PICS: Private Intersection over Committed (and reusable) Sets

Aarushi Goel
Purdue University
aarushi@purdue.edu

Peihan Miao
Brown University
peihan_miao@brown.edu

Phuoc Van Long Pham
Brown University
phuoc_pham_van_long@brown.edu

Satvinder Singh
Purdue University
sing1745@purdue.edu

***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 [EUROCRYPT 2021]), we present an efficient instantiation of PICS using lightweight cryptographic tools. We implement our protocol to demonstrate concrete efficiency. Compared to VOLE-PSI, for input sets of size 2^{24}, our communication overhead is as low as 1.1%. Our end-to-end performance overhead is 130% in the LAN setting and decreases to $80\% - 10\%$ in the WAN setting with bandwidths ranging from 200 to 5 Mbps.

1. Introduction

Private set intersection (PSI)—a special case of secure multiparty computation (MPC) [1], [2]—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 [3], testing of sequenced human genome [4], password breach detection [5], mobile private contact discovery [6], [7], and online advertising measurement [8], 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 [9], [10], [11], [12] only use lightweight cryptographic tools such as vector oblivious linear evaluation (VOLE) [13], [14], 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 [15] 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 [16], [17] 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 [5], 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:

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 [9], [10], [11], [12] with FRI-based proof systems [18] in a novel way. 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 as low as 1.1% for set size $n = 2^{24}$, while the computational overhead is $60 - 130\%$ for set sizes between 2^{16} to 2^{24}. However, since our communication overhead is minimal, this gap narrows over WAN networks. For instance, in WAN networks with 80ms RTT latency and 5Mbps bandwidth, our end-to-end runtime overhead reduces to 10% for set size $n = 2^{24}$.
4) We compare our performance with prior work on Authorized PSI [19] and client-server PSI [20], 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 "snapshot" of the set at each time period, enabling easier further audit.

The Apple's PSI Protocol. Apple introduced a PSI protocol [15] to combat child sexual abuse in 2021. At its core, the protocol finds a (fuzzy) intersection between a large dataset (provided by NCMEC and other child-safety organizations [21]) 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 [6], [7]. 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 [22], [23], [24], [25] allow users to check whether their pass-

words 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) [26], [27] 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.[1] 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.

1. 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."

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 [28]. Specifically, parties can run any standard maliciously secure PSI protocol during the Intersection Phase, while attaching a generic zero-knowledge proof [29] 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 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 [30], 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. This observation forms the foundation of our approach.

Our Strategy. To achieve a concretely efficient PICS protocol, we build upon the state-of-the-art maliciously secure VOLE-PSI protocol [9], [10], [11], [12].

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 [9]. 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 [9] is based on a Vector Oblivious Linear Evaluation (VOLE) sub-protocol. In the first step, the sender and receiver execute a VOLE sub-protocol [31], 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 their 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

$$H(t_1 \| x_1), \ldots, H(t_n \| x_n)$$

  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.

Sender's Implicit Commitment. The hash values $H(t_i \| x_i)$ implicitly bind the sender to her input. To ensure input consistency in PICS, the sender only needs to prove that the x_i's used in these hash computations match those in her initial commitment. Importantly, we do not need to prove that the t_i's were computed correctly or that the VOLE sub-protocol was executed honestly. This allows us to avoid any non-black-box use of the operations in the VOLE sub-protocol.

Skipping the verification of the t_i values is sufficient to prevent the sender from injecting new elements into her committed set during the Intersection Phase. However, it still allows the sender to execute the Intersection Phase on a strict subset of her committed inputs. As discussed in Section 1.3, this suffices for our intended applications.

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.

1.4.3. Proving Consistency Between Initial and Implicit Input Commitments. As discussed earlier, our goal is to design this proof 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. However, we instead choose to work with a *polynomial commitment* scheme. Since vectors can be cast as polynomials using polynomial interpolation, polynomial commitment also serves as an easy replacement for vector commitment.

Our decision to use a polynomial commitment instead of a vector commitment is driven by 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 sender's implicit commitment to her input set in the VOLE-PSI protocol can be reformulated (with the addition of a few extra messages) as a well-known polynomial commitment scheme based on FRI [18].

Custom Commit-and-Prove Style Proofs. Since the sender's implicit commitment aligns with FRI-based polynomial commitment, proving consistency between her initial and implicit commitments essentially reduces to a standard task in modern SNARK literature—proving consistency between two FRI-based polynomial commitments. We design our custom proofs using these techniques. The main novelty lies in how a subset of the VOLE-PSI messages can be

interpreted as a polynomial commitment. We defer details to Section 4.

The receiver's side is more challenging, as there is no direct implicit polynomial commitment to the receiver's input. 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 FRI-based polynomial commitment of the receiver's input. 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.

2. Preliminaries

In this section, we start by establishing some notation and then recall preliminary definitions.

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. 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.

Randomized 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 $1 \leq i \leq n$.

For privacy, we will also sometimes work with *randomized* low-degree extensions of polynomials. More formally, a r degree randomized extension of a polynomial $p(\cdot)$ is the polynomial $p'(x) = p(x) + (x^d - 1) \cdot R(x)$ where d is the degree of the polynomial $p(\cdot)$ and $R(x)$ is a randomly sampled polynomial of degree r.

We represent converting a vector $\mathbf{V}$ into a r degree randomized extension of the corresponding polynomial as $V(\cdot) \leftarrow \mathsf{Vec2RandPoly}(\mathbf{V}; r)$.

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.[2]

Definition 1 (Oblivious Key Value Store [10])**.** *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:*

2. The definition and constructions extend to pseudo-random values naturally.

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:*

$\mathcal{R}(k_1, \ldots, k_n)$:
  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 ([12], [32], [33], [9]) 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 $\text{IDEAL}_{f,\mathsf{Sim}(z),I}(\mathbf{x})$ 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 $\text{REAL}_{\pi,\mathcal{A}(z),I}(\mathbf{x})$ 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.

Definition 2 (Security with abort against malicious adversaries [34])**.** *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 (VOLE)

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 [35] 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:*** *Let* $\mathsf{pp} \xleftarrow{\$} \mathsf{PCS.Setup}(1^\lambda)$*. There exists a PPT simulator* Sim*, such that for any polynomial $p(\cdot)$ of degree less than d,* $(\sigma, r) \xleftarrow{\$} \mathsf{PCS.Commit}(\mathsf{pp}, d, p(\cdot))$*, if for all sets I, $(x_i, y_i, \pi_i)_{i\in I}$ such that* $\mathsf{PCS.Verify}(\mathsf{pp}, d, \sigma, x_i, y_i, \pi_i) = 1$*, then*

$$\{\mathsf{Sim}(\mathsf{pp}, d, \sigma, \{x_i, y_i\}_{i\in I})\} \approx_c \{\{\pi_i\}_{i\in I}\}$$

FRI based Polynomial Commitment Scheme Fast Reed-Solomon Interactive (FRI) [36] is a renowned and widely used oracle proof of proximity protocol that was later extended to a polynomial commitment scheme [18]. Although we use this commitment in a black-box manner, our protocol crucially relies on the structure of the $\mathsf{PCS.Commit}(\cdot)$ algorithm which we now briefly describe:

Commit: On input a polynomial $p(\cdot)$ of degree less than d, define the evaluation domain $\mathcal{D}$ to be the set of D^{th} roots of unity over $\mathbb{F}$ where $D = c \cdot d$ for some suitably chosen constant c. Let $y_1, y_2, \ldots, y_D$ be evaluations of $p(\cdot)$ on $\mathcal{D}$, that is, $y_i = p(\omega^i)$ where ω is the generator of $\mathcal{D}$. Now, the commitment of the polynomial $p(\cdot)$ is the Merkle root corresponding to the Merkle commitment for the vector $(y_1, y_2, \ldots, y_D)$ using a hash function H modeled as a random oracle.

We make black-box use of the Open and Verify algorithms. We will also omit the randomness r and degree d when describing our construction for readability.

2.6. Zero-Knowledge Proofs/Arguments

We will use the standard definition of public-coin interactive zero-knowledge arguments of knowledge [29] and later compile them into non-interactive zero-knowledge arguments of knowledge (NIZKs) in the random oracle model using the Fiat-Shamir [37] transform. We recall the definition of NIZKs in the random oracle model now.

We use H to denote the random oracle and H in the superscript to denote that the corresponding algorithm makes black-box calls to H.

