Bandwidth Efficient Partial Authorized PSI

Tjitske Koster
TU Delft
Delft, The Netherlands
tjitskekoster.com

Francesca Falzon
ETH Zürich
Zürich, Switzerland
ffalzon.github.io

Evangelia Anna Markatou
TU Delft
Delft, The Netherlands
e.a.markatou@tudelft.nl

***Abstract*—Recent attacks on private set intersection (PSI) and PSI-like protocols have demonstrated that input privacy can be compromised when parties maliciously choose their inputs—even in protocols proven secure against malicious adversaries. To counter such attacks, Authorized PSI (APSI) introduces a judge who authorizes the elements of the parties before the intersection is computed. Falzon and Markatou (PETS 2025) proposed Partial-APSI, a privacy-preserving variant of APSI that prevents revealing the entire set to a judge. Their Partial-APSI protocol requires significant bandwidth overhead due to the use of bilinear pairings and because the judge must sign each element in the input set. In this work, we present a bandwidth-efficient Partial-APSI protocol that outperforms Falzon and Markatou, both asymptotically and empirically. For example, for sets of size 2^{20}, we require about $21\times$ less bandwidth and are about $6\times$ faster over a LAN network. In addition to our protocol, we model the real-world behavior of rational parties through a game-theoretic analysis. We introduce payout mechanisms for detected cheating and establish lower bounds on their values, ensuring that the best strategy for rational parties is to provide honest input.**

***Index Terms*—PSI, APSI, PAPSI, verkle tree.**

1. Introduction

Private set intersection (PSI) is a secure multiparty computation technique that enables two or more parties to securely compute the intersection of their input sets while keeping the remainder of their inputs private. PSI has several real-world applications, including genome comparison [1], [2], profile matching [3], bot net detection [4], and private mobile contact discovery [5], [6], [7].

Another important use case that has already seen early-stage deployment is *ad conversion* [8], [9], [10]. A store and an ad provider can use PSI to privately compute the intersection of users who both viewed an ad and made a purchase. This enables the store to evaluate the ad's effectiveness without revealing sensitive customer data.

Numerous efficient PSI protocols have been proposed under various threat models. Some protocols are proven secure against semi-honest adversaries, where all parties are assumed to follow the protocol honestly but may attempt to learn additional information from the protocol transcript [11], [12], [13]. In contrast, other protocols have been proven secure against malicious adversaries, which may arbitrarily deviate from the protocol to compromise security [14], [15]. However, protocols proven secure in either threat model suffer from a key limitation: they *do not prevent* a party from maliciously selecting their input.

Several recent attacks against PSI(-like) protocols [16], [17], [18] demonstrate that when an adversarial party is allowed to choose their input arbitrarily, they can break input privacy. Additionally, many Bloom filter-based PSI protocols allow an adversarial party to infer membership of elements of the others' set not in the intersection [19].

One mitigation against such attacks is to introduce a trusted third-party Judge to authorize the parties' inputs before computing the intersection (e.g., [20], [21]). This PSI variant, called *Authorized Private Set Intersection* (APSI), reveals the entire set to the Judge. Recently, Falzon and Markatou [22] introduced *Partial Authorized Private Set Intersection* (Partial-APSI or PAPSI), where each party commits to their input set and reveals only a random subset selected by the Judge for verification.

This selective disclosure helps preserve privacy while enabling input validation. As a result, Partial-APSI is especially well-suited for complying with regulatory frameworks that mandate data minimization. For example, according to Article 5(1)(c) of the European Union's General Data Protection Regulation (GDPR), organizations must ensure that only the minimum amount of personal data necessary for a specific purpose is collected, processed, and retained [23]. Data minimization is also a core tenet of the California Consumer Privacy Act (CCPA) [24]. Partial-APSI has several further applications, including:

Financial Auditing. Returning to the ad conversion example, we wish to ensure that the store's input set contains only legitimate customers. To achieve this, a financial auditor can act as a Judge and sample a subset of the customer list to verify its validity before the intersection. This approach, called *statistical audit sampling*, is a common auditing technique. PAPSI enables this process by allowing the auditor access only to the selected records, ensuring that other customers' records remain hidden and eliminating unnecessary data sharing.

Contact matching. When a new user subscribes to a chat provider, they want to know which of their contacts use the application. The new user and the chat provider can use PSI to determine this intersection. In Authorized PSI, a telecom company can act as a Judge. To verify the users of the chat provider, they can check if the user signed the terms of service.

This work was supported by the European Commission under the Horizon Europe Program as part of the project RECITALS (Grant Agreement no. 101168490) and by Armasuisse Science and Technology.

TABLE 1. COMPARISON WITH CLOSELY RELATED WORK. HERE, n IS THE SIZE OF THE RECEIVER'S SET, m IS THE SIZE OF THE SENDER'S SET, p IS THE FRACTION OF ELEMENTS REVEALED TO THE JUDGE, AND $k < n$ IS THE BRANCHING FACTOR OF THE VERKLE TREE.

