64	5.59	6.90
	Total	0.11	0.14	0.16	0.18	0.48	0.52	0.74	0.81	1.82	2.02	2.40	2.66	7.76	8.39	9.95	11.72
WAN	Client	2.12	2.70	3.53	3.87	7.62	9.74	12.0	14.0	28.0	37.3	46.4	56.0	111.2	147.8	184.4	222.7
	Total	3.80	4.39	5.45	5.80	10.1	12.7	15.9	17.0	37.2	45.5	56.6	65.5	140.3	180.1	215.3	250.9
Restricted	Client	3.64	4.45	5.76	6.64	15.2	19.7	23.0	27.4	55.3	73.6	91.9	112.6	220.5	293.9	367.8	440.7
	Total	5.48	6.47	7.84	8.69	18.2	22.8	26.8	31.5	69.5	84.7	103.9	124.4	265.7	339.7	412.8	487.3
Comm.	Client	0.67				2.71				10.88				43.59			
	Pivot	2.30				9.21				36.88				147.59			
	Leader	3.63				14.50				58.00				232.00			
	Total	46.6	60.0	73.5	86.9	187.6	241.8	296.1	350.3	752.0	969.6	1,287	1,405	3,012	3,884	4,755	5,627

and maintenance costs, and provable security guarantees. The possible data sources, such as individuals and other entities, may have their private identifiers and payloads collected by the participants. Our MP-CPSI provides an efficient and secure solution that encourages collectors/participants to adopt the privacy-preserving solution, thereby protecting the sensitive information of data sources.

Research Process. We respect intellectual contributions and maintain open communication. Our work involves no personally identifiable information or human subjects, and no real-world user data is used in our experiments.

Security Community. Researchers in the related areas or on a wider range of applications to privacy and security could be inspired by our technical details or high-level ideas. Standardization groups could consider including our MP-CPSI in the guidelines for privacy-preserving analytics.

Potential Harms Analysis. Our efficiency relies on an honest-majority three-party committee. While collusion would break security guarantees, this risk is mitigated in practice through structural independence, strict regulations, and reputational costs. We also eliminate Cuckoo hashing to prevent leakage of input distribution information. However, the subsequent circuit designers should still be careful, given the recent membership inference attacks on outputs, potentially by integrating differential privacy mechanisms into the MPC circuits.

Ethical Principles. Our work aligns with the Menlo Report.

Beneficence & Respect: MP-CPSI facilitates useful collaborative insights over the intersection and payloads, without revealing more information beyond the circuit outputs.

Justice: The lightweight overhead lowers deployment barriers, enabling resource-constrained entities like small companies and edge devices to participate in secure collaborations.

Law & Public Interest: Our work provides an efficient and secure solution, mitigating the adoption of insecure solutions that violate data regulations or compromise privacy.

Decision to Conduct and Publish. Motivated by the prohibitive costs of existing protocols, we developed MP-CPSI to enable practical, private collaborative analytics. We share these results to encourage real-world adoption and provide a technical foundation for future secure and efficient designs.

Open Science

We have provided the full implementation of our MP-CPSI protocol used for benchmarking to support open science. The source code, including scripts to reproduce the experiments, is available at https://doi.org/10.5281/zenodo.20282807. The artifact is based on the volepsi library and mainly includes: (1) `3PC.h` that provides the generic 3PC Boolean gates, EQZ, and shuffle circuits; (2) `MpCpsi.cpp` provides the implementation of our MP-CPSI protocol and extension of MP-PSI-CA in Appendix C.2, with slightly different party indices from the protocol for ease of implementing and launching the parties; (3) `fileBased.cpp` and `perf.cpp` provide the code for executing our protocol over specified input files and benchmarking the performance using random inputs, respectively. We provide the instructions for installation, test, and benchmark in parameter settings in `README.md`.

References

[1] Toshinori Araki, Jun Furukawa, Yehuda Lindell, Ariel Nof, and Kazuma Ohara. High-throughput semi-honest secure three-party computation with an honest majority. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 805–817, 2016.

[2] Gilad Asharov, Koki Hamada, Dai Ikarashi, Ryo Kikuchi, Ariel Nof, Benny Pinkas, Katsumi Takahashi, and Junichi Tomida. Efficient secure three-party sorting with applications to data analysis and heavy hitters. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 125–138, 2022.