Definition 4 (NIZK in ROM (simplified from [38]))**.** *Let R be an NP relation corresponding to the language $\mathcal{L}_R$ i.e.,*

$$\mathcal{L}_R = \{x \mid \exists w \text{ s.t. } R(x,w) = 1\}$$

A non-interactive zero-knowledge proof system between a prover P and verifier V for $\mathcal{L}_R$ consists of a pair of PPT algorithms $(\mathsf{Prove}^H, \mathsf{Verify}^H)$ defined as follows:

- $\pi \xleftarrow{\$} \mathsf{Prove}^H(x,w)$: *On input a statement x and a witness w, this (randomized) algorithm outputs a proof π.*
- $0/1 \leftarrow \mathsf{Verify}^H(x,\pi)$: *On input a statement x and a proof π, this algorithm outputs a bit 0/1 denoting, reject/accept respectively.*

These algorithms satisfy the following properties:

- **Completeness:** For all x, w such that $R(x,w) = 1$,

$$\Pr[\mathsf{Verify}^H(x, \mathsf{Prove}^H(x,w)) \neq 1] \leq \mathsf{negl}(\lambda)$$

- **Knowledge Soundness:** There exists a PPT extractor $\mathcal{E}$, such that for any malicious PPT prover P^*, statement-proof pair $(x,\pi) \leftarrow P^*(1^\lambda)$, the following holds:

$$\Pr[R(x,w) \neq 1 \wedge \mathsf{Verify}^H(x,\pi) = 1] \leq \mathsf{negl}(\lambda),$$

  where $w \leftarrow \mathcal{E}^{P^*}(x)$.
- **Zero Knowledge:** There exists a PPT simulator Sim^H, such that for all statement-witness pairs (x,w) where $R(x,w) = 1$, the following holds:

$$\{\mathsf{Sim}^H(x)\} \approx_c \{\mathsf{Prove}^H(x,w)\}$$

Commit-and-Prove NIZK. In this work, we will use NIZKs for a specific class of languages comprising of statements corresponding to pre-committed polynomials. In particular, these are languages of the form:

$$\mathcal{L}' = \{(pp,\sigma) : \exists\ (p(\cdot), w) \text{ s.t. } \sigma = \mathsf{PCS.Commit}(pp, p(\cdot)) \wedge R(p(\cdot), w) = 1\}$$

NIZKs corresponding to such pre-committed values are often referred to as *commit-and-prove* NIZKs. Since our protocol is in the ROM, we generally omit the superscript H in our protocol descriptions.

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, i.e., we assume that the parties know the maximum number of intersections ahead of time, that they might wish to compute in the future. This simplification makes the protocol easier to explain and analyze. Meanwhile, this simplification is justified for the following reasons:

1) From a practical perspective, for sets of size n, our implementation supports up to n intersections. In other words, for input sets with a million elements, our implemented protocol supports a million intersections, which suffices for most real world use cases.
2) We discuss a simple extension of our protocol in Section 4.3 that supports an unbounded number of intersections at minimal cost.

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. They 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 $X' \subseteq X$ and $Y' \subseteq Y$, such that the output of the Intersection Phase is $X' \cap Y'$. 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. A formal description of the ideal functionality $\mathcal{F}_{\mathsf{PICS}}$, capturing these properties, is provided 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.

Definition 5 (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

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

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

Figure 2: PICS Functionality

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.

4.1.1. Sender. We first outline our approach for the sender.

Re-imagining Sender's Implicit Commitment as a FRI-Based Polynomial Commitment. Let us discuss how one can augment the sender's implicit commitment, i.e., $H(t_1\|x_1), \ldots, H(t_n\|x_n)$ (refer Section 1.4), to cast it as a FRI-based polynomial. Let $\{\delta^1, \ldots, \delta^{4cn}\}$ be the $4cn$-th[3] roots of unity[4], and let $T(x)$ be a degree $(2n-1)$ polynomial obtained by interpolating points $\{t_1, x_1, \ldots, t_n, x_n\}$, such that for each $i \in [n]$, $T(\delta^{4ci}) = x_i$ and $T(\delta^{4ci-1}) = t_i$.

In order to commit to this polynomial using the FRI-based polynomial commitment, one would start by evaluating it at $\{\delta^1, \ldots, \delta^{4cn}\}$ and then create a Merkle tree using $T(\delta^1), \ldots, T(\delta^{4cn})$ as the leaves of the tree. Observe that n of the $2cn$ nodes on the second level from bottom (i.e., the level above leaves) are of the form:

$$H(T(\delta^{4ci-1}), T(\delta^{4ci})) = H(t_i, x_i).$$

This precisely correspond to the sender's implicit commitment in the VOLE PSI protocol. Therefore, in order to send a polynomial commitment to $T(x)$, it suffices for the sender to additionally send the remaining $2cn - n$ nodes in the second level of the Merkle tree. Given these $2cn$ hash values, the receiver can compute the root of the Merkle tree which corresponds to a polynomial commitment to $T(x)$.

Explicit Initial Commitment. We again use the FRI-based polynomial commitment to commit to the sender's inputs in the Committing Phase. In particular, let $\{\omega^1, \ldots, \omega^n\}$ be the n-th roots of unity, and let $X(x)$ be a degree $n-1$ polynomial obtained by interpolating the sender's inputs $\{x_1, \ldots, x_n\}$ on these roots of unity, i.e., for each $i \in [n]$, $X(\omega^i) = x_i$. The sender then commits to this polynomial using the FRI-based polynomial commitment.

Proof of Consistency. Now that we have FRI commitments to $T(x)$ and $X(x)$, all that remains is to show that these polynomials were defined consistently using the same set of x_i values. In other words, we want to show that $\forall i \in [n]$:

$$T(\delta^{4ci}) = X(\omega^i) \tag{1}$$

Recall that since δ^i's are $4cn$-th roots of unity and ω^i's are n-th roots of unity, it must hold that for each $i \in [n]$, $\delta^{4ci} = \omega^i$. Therefore, the above check reduces to ensuring that for all $i \in [n]$, $T(\omega^i) = X(\omega^i)$. In other words, there must exist a polynomial $D(x)$ of degree at most $n-1$, such that

$$T(x) - X(x) = (x^n - 1) \cdot D(x).$$

This check can be done using a standard approach employed in the design of commit-and-prove zkSNARKs, where the prover additionally sends a polynomial commitment to $D(x)$. The verifier then samples a random evaluation point r in the field. The prover evaluates all these polynomials $T(x)$, $X(x)$, and $D(x)$ at this point and sends the corresponding polynomial opening proofs. The verifier checks if $T(r) - X(r) = (r^n - 1) \cdot D(r)$. Soundness follows from the Schwartz-Zippel Lemma ([39], [40]).

3. c is the soundness parameter for FRI polynomial commitment.
4. We explain later why we choose this specific set of roots of unity

Input-Hiding Polynomial Commitments. We note that the FRI-based polynomial commitment, on its own, does not provide any hiding guarantees. Moreover, as the committer opens evaluations of the polynomial, progressively more information about the polynomial is leaked. Since the polynomial encodes the set X, this leaks information about X. To remedy this, we use the standard technique of randomizing the polynomial to which one wishes to commit to. The degree of freedom in this polynomial is chosen depending upon the number of times the polynomial commitment is opened. This ensures that openings of the polynomial commitment do not leak information about the underlying vector. Furthermore, this randomization procedure is done outside the roots of unity at which the vector has been encoded. This is to ensure that the points of interest on this polynomial ($X(\omega^i)$ in the previous case) are not changed.

Choice of Roots of Unity. We now explain our decision to choose the $4cn$-th roots of unity in the previous argument. Note that the way we define the polynomial $T(\cdot)$ is directly related to the structure of the FRI polynomial commitment scheme. Care needs to be taken to ensure that this structure is compatible with the parameters of the FRI scheme. Observe that $T(\cdot)$ has degree $2n-1$ and we open $T(\cdot)$ at a random point r. Thus, the degree of the randomized extension of $T(\cdot)$ is $2n$. To ensure compatibility with FRI, the roots of unity chosen must have size $> 2cn$ where c is the soundness parameter for FRI. Furthermore, the order of the generator for this set of roots of unity must be divisible by n and 2 for (1) and FFT to hold respectively. The smallest such value that satisfies all these constraints is $4cn$; hence our choice for the roots of unity.

4.1.2. Receiver. We now focus our attention towards the 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 the FRI-based polynomial commitment.

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.