Protocol		Security	Assumptions	Authorize		Intersect	
Citation	(P)APSI			Rounds	Communication	Rounds	Communication
DT10 [25]	APSI	Input mal.	ROM, RSA	2	$2n+2$	2	$2+2n+m$
DKT10 [26]	APSI	Malicious	ROM, RSA, DDH	2	$2n+2$	2	$1+5n+m$
DD15 [27]	APSI	Malicious	QR	2	$2n+2$	2	$2n+m$
FM25 [22]	APSI	Mal. Receiver	DBDH	2	$2n+2$	1	$1+m$
FM25 [22]	PAPSI	Input mal.	ROM, DBDH, DDH	4	$2(p+1)n+2$	3	$1+3m$
Ours	PAPSI	Input mal.	ROM + DDH				
-Pointproof-Verkle tree			+ AGM, t-wBDH	4	$3+pn(2+\log_k(n))$	2	$1+2n+m$
-Merkle tree			-	4	$3+pn(5+\log_2(n))$	2	$1+2n+m$

Genomic Matching. Genome matching has become increasingly important for applications ranging from ancestry testing to blood transfusions. PSI offers a promising solution for genome matching. Traditionally, serological antibody-based methods have been used to match blood donors with recipients. However, complications can arise for patients with antigens for which no serologic reagents are available. Recently, DNA-based genotyping has shown significant promise as a life-saving technique in reducing complications [28]. Partial-APSI can be valuable to match a blood donor and recipient from different hospitals. Hospitals trust a blood bank as a Judge, but they also want to protect their patients' privacy. Partial-APSI allows the hospital to reveal a small percentage of patients.

The protocol proposed by Falzon and Markatou has a limitation: it consumes a high amount of bandwidth. While the improvement of only partially revealing elements is promising, the high bandwidth costs are prohibitive, thus raising the following question:

Can we design a bandwidth-efficient Partial-APSI protocol with sub-linear communication in the authorization phase?

1.1. Contributions

We propose a Partial-APSI protocol that uses minimal bandwidth and significantly outperforms Falzon and Markatou [22]. For sets of size 2^{20}, we require approximately $21\times$ less bandwidth than [22]; furthermore, we reduce the runtime over a LAN network $6\times$.

We use the same security model as [22]; we assume a semi-honest Sender and Judge (i.e., they follow the protocol honestly but aim to learn additional information about the Receiver's set) and an input-malicious or *augmented semi-honest* Receiver (i.e., the Receiver can maliciously select their input before both the authorization and intersection phases, but otherwise follows the protocol).

The goal of Partial-APSI is to authorize the input set by auditing a p-fraction of it. Prior work did not provide intuition for choosing p. We therefore include a game-theoretic analysis to fill this gap. We apply the game-theoretic Adversarial Level Agreements (ALAs) framework of George and Kamara [29] to analyze how a rational Receiver, acting in their own best interest, would behave when participating in our Partial-APSI protocol. We incorporate payouts for detected malicious inputs and compare the payout heights to the fraction of revealed elements p. As a result, we obtain an equilibrium in which a rational Receiver adheres to the protocol.

Our contributions can be summarized as follows:

- We propose a **bandwidth-efficient Partial-APSI** protocol. (Section 4)
- We support our protocol with **formal proofs of correctness and security**. (Sections 4.2 and 4.3)
- We provide an **open-source**[1] Rust implementation of our protocol and **experimentally evaluate** it; we outperform prior work. (Section 6)
- We provide a **game-theoretic analysis** of Partial-APSI and prove **lower bounds on the payouts** for p to incentivize a rational Receiver to adhere to the protocol. This analysis **generalizes** and applies to the protocol of [22]. (Section 5)
- We additionally prove that the Verkle tree commitment scheme is **hiding** and **binding**, which could be of independent interest (Section 3.3).

1.2. Related work

PSI has been extended in numerous ways to incorporate authorization mechanisms, leading to what is now commonly known as Authorized PSI (APSI). A foundational example of such an extension is the *PSI for certified* sets proposed by Camenisch and Zaverucha [20], in which a trusted entity validates participants' input. Concurrently, De Cristofaro et al. [21] explored a more targeted use case, *privacy-preserving policy-based information transfer* (PPIT), which enables Receivers to retrieve specific items held by the Sender based on third-party authorizations.

Authorized PSI (APSI) was formally introduced in [25] along with a linear-cost protocol that is secure against semi-honest adversaries. This work was later extended by De Cristofaro et al. [26], who proposed a maliciously secure scheme that also retained linear communication complexity. Kerschbaum [30] proposed an APSI protocol based on Bloom filters and homomorphic encryption. This was further optimized by Debnath et al. [27], who proposed protocols for both standard APSI and *APSI cardinality*, i.e., only the intersection size is revealed. Faber et al. [31] introduced a variant of APSI in which both parties must authorize their inputs and learn the intersection.