[3] Jianli Bai, Xiangfu Song, Xiaowu Zhang, Qifan Wang, Shujie Cui, Ee-Chien Chang, and Giovanni Russello. Mostree: Malicious secure private decision tree evaluation with sublinear communication. In *Proceedings of the 39th Annual Computer Security Applications Conference*, pages 799–813, 2023.

[4] Alexander Bienstock, Sarvar Patel, Joon Young Seo, and Kevin Yeo. Near-optimal oblivious key-value stores for efficient psi, psu and volume-hiding multi-maps. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 301–318, 2023.

[5] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. Lightweight techniques for private heavy hitters. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 762–776. IEEE, 2021.

[6] 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 *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 1182–1204, 2021.

[7] Nishanth Chandran, Divya Gupta, and Akash Shah. Circuit-psi with linear complexity via relaxed batch opprf. *Proceedings on Privacy Enhancing Technologies*, 2022.

[8] Koji Chida, Koki Hamada, Atsunori Ichikawa, Masanobu Kii, and Junichi Tomida. Communication-efficient inner product private join and compute with cardinality. In *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*, pages 678–688, 2023.

[9] Arka Rai Choudhuri, Aarushi Goel, Matthew Green, Abhishek Jain, and Gabriel Kaptchuk. Fluid mpc: secure multiparty computation with dynamic participants. In *Annual International Cryptology Conference*, pages 94–123. Springer, 2021.

[10] Ronald Cramer, Ivan Damgård, Daniel Escudero, Peter Scholl, and Chaoping Xing. SPD$\mathbb{Z}_{2^k}$: Efficient MPC mod 2^k for dishonest majority. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 769–798. Springer, 2018.

[11] Emma Dauterman, Mayank Rathee, Raluca Ada Popa, and Ion Stoica. Waldo: A private time-series database from function secret sharing. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 2450–2468. IEEE, 2022.

[12] Daniel Demmler, Thomas Schneider, and Michael Zohner. Aby-a framework for efficient mixed-protocol secure two-party computation. In *NDSS*, 2015.

[13] Minglang Dong, Yu Chen, Cong Zhang, Yujie Bai, and Yang Cao. Multi-party private set operations from predicative zero-sharing. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, pages 4079–4093, 2025.

[14] Saba Eskandarian and Dan Boneh. Clarion: Anonymous communication from multiparty shuffling protocols. *Cryptology ePrint Archive*, 2021.

[15] Ying Gao, Yuanchao Luo, Longxin Wang, Xiang Liu, Lin Qi, Wei Wang, and Mengmeng Zhou. Efficient scalable multi-party private set intersection (-variants) from bicentric zero-sharing. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 4137–4151, 2024.

[16] Gayathri Garimella, Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. Oblivious key-value stores and amplification for private set intersection. In *Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part II 41*, pages 395–425. Springer, 2021.

[17] Craig Gentry, Shai Halevi, Hugo Krawczyk, Bernardo Magri, Jesper Buus Nielsen, Tal Rabin, and Sophia Yakoubov. Yoso: You only speak once: Secure mpc with stateless ephemeral roles. In *Annual International Cryptology Conference*, pages 64–93. Springer, 2021.

[18] Feng Han, Lan Zhang, Hanwen Feng, Weiran Liu, and Xiangyang Li. Scape: Scalable collaborative analytics system on private database with malicious security. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 1740–1753. IEEE, 2022.

[19] Meng Hao, Weiran Liu, Liqiang Peng, Hongwei Li, Cong Zhang, Hanxiao Chen, and Tianwei Zhang. Unbalanced circuit-psi from oblivious key-value retrieval. In *33rd USENIX Security Symposium (USENIX Security 24)*, 2024.

[20] Yan Huang, David Evans, and Jonathan Katz. Private set intersection: Are garbled circuits better than custom protocols? In *NDSS*, 2012.

[21] Mihaela Ion, Ben Kreuter, Ahmet Erhan Nergiz, Sarvar Patel, Shobhit Saxena, Karn Seth, Mariana Raykova, David Shanahan, and Moti Yung. On deploying secure computing: Private intersection-sum-with-cardinality. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 370–389. IEEE, 2020.

[22] Harish Karthikeyan and Antigoni Polychroniadou. Opa: one-shot private aggregation with single client interaction and its applications to federated learning. In *Annual International Cryptology Conference*, pages 319–353. Springer, 2025.