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 $\mathsf{out} \in \mathbb{N}$ be a parameter, such that $\mathsf{out} \geq \log |\mathbb{F}|$.
- Let $H : \{0,1\}^{\mathsf{out}} \times \{0,1\}^{\mathsf{out}} \to \{0,1\}^{\mathsf{out}}$ 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' over $\mathbb{F}$.
- 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 soundness parameter. Let $\mathsf{pp} \leftarrow \mathsf{PCS.Setup}(1^\lambda)$.
- Input sets X and Y are treated as vectors $\mathbf{X}$ and $\mathbf{Y}$ over $\mathbb{F}$. We still refer to them as sets to improve readability. 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 $\mathsf{Vec2RandPoly}$ be an algorithm for converting vectors to randomized 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. Each party also checks that their own counter is below the corresponding threshold, i.e., $\mathsf{cnt}_{\mathsf{Sen}} < M_{\mathsf{Sen}}$ and $\mathsf{cnt}_{\mathsf{Rec}} < M_{\mathsf{Rec}}$. If any check fails, the protocol aborts. Otherwise, they proceed and increment their counters by 1. Let $m = |\mathbf{X}|$ and $n = |\mathbf{Y}|$.

We defer the proofs of the following theorems to Appendix A due to space constraints.

Theorem 1. *The protocol in Figures 3a, 3b and 3c is a protocol for private intersection over committed sets (see Definition 5) against malicious adversaries in the $\mathcal{F}_{\mathsf{VOLE}}$-hybrid model.*

Theorem 2. *The protocol in Figure 4 is a interactive zero-knowledge argument of knowledge [29] for $\mathcal{L}$ (where $\mathcal{L}$ is as in Figure 3c)*

Communication Complexity. The communication cost for our Committing Phase is $O(\lambda)$ and for our Intersection Phase is $O(\lambda \cdot (m + n + \log^2(m + M_{\mathsf{Sen}}) + \log^2(n + M_{\mathsf{Rec}}))$.

Computation Complexity. The computation cost for the sender is $O(\lambda(m + M_{\mathsf{Sen}}) \log(m + M_{\mathsf{Sen}}))$ in the Committing Phase and $O(\lambda \cdot ((m+M_{\mathsf{Sen}}) \log^2(m+M_{\mathsf{Sen}}) + n + \log^2(n+M_{\mathsf{Rec}})))$ in the Intersection Phase. The computation cost for the receiver is $O(\lambda(n+M_{\mathsf{Rec}}) \log(n+M_{\mathsf{Rec}}))$ in the Committing Phase and $O(\lambda \cdot ((n + M_{\mathsf{Rec}}) \log^2(n + M_{\mathsf{Rec}}) + m + \log^2(m + M_{\mathsf{Sen}})))$ in the Intersection Phase.

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 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 i.e., the number of future intersections is fixed beforehand during the Committing Phase, we note that you can support an unbounded number of intersections. Suppose a party P committed to their inputs 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 (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 $X \subseteq X'$

π_{Com} **(Sender)**

Input: Set $\mathbf{X}$ and maximum number of intersections M_{Sender}

1) Randomly shuffle $\mathbf{X}$. Compute

$$X(\cdot) \leftarrow \mathsf{Vec2RandPoly}(\mathbf{X}; M_{\mathsf{Sen}})$$

$$\sigma_{X(\cdot)} \leftarrow \mathsf{PCS.Commit}(\mathsf{pp}, X(\cdot)) \quad (2)$$

2) Broadcast $\sigma_{X(\cdot)}$. Set a local counter $\mathsf{cnt}_{\mathsf{Sen}} = 0$.

(a) Committing Phase (Sender)

π_{Com} **(Receiver)**

Input: Set $\mathbf{Y}$ and maximum number of intersections M_{Receiver}

1) Randomly shuffle $\mathbf{Y}$ and compute OKVS $\mathbf{P}$ as

$$\mathbf{P} = \mathsf{Encode}\left(\{(y, H^F(y)) \mid y \in \mathbf{Y}\}\right).$$

2) Compute

$$P(\cdot) \leftarrow \mathsf{Vec2RandPoly}(\mathbf{P}; M_{\mathsf{Rec}})$$

$$\sigma_{P(\cdot)} \leftarrow \mathsf{PCS.Commit}(\mathsf{pp}, P(\cdot))$$

3) 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 obtains $\Delta \in \mathbb{F}, \mathbf{B} \in \mathbb{F}^{n'}$ and Receiver obtains $\mathbf{A}, \mathbf{C} \in \mathbb{F}^{n'}$ such that $\mathbf{C} = \Delta\mathbf{A} + \mathbf{B}$.
3) Receiver: Sample $v \xleftarrow{\$} \mathbb{F}$ and define $\mathbf{A}' = \mathbf{A} + \mathbf{P}$ where $\mathbf{P}$ is the OKVS defined during the Committing Phase. Send $(v, \mathbf{A}')$ to 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 $C(\cdot) \leftarrow \mathsf{Vec2RandPoly}(\mathbf{C}; 0)$.
6) Sender: Does the following:
   a) **Verify $\mathbf{A}' = \mathbf{A} + \mathbf{P}$:** Check if

$$\mathsf{PCS.Verify}(\mathsf{pp}, \sigma_{P(\cdot)}, r, P(r), \pi_{P(r)}) = 1,$$

$$A'(r) = \Delta^{-1} \cdot (C(r) - B(r)) + P(r).$$

   If either check fails, abort.
   b) **Compute t_i values:** Define $w = u + v$ and an OKVS $\mathbf{K} = \Delta\mathbf{A}' + \mathbf{B}$. For all $x_i \in \mathbf{X}$, compute

$$t_i = \mathsf{Decode}(\mathbf{K}, x_i) - \Delta H^{\mathbb{F}}(x_i) + w$$

   c) **Generate NIZK Proof:** Let $\{\delta^1, \delta^2, \ldots, \delta^{4cm}\}$ be the $4cm$-th roots of unity. Define a random $2m$-degree polynomial $T(\cdot)$ such that, for all $1 \le i \le m$, $T(\delta^{4ci}) = x_i$ and $T(\delta^{4ci-1}) = t_i$. Define a vector $\mathbf{Z}$, a polynomial $D(\cdot)$, and a language $\mathcal{L} = \{\sigma_{X(\cdot)}, \mathbf{Z} \mid \exists\, X(\cdot), T(\cdot), D(\cdot)$ such that eqs (2), (3) and (4) hold$\}$, where

$$\mathbf{Z} = \left(H(T(\delta^{2i-1}), T(\delta^{2i})) \;\middle|\; i \in [1, 2cm]\right) \quad (3)$$

$$T(x) - X(x) = D(x) \cdot (x^m - 1). \quad (4)$$

   where $\sigma_{X(\cdot)}$ is the commitment made during the Committing Phase. Compute a NIZK proof for $\mathcal{L}$:

$$\pi_{\mathsf{NIZK}} = \mathsf{Prove}^H((\sigma_{X(\cdot)}, \mathbf{Z}), (X(\cdot), T(\cdot), D(\cdot))).$$

   (We give an efficient construction of a NIZK for $\mathcal{L}$ in Figure 4.) Send $(\mathbf{Z}, \pi_{\mathsf{NIZK}}, u)$ to the Receiver.
7) Receiver: Check if $u' = H^{\mathbb{F}}(u)$ and if the NIZK proof verifies i.e., $\mathsf{Verify}^H((\sigma_{X(\cdot)}, \mathbf{Z}), \pi_{\mathsf{NIZK}}) = 1$. If not, abort. Else, output $\{y_i \in \mathbf{Y} \mid H(\mathsf{Decode}(\mathbf{C}, y_i) + w, y_i) \in \mathbf{Z}\}$.

(c) Intersection Phase (colored text represents additional steps in our protocol compared to the base VOLE-PSI protocol)

Figure 3: Our PICS Protocol.

ZK Protocol for $\mathcal{L}$

1) P: Compute the polynomial $D(\cdot)$ such that $T(x) - X(x) = D(x) \cdot (x^m - 1)$ and $\sigma_{D(\cdot)} \leftarrow \mathsf{PCS.Commit}(\mathsf{pp}, D(\cdot))$ and send $\sigma_{D(\cdot)}$ to V.
2) V: Sample a random challenge $r \xleftarrow{\$} \mathbb{F}$ and send to P.
3) P: Open all the polynomial commitments at r i.e.,

$$(X(r), \pi_1) \leftarrow \mathsf{PCS.Open}(\mathsf{pp}, X(\cdot), \sigma_{X(\cdot)}, r)$$
$$(T(r), \pi_2) \leftarrow \mathsf{PCS.Open}(\mathsf{pp}, T(\cdot), \sigma_{T(\cdot)}, r)$$
$$(D(r), \pi_3) \leftarrow \mathsf{PCS.Open}(\mathsf{pp}, D(\cdot), \sigma_{D(\cdot)}, r)$$

Send $(X(r), \pi_1, T(r), \pi_2, D(r), \pi_3)$ to V.
4) V: Check if the openings are correct i.e.,

$$\mathsf{PCS.Verify}(\mathsf{pp}, \sigma_{X(\cdot)}, r, X(r), \pi_1) = 1$$
$$\mathsf{PCS.Verify}(\mathsf{pp}, \sigma_{T(\cdot)}, r, T(r), \pi_2) = 1$$
$$\mathsf{PCS.Verify}(\mathsf{pp}, \sigma_{D(\cdot)}, r, D(r), \pi_3) = 1$$

and if $T(r) - X(r) = D(r) \cdot (r^m - 1)$ holds. If not, reject. Else, accept.

