PICS: Private Intersection over Committed (and reusable) Sets

Aarushi Goel
Rutgers University

Peihan Miao
Brown University

Phuoc Van Long Pham
Brown University

Satvinder Singh
Purdue University

Abstract

Private Set Intersection (PSI) enables two parties to compute the intersection of their private sets without revealing any additional information. While maliciously secure PSI protocols prevent many attacks, adversaries can still exploit them by using inconsistent inputs across multiple sessions. This limitation stems from the definition of malicious security in secure multiparty computation, but is particularly problematic in PSI because: (1) real-world applications—such as Apple's PSI protocol for CSAM detection and private contact discovery in messaging apps—often require multiple PSI executions over consistent inputs, and (2) the PSI functionality makes it relatively easy for adversaries to infer additional information.

We propose *Private Intersection over Committed Sets (PICS)*, a new framework that enforces input consistency across multiple sessions via committed sets. Building on the state-of-the-art maliciously secure PSI framework (i.e., VOLE-PSI [Rindal and Schoppmann, EUROCRYPT 2021]), we present an efficient instantiation of PICS using lightweight cryptographic tools. Our protocol achieves strong receiver-side input consistency (i.e., the receiver uses the exact committed set) and weak sender-side input consistency (i.e., the sender cannot inject new elements into the committed set but can potentially use a subset of the committed set). We implement our protocol to demonstrate concrete efficiency. Compared to VOLE-PSI, our communication overhead is a small constant between $1.57-2.04\times$ for set sizes between $2^{16}-2^{24}$, and the total end-to-end running time overhead is $1.22-1.98\times$ across various network settings.

1 Introduction

Private set intersection (PSI)—a special case of secure multiparty computation (MPC) [39,72]—enables two mutually distrusting parties, each holding a private input set, to jointly compute an intersection of their sets without revealing anything beyond the intersection itself. This seemingly simple functionality has found numerous applications, including DNA testing and pattern matching [70], testing of sequenced human genome [10], password breach detection [69], mobile private contact discovery [31,45], and online advertising measurement [43], among others.

Last few decades have witnessed enormous progress towards efficient realization of PSI, both in the semi-honest and malicious security settings. State-of-the-art PSI protocols [17,37,63,64] only use lightweight cryptographic tools such as vector oblivious linear evaluation (VOLE) [19,20], cryptographic hash functions, and symmetric-key operations, resulting in extremely fast implementations.

Maliciously secure PSI protocols protect against the strongest class of adversaries, i.e., those who may arbitrarily deviate from the protocol description. While such protocols provide strong security guarantees in a single execution of PSI over a given input set, they fail to address concerns that may arise when parties are expected to use consistent inputs across multiple PSI executions. Indeed, this is a requirement in many real-world applications of PSI.

For example, when a service provider runs PSI repeatedly with many users, there is no guarantee that it uses the same input set each time. Since the standard definition of security against malicious adversaries allows them to arbitrarily choose their inputs, it may seem less critical in some MPC scenarios. However, it can have serious implications in the context of PSI. Inconsistent inputs across sessions can lead to unfair treatment, discrimination, or leaking more information about users' inputs.

Below, we illustrate these issues through examples of real-world PSI deployments. A PSI protocol typically involves a *sender* and a *receiver*, with only the receiver learning the output. We discuss how input inconsistency can potentially impact both parties.

Inconsistent Inputs from a Malicious Receiver. The PSI protocol proposed by Apple in 2021 [16] to detect child sexual abuse material (CSAM) sparked worldwide debate and controversy, leading to the eventual shutdown of the initiative. In their approach, Apple acted as the receiver running a

(fuzzy, threshold variant of) PSI, comparing a set of known CSAM images against user content stored in iCloud Photos. A critical concern raised by many [7,65] was that Apple could use different image sets for different users. This could result in discrimination among users or extraction of additional information about users' legitimate data—effectively enabling a form of client-side surveillance. Ideally, we would like to prevent Apple (the PSI receiver) from arbitrarily modifying its CSAM dataset across users, particularly by injecting benign images into it.

Inconsistent Inputs from a Malicious Sender. In password breach detection [69], a service provider uses a fixed set of breached passwords to run PSI with millions of users, each holding their own set of passwords. In this case, the service provider acts as the PSI sender, yet there is no guarantee that it uses a consistent set across all users. This inconsistency can potentially lead to discrimination or unfair treatment. As in the previous case, we aim to prevent the the PSI sender from arbitrarily modifying its set across executions, especially by injecting unbreached passwords.

These vulnerabilities stem from the standard definition of *malicious security* in MPC, which allows adversaries to choose arbitrary inputs. While this is a general limitation in MPC, it is particularly problematic in PSI because (1) a single server is often expected to use a consistent set to run PSI repeatedly with many clients, and (2) the PSI functionality makes it easy to extract additional information. This raises the question:[1]

Can we enforce input consistency across multiple PSI executions, without compromising on concrete efficiency?

1.1 Our Framework

We address the question by introducing a new framework for *Private Intersection over Committed (and reusable) Sets (**PICS**)*, which ensures that the adversary uses consistent input sets across multiple PSI executions.

Similar to a standard PSI protocol, PICS is an interactive protocol executed between two parties—a *sender* and a *receiver*. It consists of two phases:

- Committing Phase: Both parties publicly commit to their respective input sets using a *succinct* commitment scheme.
- Intersection Phase: The sender and receiver engage in a maliciously secure PSI protocol over their committed input sets while ensuring that neither party can inject new elements into their committed sets. Only the receiver gets the output of this intersection.

We emphasize that the Committing Phase is generally viewed as a one-time setup per party. Once the public commitments are established, both the sender and receiver can engage in the Intersection Phase multiple times with different counter-parties.

We remark that our framework only focuses on preventing malicious adversaries from *injecting* new elements into their committed sets. This suffices for our intended applications discussed in Section 1.3. However, one could consider a stronger requirement, where adversaries are forced to use the *exact same* set in each execution of the Intersection Phase. In fact, looking ahead, our protocol achieves this stronger guarantee against a malicious receiver.

1.2 Our Contributions

Our contributions are as follows:

1. We introduce a new framework for Private Intersection over Committed (and reusable) Sets (PICS), as elaborated above.
2. We present an efficient instantiation of this framework using lightweight cryptographic tools, while maintaining *black-box* use of cryptography. Our construction composes VOLE-based PSI [17,37,63,64] with FRI-based proof systems [13,15] in a novel way. Our protocol achieves strong receiver-side input consistency (i.e., the receiver uses the exact committed set) and weak sender-side input consistency (i.e., the sender cannot inject new elements into the committed set but can potentially use a subset of the committed set). Compared to the underlying maliciously secure VOLE-PSI, our techniques only introduce an additional polylogarithmic (in the set size) computational overhead, while preserving the round complexity.
3. We implement our protocol to demonstrate concrete efficiency. The commitment size of each party is only 32 bytes. The communication overhead of PICS, compared to VOLE-PSI, is a small constant between $1.57-2.04\times$ for set sizes between $2^{16}-2^{24}$, and the total end-to-end running time overhead is $1.22-1.98\times$ over LAN and WAN networks.
4. We compare our performance with prior work on Authorized PSI [35] and client-server PSI [66,67], which *partially* addressed this problem, although with notable limitations (see Section 5.2 for details).

1.3 Example Applications

Running PSI over committed inputs is critical in many applications. In this section, we elaborate on applications of PICS that assures both parties of interest that the other (potentially malicious) party does not inject harmful elements into their committed set. Furthermore, the succinct commitment makes it possible to store a "snapshot" of the set at each time period, enabling easier further audit.

The Apple's PSI Protocol. Apple introduced a PSI protocol [16] to combat child sexual abuse in 2021. At its core, the

[1] A detailed discussion on related works can be found in Section 6.

protocol finds a (fuzzy) intersection between a large dataset (provided by NCMEC and other child-safety organizations [8]) with the users' set of photos on iCloud. Albeit with a good intention, the protocol sparked controversy as it allows the dataset holder (i.e., Apple) to inject arbitrary data when running the protocol, and in the worst case, learn the entire image library of users. In this case, PICS provides protection against a *malicious receiver*.

Private Mobile Contact Discovery. When a new user registers for a messaging app, the app provider would like to find out which of the user's contacts have also already registered with the app. This can be achieved in a privacy-preserving manner using PSI [31,45]. However, the app provider (acting as a PSI receiver) may learn more information about the users' contacts by providing inconsistent inputs across different PSI executions. For instance, in the U.S., for a newly registered phone number, the set of all numbers that share the same area code with it (such as +1 212 xxx xxxx) is much smaller than the set of all possible ten-digits phone numbers. By injecting elements from this smaller set into the PSI protocol, the service provider can potentially infer the new user's contacts, especially those who live nearby, even if they have not registered with the app. As in the previous example, PICS protects users against a *malicious receiver*.

Password Breach Detection. Many web browsers [3–6] allow users to check whether their passwords are included in a dataset of breached passwords, in a privacy-preserving way by running PSI. Consider a threat model where a powerful adversary seizes control of such a service provider and can also monitor a user's internet activity. In this setting, the adversary may learn more information about the users' (unbreached) passwords. Specifically, the adversary injects a set of unbreached passwords (curated carefully using prior knowledge about a user) into the password dataset, and then monitors whether the user attempts to change their password on any platform. In this example, PICS protects users against a *malicious sender*, preventing this kind of attack.

California Delete Act. The California Delete Act (Senate Bill 362) [1, 2] plans to grant Californians the right to demand that data brokers erase their personal information from their records. At the core of this bill is a requirement for a *centralized deletion platform*, which provides an interface for customers to demand deletion from *all* registered data brokers.[2] To enable each data broker to learn which data to remove from their database, without learning other consumers on the list, a PSI protocol is particularly suitable. The centralized deletion platform acts as a PSI sender with a set of consumers who requested deletion, while the data brokers act as PSI receivers, each holding a set of consumers in their database.

The consistency of inputs from both the sender and the receivers is crucial to ensure a proper implementation of the legislation. Note that malicious intention to inject elements into their sets is not entirely unrealistic: the sender could inject arbitrary names into their set without consent, potentially motivated by censorship. A receiver could also lie about their set during a later audit, which violates this additional measure enforced by the SB-362 bill.

To the best of our knowledge, it is still unclear how this bill will be deployed. We position PICS as a potential technical solution for this problem, due to its lightweight set commitment (a single Merkle hash), fast committing time, and low overhead compared to state-of-the-art PSI protocols. We estimate that, given California's population of 40 million, it takes less than 5 minutes for both parties to compute set commitments on an AWS instance. Additionally, our protocol support the following extensions (see Section 4.3 for details), would further solidify other aspects of the legislation:

- *Fast and verifiable refreshing.* Each data broker is required to access the deletion mechanism at least once every 45 days, hence it requires the central platform to regularly refresh the set of consumers to be deleted. PICS' fast committing time makes timely updates practical. Furthermore, it can support proof of consistency between the two sets before and after each refresh.
- *Transparency.* With the FRI commitment scheme, PICS can additionally support membership testing, allowing a consumer to verify that their name is included in the committed set.

Remark. We note that while in the rest of this paper we present our construction assuming both the sender and receiver commit to their respective inputs apriori, our protocol can be easily stripped down to only having a single party commit to their inputs in advance, if the application so demands.

1.4 Our Techniques

In this section, we outline the key ideas behind our approach to designing a concretely efficient PICS protocol.

Strawman Approach and its Drawbacks. A natural way to instantiate our framework would be to use the standard *commit-and-prove* GMW paradigm [38]. Specifically, parties can run any maliciously secure PSI protocol during the Intersection Phase, while attaching a generic zero-knowledge proof [40] to each message to demonstrate that it was computed honestly and is consistent with the committed inputs.

While theoretically sound, this approach introduces significant computational overheads. Recall that a PSI protocol involves cryptographic operations, and proving that all messages in such a protocol were honestly computed using

[2] The bill "allows a consumer, through a single verifiable consumer request, to request that every data broker that maintains any personal information delete any personal information related to that consumer held by the data broker or associated service provider or contractor."

committed inputs typically requires a *non-black-box* use of cryptography, making the protocol impractical.

1.4.1 Starting Idea

We first argue the strawman approach that requires proving consistency for every computed message is excessive.

GMW-Paradigm over a Malicious PSI is Wasteful. The standard definition of security against a malicious adversary in a PSI protocol inherently guarantees that if the protocol ends successfully (i.e., with the receiver learning the output), then all messages sent by the adversary must be consistent with *some* input. We elaborate on this below.

In the real-ideal world security paradigm [51], this is established by demonstrating the existence of a polynomial-time simulator that can *extract* the adversary's effective input. This extraction is feasible because, after a certain number of message exchanges, the adversary's input becomes "implicitly committed" within the protocol. That is, the adversary's strategy up to that point effectively binds them to a specific input. Consequently, if the receiver obtains an output, then all subsequent messages sent by the adversary must also be consistent with the extracted input.

Since malicious security in a PSI protocol (that completes successfully) already ensures that *all* messages in the protocol must have been honestly and consistently computed with respect to some input, requiring the parties to additionally prove that *every* message is computed consistently with their committed inputs is redundant.