[23] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. Efficient batched oblivious prf with applications to private set intersection. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 818–829, 2016.

[24] 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.

[25] Nishat Koti, Varsha Bhat Kukkala, Arpita Patra, Bhavish Raj Gopal, Somya Sangal, et al. Ruffle: Rapid 3-party shuffle protocols. *Proceedings on Privacy Enhancing Technologies*, 2023.

[26] Nishat Koti, Mahak Pancholi, Arpita Patra, and Ajith Suresh. {SWIFT}: Super-fast and robust {Privacy-Preserving} machine learning. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2651–2668, 2021.

[27] Shuaishuai Li, Cong Zhang, and Dongdai Lin. Secure multiparty computation with lazy sharing. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 795–809, 2024.

[28] Guopeng Lin, Weili Han, Wenqiang Ruan, Ruisheng Zhou, Lushan Song, Bingshuai Li, and Yunfeng Shao. Ents: an efficient three-party training framework for decision trees by communication optimization. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 4376–4390, 2024.

[29] Yehuda Lindell. How to simulate it–a tutorial on the simulation proof technique. *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*, pages 277–346, 2017.

[30] Ranyang Liu, Xiaojie Guo, Tong Li, Xiaofeng Chen, and Zheli Liu. Efficient circuit-psi and extensions via distributed key-value store. *IEEE Transactions on Dependable and Secure Computing*, 2026.

[31] Xiang Liu and Ying Gao. Scalable multi-party private set union from multi-query secret-shared private membership test. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 237–271. Springer, 2023.

[32] Payman Mohassel and Peter Rindal. Aby3: A mixed protocol framework for machine learning. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 35–52, 2018.

[33] Payman Mohassel, Peter Rindal, and Mike Rosulek. Fast database joins and psi for secret shared data. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, pages 1271–1287, 2020.

[34] Ofri Nevo, Ni Trieu, and Avishay Yanai. Simple, fast malicious multiparty private set intersection. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 1151–1165, 2021.

[35] Benny Pinkas, Thomas Schneider, Gil Segev, and Michael Zohner. Phasing: Private set intersection using permutation-based hashing. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 515–530, 2015.

[36] Benny Pinkas, Thomas Schneider, Oleksandr Tkachenko, and Avishay Yanai. Efficient circuit-based psi with linear communication. In *Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part III 38*, pages 122–153. Springer, 2019.

[37] Benny Pinkas, Thomas Schneider, Christian Weinert, and Udi Wieder. Efficient circuit-based psi via cuckoo hashing. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 125–157. Springer, 2018.

[38] Benny Pinkas, Thomas Schneider, and Michael Zohner. Scalable private set intersection based on ot extension. *ACM Transactions on Privacy and Security (TOPS)*, 21(2):1–35, 2018.

[39] Srinivasan Raghuraman and Peter Rindal. Blazing fast psi from improved okvs and subfield vole. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 2505–2517, 2022.

[40] Peter Rindal and Phillipp Schoppmann. Vole-psi: fast oprf and circuit-psi from vector-ole. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 901–930. Springer, 2021.

[41] Yongha Son and Jinhyuck Jeong. Psi with computation or circuit-psi for unbalanced sets from homomorphic encryption. In *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*, pages 342–356, 2023.

[42] Lushan Song, Jiaxuan Wang, Zhexuan Wang, Xinyu Tu, Guopeng Lin, Wenqiang Ruan, Haoqi Wu, and Weili Han. Pmpl: A robust multi-party learning framework with a privileged party. In *Proceedings of the 2022 ACM SIGSAC conference on computer and communications security*, pages 2689–2703, 2022.

[43] Jiuheng Su, Zhili Chen, Haifeng Qian, and Junqing Gong. Efficient and scalable circuit-based protocol for multi-party private set intersection. In *European Symposium on Research in Computer Security*, pages 383–402. Springer, 2024.

[44] Sameer Wagh, Divya Gupta, and Nishanth Chandran. Securenn: 3-party secure computation for neural network training. *Proceedings on Privacy Enhancing Technologies*, 2019.

[45] Mingli Wu, Tsz Hon Yuen, and Kwan Yin Chan. {O-Ring} and {K-Star}: Efficient multi-party private set intersection. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 6489–6506, 2024.