Figure 4: Interactive ZK proof for $\mathcal{L}$ (defined in Figure 3c), compiled into NIZK using Fiat-Shamir.

using standard ZK techniques. A similar approach works for deletions i.e., removing elements as well. Observe that this cost is amortized across all intersections and hence, minimal. Furthermore, using techniques similar to our paper, this update check (insertions and deletions both) can be done extremely efficiently on the sender side.

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 FRI polynomial commitment to $X(\cdot)$. If a value x was included, then it must be the case that $X(\omega^i) = x$ 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. We note that this is the first implementation of PSI in Rust, which may be of independent interest. Our implementation is available at https://anonymous.4open.science/r/PICS-E54E.

5.1. Implementation Details

We choose computational security parameter $\lambda = 128$ and statistical security parameter $\sigma = 40$. Our plain-PSI implementation is based on the framework of VOLE-PSI [9], [11], yet we instantiate the framework with more recent constructions for VOLE [31] and OKVS [12]. Another important change is in the finite field, where STARK-252 is utilized for compatibility with the FRI polynomial commitment scheme.[5]

Parameters choice. The VOLE construction in [31] is based on the primal learning parity with noise (LPN) assumption [43] 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. [44], which takes into account attacks including Pooled Gauss [45], statistical decoding (SD) [46], [47], [48], [49], [50], information set decoding (ISD) [51], [52], [53], and the algebraic (AGB) attack [54].

For OKVS, we implement the scheme based on Random Band matrices (RB-OKVS) from [12], 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.[6] 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 [12].

Finally, the FRI polynomial commitment scheme implementation is rewritten based on the `lambdaworks` library [55], 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 [56] to choose parameters that achieve 128 bits of soundness security.

5.2. Experimental Results

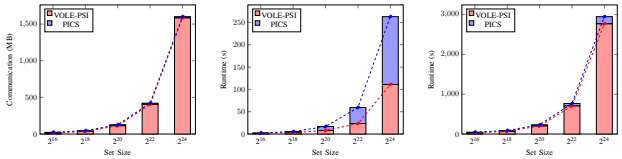
Our benchmarks are run on Amazon AWS `c5a.16xlarge` instances, with computations parallelized through 32 cores, while communication remains sequential. We simulate network bandwidth and latency with the Linux `tc` command, choosing 0.1ms RRT and 20Gbps bandwidth for LAN. Experiments in the WAN settings follow the previous work [19] with 80ms RTT, and bandwidths 200Mbps, 50Mbps, and 5Mbps, respectively. We provide in Table 5 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

5. Although recent work has extended the FRI degree test to arbitrary fields [41], [42], their concrete efficiency has not been fully analyzed.

6. The idea of hashing elements into bins is also considered in other PSI works such as [11].

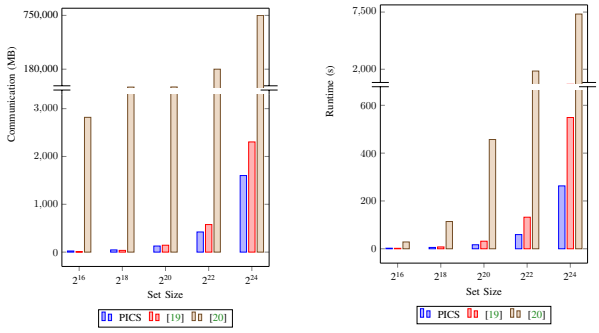
Set Size	Protocol	Sender Commit	Receiver Commit	Commitment Phase		Intersection Phase		Total	Total Runtime			
				Comm.	Comp.	Comm.	Comp.	Comm.	LAN	200Mbps	50Mbps	5Mbps
2^{16}	VOLE-PSI	✗	✗	-	-	13.9	1.41	13.9	1.41	8.35	9.27	28.7
	[19]	✗	✓	7.00	0.21	2.00	1.67	9.00	1.89	2.56	3.64	16.6
	[20]	✓	✗	0.52	0.37	2,818	28.1	2,818	28.45	141	479	4,537
	PICS	✓	✓	0.000064	0.43	23.4	1.91	23.41	2.33	11.1	12.2	46.0
2^{18}	VOLE-PSI	✗	✗	-	-	34.6	3.16	34.6	3.16	13.8	16.3	66.7
	[19]	✗	✓	28.0	0.85	8.00	6.83	36.0	7.70	9.44	13.8	65.6
	[20]	✓	✗	2.10	1.46	11GB	113	11GB	114	564	1,917	18GB
	PICS	✓	✓	0.000064	1.32	46.1	4.24	46.1	5.56	18.9	22.0	88.7
2^{20}	VOLE-PSI	✗	✗	-	-	113	8.02	113	8.02	21.3	32.9	201
	[19]	✗	✓	112	3.36	32	28.2	144	31.6	37.6	54.9	262
	[20]	✓	✗	8.39	5.85	45GB	451	45GB	457	2,255	7,665	73GB
	PICS	✓	✓	0	4.56	127	12.3	127	16.8	33.1	45.4	233
2^{22}	VOLE-PSI	✗	✗	-	-	405	23.4	405	23.4	53.2	96.3	706
	[19]	✗	✓	448	13.5	128	118	576	132	155	224	1,054
	[20]	✓	✗	33.6	23.4	180GB	1,804	180GB	1,827	9,018	31GB	290GB
	PICS	✓	✓	0.000064	18.1	421	41.2	421	59.3	92.0	136	767
2^{24}	VOLE-PSI	✗	✗	-	-	1,583	111	1,583	111	193	364	2,767
	[19]	✗	✓	1,792	54.2	512	494	2,304	549	641	917	4,235
	[20]	✓	✗	134	93.6	721GB	7,214	721GB	7,308	36GB	123GB	1,162GB
	PICS	✓	✓	0.000064	74.6	1,601	188	1,601	263	348	521	2,948

Figure 5: Experimental comparison. All communication costs are in MB unless otherwise noted, and all computation costs are in seconds.



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

Figure 6: Performance comparison of plain VOLE-PSI and PICS.



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

Figure 7: Performance comparison of PICS with [19], [20].

either the sender side [20] or the receiver side [19]. 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 [20] and [19] 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 only adding succinct 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, all proofs are succinct and only add $10-20$ MB of communication, which translates to $\approx 1.1\%$ communication overhead when the set size is 2^{24}. The computational overhead is higher due to the cost of FFT and Merkle Hash. However, the end-to-end running time overhead decreases significantly in the WAN setting, as shown in Figure 6, due to the low communication overhead in PICS. For instance, in WAN networks with bandwidth 50Mbps, the total runtime is only $30-45\%$ slower than VOLE-PSI, and the overhead is further reduced to less than 10% with bandwidth 5Mbps and sufficiently large set sizes ($n = 2^{22}$ and $n = 2^{24}$).

Comparison with [19] (receiver input consistency). In the Authorized Private Set Intersection (APSI) protocol of [19], a trusted third party (the Judge) authorizes the inputs of the receiver, preventing the receiver from injecting unauthorized elements when running PSI with the sender. APSI realizes a variant of the PICS functionality, where the receiver's set is committed. We report in Table 5 the communication cost of the Committing Phase as all messages exchanged between the receiver and the Judge, and that of the Intersection Phase as all messages between the sender and receiver afterward. We also report the computational

cost of each phase, as well as the end-to-end runtime (both phases combined), in which the LAN and WAN performance is estimated based on the communication cost.[7]

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 [19] only achieves security against a malicious receiver (and a semi-honest sender).