Policy-enhanced PSI (PPSI) was introduced by Stefanov et al. [32] to enable more complex authorization

1. https://github.com/TjitskeKoster/Bandwidth_Efficient_Partially_Authorized_PSI.git

policies. Their work is secure against malicious adversaries, provided that the underlying PSI scheme, used in a black-box manner, is maliciously secure. Nagy et al. [33] proposed *Common Friends*, a social network application where users can determine if they are friends or share mutual friends. Their construction enforces access control in a privacy-preserving way and uses Bloom filters for efficiency. D'Arco et al. [34] proved that unconditionally secure size-hiding PSI is only possible in the APSI setting.

Sun et al. [35] gave a maliciously-secure PSI protocol that allows the Sender to publish a one-time, linear-size encoding of its set and use it to compute intersections with multiple Receivers. Ghosh et al. [36] proposed *private certifier intersection*, which allows multiple parties with certificates to discover shared certifiers and verify those, without revealing any information about certifiers that are not part of the intersection.

Recently, Goel et al. [37] introduced *Private Intersection over Committed (and Reusable) Sets*, which uses commitments to ensure that parties use the same input set across multiple runs. While this ensures consistency of the input across multiple runs, it does not guarantee that the parties use an honest input.

Yang et al. [38] propose an authorization phase, where no plaintext elements of the client are revealed. However, they require an additional party, an authority, as well as the Judge. They do not provide an intersection phase.

Table 1 compares our protocol with closely related (P)APSI protocols that also comprise one Judge and two parties. Compared to the authorization phase of [22] we use less bandwidth if $p < \frac{n\log_k(n)-1}{2n-1}$, since typically $2 \leq \log_k(n)$ (see also Fig. 6).

Falzon and Markatou [22] introduced the notion of *Partial-APSI*. In their PAPSI protocol, the Receiver first commits to their elements so they can later open a few of them at the Judges' choice. The commitment is a blinding of each element x_i with a Hash function H and an exponent r, i.e., $H(x_i)^r$. The Receiver sends a commitment of each of their elements to the Judge. The Judge then samples some indices $\mathcal{I} \subseteq [n]$ and requests the Receiver to open these commitments. The Receiver provides the required plaintext elements and proves that all of them are raised to the same exponent. The Judge then signs the entire set with the secret key sk, i.e., $H(x_i)^{r\mathsf{sk}}$. In the Intersection Phase, Falzon and Markatou need OPRFs; this is a protocol in which the Receiver obliviously raises the Sender's elements to the power r. The Receiver does not learn the elements of the Sender, and the Sender does not learn r. The Receiver now possesses a set of the form $H(y_j)^r$, but needs an extra mechanism to ensure that only signed elements can be intersected. [22] achieves this with bilinear pairings. A bilinear pairing e has a strong property, namely $e(H(y_j)^r, g^{\mathsf{sk}}) = e(H(x_i)^{r\mathsf{sk}}, g)$ if an only if $y_j = x_i$. The Sender can thus send $e(H(y_j)^r, \mathsf{pk})$ to the Receiver, and the Receiver can compute the intersection with only authorized elements. Note that in this brief overview, we left out some of the blinding factors for readability. For the full secure protocol, we refer to [22]. The main limitation of this work is the high bandwidth usage during the authorization phase: each element has its own commitment and signature, leading to a communication cost of $O(2n)$.

2. Preliminaries

2.1. Problem Definition & Threat Model

Partial-APSI consists of two phases. (i) **Authorization**: a Judge $\mathcal{J}$ requests a random fraction $p \in (0, 1]$ of the set of the reCeiver $\mathcal{C}$[2] and **authorizes** it. (ii) **Intersection:** the Sender $\mathcal{S}$ verifies that $\mathcal{C}$'s elements are authorized and then $\mathcal{C}$ and $\mathcal{S}$ compute the **intersection**. We consider one-sided PSI, where only $\mathcal{C}$ learns the intersection.

- The receiver $\mathcal{C}$ is an input-malicious (or augmented semi-honest) party. $\mathcal{C}$ does not deviate from the protocol, apart from potentially modifying their input at the beginning of the authorize and intersect phases.
- The Judge $\mathcal{J}$ is modeled as a semi-honest party. They do not deviate from the protocol, but may try to extract information about the non-revealed elements of $\mathcal{C}$.
- The sender $\mathcal{S}$ is modeled as a semi-honest party. They do not deviate from the protocol, but may also try to passively extract information about the elements of $\mathcal{C}$ from the protocol transcripts.