[46] Cong Zhang, Yu Chen, Weiran Liu, Min Zhang, and Dongdai Lin. Linear private set union from {Multi-Query} reverse private membership test. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 337–354, 2023.

[47] Shuqing Zhang. Efficient vole based multi-party psi with lower communication cost. *Cryptology ePrint Archive*, 2023.

A Semi-Honest Model

For m parties $\{P_1,...,P_m\}$, let X^j denote each party P_j's input. With respect to an m-ary ideal functionality $\mathcal{F}$, we use $\mathcal{F}(X) = (\mathcal{F}_1(X^1),...,\mathcal{F}_m(X^m))$ denote the output of ideal functionality for input $X = (X^1,...,X^m)$, where $\mathcal{F}_j(X)$ is the ideal output of P_j for each $j \in [m]$. Let Π be a secure computation protocol for computing $\mathcal{F}$. We use $\mathsf{Output}^{\Pi}(X)$ denote the output of Π, and the view of P_j in protocol Π are respectively denoted by $\mathsf{View}_j^{\Pi}(X)$. For any t indices collection $\mathcal{C} = \{j_1,...,j_t\} \subseteq [m]$, let $\mathcal{F}_{\mathcal{C}}(X)$ denote the functionality outputs $(\mathcal{F}_{j_1}(X),...,\mathcal{F}_{j_t}(X))$ and $\mathsf{View}_{\mathcal{C}}^{\Pi}(X)$ denote the joint view $(\mathsf{View}_{j_1}^{\Pi}(X),...,\mathsf{View}_{j_t}^{\Pi}(X))$.

Definition 4. *Let $f = (f_1,...,f_m)$ be a m-ary functionality. We say that Π securely computes f in the presence of any semi-honest adversary if there exists a probabilistic polynomial-time algorithm Sim_i, such that for every legal $\mathcal{C} \subseteq [m]$:*

$$\{(\mathsf{Sim}(\mathcal{C},(X^{j_1},...,X^{j_t}),\mathcal{F}_{\mathcal{C}}(X)),\mathcal{F}(X))\}_X$$
$$\stackrel{c}{\equiv} \{(\mathsf{View}_{\mathcal{C}}^{\Pi}(X),\mathsf{Output}^{\Pi}(X))\}_X$$

B Implementation and Experiment Details

B.1 Equality-to-Zero Test Circuit

Given the membership-conditional triplet zero-sharings $\{(c_i^1,c_i^2,c_i^3)\}$, we can compute the membership indicators $\{\mathsf{b}_i\}$ by testing whether $c_i^1 \oplus c_i^2 \oplus c_i^3 = 0^l$ by evaluating the binary EQZ circuit. Specifically, the EQZ circuit accepts $\{c_i^j\}$ from each party P_j as input and proceeds as follows. Firstly, parties compute secret-shared XOR-sum $c_i = c_i^1 \oplus c_i^2 \oplus c_i^3$, which involves only local computations and no communication as XOR gates are free. Note that, for each l-bit c_i, it has $c_i = 0^l$ if and only if each bit $c_{i,j}$ ($j \in [l]$) equals to 0, where $(c_{i,1},...,c_{i,l})$ represents the bit decomposition of c_i. This is equivalent to testing whether all the inverted bits $\{\overline{c_{i,j}} \leftarrow \mathsf{NOT}(c_{i,j})\}$ equal to 1, where the NOT gates NOT are also free from communication. Based on that, parties can jointly compute indicator $\mathsf{b}_i = \overline{c_{i,1}} \wedge \overline{c_{i,2}} \wedge ... \wedge \overline{c_{i,l}}$, which costs $l-1$ AND gates. By employing binary-tree topology to arrange the AND gates, we can reduce the communication rounds to $\lceil \log_2(l) \rceil$. Since all the indicators are independent of each other and can be computed in parallel, the overall communication costs are $3n$ input gates and $n(l-1)$ AND gates in $\lceil \log_2(l) \rceil$ rounds.

B.2 Implementation Details

We utilize the same OKVS [39] as Dong *et al.* [13] implemented in volepsi (github.com/ladnir/volepsi), use the PRP and PRG in cryptoTools (github.com/ladnir/cryptoTools), and network communication interfaces provided in Coproto (github.com/ladnir/coproto). We also implement the necessary 3PC Boolean gates in the same framework, following the protocols in [1, 27, 32].