In terms of performance, in the Committing Phase, PICS has comparable running time with [19] and requires significantly less communication—we only require one Merkle hash whereas their communication grows linearly with the set size. Furthermore, our protocol is completely non-interactive, i.e., no interaction with a Judge is required. In the Intersection Phase, PICS falls short in communication but has more efficient computation, as it relies on lightweight cryptography while [19] requires expensive pairings operations. Considering the end-to-end performance combining both phases, as shown in Figure 5 and Figure 7, PICS has an advantage with $1.5\times$ improvement in communication cost. In terms of running time, PICS is $2.5\times$ faster in the LAN setting (which reflects the computation efficiency) and $1.5\times$ in the WAN setting with bandwidth 5Mbps (which reflects the communication efficiency).

Comparision with [20] (sender input consistency). Sun et. al. [20] proposed a PSI protocol in the client-server setting, where the server (sender) publishes a one-time encoding of its input set, which can be reused across multiple clients (receivers). This protocol can be considered as PICS with the sender's set committed. We measure the communication cost in the Committing Phase as the size of the published encoding by the sender, and define the Intersection Phase communication as all subsequent interactions between the sender and a receiver (in a single PSI execution).

We estimate their performance based on the per-element cost reported in [20].[8] In the Committing Phase, PICS and [20] have comparable runtime, while PICS provides a much more succinct commitment ($O(1)$ versus $O(n)$ in size). In the Intersection Phase, PICS achieves $20\times$ speedup in computation and more than $120\times$ reduction in communication. Overall, as shown in Figure 5 and Figure 7, the end-to-end running time of PICS is $30-390\times$ faster than [20].

5.3. Related Work

PSI was introduced by Meadows [57] and has been studied intensively from various techniques, including Diffie-Hellman [57], [58], [8], RSA [59], [60], garbled circuits [61], [62], [63], [32], fully homomorphic encryption (FHE) [64], [65], [66], oblivious transfer (OT) [67], [68], [69], [70], [71], and most recently VOLE [9], [10], [11], [12]. However, even the maliciously secure PSI protocols suffer from the vulnerabilities discussed above.

7. The source code of [19] does not simulate message-sending activities.
8. The implementation of [20] is not open-sourced.

Authorized PSI. Camenisch and Zaverucha [72] 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 [59] 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 [73] presented a more efficient APSI protocol from Bloom filters and homomorphic encryption. The work of Debnath and Dutta [74] further improved the efficiency for both APSI and APSI-Cardinality (where parties only learn the size of the intersection). Faber et al. [75] presented an APSI protocol with both sides authenticated. A more recent work [19] 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. [20]. 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 the need for a trusted third party.

Acknowledgments

Aarushi Goel would like to thank Gabriel Kaptchuk for helpful discussions on the applications of this framework. 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.

References

[1] A. C.-C. Yao, "How to generate and exchange secrets (extended abstract)," in *27th FOCS*. IEEE Computer Society Press, Oct. 1986, pp. 162–167.

[2] O. Goldreich, S. Micali, and A. Wigderson, "How to prove all NP-statements in zero-knowledge, and a methodology of cryptographic protocol design," in *CRYPTO'86*, ser. LNCS, A. M. Odlyzko, Ed., vol. 263. Springer, Berlin, Heidelberg, Aug. 1987, pp. 171–185.

[3] J. R. Troncoso-Pastoriza, S. Katzenbeisser, and M. Celik, "Privacy preserving error resilient dna searching through oblivious automata," in *ACM CCS 2007*, P. Ning, S. De Capitani di Vimercati, and P. F. Syverson, Eds. ACM Press, Oct. 2007, pp. 519–528.

[4] P. Baldi, R. Baronio, E. De Cristofaro, P. Gasti, and G. Tsudik, "Countering GATTACA: efficient and secure testing of fully-sequenced human genomes," in *ACM CCS 2011*, Y. Chen, G. Danezis, and V. Shmatikov, Eds. ACM Press, Oct. 2011, pp. 691–702.

[5] K. Thomas, J. Pullman, K. Yeo, A. Raghunathan, P. G. Kelley, L. Invernizzi, B. Benko, T. Pietraszek, S. Patel, D. Boneh, and E. Bursztein, "Protecting accounts from credential stuffing with password breach alerting," in *USENIX Security 2019*, N. Heninger and P. Traynor, Eds. USENIX Association, Aug. 2019, pp. 1556–1571.

[6] D. Demmler, P. Rindal, M. Rosulek, and N. Trieu, "PIR-PSI: Scaling private contact discovery," *PoPETs*, vol. 2018, no. 4, pp. 159–178, Oct. 2018.

[7] D. Kales, C. Rechberger, T. Schneider, M. Senker, and C. Weinert, "Mobile private contact discovery at scale," in *USENIX Security 2019*, N. Heninger and P. Traynor, Eds. USENIX Association, Aug. 2019, pp. 1447–1464.

[8] M. Ion, B. Kreuter, A. E. Nergiz, S. Patel, S. Saxena, K. Seth, M. Raykova, D. Shanahan, and M. Yung, "On deploying secure computing: Private intersection-sum-with-cardinality," in *IEEE European Symposium on Security and Privacy, EuroS&P 2020, Genoa, Italy, September 7-11, 2020*. IEEE, 2020, pp. 370–389. [Online]. Available: https://doi.org/10.1109/EuroSP48549.2020.00031

[9] P. Rindal and P. Schoppmann, "VOLE-PSI: Fast OPRF and circuit-PSI from vector-OLE," in *EUROCRYPT 2021, Part II*, ser. LNCS, A. Canteaut and F.-X. Standaert, Eds., vol. 12697. Springer, Cham, Oct. 2021, pp. 901–930.

[10] G. Garimella, B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai, "Oblivious key-value stores and amplification for private set intersection," in *CRYPTO 2021, Part II*, ser. LNCS, T. Malkin and C. Peikert, Eds., vol. 12826. Virtual Event: Springer, Cham, Aug. 2021, pp. 395–425.

[11] S. Raghuraman and P. Rindal, "Blazing fast PSI from improved OKVS and subfield VOLE," in *ACM CCS 2022*, H. Yin, A. Stavrou, C. Cremers, and E. Shi, Eds. ACM Press, Nov. 2022, pp. 2505–2517.

[12] A. Bienstock, S. Patel, J. Y. Seo, and K. Yeo, "Near-optimal oblivious key-value stores for efficient PSI, PSU and volume-hiding multi-maps," in *USENIX Security 2023*, J. A. Calandrino and C. Troncoso, Eds. USENIX Association, Aug. 2023, pp. 301–318.

[13] E. Boyle, G. Couteau, N. Gilboa, and Y. Ishai, "Compressing vector OLE," in *ACM CCS 2018*, D. Lie, M. Mannan, M. Backes, and X. Wang, Eds. ACM Press, Oct. 2018, pp. 896–912.

[14] E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, and P. Scholl, "Efficient pseudorandom correlation generators: Silent OT extension and more," in *CRYPTO 2019, Part III*, ser. LNCS, A. Boldyreva and D. Micciancio, Eds., vol. 11694. Springer, Cham, Aug. 2019, pp. 489–518.

[15] A. Bhowmick, D. Boneh, S. Myers, K. Talwar, and K. Tarbe, "The Apple PSI System," https://www.apple.com/child-safety/pdf/Apple_PSI_System_Security_Protocol_and_Analysis.pdf, 2021.

[16] S. Scheffler, A. Kulshrestha, and J. R. Mayer, "Public verification for private hash matching," in *2023 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2023, pp. 253–273.

[17] H. Abelson, R. J. Anderson, S. M. Bellovin, J. Benaloh, M. Blaze, J. Callas, W. Diffie, S. Landau, P. G. Neumann, R. L. Rivest, J. I. Schiller, B. Schneier, V. Teague, and C. Troncoso, "Bugs in our pockets: the risks of client-side scanning," *J. Cybersecur.*, vol. 10, no. 1, 2024. [Online]. Available: https://doi.org/10.1093/cybsec/tyad020

[18] E. Ben-Sasson, L. Goldberg, S. Kopparty, and S. Saraf, "DEEP-FRI: Sampling outside the box improves soundness," in *ITCS 2020*, T. Vidick, Ed., vol. 151. LIPIcs, Jan. 2020, pp. 5:1–5:32.

[19] F. Falzon and E. A. Markatou, "Re-visiting authorized private set intersection: A new privacy-preserving variant and two protocols," *Proceedings on Privacy Enhancing Technologies*, 2025.

[20] Y. Sun, J. Katz, M. Raykova, P. Schoppmann, and X. Wang, "Actively secure private set intersection in the client-server setting," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, 2024, pp. 1478–1492.