Instead, it suffices to select a *subset of messages* (which depend on the entire input) and prove their consistency with the committed inputs. The consistency of the remaining messages then follows from the malicious security of the underlying PSI protocol.

Our Strategy. To achieve a concretely efficient PICS protocol, we build upon the state-of-the-art maliciously secure VOLE-PSI protocol [17, 37, 63, 64]. Leveraging the above observation, we first identify, for both the sender and the receiver, the subset of messages in the underlying VOLE-PSI protocol that implicitly bind them to their respective inputs. We refer to these as their *implicit commitments*. We then design efficient zero-knowledge proofs that enable the sender and receiver to demonstrate that their implicit commitments are consistent with their *explicit input commitments* from the Committing Phase. Importantly, we only make a *black-box* use of cryptography.

1.4.2 Overview of VOLE-PSI [64]

Before detailing our implementation of the above strategy, we review the construction of VOLE-PSI. This protocol proceeds as follows:

- *VOLE Correlation:* As the name suggests, VOLE-PSI [64] is based on a Vector Oblivious Linear Evaluation (VOLE) sub-protocol. In the first step, the sender and receiver execute a VOLE sub-protocol [71], obtaining the following correlated values: the sender receives a scalar Δ and a vector $\mathbf{B}$, while the receiver obtains two vectors $(\mathbf{A}, \mathbf{C})$ such that $\mathbf{C} = \Delta \cdot \mathbf{A} + \mathbf{B}$.
- *Receiver's Message:* The receiver computes an Oblivious Key-Value Stores (OKVS) encoding (see Section 2.2) of his input set, denoted as $\mathbf{P}$. The receiver then sends $\mathbf{A}' = \mathbf{A} + \mathbf{P}$ to the sender.
- *Sender's Final Message:* The sender processes the receiver's message using Δ and $\mathbf{B}$, obtaining values $t_1, \ldots, t_n$, where n is the size of the sender's set. The sender then transmits $H(x_1 \| t_1), \ldots, H(x_n \| t_n)$ to the receiver, where H is a random oracle (instantiated as a hash function), and $x_1, \ldots, x_n$ are the sender's input elements. This is the only message from the sender that directly depends on her input.
- *Receiver's Output:* Finally, the receiver processes the sender's message using $\mathbf{A}, \mathbf{C}$, and $\mathbf{P}$ to compute the intersection.

We provide some intuition on the correctness and security of the VOLE-PSI protocol. After receiving $\mathbf{A}'$ from the receiver, the sender computes an OKVS encoding as $\mathbf{K} := \Delta \cdot \mathbf{A}' + \mathbf{B} = \Delta \cdot \mathbf{P} + \mathbf{C}$. For each element x_i in her set, the sender computes $t_i := \mathsf{Decode}(\mathbf{K}, x_i) - \Delta \cdot \tilde{H}(x_i)$, where Decode is the decode function of OKVS and $\tilde{H}$ is a random oracle. By the linearity of OKVS constructions, $t_i = \Delta \cdot (\mathsf{Decode}(\mathbf{P}, x_i) - \tilde{H}(x_i)) + \mathsf{Decode}(\mathbf{C}, x_i)$. On the other hand, the receiver can compute $\mathsf{Decode}(\mathbf{C}, x_i)$ for any x_i. If x_i is in the receiver's set, then $\mathsf{Decode}(\mathbf{P}, x_i) = \tilde{H}(x_i)$ by the OKVS property of $\mathbf{P}$. Thus the first term in t_i cancels out, allowing the receiver to compute t_i and identify x_i in the intersection. If x_i is not in his set, $\mathsf{Decode}(\mathbf{P}, x_i) \neq \tilde{H}(x_i)$ with overwhelming probability. Therefore, the receiver would need to guess Δ to compute t_i, which is computationally infeasible. We refer to [64] for more details of the proof.

1.4.3 Consistency of Receiver's Inputs

We begin by explaining how the receiver's implicit commitment is defined and how we ensure its consistency with the inputs committed to during the Committing Phase.

Receiver's Implicit Commitment. It is clear that $\mathbf{A}'$ is the only message sent by the receiver that depends on his entire input. Since $\mathbf{A}'$ is computed using $\mathbf{P}$, which in turn is derived from his inputs, this message binds the receiver to his inputs and can be used as an implicit commitment.

Proving Consistency Between Initial and Implicit Input Commitments. As discussed earlier, our goal is to design a mechanism that allows both parties to prove consistency between their implicit and explicit inputs, while ensuring

black-box use of the cryptographic operations involved in both commitment schemes. To achieve this, two key design choices must be made:

- *Choice of Explicit Initial Commitment:* While the implicit commitment is determined by the underlying PSI protocol, we have the flexibility to choose the commitment scheme for committing to the inputs in the Committing Phase.
- *Custom Commit-and-Prove Style Proofs:* Since we have no flexibility in choosing an implicit commitment, it is unclear whether any existing commit-and-prove style proofs can be used to ensure consistency. Therefore, we must design custom commit-and-prove style black-box consistency proofs for both the sender and the receiver.

Crucially, we aim to make these design choices while keeping the underlying VOLE-PSI protocol *intact*, augmenting it only with additional messages and introducing minimal overhead.

Choice of Explicit Initial Commitment Scheme. Observe that we need to commit to a *set* of elements in the Committing Phase. For this, we would require a *vector commitment* scheme. Since vectors can be cast as polynomials using polynomial interpolation, polynomial commitment also serves as an easy replacement for vector commitment.

We use a polynomial commitment instead of a vector commitment on the receiver side due to two key reasons: (1) recent advancements in the field of succinct non-interactive arguments of knowledge (SNARKs) have demonstrated that it is possible to design efficient proofs for proving statements about values committed using a polynomial commitment scheme, while maintaining black-box use of cryptography; (2) we observe that the receiver's implicit commitment to his input set in the VOLE-PSI protocol satisfies nice linearity properties due to VOLE. These, when formulated in terms of polynomials, allow the design of succinct and efficient proofs.

We have the receiver commit to its OKVS-encoded input vector $\mathbf{P}$ in the Committing Phase using a polynomial commitment scheme. This serves as the receiver's *explicit* commitment. While our protocol is compatible with any polynomial commitment scheme, we instantiate our framework using the FRI-based scheme of [13], which relies solely on symmetric-key operations in the random oracle model.

Custom Commit-and-Prove Style Proofs. Next, we need to design a proof that demonstrates that the polynomial commitment to $\mathbf{P}$ is consistent with the receiver's *implicit* commitment. However, it is not immediately clear how to achieve this, since the receiver's input is not implicitly committed via a polynomial commitment. Instead, his implicit commitment is $\mathbf{A}' = \mathbf{A} + \mathbf{P}$, where $\mathbf{A}$ comes from VOLE and $\mathbf{P}$ is an OKVS encoding. We now need to prove consistency between $\mathbf{A}'$ and an initial polynomial commitment to $\mathbf{P}$. Crucially, we must avoid non-black-box use of the cryptographic operations involved in the OKVS and VOLE constructions. In Section 4, we explain how we overcome these challenges by exploiting the linearity of the VOLE correlation and the security guarantees of VOLE.

1.4.4 Consistency of Sender's Inputs

While one could adopt a similar commit-and-prove template to ensure consistency between the sender's inputs in the Committing Phase and those used in the Intersection Phase, a much simpler approach suffices. Recall that in the PSI functionality, the receiver's output is merely a subset of the sender's inputs. Therefore, enforcing consistency for *all* of the sender's inputs is unnecessary and inefficient. Instead, it suffices for the receiver to verify that the output elements appear in the sender's initial explicit commitment. In other words, the sender and receiver can first run the plain VOLE-PSI protocol to obtain the output, after which the receiver verifies that each output element appears in the sender's commitment.

To implement this idea, we have the sender commit to $H'(x_i|r_i)$ for randomly chosen r_i using a vector commitment in the Committing Phase. The hash values $H(x_i\|t_i)$ computed during the Intersection Phase implicitly bind the sender to her input and hence we use this as her implicit commitment. Verifying consistency between the explicit and implicit commitments can be done via a simple and efficient de-randomization trick that naturally integrates with the VOLE-PSI protocol. We refer the reader to Section 4 for details.

2 Preliminaries

2.1 Notation

λ is used to denote the computational security parameter and κ the statistical security parameter. $x \xleftarrow{\$} S$ denotes that x is sampled uniformly at random from the set S. Boldface alphabets of the form $\mathbf{A}$ are used for denoting vectors. We represent the i^{th} element of a vector $\mathbf{V}$ as $\mathbf{V}[i]$. For $n \in \mathbb{N}$, let $[n]$ represent the set $\{1,2,\ldots,n\}$. We assume that all algorithms/functions implicitly take the security parameters as input, but generally omit it for brevity.

Polynomial Encodings. We will often encode vectors as polynomials to facilitate their commitment using a polynomial commitment scheme. Let $\{\omega^1,\omega^2,\ldots,\omega^n\}$ represent the n^{th} roots of unity i.e., roots of the equation $x^n = 1$ over $\mathbb{F}$. We define a polynomial encoding of a vector $\mathbf{A} = (a_1,a_2,\ldots,a_n)$ as a polynomial $A(\cdot)$ of degree less than n such that $A(\omega^i) = a_i$ for all $1 \le i \le n$. We represent converting a vector $\mathbf{V}$ into the corresponding polynomial as $V(\cdot) = \mathsf{Vec2Poly}(\mathbf{V})$.

2.2 Oblivious Key-Value Stores (OKVS)

OKVS is a data structure that is used to encode a set of key-value pairs. OKVS hides the keys associated with random

values.[3]

Definition 1 (Oblivious Key Value Store [37])**.** *Let $\mathcal{K}$ be the set of keys and $\mathcal{V}$ the set of values. An oblivious key-value store (OKVS) scheme consists of a pair of PPT algorithms* $(\mathsf{Encode}, \mathsf{Decode})$ *defined as follows:*

1. $\mathbf{P} \leftarrow \mathsf{Encode}(\{(k_i, v_i)\}_{i\in[n]})$*: The encode function on input a set of key-value pairs $\{(k_i, v_i)\}_{i\in[n]}$ outputs a vector $\mathbf{P}$ representing the OKVS data structure.*
2. $v \leftarrow \mathsf{Decode}(\mathbf{P}, k)$*: The decode function on input an OKVS data structure $\mathbf{P}$ and a key k outputs a value v.*

These algorithms satisfy the following properties:

- ***Correctness:*** *For each subset $\mathcal{A} \subseteq \mathcal{K} \times \mathcal{V}$ comprising of key-value pairs with distinct keys (i.e., for each pair $(k,v),(k',v') \in \mathcal{A}$, $k \neq k'$), the following holds for each $(k,v) \in \mathcal{A}$:* $\Pr[\mathsf{Decode}(\mathsf{Encode}(\mathcal{A}), k) \neq v] \leq negl(\lambda)$
- ***Obliviousness:*** *For any two distinct key sets $\{k_1^0, \ldots, k_n^0\}$ and $\{k_1^1, \ldots, k_n^1\}$, distributions $\mathcal{R}(k_1^0, \ldots, k_n^0)$ and $\mathcal{R}(k_1^1, \ldots, k_n^1)$ are computationally indistinguishable, where $\mathcal{R}$ is defined as:*

$\underline{\mathcal{R}(k_1, \ldots, k_n)}$:
for $i \in [n]$: $v_i \xleftarrow{\$} \mathcal{V}$
return $\mathsf{Encode}(\{(k_1, v_1), \ldots, (k_n, v_n)\})$

Looking ahead, we additionally require the Decode function to be linear in $\mathbf{P}$ and that the OKVS $\mathbf{P}$ is a vector of field elements. Several existing OKVS constructions ([17, 57, 59, 64]) satisfy these constraints.

2.3 Secure Multi-Party Computation

Looking ahead, we see that it is easy to define our framework PICS as well as underlying cryptographic primitives like VOLE as ideal functionalities and then argue security. Since our framework can be seen as a special form of secure multi-party computation, we follow standard MPC literature and consider the real-ideal paradigm. Secure multi-party computation allows mutually distrusting parties to compute a function on their joint private inputs without revealing anything but the output. In the ideal world, there is a trusted third party (TTP) that computes a function f that cannot be corrupted. We denote by $\mathrm{IDEAL}_{f,\mathsf{Sim}(z),I}(\mathbf{x})$ the joint output of the parties where $\mathbf{x}$ is the vector of inputs of all parties and Sim is an adversary that corrupts parties in the set I and receives auxiliary input z. In the real world, parties interact with each other according to some protocol π. We denote by $\mathrm{REAL}_{\pi,\mathcal{A}(z),I}(\mathbf{x})$ the joint output of the parties where $\mathbf{x}$ is the vector of inputs of all parties and $\mathcal{A}$ is a malicious adversary that corrupts parties in the set I and receives auxiliary input z.

[3]The definition and constructions extend to pseudo-random values naturally.