While the original implementation of Dong *et al.* [13] does not provide the implementation of their MP-CPSI protocol, we evaluate the baselines by separately evaluating: (1) the generation and aggregation of pairwise OPPRF-based zero-sharings, which have been provided in their implementation of MP-PSI; (2) the EQZ circuit phase using generic m-party computations, which we emulate in MP-SPDZ framework. The functionality $\mathcal{F}_{\mathsf{bpMZS}}$ in [13], which outputs membership zero-sharings over a field $\mathbb{F}$ among m parties. Specifically, we use $\mathbb{F} = GF(2^{64})$ to enable more efficient binary computations for the non-linear EQZ circuits than arithmetic computations. We use the binary honest-majority protocol `ccd-party` for $\mathsf{PairZS_{hm}}$ and binary dishonest-majority protocol `semi-bin-party` for $\mathsf{PairZS_{dm}}$. The suggested honest-

majority baseline $\mathsf{PairZS}_{\mathsf{hm}}$ achieves better performance than the prior honest-majority protocol [6]. Specifically, $\mathsf{PairZS}_{\mathsf{hm}}$ requires about 89.3 MB and 321.7 MB with $n = 2^{16}$ for $m = 5$ and $m = 10$, respectively, while the *partial evaluation* of [6] reports already 290.68 MB and 1166.28 MB, respectively (their Table 5 reports the performance of their Step 1-4 and omit the cost for Step 5-6).

B.3 Metrics Measurement

It is important to note that our emulation imposes a *shared bandwidth constraint*. The bandwidth limit set by `tc` applies to the total traffic on the interface. Consequently, in an experiment with m parties, the effective pairwise bandwidth is roughly $1/m$ of the nominal capacity, and they queue to send messages on the interface. This setup creates a highly congested network environment, serving as a rigorous stress test for protocol resilience in real-world scenarios such as crowded edge networks. Nevertheless, the previous works conducted their evaluations using the same `tc` command and settings (10Gbps LAN and 200Mbps WAN), and we believe that making traffic controls more accurate for multi-party protocols is a valuable practical problem.

Since the total bandwidth on the loopback device is shared in the emulated networks, the clients queue to send their messages, and they *do not* finish sending simultaneously. We think this could match the real-world concurrent scenario, where the leader communicates with many clients, but with only limited bandwidth and a competitive receiving interface. To accurately reflect the performance, we measure the "Client RunTime" as the *maximum* latency observed among all clients. Specifically, the timer starts when the protocol begins and stops when the leader confirms the receipt of the *last client's message*. We believe this metric can capture the tail latency of the slowest client and the parallel processing capability of the leader.

B.4 Comparison to Online-Only $\mathsf{PairZS}_{\mathsf{dm}}$

The work of Dong *et al.* [13] reports the online costs of their multi-party protocols and omits the prohibitive $O(m^2n)$ preprocessing costs for generating beaver triples and shuffle correlations, following the assumption that the necessary correlations are pre-computed by the m parties themselves (which also incurs computation and communication bottlenecks in large-scale settings as shown in Table 3), or distributed by a Trusted Third Dealer [31, 46] (a comparatively strong assumption that has less been justified in MPC). In contrast, our protocol assumes a well-established honest-majority three-party committee and decouples the circuit complexity from m, requiring no preprocessing overhead (except for the negligible constant costs of distributing small seeds). As shown in Table 5, our protocol requires about 1.7× less communication costs for $m = 3$ parties and 7.8× for $m = 30$.

Table 5: Comparison of communication costs to online-only $\mathsf{PairZS}_{\mathsf{dm}}$ (following the Trusted Dealer assumption of Dong *et al.* [13] and ignoring the prohibitive $O(m^2n)$ preprocessing costs for generating multiplication triples).