[21] Apple Inc., "Csam detection technical summary," Apple Inc., Tech. Rep., 2021, accessed: 2025-05-22. [Online]. Available: https://www.apple.com/child-safety/pdf/CSAM_Detection_Technical_Summary.pdf

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

[23] "Firefox Password Manager – Alerts for breached websites," https://support.mozilla.org/sl/kb/firefox-password-manager-alerts-breached-websites.

[24] "Password Monitor: Safeguarding passwords in Microsoft Edge," https://www.microsoft.com/en-us/research/blog/password-monitor-safeguarding-passwords-in-microsoft-edge/.

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

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

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

[28] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game or A completeness theorem for protocols with honest majority," in *19th ACM STOC*, A. Aho, Ed. ACM Press, May 1987, pp. 218–229.

[29] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof-systems (extended abstract)," in *17th ACM STOC*. ACM Press, May 1985, pp. 291–304.

[30] Y. Lindell, "How to simulate it - A tutorial on the simulation proof technique," in *Tutorials on the Foundations of Cryptography*, Y. Lindell, Ed. Springer International Publishing, 2017, pp. 277–346. [Online]. Available: https://doi.org/10.1007/978-3-319-57048-8_6

[31] C. Weng, K. Yang, J. Katz, and X. Wang, "Wolverine: Fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits," in *2021 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2021, pp. 1074–1091.

[32] B. Pinkas, T. Schneider, O. Tkachenko, and A. Yanai, "Efficient circuit-based PSI with linear communication," in *EUROCRYPT 2019, Part III*, ser. LNCS, Y. Ishai and V. Rijmen, Eds., vol. 11478. Springer, Cham, May 2019, pp. 122–153.

[33] B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai, "PSI from PaXoS: Fast, malicious private set intersection," in *EUROCRYPT 2020, Part II*, ser. LNCS, A. Canteaut and Y. Ishai, Eds., vol. 12106. Springer, Cham, May 2020, pp. 739–767.

[34] Y. Lindell, "How to simulate it - A tutorial on the simulation proof technique," Cryptology ePrint Archive, Report 2016/046, 2016. [Online]. Available: https://eprint.iacr.org/2016/046

[35] A. Kate, G. M. Zaverucha, and I. Goldberg, "Constant-size commitments to polynomials and their applications," in *ASIACRYPT 2010*, ser. LNCS, M. Abe, Ed., vol. 6477. Springer, Berlin, Heidelberg, Dec. 2010, pp. 177–194.

[36] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Fast reed-solomon interactive oracle proofs of proximity," in *ICALP 2018*, ser. LIPIcs, I. Chatzigiannakis, C. Kaklamanis, D. Marx, and D. Sannella, Eds., vol. 107. Schloss Dagstuhl, Jul. 2018, pp. 14:1–14:17.

[37] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *CRYPTO'86*, ser. LNCS, A. M. Odlyzko, Ed., vol. 263. Springer, Berlin, Heidelberg, Aug. 1987, pp. 186–194.

[38] A. Chiesa, D. Ojha, and N. Spooner, "Fractal: Post-quantum and transparent recursive proofs from holography," in *EUROCRYPT 2020, Part I*, ser. LNCS, A. Canteaut and Y. Ishai, Eds., vol. 12105. Springer, Cham, May 2020, pp. 769–793.

[39] J. T. Schwartz, "Fast probabilistic algorithms for verification of polynomial identities," *J. ACM*, vol. 27, no. 4, pp. 701–717, 1980. [Online]. Available: https://doi.org/10.1145/322217.322225

[40] R. Zippel, "Probabilistic algorithms for sparse polynomials," in *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposiumon Symbolic and Algebraic Computation, Marseille, France, June 1979, Proceedings*, ser. Lecture Notes in Computer Science, E. W. Ng, Ed., vol. 72. Springer, 1979, pp. 216–226. [Online]. Available: https://doi.org/10.1007/3-540-09519-5_73

[41] E. Ben-Sasson, D. Carmon, S. Kopparty, and D. Levit, "Scalable and transparent proofs over all large fields, via elliptic curves - (ECFFT part II)," in *TCC 2022, Part I*, ser. LNCS, E. Kiltz and V. Vaikuntanathan, Eds., vol. 13747. Springer, Cham, Nov. 2022, pp. 467–496.

[42] H. Zeilberger, B. Chen, and B. Fisch, "BaseFold: Efficient field-agnostic polynomial commitment schemes from foldable codes," in *CRYPTO 2024, Part X*, ser. LNCS, L. Reyzin and D. Stebila, Eds., vol. 14929. Springer, Cham, Aug. 2024, pp. 138–169.

[43] A. Blum, M. L. Furst, M. J. Kearns, and R. J. Lipton, "Cryptographic primitives based on hard learning problems," in *CRYPTO'93*, ser. LNCS, D. R. Stinson, Ed., vol. 773. Springer, Berlin, Heidelberg, Aug. 1994, pp. 278–291.

[44] H. Liu, X. Wang, K. Yang, and Y. Yu, "The hardness of LPN over any integer ring and field for PCG applications," in *EUROCRYPT 2024, Part VI*, ser. LNCS, M. Joye and G. Leander, Eds., vol. 14656. Springer, Cham, May 2024, pp. 149–179.

[45] A. Esser, R. Kübler, and A. May, "LPN decoded," in *CRYPTO 2017, Part II*, ser. LNCS, J. Katz and H. Shacham, Eds., vol. 10402. Springer, Cham, Aug. 2017, pp. 486–514.

[46] A. K. A. Jabri, "A statistical decoding algorithm for general linear block codes," in *Cryptography and Coding, 8th IMA International Conference, Cirencester, UK, December 17-19, 2001, Proceedings*, ser. Lecture Notes in Computer Science, B. Honary, Ed., vol. 2260. Springer, 2001, pp. 1–8. [Online]. Available: https://doi.org/10.1007/3-540-45325-3_1

[47] R. Overbeck, "Statistical decoding revisited," in *Information Security and Privacy, 11th Australasian Conference, ACISP 2006, Melbourne, Australia, July 3-5, 2006, Proceedings*, ser. Lecture Notes in Computer Science, L. M. Batten and R. Safavi-Naini, Eds., vol. 4058. Springer, 2006, pp. 283–294. [Online]. Available: https://doi.org/10.1007/11780656_24

[48] M. P. C. Fossorier, K. Kobara, and H. Imai, "Modeling bit flipping decoding based on nonorthogonal check sums with application to iterative decoding attack of mceliece cryptosystem," *IEEE Trans. Inf. Theory*, vol. 53, no. 1, pp. 402–411, 2007. [Online]. Available: https://doi.org/10.1109/TIT.2006.887515

[49] T. Debris-Alazard and J. Tillich, "Statistical decoding," in *2017 IEEE International Symposium on Information Theory, ISIT 2017, Aachen, Germany, June 25-30, 2017*. IEEE, 2017, pp. 1798–1802. [Online]. Available: https://doi.org/10.1109/ISIT.2017.8006839

[50] K. Carrier, T. Debris-Alazard, C. Meyer-Hilfiger, and J.-P. Tillich, "Statistical decoding 2.0: Reducing decoding to LPN," in *ASIACRYPT 2022, Part IV*, ser. LNCS, S. Agrawal and D. Lin, Eds., vol. 13794. Springer, Cham, Dec. 2022, pp. 477–507.

[51] E. Prange, "The use of information sets in decoding cyclic codes," *IRE Trans. Inf. Theory*, vol. 8, no. 5, pp. 5–9, 1962. [Online]. Available: https://doi.org/10.1109/TIT.1962.1057777

[52] A. May, A. Meurer, and E. Thomae, "Decoding random linear codes in $\tilde{\mathcal{O}}(2^{0.054n})$," in *ASIACRYPT 2011*, ser. LNCS, D. H. Lee and X. Wang, Eds., vol. 7073. Springer, Berlin, Heidelberg, Dec. 2011, pp. 107–124.

[53] A. Becker, A. Joux, A. May, and A. Meurer, "Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding," in *EUROCRYPT 2012*, ser. LNCS, D. Pointcheval and T. Johansson, Eds., vol. 7237. Springer, Berlin, Heidelberg, Apr. 2012, pp. 520–536.

[54] P. Briaud and M. Øygarden, "A new algebraic approach to the regular syndrome decoding problem and implications for PCG constructions," in *EUROCRYPT 2023, Part V*, ser. LNCS, C. Hazay and M. Stam, Eds., vol. 14008. Springer, Cham, Apr. 2023, pp. 391–422.

[55] lambdaworks contributors, "lambdaworks," 2023. [Online]. Available: https://github.com/lambdaclass/lambdaworks

[56] S. Team, "ethstark documentation–version 1.1," IACR preprint archive 2021, Tech. Rep., 2021.