Definition 2 (Security with abort against malicious adversaries [50])**.** *We say that a protocol π computing a two-party functionality f achieves security with abort against malicious adversaries if for every non-uniform PPT adversary $\mathcal{A}$, there exists a non-uniform PPT simulator* Sim*, such that for every $I \subsetneq [n]$, for all possible inputs $\mathbf{x}$, for all auxiliary input z,*

$$IDEAL_{f,\mathsf{Sim}(z),I}(\mathbf{x}) \approx REAL_{\pi,\mathcal{A}(z),I}(\mathbf{x})$$

2.4 Vector Oblivious Linear Evaluation

Vector oblivious linear evaluation (VOLE) is an input-less secure two-party computation protocol that allows the sender to obtain random values $\Delta \in \mathbb{F}$ and $\mathbf{B} \in \mathbb{F}^m$ and the receiver to obtain values $\mathbf{A}, \mathbf{C} \in \mathbb{F}^m$, such that $\mathbf{C} = \Delta\mathbf{A} + \mathbf{B}$. Formally speaking, a VOLE protocol π_{VOLE} securely realizes the following ideal functionality $\mathcal{F}_{\mathsf{VOLE}}$ according to Definition 2.

Functionality $\mathcal{F}_{\mathsf{VOLE}}$

- Sender sends $(\mathsf{Sender}, \mathsf{id})$ and Receiver sends $(\mathsf{Receiver}, \mathsf{id})$ to instantiate $\mathcal{F}_{\mathsf{VOLE}}$
- $\mathcal{F}_{\mathsf{VOLE}}$ now does the following:
  - If the Sender is malicious, wait to receive $\Delta \in \mathbb{F}$ and $\mathbf{B} \in \mathbb{F}^m$. Sample $\mathbf{A} \xleftarrow{\$} \mathbb{F}^m$ and set $\mathbf{C} = \Delta\mathbf{A} + \mathbf{B}$
  - Else if the Receiver is malicious, wait to receive $\mathbf{A}, \mathbf{C} \in \mathbb{F}^m$. Sample $\Delta \xleftarrow{\$} \mathbb{F}$ and set $\mathbf{B} = \mathbf{C} - \Delta\mathbf{A}$
  - Else, sample $\Delta \xleftarrow{\$} \mathbb{F}$ and $\mathbf{A}, \mathbf{B} \xleftarrow{\$} \mathbb{F}^m$ and set $\mathbf{C} = \Delta\mathbf{A} + \mathbf{B}$
- Send $(\Delta, \mathbf{B})$ to Sender and $(\mathbf{A}, \mathbf{C})$ to Receiver

Figure 1: VOLE Functionality

2.5 Polynomial Commitment Schemes

A polynomial commitment scheme [46] is a type of functional commitment that enables one to commit to polynomials and later open the commitment at any evaluation point, accompanied by a proof of consistency with the committed polynomial.

Definition 3 (Polynomial Commitment)**.** *A polynomial commitment for polynomials over a field $\mathbb{F}$ consists of a tuple of PPT algorithms* $(\mathsf{PCS.Setup}, \mathsf{PCS.Commit}, \mathsf{PCS.Open}, \mathsf{PCS.Verify})$ *defined as follows:*

- $\mathsf{pp} \xleftarrow{\$} \mathsf{PCS.Setup}(1^\lambda)$ *: The setup algorithm takes as input the security parameter λ and outputs public parameters* pp*.*
- $(\sigma, r) \xleftarrow{\$} \mathsf{PCS.Commit}(\mathsf{pp}, d, p(\cdot))$ *: The commit algorithm takes as input the public parameters* pp *along with a polynomial p of degree less than d, and outputs a commitment σ along with the randomness used r.*

- $(y,\pi) \leftarrow \mathsf{PCS.Open}(\mathsf{pp}, d, p(\cdot), \sigma, r, x)$: *The open algorithm takes as input the public parameters* pp*, a polynomial* p *of degree less than* d *along with its commitment* σ*, randomness* r *used during commit, and an evaluation point* x *and outputs evaluation* $y = p(x)$ *along with a proof* π *certifying that the evaluation is correct with respect to the commitment* σ*.*
- $0/1 \leftarrow \mathsf{PCS.Verify}(\mathsf{pp}, d, \sigma, x, y, \pi)$: *The verify algorithm takes as input the public parameters* pp*, a commitment* σ*, an evaluation point* x *along with the evaluation* y*, and a proof* π *and outputs* $0/1$ *depending upon whether it accepts/rejects the proof.*

These algorithms satisfy the following properties:

- ***Correctness:*** *Let* $\mathsf{pp} \xleftarrow{\$} \mathsf{PCS.Setup}(1^\lambda)$*. The following holds for any polynomial* $p(\cdot)$ *of degree less than* d *defined over* $\mathbb{F}$ *and for each* $x \in \mathbb{F}$*:*

$$\Pr[\mathsf{PCS.Verify}(\mathsf{pp}, d, \sigma, x, y, \pi)) \neq 1] \leq \mathsf{negl}(\lambda),$$

  where $(\sigma, r) \xleftarrow{\$} \mathsf{PCS.Commit}(\mathsf{pp}, d, p(\cdot))$*, and* $(y,\pi) \leftarrow \mathsf{PCS.Open}(\mathsf{pp}, d, p(\cdot), \sigma, r, x)$*.*
- ***Polynomial Binding:*** *Let* $\mathsf{pp} \xleftarrow{\$} \mathsf{PCS.Setup}(1^\lambda)$*. There exists a PPT extractor* $\mathcal{E}$*, such that for any n.u. PPT adversary* $\mathcal{A}$*,* $(d, \sigma) \leftarrow \mathcal{A}(pp)$*,* $X = \{x_1, x_2, \ldots, x_{d+1}\}$ *where each* $x_i \xleftarrow{\$} \mathbb{F}$ *and* $(y_i, \pi_i) \leftarrow \mathcal{A}(pp, d, \sigma, x_i)$*, the following holds:*

$$\Pr\Big[\big\{\mathsf{PCS.Verify}(\mathsf{pp}, d, \sigma, x_i, y_i, \pi_i) = 1\big\}_{i\in[d+1]} \wedge P(\cdot) \neq \mathsf{intp}\big(Z\big)\Big] \leq \mathsf{negl}(\lambda)$$

  where $P(\cdot) = \mathcal{E}^{\mathcal{A}}(\mathsf{pp}, d, \sigma)$*,* $Z = \{(x_i, y_i)\}_{i\in[d+1]}$*, and* intp *is an algorithm that takes as input a set of* $(d+1)$ *polynomial evaluations and outputs the interpolated polynomial* $P(\cdot)$ *of degree less than* d *satisfying the given evaluations (*$\perp$ *if does not exist).*
- ***Hiding:*** *There exists non-uniform PPT simulators* $(\mathsf{Sim}_1, \mathsf{Sim}_2)$ *such that no non-uniform PPT adversary* $\mathcal{A}$ *can win the following game with probability more than* $1/2 + \mathsf{negl}(\lambda)$*:*
  1. Ch *sends* $\mathsf{pp} \xleftarrow{\$} \mathsf{PCS.Setup}(1^\lambda)$ *to* $\mathcal{A}$*.*
  2. $\mathcal{A}$ *sends a vector* $\mathbf{P}$ *of length* d*, and maximum number of future queries* M *to be made.*
  3. Ch *initializes state* st *and samples* $b \xleftarrow{\$} \{0,1\}$*.*
     - *If* $b = 0$*, sample* $\mathbf{Q} \xleftarrow{\$} \mathbb{F}^{c \cdot M}$ *where* c *is a parameter of the Polynomial Commitment Scheme and define* $\mathbf{P'} = \mathbf{P} \| \mathbf{Q}$*. Let* $P'(\cdot) = \mathsf{Vec2Poly}(\mathbf{P'})$ *and compute* $(\sigma, r) \leftarrow \mathsf{PCS.Commit}(\mathsf{pp}, d, P'(\cdot))$*.*
     - *If* $b = 1$*, compute* $(\mathsf{st}, \sigma) \leftarrow \mathsf{Sim}_1(\mathsf{st}, \mathsf{pp}, d, M)$
  4. Ch *sends* σ *to* $\mathcal{A}$*.*
  5. *Do* M *times:*
     (a) $\mathcal{A}$ *sends evaluation point* x*.*
     (b) Ch*: If* $b = 0$*, compute* $(y,\pi) \leftarrow \mathsf{PCS.Open}(\mathsf{pp}, d, P'(\cdot), \sigma, r, x)$*. Else if* $b = 1$*, compute* $(y, \pi, \mathsf{st}) \leftarrow \mathsf{Sim}_2(\mathsf{st}, \mathsf{pp}, d, \sigma, x)$*.*
     (c) Ch *sends* (y,π) *to* $\mathcal{A}$*.*
  6. $\mathcal{A}$ *guesses the bit* b *and wins if correct.*

FRI based Polynomial Commitment Scheme Fast Reed-Solomon Interactive (FRI) [13] is a renowned and widely used oracle proof of proximity protocol that was later extended to a polynomial commitment scheme [15]. We make black-box use of the FRI polynomial commitment scheme and omit the randomness r and degree d when describing our construction for readability. Note that we tailor the definition of the hiding property of polynomial commitment schemes to be directly compatible with the way we use it. Looking ahead, the receiver in our protocol will open the commitment at M_{Rec} evaluations and so, the length of the random vector $\mathbf{Q}$ will be $c \cdot M_{\mathsf{Rec}}$ where c is a parameter of the FRI scheme. It can be easily seen that the FRI-based polynomial commitment scheme satisfies our definition when choosing parameters carefully. We defer a proof for the hiding property of the scheme to the full version of the paper.

3 Defining Private Intersection over Committed Sets (PICS)

In this section, we present a formal definition of our framework. In this definition, we work in the bounded intersection model on the receiver side, i.e., we assume that the receiver knows the maximum number of intersections ahead of time, that he might wish to compute in the future. This simplification makes the protocol easier to explain and analyze. In Section 4.3, we discuss a simple extension of our protocol that supports an unbounded number of intersections at minimal cost. Furthermore, the number of supported intersections is a parameter that can be set depending upon application requirements, providing a trade-off with respect to the cost of the protocol.

As discussed previously, a protocol in this framework of private intersection over committed sets proceeds in two phases:

1. Committing Phase: In this phase, the parties commit to their input sets that will later be used to compute the intersection. More formally, a party P will commit to their role as sender or receiver together with their input set. The receiver will also specify the maximum number of subsequent intersections M that they want to compute.
2. Intersection Phase: Two mutually distrusting parties who then wish to securely compute the intersection on their respective input sets, can now do so while ensuring consistency with the inputs committed to during the Committing Phase.

Let X and Y be the respective inputs of the two parties that were committed during the Committing Phase. Our definition ensures that when these parties compute an intersection during the Intersection Phase, they are unable to inject new elements into the committed sets (without causing the protocol to fail). Observe that this definition allows them to choose arbitrary subsets X' and Y', such that the output of the Intersection Phase is $X \cap X' \cap Y \cap Y'$. As we shall see later, our protocol guarantees strong input consistency on the receiver side, i.e., $Y' = Y$. It is an interesting open question whether we can enforce $X' = X$ to achieve strong input consistency on the sender side.

Since our PICS framework supports the ability for multiple parties to commit to their inputs and later perform an intersection between any pair of parties, we give a general definition involving multiple parties. We present a formal description of the ideal functionality $\mathcal{F}_{\mathsf{PICS}}$ in Figure 2. It is parameterized by two values, $m, n \in \mathbb{N}$, where m denotes the input set size of the sender and n that of the receiver.

Functionality $\mathcal{F}_{\mathsf{PICS}}$

- Committing Phase:
  1. Sender: Send $(X, \mathsf{id}_{\mathsf{Sen}})$ to TTP.
  2. Receiver: Send $(Y, \mathsf{id}_{\mathsf{Rec}}, M_{\mathsf{Rec}})$ to TTP.
  3. TTP: If $\mathsf{id}_{\mathsf{Sen}}$ or $\mathsf{id}_{\mathsf{Rec}}$ (respectively) doesn't already exist, then store the corresponding data and initialize $\mathsf{cnt}_{\mathsf{Rec}}$ to 0 . Else, ignore.
- Intersection Phase:
  1. Receiver: Send $(\mathsf{id}_{\mathsf{Sen}}, \mathsf{id}_{\mathsf{Rec}}, Y')$ to TTP
  2. TTP: Check if $\mathsf{cnt}_{\mathsf{Rec}} < M_{\mathsf{Rec}}$ If not, abort. Else, send $(\mathsf{id}_{\mathsf{Sen}}, \mathsf{id}_{\mathsf{Rec}})$ to Sender.
  3. Sender: Send (Accept, X') or $\perp$ to TTP depending upon whether she wishes to compute an intersection or not.
  4. TTP: If the Sender sent abort, send $\perp$ to Receiver. Check if $|X| \leq m$ and $|Y| \leq n$. If any check fails, abort. Else, send $(X \cap X' \cap Y \cap Y')$ to the Receiver, $(|X|, |Y|, M_{\mathsf{Rec}})$ to both the Sender and Receiver, and update $\mathsf{cnt}_{\mathsf{Rec}} += 1$.