Set Size	**Party**	$\mathsf{PairZS}_{\mathsf{dm}}$ (Trusted Dealer)			**Ours** (HM 3P-Committee)			
(n)	(m)	Client	Leader	**Total**	Client	Pivot	Leader	**Total**
2^{16}	3	3.8	6.6	**14.3**	0	2.3	3.6	**8.2**
	5	3.8	13.2	**28.5**	0.7	2.3	3.6	**9.6**
	10	3.8	29.6	**64.2**	0.7	2.3	3.6	**12.9**
	30	3.8	95.5	**206.9**	0.7	2.3	3.6	**26.4**
2^{18}	3	15.4	26.5	**57.3**	0	9.2	14.5	**32.9**
	5	15.4	53.0	**114.6**	2.7	9.2	14.5	**38.4**
	10	15.4	119.3	**257.8**	2.7	9.2	14.5	**51.9**
	30	15.4	383.3	**828.6**	2.7	9.2	14.5	**106.2**
2^{20}	3	61.6	106.4	**229.5**	0	36.9	58.0	**131.8**
	5	61.6	212.8	**459.0**	10.9	36.9	58.0	**153.5**
	10	61.6	478.9	**1032.8**	10.9	36.9	58.0	**207.9**
	30	61.6	1539.0	**3320.1**	10.9	36.9	58.0	**425.5**

Parameters: There are m input parties $\{P_j\}_{j\in[m]}$ with input sets $\{X^j\}_{j\in[m]}$ and payload sets $\{Y^j\}_{j\in[m]}$ of size n.
Functionality:

1. Wait for input $X^j = \{x_1^j, ..., x_n^j\}$ and $Y^j = \{y_1^j, ..., y_n^j\}$ from P_j for $j \in [m]$;
2. For $i \in [n]$, sample $(\mathsf{b}_i^1, \mathsf{b}_i^2, \mathsf{b}_i^3) \in (\{0,1\})^3$ uniformly at random, such that $\mathsf{b}_i^1 \oplus \mathsf{b}_i^2 \oplus \mathsf{b}_i^3 = 1$ if $x_i^3 \in \bigcap_{j\in[m]} X^j$, and $\mathsf{b}_i^1 \oplus \mathsf{b}_i^2 \oplus \mathsf{b}_i^3 = 0$ otherwise;
3. For $i \in [n]$ and $j \in [m]$, computes $\{\mathsf{p}_i^j := y_{i'}^j\}$ if $\exists i' \in [n]$ such that $x_{i'}^j = x_i^3$ and $x_{i'}^j \in \bigcap_{j\in[m]} X^j$; otherwise, sets $\{\mathsf{p}_i^j := 0\}$. Samples $(\mathsf{p}_i^{j,1}, \mathsf{p}_i^{j,2}, \mathsf{p}_i^{j,3})$ uniformly at random, such that $\mathsf{p}_i^{j,1} \oplus \mathsf{p}_i^{j,2} \oplus \mathsf{p}_i^{j,3} = \mathsf{p}_i^j$.
4. Output $\{(\mathsf{b}_i^g, \mathsf{b}_i^{g+1})\}_{i\in[n]}$ and $\{(\mathsf{p}_i^{j,g}, \mathsf{p}_i^{j,g+1})\}$ to P_g for $g \in [3]$.

Figure 7: MP-CPSI Protocol with Payload Ideal Functionality $\mathcal{F}_{\mathsf{mp\text{-}cpsi\text{-}payload}}$ with additional parameters and steps marked in color green.

C MP-CPSI Extensions

C.1 Functionality and Protocol with Payloads

We provide the formal ideal functionality in Figure 7 and protocol in Figure 8. The additional procedures for supporting the associated payloads over the MP-CPSI functionality and protocol without payloads are highlighted in green.

C.2 Example Operation Extensions

Plain Multi-Party PSI. The most basic extension is to reveal the plaintext intersection items. In our protocol, the leader P_3 and pivot P_1 already hold the shares b_i^3 and b_i^1 of the indicator b_i. To reconstruct the intersection, the other pivot P_2 simply sends its share $\{b_i^2\}_{i\in[n]}$ to the leader. The leader computes $\mathsf{b}_i = \mathsf{b}_i^1 \oplus \mathsf{b}_i^2 \oplus \mathsf{b}_i^3$ and outputs $I = \{x_i^3 \mid \mathsf{b}_i = 1\}$. The reconstruction only requires sending n bits in total, hence the performance is almost the same as our MP-CPSI.