We prove security using the real-ideal paradigm. In the real world, the parties execute our protocol. In the ideal world, a trusted party executes the ideal functionality (Fig. 1) on inputs chosen by the receiver and the sender.

Definition 2.1. *Let* $\mathsf{View}^{\Pi}_{\mathcal{C}}(\mathcal{X}, \mathcal{Y})$, $\mathsf{View}^{\Pi}_{\mathcal{J}}(\mathcal{X}, \mathcal{Y})$, *and* $\mathsf{View}^{\Pi}_{\mathcal{S}}(\mathcal{X}, \mathcal{Y})$ *be the views of* $\mathcal{C}$, $\mathcal{J}$ *and* $\mathcal{S}$ *in the protocol* Π, *respectively. Let* $\mathsf{Out}^{\Pi}(\mathcal{X}, \mathcal{Y})$ *be the output of* Π *(i.e., the intersection) and* $O = f(\mathcal{X}, \mathcal{Y}, \mathcal{R})$ *be the output of the ideal functionality of Fig. 1. The protocol* Π *is* ***semi-honest secure for the*** $\mathcal{S}$ ***and*** $\mathcal{J}$ *and* ***augmented semi-honest secure for the*** $\mathcal{C}$ *if there exists PPT simulators* $\mathsf{Sim}_{\mathcal{C}}$, $\mathsf{Sim}_{\mathcal{S}}$ *and* $\mathsf{Sim}_{\mathcal{J}}$ *with input* 1^{λ}, *such that for all inputs* $\mathcal{X}, \mathcal{Y}$, *for all public parameters* $\mathcal{R}$ *and* p, *and for all index sets* $\mathcal{I} \subseteq [n]$ *such that* $|\mathcal{I}| = pn$:

$$\Big(\mathsf{View}^{\Pi}_{\mathcal{J}}(\mathcal{X}, \mathcal{Y}), \mathsf{Out}^{\Pi}(\mathcal{X}, \mathcal{Y})\Big) \stackrel{c}{\approx} (\mathsf{Sim}_{\mathcal{J}}(\{x_i\}_{i\in\mathcal{I}}, \mathcal{I}, |\mathcal{X}|), O)$$

$$\Big(\mathsf{View}^{\Pi}_{\mathcal{S}}(\mathcal{X}, \mathcal{Y}), \mathsf{Out}^{\Pi}(\mathcal{X}, \mathcal{Y})\Big) \stackrel{c}{\approx} (\mathsf{Sim}_{\mathcal{S}}(\mathcal{Y}, |\mathcal{X}|), O)$$

$$\mathsf{View}^{\Pi}_{\mathcal{C}}(\mathcal{X}, \mathcal{Y}) \stackrel{c}{\approx} \mathsf{Sim}_{\mathcal{C}}(\mathcal{X}, |\mathcal{Y}|, O)$$

2.2. Computational assumptions

Our protocol's security depends on the Decisional Diffie-Hellman assumption in the random oracle model (ROM). We use vector commitments as a black box. Additional assumptions can be found in Appendix B.

Definition 2.2. Decisional Diffie-Hellman (DDH) assumption [39]*: Given a generator* g *of group* $\mathbb{G}$ *of prime order,* $a, b, c \in \mathbb{Z}$ *and* $c \neq ab$. *It holds that for all probabilistic polynomial time algorithms* $\mathcal{A}$ *and for security parameter* $\mathsf{negl}(\lambda)$

$$\Big|\Pr[\mathcal{A}(g, g^a, g^b, g^{ab}) = \text{true}] - \Pr[\mathcal{A}(g, g^a, g^b, g^c) = \text{true}]\Big| \leq \mathsf{negl}(\lambda).$$

2. We use $\mathcal{C}$ to align with [22] and R is already extensively used.

Ideal Functionality for Partial-APSI

Parties and inputs:
Receiver $\mathcal{C}$ inputs sets $\mathcal{X}$ for Authorize and $\mathcal{X}'$ for Intersect, sender $\mathcal{S}$ inputs set $\mathcal{Y}$; judge $\mathcal{J}$ inputs $\perp$.

Parameters:
Set sizes $|\mathcal{X}|$, $|\mathcal{Y}|$, probability $p \in (0, 1]$, regulation knowledge $\mathcal{R}$

Functionality:
$\mathbf{T}$ receives an Authorize request of $\mathcal{X}$ from $\mathcal{C}$:

- $\mathbf{T}$ samples $\mathcal{I} \subseteq_{\$} [n]$ such that $|\mathcal{I}| = \lceil pn \rceil$ where $n = |X|$, and then either accepts or rejects the elements in $\{x_i \mid x_i \in \mathcal{X}\}_{i \in \mathcal{I}}$ based on $\mathcal{R}$.
- If the elements are consistent with $\mathcal{R}$, $\mathbf{T}$ replies `Accept` to $\mathcal{C}$ and remembers that $\mathcal{X}$ is authorized. Otherwise, $\mathbf{T}$ replies `Reject` to $\mathcal{C}$.