Figure 2: PICS Functionality

Definition 4 (Private Intersection over Committed Sets)**.** *Let $\pi = (\pi_{\mathsf{Com}}, \pi_{\mathsf{Int}})$, where π_{Com} and π_{Int} are interactive protocols for the* Committing Phase *and* Intersection Phase *respectively. Suppose $n \in \mathbb{N}$ parties execute k combined instances of π_{Com} and π_{Int} in an arbitrary order. Let $\mathcal{A}$ be a malicious, n.u. PPT adversary corrupting at most $n-1$ parties. We say that π is a protocol for private intersection over committed sets if the combined k executions of π_{Com} and π_{Int} satisfy Definition 2 with respect to $\mathcal{F}_{\mathsf{PICS}}$ and $\mathcal{A}$.*

4 Constructing Private Intersection over Committed Sets (PICS)

We now describe our construction of a PICS protocol. We start by discussing the main ideas in Section 4.1 and present the formal details in Section 4.2. Finally, we conclude with some extensions in Section 4.3.

4.1 Main Ideas

Recall from Section 1.4, that we refer to the commitments computed by the parties during the Committing Phase as their *initial commitments* and their messages in the underlying VOLE-PSI protocol that implicitly bind them to their respective inputs as their *implicit commitments*. Our main goal is to design efficient consistency checks between these two forms of commitments. For simplicity, in this section, we assume both parties have input sets of size n. Next, we outline our approach for ensuring consistency of the receiver's and sender's inputs, respectively.

4.1.1 Receiver

Receiver's Input-Binding Message. Recall from Section 1.4.2 that the only input-dependent message sent by the receiver in VOLE-PSI is $\mathbf{A}' = \mathbf{A} + \mathbf{P}$, where $\mathbf{A}$ is the output of the VOLE sub-protocol and $\mathbf{P}$ is an OKVS encoding of the receiver's input. To ensure input consistency, it suffices for the receiver to convince the sender that $\mathbf{A}'$ was computed correctly using the inputs committed to during the Committing Phase. In other words, the receiver has to essentially prove the following statement:

There exists $\mathbf{A}, \mathbf{P}$, where $\mathbf{P}$ is an OKVS encoding of the inputs within the initial commitment and $\mathbf{A}$ is the VOLE output, such that $\mathbf{A}' = \mathbf{A} + \mathbf{P}$.

Designing a proof for this statement presents the following challenges:

1. *Challenge 1:* Computing OKVS encodings is computationally intensive, and proving that this encoding was computed honestly would introduce significant overheads. Furthermore, such proofs typically involve non-black box use of the underlying crypto primitives.
2. *Challenge 2:* VOLE is a cryptographic primitive. At first, it is unclear how the receiver can convince the sender that $\mathbf{A}$ was indeed the output of the VOLE sub-protocol, without making non-black-box use of the VOLE sub-protocol.

We now address these challenges one by one.

Explicit Initial Commitment. Since the computation of $\mathbf{P}$ is independent of any other messages exchanged in the VOLE-

PSI protocol and can be re-used across multiple PSI executions, our idea for overcoming the first challenge is to use a commitment to $\mathbf{P}$ as the initial commitment. Specifically, during the Committing Phase, instead of just committing to his inputs $\{y_1,\ldots,y_n\}$, the receiver encodes a set of key-value pairs $\{(y_1,H(y_1)),\ldots,(y_n,H(y_n))\}$ into an OKVS data structure $\mathbf{P}$ and then commits to $\mathbf{P}$ using a polynomial commitment scheme. As discussed earlier, while our ideas can be implemented using any polynomial commitment scheme, we choose to use the FRI-based polynomial commitment scheme [13].

Since the initial commitment to receiver's input set now is the commitment to $\mathbf{P}$ itself, all that remains is for the receiver to convince the sender that there exists $\mathbf{A}$, which is the VOLE output and that $\mathbf{A}' = \mathbf{A}+\mathbf{P}$.

Proof of Consistency. Before proceeding with our construction of this consistency proof, let us review what the sender and receiver obtain from the VOLE sub-protocol. Recall from Section 1.4.2 that the sender receives $\Delta,\mathbf{B}$, while the receiver gets $\mathbf{A},\mathbf{C}$, which are correlated as follows: $\mathbf{C} = \Delta\cdot\mathbf{A}+\mathbf{B}$, or equivalently, $\mathbf{A} = \Delta^{-1}\cdot(\mathbf{C}-\mathbf{B})$. Thus, to prove that $\mathbf{A}' = \mathbf{A}+\mathbf{P}$, it suffices to show that $\mathbf{A}' = \Delta^{-1}\cdot(\mathbf{C}-\mathbf{B})+\mathbf{P}$. Among these vectors, the sender knows $\mathbf{B},\Delta,\mathbf{A}'$ and has a polynomial commitment to $\mathbf{P}$.

Let us also view $\mathbf{A}',\mathbf{B}$, and $\mathbf{C}$ as polynomials $A'(x),B(x)$, and $C(x)$, respectively, obtained via deterministic polynomial interpolation, as explained in Section 2.1. The relation that we want to check can now be rewritten in terms of these polynomials as: $A'(x) = \Delta^{-1}(C(x)-B(x))+P(x)$. To verify this, it suffices to check whether this equation holds at a randomly chosen evaluation point. In other words, using the Schwartz-Zippel lemma, we can conclude that it suffices to check whether the following holds: $A'(r) = \Delta^{-1}(C(r)-B(r))+P(r)$, for a random challenge r sampled by the sender.

For this check, recall that $A'(x),B(x)$, and Δ are already known to the sender, and she can obtain $P(r)$ by asking the receiver to open the commitment to $P(x)$ at r. All that remains is $C(r)$, which the receiver sends in the clear to the sender.

However, since we did not pre-commit to $C(x)$, the receiver could potentially send an incorrect value $C'(r)$ instead of $C(r)$. We now show that even if the receiver sends an incorrect $C'(r)$, the above check will fail with all but negligible probability if $\mathbf{A}' \neq \mathbf{A}+\mathbf{P}$.

Soundness. Assume, for the sake of contradiction, that the check passes, i.e., $A'(r) = \Delta^{-1}(C'(r)-B(r))+P(r)$. Substituting $B(r) = C(r)-\Delta A(r)$ into this equation and rearranging, we get

$$\Delta = \frac{C'(r)-C(r)}{A'(r)-A(r)-P(r)}.$$

Note that since $\mathbf{A}' \neq \mathbf{A}+\mathbf{P}$, $A'(r)-A(r)-P(r)$ is non-zero with all but negligible probability, due to the Schwartz-Zippel lemma, which ensures that we can safely divide by it. Now, all values on the right-hand side of the equation are known to the receiver. Therefore, the receiver can compute Δ. However, this violates the security of the VOLE sub-protocol, since the only value sent after the VOLE execution was the random challenge r, which is independent of the VOLE computation.

Thus, the check passes only if the receiver can break the security of VOLE or if the randomly chosen challenge r is "lucky," i.e., $A'(r)-A(r)-P(r) = 0$. Both of these events occur with negligible probability.

Input-Hiding Polynomial Commitments. We note that the FRI-based polynomial commitment scheme, on its own, does not provide any hiding guarantees. Moreover, each time the committed polynomial is opened at a new point, additional information about the polynomial—and thus about the encoded vector—is leaked. To mitigate this leakage, we apply a standard technique: randomizing the polynomial prior to commitment. The amount of added randomness is chosen based on the number of anticipated openings, ensuring that these openings do not compromise the privacy of the underlying vector.

Among the various techniques for randomization, we use the following simple approach. Let $\mathbf{P}$ be the vector that the receiver wishes to interpolate into a polynomial before committing to it in the Committing Phase. To support M intersections [4], we need to introduce M degrees of randomness. To do this, we sample a random vector $\mathbf{Q}\in\mathbb{F}^{c\cdot M}$ and define $\mathbf{P}' = \mathbf{P}\|\mathbf{Q}$ where c is a parameter for the FRI polynomial commitment scheme. We then interpolate $\mathbf{P}'$ into a polynomial, as described in Section 2.1, and commit to it using the FRI-based polynomial commitment scheme.

This padding increases the degree of the committed polynomial by $c\cdot M$, and to maintain compatibility with the protocol, we must accordingly increase the size of the VOLE instance by $c\cdot M$. Specifically, the sender must generate a VOLE instance of length $|\mathbf{P}|+c\cdot M$. Since the degrees of all relevant polynomials remain consistent, the soundness arguments from earlier still apply. However, to preserve privacy, we discard the final $c\cdot M$ outputs of the VOLE instance and use only the first $|\mathbf{P}|$ in the subsequent steps of the Intersection Phase.

4.1.2 Sender

Sender's Input-Binding Message. Recall from Section 1.4 that the sender's implicit commitment consists of the values $H(x_1\|t_1),\ldots,H(x_n\|t_n)$. To ensure input consistency, it suffices for the sender to convince the receiver that no additional values of the form $H(x'\|t')$ are sent for elements $x'\notin X$. At a high level, our approach is as follows: the receiver commits to its input set X using a simple vector commitment during the Committing Phase. The parties then execute a maliciously secure VOLE-PSI protocol, augmented with the input consistency checks for the receiver as described earlier.

[4] We discuss how to extend our protocol to support an unbounded number of intersections in Section 4.3

After learning the output, the receiver verifies that each output element is included in the sender's explicit commitment.

Explicit Initial Commitment. The sender's initial commitment must be hiding. To achieve this, we commit to $H'(x_1\|r_1), H'(x_2\|r_2), \ldots, H'(x_n\|r_n)$ using a Merkle tree, where each r_i is sampled uniformly at random and H' is an independent random oracle. After the VOLE-PSI protocol completes, the sender must somehow reveal the r_i values corresponding to elements in the intersection. Using these, the receiver can verify that the revealed elements were part of the committed set.

Checking Consistency. In the Intersection Phase, at the end of the VOLE-PSI protocol, the sender sends $H'(x_1\|r_1), H'(x_2\|r_2), \ldots, H'(x_n\|r_n)$. Additionally, for each element x_i in the sender's set, she includes an additional value $m_i = H'(x_i\|t_i) + r_i$ [5]. For elements in the intersection ($x_i = y_j$), the receiver knows $x_i\|t_i$ and can compute $H'(x_i\|t_i)$, allowing recovery of r_i from m_i. The receiver then checks whether the corresponding $H'(x_i\|r_i)$ is honestly computed and appeared in the initial Merkle commitment. For non-intersecting elements, $x_i\|t_i$ is unknown and computationally hard to guess, so $H'(x_i\|t_i)$ masks r_i like a one-time pad, ensuring hiding.

We remark that this approach only achieves the weaker notion of input consistency for the sender, who cannot *inject* new elements into her committed set. Specifically, if the sender sends an incorrect hash value $h_i \neq H'(x_i\|r_i)$ in the Intersection Phase, the corresponding element x_i will not be matched even if it was in the receiver's set. This effectively allows the sender to choose a subset of her committed set to use in the Intersection Phase. To achieve strong input consistency, one would need to prove that the same x_i was fed as part of the input to H in the Committing Phase and to H' in the Intersection Phase, which appears challenging with only black-box use of the random oracle. We leave it as an interesting open problem to design a protocol that also achieves strong sender-side input consistency.

4.2 Our Construction

In this section, we give a formal description our protocol. First, we state the parameters and building blocks required in this construction. We work over a field $\mathbb{F}$ of size at least 2^λ, where $|\mathbb{F}| - 1$ is a perfect power of 2. Let $H, H' : \mathbb{F} \times \mathbb{F} \to \mathbb{F}$ and $H^{\mathbb{F}} : \mathbb{F} \to \mathbb{F}$ be random oracles. Let $(\mathsf{Encode}, \mathsf{Decode})$ be an OKVS (see Definition 1) that encodes n key-value pairs to encodings of length n'. Let $(\mathsf{PCS.Setup}, \mathsf{PCS.Commit}, \mathsf{PCS.Open}, \mathsf{PCS.Verify})$ be the FRI-based polynomial commitment scheme (see Section 2.5) and c the associated FRI soundness parameter. Let $\mathsf{pp} \leftarrow \mathsf{PCS.Setup}(1^\lambda)$. Input sets X and Y with elements in $\mathbb{F}$. Note that, typically sets $X, Y \subseteq \{0,1\}^*$; however, we can map them to $\mathbb{F}$ using a suitable random oracle $H^* : \{0,1\}^* \to \mathbb{F}$. Let Vec2Poly be an algorithm for converting vectors to polynomials, as explained in Section 2.1.

The Committing Phase is described in Figure 3a and Figure 3b, and the Intersection Phase in Figure 3c. Before proceeding, both the sender and receiver verify that the other party participated in the Committing Phase with the correct role and committed their inputs.

We defer the proof of the following theorem to Appendix A.

Theorem 1. *The protocol in Figures 3a, 3b and 3c is a protocol for private intersection over committed sets (see Definition 4) against malicious adversaries in the $\mathcal{F}_{\mathsf{VOLE}}$-hybrid model.*

Communication Complexity. The communication cost for our Committing Phase is $O(\lambda)$ and for our Intersection Phase is $O(\lambda \cdot (m + n + M_{\mathsf{Rec}}))$.

Computation Complexity. The computation cost for the sender is $O(m)$ in the Committing Phase and $O(m + (n + M_{\mathsf{Rec}}) \cdot \log(n + M_{\mathsf{Rec}}))$ in the Intersection Phase. The computation cost for the receiver is $O((n + M_{\mathsf{Rec}}) \cdot \log(n + M_{\mathsf{Rec}}))$ in the Committing Phase and $O((n + M_{\mathsf{Rec}}) \cdot \log(n + M_{\mathsf{Rec}}) + m)$ in the Intersection Phase.