Parameters: (...) Payload length ρ, payload space $\mathcal{P} \leftarrow \{0,1\}^\rho$, and a PRG $G' : \{0,1\}^\kappa \to \mathcal{P}^\beta$.
Input: P_j inputs set $X^j = \{x_1^j, ..., x_n^j\}$ and corresponding payload $Y^j = \{y_1^j, ..., y_n^j\}$.
Protocol:

Execution 1

1. (...)
2. (...)
3. (...) P_1 and P_3 also expand the seed s_1^3 and obtain $E_1^3 \leftarrow G'(s_1^3)$;
4. (...)
5. (...) P_1 encodes $E_1^2 \leftarrow E_1^3 \oplus \mathsf{Encode}(\{(F(k_1', x_i^1), y_i^1\})$ and sends to P_2;
6. (...) P_j encodes input-payload KVS object $E_j \leftarrow \mathsf{Encode}(\{(F(k_1, x_i^j), y_i^j)\}_{i\in[n]})$, secret-shares E_j as E_1^j and E_j^3 such that $E_j^1 \oplus E_j^3 = E_j$, and sends E_j^1 to P_1 and E_j^3 to P_3. P_1 encodes $Q_j \leftarrow \mathsf{Encode}(\{(F(k_1', x_i^1), \mathsf{Decode}(E_j^1, F(k_1, x_i^1)))\}_{i\in[n]})$ for $j \in \mathcal{G}_1$, continually expands the seed s_1^3 and obtains $Q_j^3 \leftarrow G'(s_1^3)$, and sends $Q_j^2 := Q_j \oplus Q_j^3$ to P_2. P_3 locally continues expand the seed s_1^3 and obtains Q_j^3;
7. (...) For $j \in \mathcal{G}_1$, P_2 decodes $\{q_i^j \leftarrow \mathsf{Decode}(Q_j^2, F(k_1', x_i^3))\}_{i\in[n]}$ and P_3 decodes $\{p_i^j \leftarrow \mathsf{Decode}(Q_j^3, F(k_1', x_i^3)) \oplus \mathsf{Decode}(E_j^3, F(k_1, x_i^3))\}_{i\in[n]}$. For P_1's payloads, P_2 decodes $\{q_i^j \leftarrow \mathsf{Decode}(E_1^2, F(k_1', x_i^3))\}_{i\in[n]}$ and P_3 decodes $\{p_i^j \leftarrow \mathsf{Decode}(E_1^3, F(k_1', x_i^3))\}_{i\in[n]}$;
8. (...)

Execution 2

1. (...)
2. (...)
3. (...) P_2 and P_3 also expand the seed s_2^3 and obtain $E_2^3 \leftarrow G'(s_2^3)$;
4. (...)
5. (...) P_2 encodes $E_2^1 \leftarrow E_2^3 \oplus \mathsf{Encode}(\{(F(k_2', x_i^2), y_i^2\})$ and sends to P_1;
6. (...) P_j encodes input-payload KVS object $E_j \leftarrow \mathsf{Encode}(\{(F(k_2, x_i^j), y_i^j)\}_{i\in[n]})$, secret-shares E_j as E_2^j and E_j^3 such that $E_j^2 \oplus E_j^3 = E_j$, and sends E_j^2 to P_2 and E_j^3 to P_3. P_2 encodes $Q_j \leftarrow \mathsf{Encode}(\{(F(k_2', x_i^2), \mathsf{Decode}(E_j^2, F(k_2, x_i^2)))\}_{i\in[n]})$ for $j \in \mathcal{G}_2$, continually expands the seed s_2^3 and obtains $Q_j^3 \leftarrow G'(s_2^3)$, and sends $Q_j^1 := Q_j \oplus Q_j^3$ to P_1. P_3 locally continues expand the seed s_2^3 and obtains Q_j^3;
7. (...) For $j \in \mathcal{G}_2$, P_1 decodes $\{q_i^j \leftarrow \mathsf{Decode}(Q_j^1, F(k_2', x_i^3))\}_{i\in[n]}$ and P_3 decodes $\{p_i^j \leftarrow \mathsf{Decode}(Q_j^3, F(k_2', x_i^3)) \oplus \mathsf{Decode}(E_j^3, F(k_2, x_i^3))\}_{i\in[n]}$. For P_2's payloads, P_1 decodes $\{q_i^j \leftarrow \mathsf{Decode}(E_2^1, F(k_2', x_i^3))\}_{i\in[n]}$ and P_3 decodes $\{p_i^j \leftarrow \mathsf{Decode}(E_2^3, F(k_2', x_i^3))\}_{i\in[n]}$;