$\mathbf{T}$ receives a request from party $\mathcal{C}$ with input $\mathcal{X}'$ to perform set Intersection with the sender $\mathcal{S}$.

- If $\mathcal{X}' = \mathcal{X}$, $\mathbf{T}$ forwards the request to $\mathcal{S}$.
- $\mathcal{S}$ sends a set $\mathcal{Y}$ to $\mathbf{T}$.
- $\mathbf{T}$ computes the intersection $\mathcal{X} \cap \mathcal{Y}$ and sends this and the set size $|\mathcal{Y}|$ to $\mathcal{C}$.

Figure 1. Ideal Functionality for Partial-APSI, $\mathcal{F}_{\mathsf{Partial\text{-}APSI}}$. Where $\mathbf{T}$ is a trusted third-party.

2.3. Signature Scheme

Authorized elements will be marked with a signature.

Definition 2.3. *A* digital signature scheme *consists of three algorithms* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$*:*

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ *takes as input a security parameter* λ *and outputs a public-private key pair* $(\mathsf{pk}, \mathsf{sk})$.
- $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$ *takes as input a secret key* sk *and message* m*, and outputs a signature* σ.
- $b \leftarrow \mathsf{Verify}(\mathsf{pk}, \sigma, m)$ *takes as input a signature* σ*, a message* m*, and public key* pk *and outputs a bit* $b \in \{1, 0\}$.

A digital signature scheme is correct if it holds for any message m that $\mathsf{Verify}(\mathsf{pk}, \mathsf{Sign}(\mathsf{sk}, m), m) = 1$. We require the digital signature scheme to be existential unforgeable under chosen-message attacks (EUF-CMA), meaning that an adversary has at most negligible probability of forging a valid signature on a message they have not seen before.

3. Vector commitment schemes

Commitment schemes are designed so that a party (the prover) can commit to a secret value. Later, the prover can reveal the secret value and prove that the commitment belongs to this value. It is also required that the commitment is binding; the prover cannot provide a different value with the same commitment. Additionally, a commitment can be hiding, i.e., the verifier should not be able to distinguish commitments of two different values.

As an extension of commitment schemes, Catalano and Fiore [40] introduced vector commitment schemes (VCS) that allow for committing to a vector of values.

Definition 3.1. *A vector commitment scheme is a tuple of algorithms* $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{V})_{vcs}$

- $\mathsf{pp}_{vcs} \leftarrow \mathsf{Setup}_{vcs}(k)$ *This algorithm outputs the public parameters supporting* k *messages.*
- $\mathsf{C}, \mathsf{aux} \leftarrow \mathsf{Commit}_{vcs}(m_1, .., m_k, \mathsf{pp}_{vcs})$ *On a sequence of* k *messages* $m_1 ... m_k$ *and the public parameters* pp_{vcs}*, this algorithm outputs a commitment string* C *and auxiliary information*
- $\Lambda_i \leftarrow \mathsf{Open}_{vcs}(m, i, \mathsf{aux})$ *On input of message* m*, index* i *and auxiliary information, the open algorithm returns a proof* Λ_i.
- $\{0, 1\} \leftarrow \mathsf{V}_{vcs}(\mathsf{C}, m, i, \Lambda_i)$ *This algorithm outputs 1 if and only if* Λ_i *is a valid proof that* C *was created with message* m *on the* i*'th index.*

A vector commitment scheme is correct *if*

$$\Pr\left[\begin{array}{c} \mathsf{V}_{vcs}(\mathsf{Commit}_{vcs}(m_1, .., m_k, \mathsf{pp}_{vcs}), \\ m_i, i, \mathsf{Open}_{vcs}(m_i, i, \mathsf{aux}) = 1) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

We formally define *position-binding* and *hiding* below.

Definition 3.2. *A vector commitment scheme VCS* $= (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{V})_{vcs}$ *is* position binding *if for every PPT adversary* $\mathcal{A}$*,* $(\mathsf{C}, i, m, m', \Lambda, \Lambda') \leftarrow \mathcal{A}$ *and* $m \neq m'$ *it holds*

$$\Pr\left[\begin{array}{c} \mathsf{V}_{vcs}(\mathsf{C}, m, i, \Lambda) = 1 \\ \wedge \\ \mathsf{V}_{vcs}(\mathsf{C}, m', i, \Lambda') = 1 \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Definition 3.3. *A vector scheme VCS* $= (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{V})_{vcs}$ *is* hiding *if for every PPT adversary* $\mathcal{A}$ *and any two sequences of messages* $V_1 = [m_1, ..., m_k]$ *and* $V_2 = [m'_1, ..., m'_k]$ *it holds that*

$$\Pr\left[\, b = b' \;\middle|\; \begin{array}{l} \mathsf{pp}_{vcs} \leftarrow \mathsf{Setup}_{vcs}(k) \\ (V_1, V_2) \leftarrow \mathcal{A} \\ b \leftarrow\!\!\$\, \{0, 1\} \\ \mathsf{C} \leftarrow \mathsf{Commit}_{vcs}(V_i, \mathsf{pp}_{vcs}) \\ b' \leftarrow \mathcal{A}(\mathsf{C}) \end{array} \right] \leq \mathsf{negl}(\lambda),$$

even after opening one or more positions.