Unbalanced Setting. Our PICS protocol follows the VOLE-PSI framework and is primarily designed for the balanced setting where the two sets are of the same size. Nevertheless, it can be applied directly to the unbalanced setting in which one set is significantly larger than the other (i.e., $m \gg n$ or $m \ll n$. The computation and communication complexities remain as detailed above.

4.3 Extensions

In this section, we discuss suitable extensions of our protocol that are relevant for certain applications, as discussed in Section 1.3. The ease and efficiency of adding such desired modifications, depending upon the relevant use case, also enforces the generality and flexibility of our framework as a whole.

Unbounded Intersections. While our protocol works in the bounded intersection model for the receiver, i.e., the number of future intersections is fixed beforehand during the Committing Phase, we note that we can support an unbounded number of intersections. Suppose a receiver P committed to their input with the number of supported intersections being M. Before exhausting all the M supported intersections, P can create a new commitment with supported intersections M' and prove consistency of this new commitment with respect to the old commitment using standard ZK techniques. Note that this cost is amortized for all the M' intersections and hence, minimal. Now, P can use this new commitment and support an additional $(M' - 1)$ number of intersections

[5] We assume this operation is defined over the underlying field.

π_{Com} (Sender)

Input: Set X of size m

1. Randomly shuffle X to get $\mathbf{X}$ and sample $r_1, r_2, \ldots, r_m \xleftarrow{\$} \mathbb{F}$.
2. Compute the Merkle root R corresponding to the Merkle commitment for the vector
$$\mathbf{L} = (H'(x_1 \| r_1), H'(x_2 \| r_2), \ldots, H'(x_m \| r_m)).$$
3. Broadcast R.

(a) Committing Phase (Sender)

π_{Com} (Receiver)

Input: Set Y of size n and maximum number of intersections M_{Rec}

1. Randomly shuffle Y and compute OKVS $\mathbf{P}$ as
$$\mathbf{P} = \mathsf{Encode}\left(\{(y, H^{\mathbb{F}}(y)) \mid y \in Y\}\right).$$
2. Sample $\mathbf{Q} \xleftarrow{\$} \mathbb{F}^{c \cdot M_{\mathsf{Rec}}}$ and define $\mathbf{P'} = \mathbf{P} || \mathbf{Q}$ where c is the FRI parameter.
3. Compute $P'(\cdot) = \mathsf{Vec2Poly}(\mathbf{P'})$ and $\sigma_{P'(\cdot)} \leftarrow \mathsf{PCS.Commit}(\mathsf{pp}, P'(\cdot))$
4. Broadcast $\sigma_{P'(\cdot)}$. Set a local counter $\mathsf{cnt}_{\mathsf{Rec}} = 0$.

(b) Committing Phase (Receiver)

π_{Int}

1. Sender: Sample $u \xleftarrow{\$} \mathbb{F}$ and send $u' = H^{\mathbb{F}}(u)$ to the Receiver.
2. Invoke $\mathcal{F}_{\mathsf{VOLE}}$: Sender gets $\Delta \in \mathbb{F}, \mathbf{B} \in \mathbb{F}^{n'+c \cdot M_{\mathsf{Rec}}}$ and Receiver gets $\mathbf{A}, \mathbf{C} \in \mathbb{F}^{n'+c \cdot M_{\mathsf{Rec}}}$ such that $\mathbf{C} = \Delta \mathbf{A} + \mathbf{B}$.
3. Receiver: Check if $\mathsf{cnt}_{\mathsf{Rec}} < M_{\mathsf{rec}}$. If not, abort. Sample $v \xleftarrow{\$} \mathbb{F}$ and define $\mathbf{A'} = \mathbf{A} + \mathbf{P'}$ where $\mathbf{P'}$ is defined during the Committing Phase. Send $(v, \mathbf{A'})$ to the Sender.
4. Sender: Sample $r \xleftarrow{\$} \mathbb{F}$ and send it to the Receiver.
5. Receiver: Compute $(P'(r), \pi_{P'(r)}) \leftarrow \mathsf{PCS.Open}(\mathsf{pp}, P'(\cdot), \sigma_{P'(\cdot)}, r)$, where $\sigma_{P'(\cdot)}$ is the commitment made during the Committing Phase. Send $(P'(r), \pi_{P'(r)}, C(r))$ to the Receiver where $C(\cdot) = \mathsf{Vec2Poly}(\mathbf{C})$.
6. Sender: Check if:
$$\mathsf{PCS.Verify}(\mathsf{pp}, \sigma_{P'(\cdot)}, r, P'(r), \pi_{P'(r)}) = 1, \quad \text{and} \quad A'(r) = \Delta^{-1} \cdot (C(r) - B(r)) + P'(r).$$
If either check fails, abort. Define $w = u + v$ and OKVS $\mathbf{K}$ as the first n' elements of $\Delta \mathbf{A'} + \mathbf{B}$. Compute
$$t_i = \mathsf{Decode}(\mathbf{K}, x_i) - \Delta H^{\mathbb{F}}(x_i) + w$$
for all $x_i \in \mathbf{X}$ and $\mathbf{Z} = \{H(x_i \| t_i) \mid x_i \in \mathbf{X}\}$. Compute $\mathbf{Z'} = \{H'(x_i \| t_i) + r_i \mid x_i \in \mathbf{X}\}$ where r_i is the randomness sampled during the Committing Phase. Send $(u, \mathbf{Z}, \mathbf{Z'}, \mathbf{L})$ to Receiver where $\mathbf{L}$ is defined during the Committing Phase.
7. Receiver: Check if $u' = H^{\mathbb{F}}(u)$ and if R corresponds to the Merkle root for $\mathbf{L}$ where R is defined during the Committing Phase. Define $s_i = \mathsf{Decode}(\mathbf{C'}, y_i) + w$ for all $y_i \in Y$ where $\mathbf{C'}$ is the first n' elements of $\mathbf{C}$. Output all $y_i \in Y$ such that there exists $z_j \in \mathbf{Z}$ satisfying $z_j = H(y_i \| s_i)$ and $H'(y_i \| (H'(y_i \| s_i) + Z'[j])) \in \mathbf{L}$. Finally, increment $\mathsf{cnt}_{\mathsf{Rec}}$ by 1.

(c) Intersection Phase (Additional steps in our protocol compared to the base VOLE-PSI protocol are colored green for Receiver set consistency check and blue for Sender set consistency check)

Figure 3: Our PICS Protocol.

(1 degree of freedom is used in performing the consistency check). This process can be repeated an arbitrary number of times to support an unbounded number of intersections.

Updating Sets. We also note that parties can update their sets i.e., add or remove elements. This is important for applications like password breach detection and the California Delete Act that we discuss in Section 1.3. Suppose a party P wishes to update the set X to X' by adding elements. P can create a fresh commitment to X' and prove that $X \subseteq X'$ using standard ZK techniques. A similar approach works for removing elements as well. Observe that this cost is amortized across all intersections and hence, minimal.

Set Membership Verification. For certain applications like the California Delete Act (see Section 1.3), a verification process may be required that allows a consumer to verify the inclusion of their name in the public commitment. We note

that such checks can be done in our framework. Observe that on the sender side (as is the case for this application), the public commitment is a Merkle commitment to $\mathbf{L}$. If a value x was included, then it must be the case that $H(x_i \| r_i) \in \mathbf{L}$ for some i. Hence, we can simply send the path in the Merkle tree corresponding to this leaf node along with its neighbors. The consumer can then verify all hash computations along this path to confirm the inclusion of their name. Such a membership verification is extremely efficient.

5 Implementation and Evaluation

We implement our PICS protocol in Rust and report the performance in this section.

5.1 Implementation Details

We choose computational security parameter $\lambda = 128$ and statistical security parameter $\kappa = 40$. Our plain-PSI implementation is based on the framework of VOLE-PSI [63, 64], yet we instantiate the framework with more recent constructions for VOLE [71] and OKVS [17]. Another important change is in the finite field, where STARK-252 is utilized for compatibility with the FRI polynomial commitment scheme.[6]

Parameter choice. The VOLE construction in [71] is based on the primal learning parity with noise (LPN) assumption [18] with regular noise distribution in a prime field of bit size 252. We compute the LPN parameters achieving 128 bits of security using the Python script provided by Liu et al. [52], which takes into account attacks including Pooled Gauss [33], statistical decoding (SD) [23, 30, 36, 44, 55], information set decoding (ISD) [12, 53, 62], and the algebraic (AGB) attack [21].

For OKVS, we implement the scheme based on Random Band matrices (RB-OKVS) from [17], modifying into a "binned" version to support parallelization. More specifically, suppose we want to divide the workload equally among d threads, the n inputs elements are first hashed into d bins, and then the encoding algorithm of RB-OKVS is performed for each bin in parallel.[7] We set the OKVS rate to 2 (namely $n' = 2n$ in our protocol description), and set the band-width to be 80, which satisfies the 2^{-40} statistical failure probability. The proof of security for this variant (which is omitted due to space) begins with a simple balls-into-bin argument to prove bound on each bin's size (which affects the OKVS rate in each bin), then the failure probability of each bin follows [17].

Finally, the FRI polynomial commitment scheme implementation is rewritten based on the `lambdaworks` library [49], with support for multi-threading in FFT, folding, and building Merkle trees. We set $c = 2$ as the blowup factor and follow the ethSTARK documentation [68] to choose parameters that achieve 128 bits of soundness security.

5.2 Experimental Results

Our benchmarks are run on Amazon AWS `c5a.16xlarge` instances, with computations parallelized through 32 cores, while communication remains sequential. We simulate network bandwidth and latency with the Linux `tc` command, choosing 0.1ms RRT and 20Gbps bandwidth for LAN. Experiments in the WAN settings follow the previous work [35] with 80ms RTT, and bandwidths 200Mbps, 50Mbps, and 5Mbps, respectively. We provide in Table 1 results of experiments for our protocol, along with our implementation of VOLE-PSI in STARK-252, and two previous PSI works that achieve input consistency on either the receiver side [35] or the sender side [66].[8] We also compare with a concurrent work [67] that achieves receiver-side input consistency. All experiments are run with the same set size for the sender and receiver.

Except for VOLE-PSI, we divide each protocol into two phases: Committing Phase and Intersection Phase, and report the communication and computation costs of each phase. We also report the end-to-end communication cost and running time (including both phases) of each protocol under the four network settings mentioned above.

Since the protocols in [35, 66, 67] consider different settings and (weaker) adversarial models, defining and evaluating the Committing Phase and Intersection Phase imposes a challenge. In each paragraph below, we first provide a short description of the setting in each protocol before analyzing the experimental results.

Comparison with plain maliciously secure VOLE-PSI. Our PICS protocol is built on VOLE-PSI, with the original protocol intact while adding commitments and proofs. We discuss the overhead of our protocol compared to VOLE-PSI. In the Committing Phase, each party only needs to send a short commitment of 32 bytes (one Merkle hash). In the Intersection Phase, PICS incurs only marginal computational overhead compared to VOLE-PSI. Note that the cost of computing OKVS is incurred in the Intersection Phase of VOLE-PSI, but is moved to the Committing Phase in PICS. The communication overhead is higher, yet remains a small constant between $1.57 - 2.04\times$. The end-to-end total running time (including both Committing Phase and Intersection Phase) overhead is $1.22 - 1.98\times$, as shown in Table 1 and Figure 4.

Comparison with [35] (receiver input consistency). In the Authorized Private Set Intersection (APSI) protocol of [35], a trusted third party (the Judge) authorizes the receiver's inputs, preventing injection of unauthorized elements in PSI.

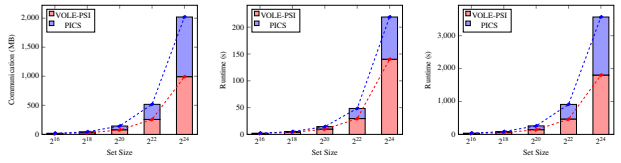
[6] Although recent work has extended the FRI degree test to arbitrary fields [14], [73], their concrete efficiency has not been fully analyzed.

[7] The idea of hashing elements into bins is also considered in other PSI works such as [63].

[8] We include a more detailed discussion on the related work in Section 6.