[57] C. Meadows, "A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party," in *Proceedings of the 1986 IEEE Symposium on Security and Privacy, Oakland, California, USA, April 7-9, 1986*. IEEE Computer Society, 1986, pp. 134–137. [Online]. Available: https://doi.org/10.1109/SP.1986.10022

[58] B. A. Huberman, M. K. Franklin, and T. Hogg, "Enhancing privacy and trust in electronic communities," in *Proceedings of the First ACM Conference on Electronic Commerce (EC-99), Denver, CO, USA, November 3-5, 1999*, S. I. Feldman and M. P. Wellman, Eds. ACM, 1999, pp. 78–86. [Online]. Available: https://doi.org/10.1145/336992.337012

[59] E. De Cristofaro and G. Tsudik, "Practical private set intersection protocols with linear complexity," in *FC 2010*, ser. LNCS, R. Sion, Ed., vol. 6052. Springer, Berlin, Heidelberg, Jan. 2010, pp. 143–159.

[60] G. Ateniese, E. De Cristofaro, and G. Tsudik, "(If) size matters: Size-hiding private set intersection," in *PKC 2011*, ser. LNCS, D. Catalano, N. Fazio, R. Gennaro, and A. Nicolosi, Eds., vol. 6571. Springer, Berlin, Heidelberg, Mar. 2011, pp. 156–173.

[61] Y. Huang, D. Evans, and J. Katz, "Private set intersection: Are garbled circuits better than custom protocols?" in *NDSS 2012*. The Internet Society, Feb. 2012.

[62] B. Pinkas, T. Schneider, G. Segev, and M. Zohner, "Phasing: Private set intersection using permutation-based hashing," in *USENIX Security 2015*, J. Jung and T. Holz, Eds. USENIX Association, Aug. 2015, pp. 515–530.

[63] B. Pinkas, T. Schneider, C. Weinert, and U. Wieder, "Efficient circuit-based PSI via cuckoo hashing," in *EUROCRYPT 2018, Part III*, ser. LNCS, J. B. Nielsen and V. Rijmen, Eds., vol. 10822. Springer, Cham, Apr. / May 2018, pp. 125–157.

[64] H. Chen, K. Laine, and P. Rindal, "Fast private set intersection from homomorphic encryption," in *ACM CCS 2017*, B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, Eds. ACM Press, Oct. / Nov. 2017, pp. 1243–1255.

[65] H. Chen, Z. Huang, K. Laine, and P. Rindal, "Labeled PSI from fully homomorphic encryption with malicious security," in *ACM CCS 2018*, D. Lie, M. Mannan, M. Backes, and X. Wang, Eds. ACM Press, Oct. 2018, pp. 1223–1237.

[66] K. Cong, R. C. Moreno, M. B. da Gama, W. Dai, I. Iliashenko, K. Laine, and M. Rosenberg, "Labeled PSI from homomorphic encryption with reduced computation and communication," in *ACM CCS 2021*, G. Vigna and E. Shi, Eds. ACM Press, Nov. 2021, pp. 1135–1150.

[67] C. Dong, L. Chen, and Z. Wen, "When private set intersection meets big data: an efficient and scalable protocol," in *ACM CCS 2013*, A.-R. Sadeghi, V. D. Gligor, and M. Yung, Eds. ACM Press, Nov. 2013, pp. 789–800.

[68] B. Pinkas, T. Schneider, and M. Zohner, "Faster private set intersection based on OT extension," in *USENIX Security 2014*, K. Fu and J. Jung, Eds. USENIX Association, Aug. 2014, pp. 797–812.

[69] V. Kolesnikov, R. Kumaresan, M. Rosulek, and N. Trieu, "Efficient batched oblivious PRF with applications to private set intersection," in *ACM CCS 2016*, E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, Eds. ACM Press, Oct. 2016, pp. 818–829.

[70] B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai, "SpOT-light: Lightweight private set intersection from sparse OT extension," in *CRYPTO 2019, Part III*, ser. LNCS, A. Boldyreva and D. Micciancio, Eds., vol. 11694. Springer, Cham, Aug. 2019, pp. 401–431.

[71] M. Chase and P. Miao, "Private set intersection in the internet setting from lightweight oblivious PRF," in *CRYPTO 2020, Part III*, ser. LNCS, D. Micciancio and T. Ristenpart, Eds., vol. 12172. Springer, Cham, Aug. 2020, pp. 34–63.

[72] J. Camenisch and G. M. Zaverucha, "Private intersection of certified sets," in *FC 2009*, ser. LNCS, R. Dingledine and P. Golle, Eds., vol. 5628. Springer, Berlin, Heidelberg, Feb. 2009, pp. 108–127.

[73] F. Kerschbaum, "Outsourced private set intersection using homomorphic encryption," in *ASIACCS 12*, H. Y. Youm and Y. Won, Eds. ACM Press, May 2012, pp. 85–86.

[74] S. K. Debnath and R. Dutta, "Secure and efficient private set intersection cardinality using bloom filter," in *ISC 2015*, ser. LNCS, J. Lopez and C. J. Mitchell, Eds., vol. 9290. Springer, Cham, Sep. 2015, pp. 209–226.

[75] S. Faber, R. Petrlic, and G. Tsudik, "Unlinked: Private proximity-based off-line OSN interaction," in *Proceedings of the 14th ACM Workshop on Privacy in the Electronic Society, WPES 2015, Denver, Colorado, USA, October 12, 2015*, I. Ray, N. Hopper, and R. Jansen, Eds. ACM, 2015, pp. 121–131. [Online]. Available: https://doi.org/10.1145/2808138.2808149

Appendix

1. Proof of Theorem 1

Proof. 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) $\mathsf{Sim}_{\mathsf{Sen}}$ sends a polynomial commitment σ' to a random polynomial of degree $n+M_{\mathsf{Rec}}$ to the sender and receives back a commitment σ.
  2) $\mathsf{Sim}_{\mathsf{Sen}}$ extracts the input $\mathbf{X}$ based upon σ. More formally, recall that to commit to a polynomial, a party commits to the Merkle root corresponding to the Merkle tree on the evaluations of the polynomial at roots of unity. Thus, $\mathbf{X}$ can be extracted based upon the 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 some value 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 $\Delta \in \mathbb{F}, \mathbf{B} \in \mathbb{F}^{n'}$ from the sender, the simulator samples random $\mathbf{A} \xleftarrow{\$} \mathbb{F}^{n'}$ and sets $\mathbf{C} = \Delta\mathbf{A} + \mathbf{B}$.
  3) $\mathsf{Sim}_{\mathsf{Sen}}$ samples random $\mathbf{A}' \xleftarrow{\$} \mathbb{F}^{n'}$ and $v \xleftarrow{\$} \mathbb{F}$ and sends it to the sender.
  4) Upon receiving r from the sender, the simulator opens the commitment σ' at r, to say a value β, and sends $C(r) = (A'(r) - \beta)\Delta + B(r)$ to the sender.
  5) Observe that the check $A'(r) = \Delta^{-1} \cdot (C(r) - B(r)) + P(r)$ passes by our definition of $C'(r)$. $\mathsf{Sim}_{\mathsf{Sen}}$ now receives $(\mathbf{Z}, u, \pi)$ from the sender.
  6) If the NIZK proof is accepted, $\mathsf{Sim}_{\mathsf{Sen}}$ runs the knowledge soundness extractor to obtain the witness $(\mathbf{X}'', \mathbf{T})$. Based on the $H(T(\delta^{2i-1}), T(\delta^{2i}))$ queries made to the RO, the simulator can also extract $\mathbf{X}'$ i.e., compute all x' such that the sender queries (t', x') to the RO where $t' = \mathsf{Decode}(\mathbf{K}, x') - \Delta H^{\mathbb{F}}(x') + w$ If either $\mathbf{X} \neq \mathbf{X}''$ or $\mathbf{X}'$ is not a subset of $\mathbf{X}$, abort.
  7) $\mathsf{Sim}_{\mathsf{Sen}}$ sends $\mathbf{X}'$ to the ideal functionality $\mathcal{F}_{\mathsf{PICS}}$.