We additionally require the vector commitment scheme to be *concise*.

Definition 3.4. *The vector scheme VCS is* concise *if the output sizes of* Commit_{vcs} *and* Open_{vcs} *are constant and independent of the length of the vector.*

In the following subsections, we give examples of (concise) vector commitment schemes.

3.1. Merkle tree

In 1989, Merkle [41] proposed a digital signature using a complete binary tree. The tree is position binding and can thus be used as a vector commitment scheme. The prover constructs the tree from bottom to top. Each leaf node i contains the value x_i. Then the prover hashes each pair of children to form the parent nodes. This is iterated until the root. The vector's commitment is the value of the root. If the verifier queries an index i, the prover sends the

value x_i and the sibling node, along with all hash values of the sibling nodes on the path to the root. To verify, the verifier starts by hashing the two leaf values. Then, the verifier iteratively hashes the new value with the sibling node's value. The iteration ends at the root. The verifier finally compares the last hash to the commitment; if they are equal, the verifier approves the proof. The Merkle tree is binding if the hash functions are modeled as random oracles [41]. One downside is that the tree is not hiding, as the sibling node of the leaf node is revealed. One can solve this by blinding the values before inserting them as leaves. The second disadvantage is that the proof of one element is logarithmic in the vector's size, and hence this commitment scheme is not concise.

3.2. Pointproof vector commitment

Gorbunov et al. [42] proposed a concise pairing-based scheme based on the vector commitment of Libert and Yung [43]. Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ be a bilinear group and let g_1 be a generator of $\mathbb{G}_1$. To commit to a vector $V = [m_1, \ldots, m_k]$, we construct a polynomial where the coefficient of x^i is m_i for each $i \in [k]$. The commitment to V is given by g_1 raised to the value of this polynomial evaluated at a secret point. The scheme is binding in the algebraic group model, random oracle model, and under the t-weak bilinear Diffie-Hellman assumption (for definitions, see Appendix B). Proofs of binding and hiding can be found in [42].

3.3. Verkle tree commitment scheme

A Verkle tree [44] is similar to a Merkle tree; the difference is that the parent node is a vector commitment to its children. Furthermore, the Verkle tree does not have to be a binary tree, and any branching number k can be chosen. This choice determines the proof size and computation time. To the best of our knowledge, no formal proof exists of the security of a Verkle tree, and we thus provide the proof in this section.

To construct a Verkle tree, one first chooses a concise vector commitment scheme and a branching factor k. Then one takes the messages m and hashes them to the field over which the VCS is defined. The parent node of k successive (leaf) nodes is defined as the single token commitment of the VCS commitment of these nodes. An example is pictured in Fig. 2. In this example, $n = 9$, the width k is 3, and the messages m_i are hashed to the leaf nodes $\hat{x}_i$. In the next layer, C_1 is the vector commitment of $[\hat{x}_1, \hat{x}_2, \hat{x}_3]$ similary for C_2 and C_3, finally the root is a commitment of $\mathsf{C}_1, \mathsf{C}_2, \mathsf{C}_3$. To prove the presence of m_7 in the tree, the prover provides a proof

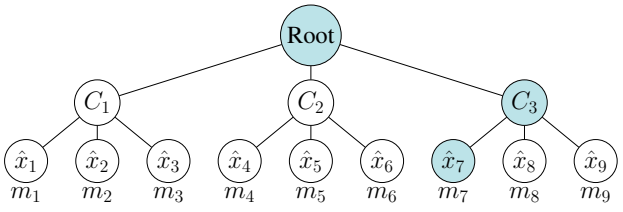


Figure 2. Verkle tree example with branching factor 3.

$\lambda_1 = \mathsf{Open}_{vcs}(\hat{x}_7, 1)$, and $\lambda_2 = \mathsf{Open}_{vcs}(\mathsf{C}_3, 3)$. The verifier verifies $\mathsf{V}_{vcs}(\mathsf{C}_3, \hat{x}_7, 1, \lambda_1)$ and $\mathsf{V}_{vcs}(\mathsf{R}, \mathsf{C}_3, 3, \lambda_2)$. The proof path is blue in Fig. 2.