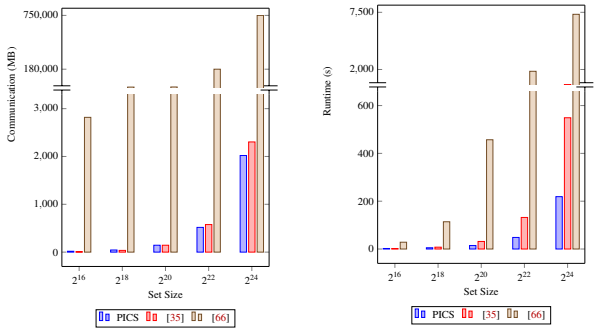
Set Size	Protocol	Sender Commit	Receiver Commit	Committing Phase Comm.	Committing Phase Comp.	Intersection Phase Comm.	Intersection Phase Comp.	Total Comm.	Total Runtime LAN	Total Runtime 200Mbps	Total Runtime 50Mbps	Total Runtime 5Mbps
2^{16}	VOLE-PSI	✗	✗	-	-	11.5	1.62	11.5	1.62	8.25	8.94	24.9
	[35]	✗	✓	7.00	0.21	2.00	1.67	9.00	1.89	2.56	3.64	16.6
	[66]	✓	✗	0.52	0.37	2,818	28.1	2,818	28.45	141	479	4,537
	[67]	✗	✓	1.14	0.05	2.40	0.27	9.8	0.33	0.37	0.49	1.90
	PICS	✓	✓	0.000064	0.45	18.0	1.75	18.0	2.20	10.1	11.7	36.9
2^{18}	VOLE-PSI	✗	✗	-	-	25.3	3.70	25.3	3.70	13.7	15.2	51.5
	[35]	✗	✓	28.0	0.85	8.00	6.83	36.0	7.70	9.44	13.8	65.6
	[66]	✓	✗	2.10	1.46	11GB	113	11GB	114	564	1,917	18K
	[67]	✗	✓	5.29	0.19	5.62	0.47	15.92	0.67	1.31	3.22	26.1
	PICS	✓	✓	0.000064	1.55	44.4	3.70	44.4	5.25	17.5	20.4	85.3
2^{20}	VOLE-PSI	✗	✗	-	-	76.1	9.58	76.1	9.58	21.5	27.9	140
	[35]	✗	✓	112	3.36	32.1	28.2	144	31.6	37.6	54.9	262
	[66]	✓	✗	8.39	5.85	45GB	451	45GB	457	2,255	7,665	73K
	[67]	✗	✓	21.9	0.78	18.5	1.23	40.4	2.01	3.63	8.47	66.7
	PICS	✓	✓	0.000064	4.66	144	9.84	144	14.5	29.9	43.1	258
2^{22}	VOLE-PSI	✗	✗	-	-	257	29.4	257	29.4	52.6	76.3	461
	[35]	✗	✓	448	13.5	128	118	576	132	155	224	1,054
	[66]	✓	✗	33.6	23.4	180GB	1,804	180GB	1,827	9,018	31K	290K
	[67]	✗	✓	88.4	3.34	68.9	4.29	157.3	7.63	13.9	32.8	259
	PICS	✓	✓	0.000064	18.2	517	29.8	517	48.4	80.3	131	911
2^{24}	VOLE-PSI	✗	✗	-	-	991	140	991	140	196	288	1,798
	[35]	✗	✓	1,792	54.2	512	494	2,304	549	641	917	4,235
	[66]	✓	✗	134	93.6	721GB	7,214	721GB	7,308	36K	123K	1,162K
	[67]	✗	✓	354	12.4	276	16.6	530	29.0	50.2	114	877
	PICS	✓	✓	0.000064	77.5	2,020	141.5	2,020	219	308	504	3,567

Table 1: Experimental comparison. All communication costs are in MB unless otherwise noted, and all computation costs are in seconds. The reported performance of [66, 67] is based on the reported numbers in their work and our estimates.



(a) Communication (b) Runtime (LAN) (c) Runtime (5Mbps)

Figure 4: Performance comparison of plain VOLE-PSI and PICS (both phases combined).



(a) Total Communication (b) Total Runtime (LAN)

Figure 5: Performance comparison of PICS with [35, 66].

APSI realizes a variant of the PICS functionality, where only the receiver's set is committed. In Table 1, we report (i) the communication cost of the Committing Phase (receiver $\leftrightarrow$ Judge messages) and the Intersection Phase (sender $\leftrightarrow$ receiver messages), (ii) the computational cost of each phase, and (iii) the end-to-end runtime under LAN and WAN assumptions (estimated from communication cost).[9]

The advantage of PICS is three-fold: (1) we don't require a trusted third-party to authorize the set, (2) we achieve input consistency on both sides, and (3) we achieve malicious security against both parties while APSI [35] only achieves security against a malicious receiver (and a semi-honest sender).

In terms of performance, during the Committing Phase, PICS achieves a running time comparable to [35] while requiring significantly less communication—only a single Merkle hash, compared to communication that grows linearly with the set size in [35]. Moreover, PICS is fully non-interactive, requiring no interaction with a Judge. In the Intersection Phase, PICS incurs higher communication cost but achieves more efficient computation, relying solely on lightweight cryptography, whereas [35] requires costly pairing operations. Considering end-to-end performance across both phases (see Table 1 and Figure 5), PICS yields a 1.14× reduction in communica-

[9]The implementation of [35] does not simulate message transmission.

tion cost for set size 2^{24}. In terms of running time, PICS is $2.51\times$ faster over LAN (reflecting computational efficiency) and $1.19\times$ faster over WAN with bandwidth 5 Mbps.

Comparision with [66] (sender input consistency). Sun et. al. [66] proposed a PSI protocol in the client–server setting, where the server (sender) publishes a one-time encoding of its input set that can be reused across multiple clients (receivers). This protocol can be viewed as a PICS variant with the sender's set committed. We define the communication cost of the Committing Phase as the size of the sender's published encoding, and that of the Intersection Phase as the total communication between the sender and a receiver during a single PSI execution.

We estimate their performance based on the per-element cost reported in [66].[10] In the Committing Phase, PICS and [66] have comparable runtime, while PICS provides a much more succinct commitment ($O(1)$ versus $O(n)$ in size). In the Intersection Phase, PICS achieves $16-60\times$ speedup in computation and $157-357\times$ reduction in communication. Overall, as shown in Table 1 and Figure 5, the end-to-end running time of PICS is $13-326\times$ faster than [66].

Comparison with [67] (receiver input consistency). In a concurrent work, Sun et al. [67] follow the client–server setting as in [66], with the difference that the server learns the PSI output. Their protocol can thus be viewed as a PICS variant with the receiver's set committed. It is built on a new primitive called committed vector oblivious linear evaluation (C-VOLE). Their construction for C-VOLE relies crucially on the underlying LPN-based VOLE construction, making non-black-box use of it to be compatible with their carefully designed LPN-based commitment scheme. In contrast, our construction is agnostic to the underlying VOLE protocol. Furthermore, the commitment in their protocol grows linearly with the receiver's set size, wheres we provide succinct commitments in the Committing Phase. We estimate their performance in Table 1 based on the numbers reported in [67].[11]

6 Related Work

PSI was introduced by Meadows [54] and has been studied intensively from various techniques, including Diffie-Hellman [42, 43, 54], RSA [9, 28], garbled circuits [41, 58–60], fully homomorphic encryption (FHE) [25–27], oblivious transfer (OT) [24, 32, 48, 56, 61], and most recently VOLE [17, 37, 63, 64]. However, even the maliciously secure PSI protocols suffer from the vulnerabilities discussed above.

Authorized PSI. Camenisch and Zaverucha [22] proposed a notion called PSI for certified sets, where they introduce a trusted third party (certification authority) to certify the input sets, ensuring that the inputs are valid and binding them to each party. However, their protocol requires quadratic communication and computational complexity in the set size. De Cristofaro and Tsudik [28] introduced the notion of Authorized PSI (APSI), where they identify one party as the server (providing PSI service) and the other party as the client (receiving the PSI result). Each element in the client set must be authorized (signed) by a trusted authority. Their protocol achieves linear communication and computational complexity. A follow-up work by Kerschbaum [47] presented a more efficient APSI protocol from Bloom filters and homomorphic encryption. The work of Debnath and Dutta [29] further improved the efficiency for both APSI and APSI-Cardinality (where parties only learn the size of the intersection). Faber et al. [34] presented an APSI protocol with both sides authenticated. A more recent work [35] presented an APSI protocol with a single round of communication from the server to the client in the online phase, achieving security against a malicious client and semi-honest server. They also introduced the notion of Partial APSI which allows for partial verification of the sets.

However, all these works crucially rely on a trusted third party to (fully or partially) access and authorize the sets, which defeats the purpose of PSI. Moreover, they require a co-design of the authorization process and the PSI protocol. In contrast, our new framework separates the validation and intersection phases, allowing them to be designed independently while connecting them through a succinct commitment. From a technical perspective, all the existing approaches for APSI rely on computationally expensive public-key operations, such as Diffie-Hellman-based OPRF, homomorphic encryption, or bilinear pairings, which make the online PSI protocol orders of magnitude slower than state-of-the-art ones that primarily use symmetric-key cryptography. Finally, even under the APSI framework, our new PICS protocol provides an extremely efficient way for a trusted party to authorize the set, namely, by generating a digital signature on the succinct commitment.

Client-Server PSI. Another related direction is reusable PSI in the client-server setting, introduced by Sun et al. [66]. In this work, a server publishes a one-time, linear-sized encoding of its set. Afterwards, multiple clients can independently execute a PSI protocol with the server, with complexity linear in the size of each client's set. The published encoding can be viewed as a set commitment that ensures input consistency of the server, without needing a trusted third party. Recently, there have been two *concurrent works* [11, 67] in this setting. All these works only achieve one-sided input consistency compared to two-sided input consistency in ours. Specifically, [66] achieves sender-side input consistency while [11, 67] achieve receiver-side input consistency. We provide detailed comparison with [66, 67] in Section 5.2. The work by Bartusek et al. [11] relies on extensive public key operations whereas we use efficient symmetric-key operations. Furthermore, they achieve a weaker, non-standard notion of security.

[10] The implementation of [66] is not open-sourced.
[11] The implementation of [67] is not open-sourced.

Acknowledgments

Part of this work was conducted while Aarushi Goel was at Purdue University and was visiting Simons Institute for the Theory of Computing, and this research was supported in part by an Amazon Research Award, Spring 2025. Peihan Miao and Phuoc Van Long Pham were supported in part by NSF SaTC Award 2247352, NSF CAREER Award 2442384, Meta Research Award, Google Research Scholar Award, and Amazon Research Award. Satvinder Singh was supported in part by Supra Labs. The authors would like to thank Gabriel Kaptchuk for helpful discussions on the applications of this framework.

Ethical Considerations

Private set intersection (PSI) is one of the most decorated problems in secure multiparty computation (MPC) in terms of industrial applications. However, the party with more resources (i.e., a service provider) in the protocol may abuse their power (e.g., by using inconsistent sets across multiple sessions) to learn more information about individual users.

Such vulnerabilities can be mitigated using the PICS protocol. We consider the implications of using PICS in several example applications. The Apple CSAM incident was a clear example of when standard PSI failed to address the ethical concerns stakeholders might have in the real world: iCloud users wanted provable guarantees on the dataset used by Apple. Our PICS framework addresses these concerns by allowing Apple to commit to the CSAM dataset and enforcing the use of the same committed dataset across all users when running PSI. As discussed in Section 1.3, our framework provides similar guarantees for other applications such as the California Delete Act and private contact discovery.

Nevertheless, our work does not eliminate all potential misuse of PSI. Specifically, although PICS ensures that each party uses a consistent set across executions, it does not guarantee the validity of the committed sets. For instance, in the Apple CSAM example, the protocol alone cannot ensure that Apple initially commits to a set only consisting of CSAM images. Such guarantees may be difficult to achieve using cryptographic methods alone. In practice, additional safeguards such as auditing (e.g., by the government or non-profit third parties) should be considered to further reduce the risk of misuse. In certain applications such as private contact discovery and California Delete Act, augmenting PICS with other mechanisms such as rate-limiting is recommended. We also leave open the problem of strengthening PICS to achieve strong sender-side input consistency.

That being said, by advancing the line of research on committed PSI, we believe that this work takes a step toward safer PSI protocols with positive social utility, as it provides strictly stronger security guarantees than the prior art.

Open Science

The artifacts of this paper involve the source code for the protocol and the code for generating synthetic data. In accordance with the open source policy, the repository containing the code can be found through the Zenodo record: https://zenodo.org/records/17958838.

References

[1] Senate Bill No. 362. https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=202320240SB362.

[2] California's Data Deletion Law: Understanding the California Delete Act for Regulating Data Brokers. https://secureprivacy.ai/blog/california-delete-act-guide.

[3] Firefox Password Manager – Alerts for breached websites. (link).

[4] Password Monitor: Safeguarding passwords in Microsoft Edge. (link).

[5] Password Monitoring – Apple Platform Security. https://support.apple.com/en-al/guide/security/sec78e79fc3b/web.

[6] Protect your accounts from data breaches with Password Checkup. https://security.googleblog.com/2019/02/protect-your-accounts-from-data.html.

[7] Harold Abelson, Ross J. Anderson, Steven M. Bellovin, Josh Benaloh, Matt Blaze, Jon Callas, Whitfield Diffie, Susan Landau, Peter G. Neumann, Ronald L. Rivest, Jeffrey I. Schiller, Bruce Schneier, Vanessa Teague, and Carmela Troncoso. Bugs in our pockets: the risks of client-side scanning. *J. Cybersecur.*, 10(1), 2024.

[8] Apple Inc. Csam detection technical summary. Technical report, Apple Inc., 2021. Accessed: 2025-05-22.

[9] Giuseppe Ateniese, Emiliano De Cristofaro, and Gene Tsudik. (If) size matters: Size-hiding private set intersection. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 156–173. Springer, Berlin, Heidelberg, March 2011.