We first argue correctness i.e., that the receiver receives $X' \cap Y'$ if an abort did not happen. Observe that the soundness of our NIZK proof implies that $T(x) - X(x) = D(x).(x^m - 1)$ holds. This implies that the m-th roots of unity are roots of $T(x) - X(x)$. Thus, $T(\omega^i) = X(\omega^i) = x_i$. Recall that $\omega^i = \delta^{4ci}$ and thus $T(\delta^{4ci}) = x_i$. This implies that the values in $\mathbf{Z}$ are of the form (t'_i, x_i). Observe that only the x_i such that $t'_i = t_i$ occur in the output because X' satisfies this condition. More formally, for all t_i, for all j, $i = 2c \cdot j$, we have

$$H(T(\delta^{2i-1}), T(\delta^{2i})) = H(T(\delta^{4cj-1}), T(\delta^{4cj}))$$
$$= H(t_j, x_j) = H(\mathsf{Decode}(\mathbf{K}, x_j) - \Delta H^{\mathbb{F}}(x_j) + w, x_j)$$

Now, if $x_j = y$ for some $y \in Y$, we have

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

Therefore, we have $H(T(\delta^{2i-1}), T(\delta^{2i})) = H(\mathsf{Decode}(\mathbf{C}, y) + w, y)$ and so, x_j appears in the output. By a similar argument, it can be seen that all $x_j \notin Y$ do not appear in the output ($\because \mathsf{Decode}(\mathbf{P}, x_j) \neq H^{\mathbb{F}}(x_j)$ and $\Delta \neq 0$ both hold with very high probability).
All that remains to be shown is 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 substitute $\sigma_{P()}$ with σ' as in $\mathsf{Sim}_{\mathsf{Sen}}$. Also, replace the actual opening with β. and the corresponding opening proof with the output of the simulator for the polynomial commitment scheme (hiding property). Observe that we now arrive at the output 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 if abort does not happen.
- $\mathcal{H}_1 \approx_c \mathcal{H}_2$: Since $\mathbf{A}$ is a random n' length 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 value $P(r)$ sent is identical in both distributions since we work with M_{Rec} randomized degree extensions. The opening proof is identically distributed as well via the hiding property of the polynomial commitment scheme.

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

- Committing Phase:
  1) $\mathsf{Sim}_{\mathsf{Rec}}$ sends a polynomial commitment σ' to a random polynomial of degree $n + M_{\mathsf{Sen}}$ to the receiver and receives back a commitment σ.
  2) $\mathsf{Sim}_{\mathsf{Rec}}$ extracts the input $\mathbf{Y}$ of the receiver based upon σ. Similar to $\mathsf{Sim}_{\mathsf{Sen}}$, the OKVS $\mathbf{P}$ can be extracted and Y can be extracted from $\mathbf{P}$ based on RO queries made and consistency checks with $\mathbf{P}$.
  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 $\mathbf{A}, \mathbf{C} \in \mathbb{F}^{n'}$ from the receiver, sample $\Delta \xleftarrow{\$} \mathbb{F}$ and sets $\mathbf{B} = \mathbf{C} - \Delta\mathbf{A}$.
  3) Upon receiving $(\mathbf{A}', v)$ from the receiver, extract $\mathbf{P} = \mathbf{A}' - \mathbf{A}$ and check consistency with $\mathbf{Y}$. If inconsistent, $\mathsf{Sim}_{\mathsf{Rec}}$ finds r such that the check fails and sends it to the receiver. Else, 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 last step.
  5) $\mathsf{Sim}_{\mathsf{Rec}}$ sends $\mathbf{Y}$ 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 $\mathbf{K} = \Delta\mathbf{A}' + \mathbf{B}$. For $o_i \in \mathbf{O}$, compute $H(t_i, o_i)$ (where $t_i = \mathsf{Decode}(\mathbf{K}, o_i) - \Delta H^{\mathbb{F}}(o_i) + w$) and interpolate these values to obtain the polynomial $T(\cdot)$ (If degree is less, pad random evaluations.) $\mathsf{Sim}_{\mathsf{Rec}}$ can now compute $\mathbf{Z}$ using $T(\cdot)$. Generate π_{NIZK} using the simulator for the NIZK (zero knowledge property). Send all these values to the receiver.

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 send random α at the start and later choose a random u and program $H^{\mathbb{F}}(u) = \alpha$
- $\mathcal{H}_2$: This hybrid is identical to the previous one, except that we now change the way the set $\mathbf{Z}$ is defined
- $\mathcal{H}_3$: This hybrid is identical to the previous one, except that we replace the ZK proof with the one in $\mathsf{Sim}_{\mathsf{Rec}}$
- $\mathcal{H}_4$: This hybrid is identical to the previous one, except that we now replace r and the abort condition
- $\mathcal{H}_5$: This hybrid is identical to the previous one, except that we now replace $\sigma_{X()}$ with σ'. Also, replace the opening proof with the output of the simulator for the polynomial commitment scheme (hiding property). Observe that we now arrive at the output of $\mathsf{Sim}_{\mathsf{Rec}}$.

Now,

- $\mathcal{H}_0 \approx_c \mathcal{H}_1$: 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.
- $\mathcal{H}_1 \approx_c \mathcal{H}_2$: This follows since u is sampled randomly at the last step, w is independently and randomly sampled, and thus all the elements of $\mathbf{O}$ corresponding to the non-output entries look random since Δ is sampled u.a.r. Interested readers can refer to [9] for more details.
- $\mathcal{H}_2 \approx_c \mathcal{H}_3$: This follows from ZK property of NIZK.
- $\mathcal{H}_3 \approx_c \mathcal{H}_4$: Suppose the receiver committed to a polynomial $P(x)$ during the Committing Phase. 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 passed. Thus, we can safely change the abort condition.
- $\mathcal{H}_4 \approx_c \mathcal{H}_5$: Observe that $X(r)$ is identical in both distributions since we work with M_{Sen} randomized degree extensions and $\mathsf{cnt}_{\mathsf{Sen}} < M_{\mathsf{Sen}}$. The opening proof is identically distributed as well via the hiding property of the polynomial commitment scheme.

To extend this proof to Definition 5, we can compose the simulators directly 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 used in computing the output involve the use of 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) The polynomials committed during the Committing Phase have enough randomness to support M_{Sen} and M_{Rec} intersections respectively.

Thus, the ZK property and hiding property of polynomial commitments hold across all executions. □

2. Proof of Theorem 2

Proof. We show that the protocol satisfies all properties:
Completeness: Follows trivially.
Knowledge Soundness: Consider the following extractor $\mathcal{E}$:

1) $\mathcal{E}$ interacts with P^* to get a commitment σ_D
2) Sample r u.a.r. and send it to P^* who sends back $X(r), T(r), D(r)$.
3) Keep rewinding to the start of step 2 until you receive $\max \deg(X(\cdot), T(\cdot), D(\cdot))$ many values that satisfy the check. Use these to interpolate all polynomials. Abort if you rewind more than a polynomial number of times.

Note that the max degree is polynomial in $|X|$. Furthermore, we abort after some polynomial number of rewinds. Thus, $\mathcal{E}$ is clearly poly-time. It is also easy to see that if an adversary succeeds in coming up with a valid proof with non-negligible probability, then our extractor that rewinds a polynomial number of times also succeeds with non-negligible probability.
Zero Knowledge: Consider the following simulator $\mathcal{S}$:

1) $\mathcal{S}$ commits to a random polynomial $D(\cdot)$.
2) $\mathcal{S}$ queries V on the statement to get r. Program the RO such that $D(r)$ opens to $(T(r) - X(r))/(r^m - 1)$. We note that this can be done in the FRI polynomial commitment scheme. For the sake of completeness, we give a high-level idea for the same: The FRI opening proof can be seen as a $O(\log d)$ round interactive protocol (d is the degree of the polynomial committed), where in each round, a set of $O(\lambda)$ checks are done. Each check requires opening authentication paths in the Merkle tree corresponding to certain roots of unity. Now, we can program all values along these authentication path such that each check passes. More concretely, the check is a linear relation between $D(\omega^i), D(-\omega^i)$ and $D'(\omega^2)$. Irrespective of $D(\omega^i)$ and $D'(\omega^2)$ [9], we can find a value V such that the check passes. We program the parent of the leaf node corresponding to $D(-\omega^i)$ to be a valid evaluation of the RO on either $V||R$ or $L||V$, where L, R are the respective left and right nodes, depending upon the position of V.
3) $\mathcal{S}$ opens $X(r), T(r), D(r)$

Observe that the check done by the verifier clearly passes. Since all polynomials are suitable randomized degree extensions, revealing evaluation at a single point does not reveal anything about the witness. The opening proofs also leak nothing via the hiding property of the polynomial commitment scheme. □

9. This polynomial $D'(\cdot)$ is dependent on $D(\cdot)$. We refer the reader to [18] for further details.