A Verkle tree commitment scheme is a tuple of algorithms $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{V})$ defined as:

- $(\mathsf{pp}_{vcs}, k) \leftarrow \mathsf{Setup}_{tree}(k)$ On input k, the width of the tree, this algorithm outputs the public parameters pp_{vcs} of the underlying vector commitment scheme and the width of the tree.
- $\mathsf{R} \leftarrow \mathsf{Commit}_{tree}(m_1, .., m_n, \mathsf{pp})$ On a vector of n messages this algorithm calculates Commit_{vcs} of the underlying vector commitment scheme for each k sequential leaves. Then the algorithm calculates Commit_{vcs} for k sequential parent nodes. This tree structure is repeated until the root. The output of this algorithm is the root R.
- $\Lambda_i \leftarrow \mathsf{Open}_{tree}(m, i, \mathsf{aux}_{vcs})$ The algorithm initializes an empty list Λ_i. On input of a message and an index, this algorithm performs iteratively $\Lambda_j \leftarrow \mathsf{Open}_{vcs}$ for each parent-child pair along the treepath of the requested leaf i. For each opening on commitment node C with child node c on index j of the vector commitment, the algorithm adds the tuple $(\mathsf{C}, c, j, \Lambda_j)$ to Λ_i. The output is Λ_i.
- $\{0, 1\} \leftarrow \mathsf{V}_{tree}(\mathsf{R}, m, i, \Lambda_i)$. On input of a message, an index, a proof, and the root, the algorithm performs V_{vcs} for each parent-child pair along the tree path. The algorithm outputs 1 if and only if all proofs are correct and the indices follow the path in the tree.

Remark. Note that to ensure security, a fully populated Verkle tree is required. To solve this, either dummy messages need to be added. Alternatively, a forest of trees could be used, where the n messages are split across several filled trees.

Unfortunately, this vector commitment scheme is not concise as the proof is logarithmic in n, ie $O(\log_k(n))$.

Lemma 3.5. *The Verkle tree is* position binding *if and only if the underlying vector commitment scheme is.*

Proof. Assume towards a contradiction that the Verkle tree is not position binding. There exists an adversary $\mathcal{A}$ who, in polynomial time, produces two valid openings Λ, Λ' to two different messages at the same position:

$$(\mathsf{R}, i, m, m', \Lambda, \Lambda') \leftarrow \mathcal{A}.$$

We can then construct an adversary $\mathcal{B}$ that breaks the VCS's position binding. $\mathcal{B}$ selects the width of the verkle tree, k, and samples public parameters for the vector commitment scheme accordingly. $\mathcal{B}$ forwards the public parameters to adversary $\mathcal{A}$. $\mathcal{A}$ can break the position binding property of the verkle tree, and returns $(\mathsf{R}, i, m, m', \Lambda, \Lambda')$ for some $i \leq n$. Λ and Λ' are lists of proofs of commitments along the tree paths, and $\mathcal{B}$ checks the proofs one by one. The first proof λ_i in Λ is a tuple of the form $(\mathsf{R}, c, i_1, \Lambda_c)$ the first tuple of Λ' is $(\mathsf{R}, c', i_1, \Lambda'_{c'})$. Note that the values of R and i are the same, because the tree path is unique. If the values c and c' or the values Λ_c and $\Lambda'_{c'}$ differ, then $\mathcal{B}$ outputs the tuple $(\mathsf{R}, i_1, c, c', \Lambda_c, \Lambda'_{c'})$. If, on the contrary, both tuples are equal, then $\mathcal{B}$ continues its search from the root to the leaf. At least one of the