[10] Pierre Baldi, Roberta Baronio, Emiliano De Cristofaro, Paolo Gasti, and Gene Tsudik. Countering GATTACA: efficient and secure testing of fully-sequenced human genomes. In Yan Chen, George Danezis, and Vitaly Shmatikov, editors, *ACM CCS 2011*, pages 691–702. ACM Press, October 2011.

[11] James Bartusek, Sanjam Garg, Abhishek Jain, and Guru-Vamsi Policharla. Laconic PSI on authenticated inputs and applications. Cryptology ePrint Archive, Paper 2025/1108, 2025.

[12] Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding random binary linear codes in $2^{n/20}$: How 1 + 1 = 0 improves information set decoding. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 520–536. Springer, Berlin, Heidelberg, April 2012.

[13] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast reed-solomon interactive oracle proofs of proximity. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *ICALP 2018*, volume 107 of *LIPIcs*, pages 14:1–14:17. Schloss Dagstuhl, July 2018.

[14] Eli Ben-Sasson, Dan Carmon, Swastik Kopparty, and David Levit. Scalable and transparent proofs over all large fields, via elliptic curves - (ECFFT part II). In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022, Part I*, volume 13747 of *LNCS*, pages 467–496. Springer, Cham, November 2022.

[15] Eli Ben-Sasson, Lior Goldberg, Swastik Kopparty, and Shubhangi Saraf. DEEP-FRI: Sampling outside the box improves soundness. In Thomas Vidick, editor, *ITCS 2020*, volume 151, pages 5:1–5:32. LIPIcs, January 2020.

[16] Abhishek Bhowmick, Dan Boneh, Steve Myers, Kunal Talwar, and Karl Tarbe. The Apple PSI System. https://www.apple.com/child-safety/pdf/Apple_PSI_System_Security_Protocol_and_Analysis.pdf, 2021.

[17] 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 Joseph A. Calandrino and Carmela Troncoso, editors, *USENIX Security 2023*, pages 301–318. USENIX Association, August 2023.

[18] Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 278–291. Springer, Berlin, Heidelberg, August 1994.

[19] Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 896–912. ACM Press, October 2018.

[20] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 489–518. Springer, Cham, August 2019.

[21] Pierre Briaud and Morten Øygarden. A new algebraic approach to the regular syndrome decoding problem and implications for PCG constructions. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 391–422. Springer, Cham, April 2023.

[22] Jan Camenisch and Gregory M. Zaverucha. Private intersection of certified sets. In Roger Dingledine and Philippe Golle, editors, *FC 2009*, volume 5628 of *LNCS*, pages 108–127. Springer, Berlin, Heidelberg, February 2009.

[23] Kevin Carrier, Thomas Debris-Alazard, Charles Meyer-Hilfiger, and Jean-Pierre Tillich. Statistical decoding 2.0: Reducing decoding to LPN. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part IV*, volume 13794 of *LNCS*, pages 477–507. Springer, Cham, December 2022.

[24] Melissa Chase and Peihan Miao. Private set intersection in the internet setting from lightweight oblivious PRF. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 34–63. Springer, Cham, August 2020.

[25] Hao Chen, Zhicong Huang, Kim Laine, and Peter Rindal. Labeled PSI from fully homomorphic encryption with malicious security. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 1223–1237. ACM Press, October 2018.

[26] Hao Chen, Kim Laine, and Peter Rindal. Fast private set intersection from homomorphic encryption. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1243–1255. ACM Press, October / November 2017.

[27] Kelong Cong, Radames Cruz Moreno, Mariana Botelho da Gama, Wei Dai, Ilia Iliashenko, Kim Laine, and Michael Rosenberg. Labeled PSI from homomorphic encryption with reduced computation and communication. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 1135–1150. ACM Press, November 2021.

[28] Emiliano De Cristofaro and Gene Tsudik. Practical private set intersection protocols with linear complexity. In Radu Sion, editor, *FC 2010*, volume 6052 of *LNCS*,

pages 143–159. Springer, Berlin, Heidelberg, January 2010.

[29] Sumit Kumar Debnath and Ratna Dutta. Secure and efficient private set intersection cardinality using bloom filter. In Javier Lopez and Chris J. Mitchell, editors, *ISC 2015*, volume 9290 of *LNCS*, pages 209–226. Springer, Cham, September 2015.

[30] Thomas Debris-Alazard and Jean-Pierre Tillich. Statistical decoding. In *IEEE ISIT*, pages 1798–1802. IEEE, 2017.

[31] Daniel Demmler, Peter Rindal, Mike Rosulek, and Ni Trieu. PIR-PSI: Scaling private contact discovery. *PoPETs*, 2018(4):159–178, October 2018.

[32] Changyu Dong, Liqun Chen, and Zikai Wen. When private set intersection meets big data: an efficient and scalable protocol. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 789–800. ACM Press, November 2013.

[33] Andre Esser, Robert Kübler, and Alexander May. LPN decoded. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 486–514. Springer, Cham, August 2017.

[34] Sky Faber, Ronald Petrlic, and Gene Tsudik. Unlinked: Private proximity-based off-line OSN interaction. In *ACM WPES*, pages 121–131. ACM, 2015.

[35] Francesca Falzon and Evangelia Anna Markatou. Revisiting authorized private set intersection: A new privacy-preserving variant and two protocols. *Proceedings on Privacy Enhancing Technologies*, 2025.

[36] Marc P. C. Fossorier, Kazukuni Kobara, and Hideki Imai. Modeling bit flipping decoding based on nonorthogonal check sums with application to iterative decoding attack of mceliece cryptosystem. *IEEE Trans. Inf. Theory*, 53(1):402–411, 2007.

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

[38] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.

[39] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to prove all NP-statements in zero-knowledge, and a methodology of cryptographic protocol design. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 171–185. Springer, Berlin, Heidelberg, August 1987.

[40] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th ACM STOC*, pages 291–304. ACM Press, May 1985.

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

[42] Bernardo A. Huberman, Matthew K. Franklin, and Tad Hogg. Enhancing privacy and trust in electronic communities. In *ACM EC*, pages 78–86. ACM, 1999.

[43] 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 *IEEE EuroS&P*, pages 370–389, 2020.

[44] A. Kh. Al Jabri. A statistical decoding algorithm for general linear block codes. In *Cryptography and Coding, IMA*, volume 2260, pages 1–8, 2001.

[45] Daniel Kales, Christian Rechberger, Thomas Schneider, Matthias Senker, and Christian Weinert. Mobile private contact discovery at scale. In Nadia Heninger and Patrick Traynor, editors, *USENIX Security 2019*, pages 1447–1464. USENIX Association, August 2019.

[46] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194. Springer, Berlin, Heidelberg, December 2010.

[47] Florian Kerschbaum. Outsourced private set intersection using homomorphic encryption. In Heung Youl Youm and Yoojae Won, editors, *ASIACCS 12*, pages 85–86. ACM Press, May 2012.

[48] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. Efficient batched oblivious PRF with applications to private set intersection. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 818–829. ACM Press, October 2016.

[49] lambdaworks contributors. lambdaworks, 2023.

[50] Yehuda Lindell. How to simulate it - A tutorial on the simulation proof technique. Cryptology ePrint Archive, Report 2016/046, 2016.

[51] Yehuda Lindell. How to simulate it - A tutorial on the simulation proof technique. In *Tutorials on the Foundations of Cryptography*, pages 277–346. 2017.

[52] Hanlin Liu, Xiao Wang, Kang Yang, and Yu Yu. The hardness of LPN over any integer ring and field for PCG applications. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VI*, volume 14656 of *LNCS*, pages 149–179. Springer, Cham, May 2024.

[53] Alexander May, Alexander Meurer, and Enrico Thomae. Decoding random linear codes in $\tilde{O}(2^{0.054n})$. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 107–124. Springer, Berlin, Heidelberg, December 2011.

[54] Catherine Meadows. A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In *IEEE S&P*, pages 134–137. IEEE Computer Society, 1986.

[55] Raphael Overbeck. Statistical decoding revisited. In *ACISP*, volume 4058, pages 283–294, 2006.

[56] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. SpOT-light: Lightweight private set intersection from sparse OT extension. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 401–431. Springer, Cham, August 2019.

[57] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. PSI from PaXoS: Fast, malicious private set intersection. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 739–767. Springer, Cham, May 2020.

[58] Benny Pinkas, Thomas Schneider, Gil Segev, and Michael Zohner. Phasing: Private set intersection using permutation-based hashing. In Jaeyeon Jung and Thorsten Holz, editors, *USENIX Security 2015*, pages 515–530. USENIX Association, August 2015.

[59] Benny Pinkas, Thomas Schneider, Oleksandr Tkachenko, and Avishay Yanai. Efficient circuit-based PSI with linear communication. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 122–153. Springer, Cham, May 2019.

[60] Benny Pinkas, Thomas Schneider, Christian Weinert, and Udi Wieder. Efficient circuit-based PSI via cuckoo hashing. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 125–157. Springer, Cham, April / May 2018.

[61] Benny Pinkas, Thomas Schneider, and Michael Zohner. Faster private set intersection based on OT extension. In Kevin Fu and Jaeyeon Jung, editors, *USENIX Security 2014*, pages 797–812. USENIX Association, August 2014.

[62] Eugene Prange. The use of information sets in decoding cyclic codes. *IRE Trans. Inf. Theory*, 8(5):5–9, 1962.

[63] Srinivasan Raghuraman and Peter Rindal. Blazing fast PSI from improved OKVS and subfield VOLE. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 2505–2517. ACM Press, November 2022.

[64] Peter Rindal and Phillipp Schoppmann. VOLE-PSI: Fast OPRF and circuit-PSI from vector-OLE. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 901–930. Springer, Cham, October 2021.

[65] Sarah Scheffler, Anunay Kulshrestha, and Jonathan R. Mayer. Public verification for private hash matching. In *2023 IEEE Symposium on Security and Privacy*, pages 253–273. IEEE Computer Society Press, May 2023.

[66] Yunqing Sun, Jonathan Katz, Mariana Raykova, Phillipp Schoppmann, and Xiao Wang. Actively secure private set intersection in the client-server setting. In *ACM CCS*, pages 1478–1492, 2024.

[67] Yunqing Sun, Hanlin Liu, Kang Yang, Yu Yu, Xiao Wang, and Chenkai Weng. Committed vector oblivious linear evaluation and its applications. In *ACM CCS*, pages 3635–3648, 2025.

[68] StarkWare Team. ethstark documentation–version 1.1. Technical report, IACR preprint archive 2021, 2021.

[69] Kurt Thomas, Jennifer Pullman, Kevin Yeo, Ananth Raghunathan, Patrick Gage Kelley, Luca Invernizzi, Borbala Benko, Tadek Pietraszek, Sarvar Patel, Dan Boneh, and Elie Bursztein. Protecting accounts from credential stuffing with password breach alerting. In Nadia Heninger and Patrick Traynor, editors, *USENIX Security 2019*, pages 1556–1571. USENIX Association, August 2019.

[70] Juan Ramón Troncoso-Pastoriza, Stefan Katzenbeisser, and Mehmet Celik. Privacy preserving error resilient dna searching through oblivious automata. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *ACM CCS 2007*, pages 519–528. ACM Press, October 2007.

[71] Chenkai Weng, Kang Yang, Jonathan Katz, and Xiao Wang. Wolverine: Fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits. In *2021 IEEE Symposium on Security*

and Privacy, pages 1074–1091. IEEE Computer Society Press, May 2021.

[72] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.

[73] Hadas Zeilberger, Binyi Chen, and Ben Fisch. BaseFold: Efficient field-agnostic polynomial commitment schemes from foldable codes. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part X*, volume 14929 of *LNCS*, pages 138–169. Springer, Cham, August 2024.

A Proof of Theorem 1

We first argue for a single execution of the Committing Phase and the Intersection Phase and later show how to extend to the general definition.