8. (...)
9. (...) P_1 inputs $\{q_i^j\}_{i\in[n], j\in\mathcal{G}_2\cup\{2\}}$, P_2 inputs $\{q_i^j\}_{i\in[n], j\in\mathcal{G}_1\cup\{1\}}$, P_3 inputs $\{p_i^j\}_{i\in[n], j\in[m]\setminus\{3\}}$ and $\{y_i^3\}_{i\in[n]}$ into $\mathcal{F}_{\mathsf{3pc}}$, which computes $\{\mathsf{p}_i^j := \mathsf{b}_i \text{ AND } (p_i^j \oplus q_i^j)\}_{i\in[n], j\in[m]\setminus\{3\}}$ and $\{\mathsf{p}_i^3 := \mathsf{b}_i \text{ AND } y_i^3\}_{i\in[n]}$. For $g \in \{1,2,3\}$, P_g obtains $\{(\mathsf{p}_i^{j,g}, \mathsf{p}_i^{j,g+1})\}_{i\in[n], j\in[m]}$ with random shares $\mathsf{p}_i^{j,1} \oplus \mathsf{p}_i^{j,2} \oplus \mathsf{p}_i^{j,3} = \mathsf{p}_i^j$.

Figure 8: MP-CPSI Protocol with Payload $\Pi_{\mathsf{mp\text{-}cpsi\text{-}payload}}$ extended from $\Pi_{\mathsf{mp\text{-}cpsi}}$ in Figure 6. The additional parameters and procedures in each step are marked in color green, while the common are omitted and denoted as "(...)".

Multi-Party PSI Cardinality (MP-PSI-CA). This functionality computes the size of the intersection $|I| = \sum \mathsf{b}_i$ without revealing the items. A naive circuit is to convert the Boolean shares $\{\mathsf{b}_i\}$ to arithmetic shares via B2A and sum them up. However, a more efficient approach is to *shuffle* the Boolean shares and reveal the shuffled bits to the leader. The leader counts the number of 1s to learn the cardinality, while the shuffle breaks the link to hide the intersection. We can execute a *3-party secret-shared shuffle* on the final aggregated indicators using the efficient protocol of Asharov et al. [2]. This only requires $6n$ bits of communication in 3 rounds.

Differentially Private Membership Indicators. It is natural to extend MP-PSI to satisfy differential privacy by perturbing the membership indicators, ensuring that the presence of any single individual intersection item is plausibly deniable.

In our setting, we can add noise locally without complex MPC noise-generation circuits. Recall that the leader holds shares $\mathsf{b}_i^1, \mathsf{b}_i^3$. The pivots P_1 and P_2 hold b_i^2, which is unknown to the leader. Hence, P_1 and P_2 can locally apply a randomized response mechanism (flipping the bit with probability p) to generate perturbed shares $\widetilde{\mathsf{b}_i^2}$. When the shares are reconstructed, the indicator $\widetilde{\mathsf{b}_i} = \mathsf{b}_i^1 \oplus \widetilde{\mathsf{b}_i^2} \oplus \mathsf{b}_i^3$ is the result of a randomized response on the secret indicator. While this approach requires no communication, the pivots know which indicators are perturbed and should be used carefully in subsequent computations. For example, in the MP-PSI and cardinality extensions, the final results are only learned by the leader. Based on the *post-processing property* of DP, which states that any computation performed on the output of a DP mechanism remains differentially private, we can obtain the differentially private MP-PSI(-CA) protocol combining the above extensions.

General Computation over Intersection with Payload. By supporting associated payloads, our MP-CPSI enables general subsequent computations in the downstream tasks. For example, database-style queries (e.g., "`Sum` P_A`.Amount of intersection items where` P_B`.Attribute = target`") can be implemented by combining the intersection indicators with different parties' payloads, using mixed-mode computations. Utilizing the payload extension described above, parties input their payload shares into a comparison circuit to generate attribute condition bits c_i. The composed selection bit is computed as $sel_i = \mathsf{b}_i \wedge c_i$ using the AND gate, and the final aggregation can be computed as $\sum sel_i \cdot u_i$ using Bit-to-Arithmetic conversion (Bit2A) and multiplication gates, where u_i is the shared Amount payloads in arithmetic form. This facilitates fine-grained analysis over the multi-party intersection without revealing non-matching items. The important advantage is that all these general computations can be performed using efficient honest-majority three-party mixed-mode computations [32].