Corrupt sender: Consider the simulator $\mathsf{Sim}_{\mathsf{Sen}}$:

- Committing Phase:
  1. $\mathsf{Sim}_{\mathsf{Sen}}$ initializes state st and runs simulator 1 from the hiding property of polynomial commitment schemes to get $\sigma_{P'(\cdot)}$ and sends it to the sender i.e. $(\mathsf{st}, \sigma_{P'(\cdot)}) \leftarrow \mathsf{Sim}_1(\mathsf{st}, \mathsf{pp}, n', M_{\mathsf{Rec}})$
  2. $\mathsf{Sim}_{\mathsf{Sen}}$ extracts the input $\mathbf{X}$ along with r_i values based upon the Merkle root R that it receives. More formally, $\mathbf{X}$ can be extracted based upon queries made to the RO by the sender in a top-down fashion in the Merkle tree.
  3. $\mathsf{Sim}_{\mathsf{Sen}}$ sends $\mathbf{X}$ to the ideal functionality $\mathcal{F}_{\mathsf{PICS}}$.
- Intersection Phase:
  1. The simulator receives α from the sender.
  2. When the sender invokes $\mathcal{F}_{\mathsf{VOLE}}$, $\mathsf{Sim}_{\mathsf{Sen}}$ plays the role of $\mathcal{F}_{\mathsf{VOLE}}$. Given $\Delta \in \mathbb{F}, \mathbf{B} \in \mathbb{F}^{n'+c\cdot M_{\mathsf{Rec}}}$ from the sender, the simulator samples random $\mathbf{A} \xleftarrow{\$} \mathbb{F}^{n'+c\cdot M_{\mathsf{Rec}}}$ and sets $\mathbf{C} = \Delta\mathbf{A} + \mathbf{B}$.
  3. $\mathsf{Sim}_{\mathsf{Sen}}$ samples random $\mathbf{A}' \xleftarrow{\$} \mathbb{F}^{n'+c\cdot M_{\mathsf{Rec}}}$ and $v \xleftarrow{\$} \mathbb{F}$ and sends it to the sender.
  4. Upon receiving r from the sender, the simulator opens the commitment $\sigma_{P'(\cdot)}$ at r using simulator 2 from the hiding property of polynomial commitment schemes, to say a value β, i.e. $(\beta, \pi_\beta, \mathsf{st}) \leftarrow \mathsf{Sim}_2(\mathsf{st}, \mathsf{pp}, n', \sigma_{P'(\cdot)}, r)$ and sends $(A'(r) - \beta)\Delta + B(r)$ instead of $C(r)$.
  5. $\mathsf{Sim}_{\mathsf{Sen}}$ now receives $(\mathbf{Z}, u, \mathbf{Z}', \mathbf{L})$ from the sender.
  6. The simulator checks if $\mathbf{L}$ is consistent with R. Based on the RO queries made, the simulator extracts all x_i values with correct t_i from $\mathbf{Z}$ i.e. $t_i = \mathsf{Decode}(\mathbf{K}, x_i) - \Delta H^{\mathbb{F}}(x_i) + w$. It then computes all x_i values (with correct t_i) that satisfy $(H'(x_i \| t_i) + Z'[i]) = r_i$ where r_i was extracted during the Committing Phase. Call the set of these values $\mathbf{X}'$.
  7. $\mathsf{Sim}_{\mathsf{Sen}}$ sends $\mathbf{X}'$ to the ideal functionality $\mathcal{F}_{\mathsf{PICS}}$.

We now show that the output of $\mathsf{Sim}_{\mathsf{Sen}}$ is computationally indistinguishable from the view of the sender. We show this via a sequence of hybrids:

- $\mathcal{H}_0$: Transcript is generated as in the real protocol.
- $\mathcal{H}_1$: This hybrid is identical to the previous one, except that we replace $C(r)$ with the value in $\mathsf{Sim}_{\mathsf{Sen}}$.
- $\mathcal{H}_2$: This hybrid is identical to the previous one, except that we change $\mathbf{A}'$ to a random value as in $\mathsf{Sim}_{\mathsf{Sen}}$.
- $\mathcal{H}_3$: This hybrid is identical to the previous one, except that we change the opening proof and commitment. Observe that we now arrive at the distribution of $\mathsf{Sim}_{\mathsf{Sen}}$.

Now,

- $\mathcal{H}_0 \approx_c \mathcal{H}_1$: Even though $C(r)$ was defined differently, it is the same value.
- $\mathcal{H}_1 \approx_c \mathcal{H}_2$: Since $\mathbf{A}$ is a random vector, $\mathbf{A} + \mathbf{P}$ hides $\mathbf{P}$ similar to a one-time pad.
- $\mathcal{H}_2 \approx_c \mathcal{H}_3$: Observe that the opening proof and commitments are generated with the actual input in $\mathcal{H}_2$ whereas they are generated using the simulators $(\mathsf{Sim}_1, \mathsf{Sim}_2)$ in $\mathcal{H}_3$. Indistinguishability between these two hybrids then follows directly from the hiding property of polynomial commitment schemes.

Observe that the check $A'(r) = \Delta^{-1} \cdot (C(r) - B(r)) + P(r)$ passes by our definition of $C(r)$. Also, the opening proof passes by hiding property. Thus, the probability of abort in the ideal and real world are same. Now, if an abort did not happen, it is easy to see that the output of honest receiver in real world is same as in ideal world. This is because the checks done by the simulator in step 6 are precisely the ones an honest receiver would do for his points. More formally, we have that if $x_j = y$ for some $y \in Y$,

$$\begin{aligned}
&\mathsf{Decode}(\mathbf{K}, x_j) - \Delta H^{\mathbb{F}}(x_j) + w \\
=&\mathsf{Decode}(\Delta\mathbf{A}' + \mathbf{B}, x_j) - \Delta H^{\mathbb{F}}(x_j) + w \\
=&\mathsf{Decode}(\Delta\mathbf{A} + \mathbf{B}, x_j) + \mathsf{Decode}(\Delta\mathbf{P}, x_j) - \Delta H^{\mathbb{F}}(x_j) + w \\
=&\mathsf{Decode}(\mathbf{C}, x_j) + \Delta\mathsf{Decode}(\mathbf{P}, x_j) - \Delta H^{\mathbb{F}}(x_j) + w \\
=&\mathsf{Decode}(\mathbf{C}, y) + \Delta\mathsf{Decode}(\mathbf{P}, y) - \Delta H^{\mathbb{F}}(y) + w \\
=&\mathsf{Decode}(\mathbf{C}, y) + w
\end{aligned}$$

and so, x_j appears in the intersection if the corresponding entry in $\mathbf{Z}'$ is masked with r_j. It is also easy to see that all x' such that x' was not committed to initially i.e. $H'(x' \| r') \notin \mathbf{L}$ do not appear in the intersection ($\because H'(x' \| r') \neq H'(x_j \| r_j)$ for all j with all but negligible probability) and so, the receiver does not include it in the output.

Corrupt receiver: Consider the simulator $\mathsf{Sim}_{\mathsf{Rec}}$:

- Committing Phase:

1. $\mathsf{Sim}_{\mathsf{Rec}}$ samples a random vector $\mathbf{L}$ of length m and sends the Merkle root R for the Merkle tree with $\mathbf{L}$ and receives a commitment σ.
2. $\mathsf{Sim}_{\mathsf{Rec}}$ extracts the input $\mathbf{Y}$ of the receiver based upon σ using the extractor from the binding property of the polynomial commitment scheme i.e. $P'(\cdot) = \mathcal{E}^{\mathcal{A}}(\mathsf{pp}, n + c \cdot M_{\mathsf{Rec}}, \sigma)$. The OKVS $\mathbf{P}$ can be extracted as the first n' elements of the corresponding vector. $\mathbf{Y}$ can then be extracted from $\mathbf{P}$ based on RO queries made and consistency checks with $\mathbf{P}$ i.e. $\mathbf{Y}$ consists of all RO queries y made such that $\mathsf{Decode}(\mathbf{P}, y) = H^{\mathbb{F}}(y)$.
3. $\mathsf{Sim}_{\mathsf{Rec}}$ sends $\mathbf{Y}$ to the ideal functionality $\mathcal{F}_{\mathsf{PICS}}$.

- Intersection Phase:
  1. $\mathsf{Sim}_{\mathsf{Rec}}$ sends $\alpha \xleftarrow{\$} \mathbb{F}$ to the receiver.
  2. When the receiver invokes $\mathcal{F}_{\mathsf{VOLE}}$, $\mathsf{Sim}_{\mathsf{Rec}}$ plays the role of $\mathcal{F}_{\mathsf{VOLE}}$. Given $\mathbf{A}, \mathbf{C} \in \mathbb{F}^{n' + c \cdot M_{\mathsf{Rec}}}$ from the receiver, sample $\Delta \xleftarrow{\$} \mathbb{F}$ and sets $\mathbf{B} = \mathbf{C} - \Delta\mathbf{A}$.
  3. Upon receiving $(\mathbf{A}', v)$ send $r \xleftarrow{\$} \mathbb{F}$.
  4. When the receiver opens the commitment σ at r and sends $C(r)$, $\mathsf{Sim}_{\mathsf{Rec}}$ aborts based on the checks done in the real protocol.
  5. $\mathsf{Sim}_{\mathsf{Rec}}$ checks if $\mathbf{P}'$ equals $\mathbf{A}' - \mathbf{A}$. If not, it aborts. Else, it sends $\mathbf{P}$ to $\mathcal{F}_{\mathsf{PICS}}$ and gets back the output $\mathbf{O}$. Sample a random $u \xleftarrow{\$} \mathbb{F}$ and program the RO such that $H^{\mathbb{F}}(u) = \alpha$. Define $w = u + v$ and $\mathbf{K}$ as the first n' elements of $\Delta\mathbf{A}' + \mathbf{B}$. Define the set $\mathbf{Z}$ to be the set of all $H(o_i, t_i)$ values where $t_i = \mathsf{Decode}(\mathbf{K}, o_i) - \Delta H^{\mathbb{F}}(o_i) + w$ and $o_i \in \mathbf{O}$. Sample $r_i \xleftarrow{\$} \mathbb{F}$ and program RO so that the Merkle root R is consistent with $\mathbf{L}$ where $\mathbf{L}[j] = H'(o_i \| r_i)$ for randomly sampled j. Now, define $\mathbf{Z}' = \{H'(o_i \| t_i) + r_i \mid o_i \in \mathbf{O}\}$ Send all these values to the receiver after randomly padding to appropriate size.

Since the sender receives no output, correctness follows directly. We now show that the output of $\mathsf{Sim}_{\mathsf{Rec}}$ is computationally indistinguishable from the view of the receiver. We show this via a sequence of hybrids:

- $\mathcal{H}_0$: Transcript is generated as in the real protocol.
- $\mathcal{H}_1$: This hybrid is identical to the previous one, except that we add the check in step 5 and abort if it does not hold.
- $\mathcal{H}_2$: This hybrid is identical to the previous one, except that we send random α at the start and later choose a random u and program $H^{\mathbb{F}}(u) = \alpha$
- $\mathcal{H}_3$: This hybrid is identical to the previous one, except that we now sample r_i values later and program RO so that values are consistent with Merkle tree
- $\mathcal{H}_4$: This hybrid is identical to the previous one, except that we replace the set $\mathbf{Z}'$
- $\mathcal{H}_5$: This hybrid is identical to the previous one, except that we now replace the set $\mathbf{Z}$. Observe that we now arrive at the output of $\mathsf{Sim}_{\mathsf{Rec}}$.

Now,

- $\mathcal{H}_0 \approx_c \mathcal{H}_1$: Suppose the receiver committed to a polynomial $P'(x)$ during the Committing Phase. Observe that this must be the value extracted during the Committing Phase by the binding property of the polynomial commitment scheme. Assume that the receiver sent $\mathbf{A}' \neq \mathbf{A} + \mathbf{P}'$ and some potentially incorrect value $C'(r)$. Now, assume the check passed. Then, $\Delta(A'(r) - P'(r)) = C'(r) - B(r)$ Substituting and rearranging, we get $\Delta = \frac{C'(r) - C(r)}{A'(r) - A(r) - P'(r)}$

  Observe that the polynomial $A'(x) - A(x) - P'(x)$ is non-zero since $\mathbf{A}' \neq \mathbf{A} + \mathbf{P}'$ and so by the Schwartz-Zippel lemma, $A'(r) - A(r) - P'(r) \neq 0$ which is why we can divide in the above equation. Observe that all values in the RHS are known to the receiver. This allows him to compute Δ, but the only value given after the VoLE execution was a randomly sampled r. This violates the security of VOLE. Therefore, $\mathbf{A}'$ must be equal to $\mathbf{A} + \mathbf{P}'$ with very high probability if the check in step 4 passed.
- $\mathcal{H}_1 \approx_c \mathcal{H}_2$: Observe that since u is sampled u.a.r, it is never queried to the RO with very high probability, in which case both distributions are identical.
- $\mathcal{H}_2 \approx_c \mathcal{H}_3$: This follows because $o_i \| r_i$ values are never queried to the RO with high probability and so programming the RO can be done.
- $\mathcal{H}_3 \approx_c \mathcal{H}_4$: Observe that the values in $\mathbf{Z}'$ for elements in the intersection are the same. For elements x_i not in the intersection, $H'(x_i \| t_i)$ is random since it is never queried (guessing t_i for such x_i would imply either guessing Δ or encoding more than n elements into the OKVS; both of which happen with negligible probability. Since this reduction follows from a similar reduction in the standard VOLE-PSI protocol, we refer interested readers to [64] for a more detailed argument.)
- $\mathcal{H}_4 \approx_c \mathcal{H}_5$: Observe that the values in $\mathbf{Z}$ for elements in the intersection are the same. Similar to the previous argument, $x_i \| t_i$ values for elements x_i not in the intersection are never queried to the RO H and so $H(x_i \| t_i)$ look random.

To extend this proof to Definition 4, we can compose the simulators directly with a few subtleties i.e., for each event E_i, if E_i is a Committing Phase (or Intersection Phase), we run the corresponding simulator respectively. This works because:

1. All hash queries during the Intersection Phase involve using a random coin w which is different for each execution with high probability and so, programming the RO can be done independently during each execution.
2. Observe that the hiding property of polynomial commitment scheme is valid as long as the number of queried points is less than $M = M_{\mathsf{Rec}}$ which is true in our case.
3. Note that we do not re-program repeated $x_i \| r_i$ values. The simulator remembers the programmed values and uses them during subsequent executions.