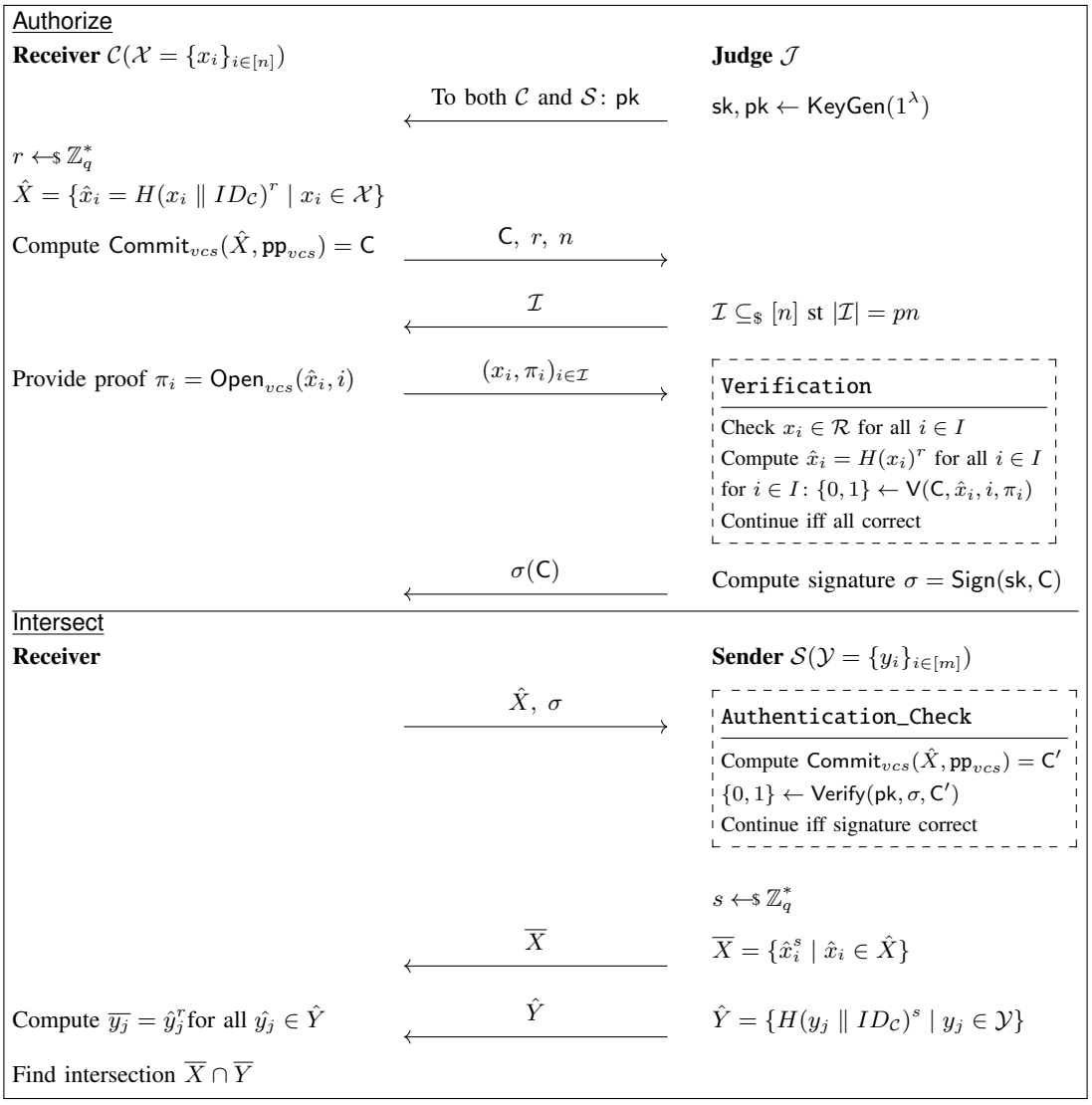


Figure 3. Partially Authorized PSI, with as public parameters: prime number q, hash function $H\colon \{0,1\}^* \to \mathbb{Z}_q^*$, public regulation knowledge $\mathcal{R}$. p is the percentage of elements the Judge will verify. Moreover, the used vector commitment scheme VCS together with pp_{vcs} should be public.

proofs should be different, and thus $\mathcal{B}$ breaks the position binding of the VCS. The adversary $\mathcal{B}$ works in polynomial time as the tree path is logarithmic to the input, and $\mathcal{A}$ works in polynomial time. □

Lemma 3.6. *The Verkle tree is* hiding *if and only if the underlying vector commitment scheme is.*

This lemma can be proven similarly to Lemma 3.5.

4. Partial-APSI Protocol

We now present our partially authorized private set intersection scheme. Partial authorization requires the Receiver to commit to the set $\mathcal{X}$ and send the commitment to the Judge. The Judge then challenges the Receiver to reveal a $p \in (0,1]$-fraction subset of its input. The previous Partial-APSI scheme [22] incurs significant bandwidth overhead in the authorization phase, since commitments and signatures are linear in $|\mathcal{X}|$. To optimize, we use a concise commitment scheme that yields a single token for both the commitment and the signature.

4.1. Protocol overview

Our protocol's public parameters are: a cryptographically secure hash function H needed to blind the elements of the Receiver and Sender, and the public parameters of the vector commitment scheme pp_{vcs}. Before the protocol, the Judge generates a secret key sk and public key pk for the signature. The Receiver has input $\mathcal{X}$ of size n and the Sender has input $\mathcal{Y}$ of size m, their elements are from $\{0,1\}^*$. Whether an element of the Receiver is authorized depends on public regulation knowledge $\mathcal{R}$.

Here, we provide an overview of the protocol; a detailed version is shown in Fig. 3. The authorization phase:

1) The Receiver blinds their elements by first hashing them and then raising them to a random r.
2) Then, the Receiver commits the blinded values in the VCS, and produces a vector commitment C.
3) After the Judge receives the commitment, C, they sample a challenge of indices $\mathcal{I}$ and forward it to the Receiver.
4) The Receiver sends the requested items in plaintext and proves that they committed to these.

5) If the Judge approves the elements and verifies the commitment proofs, the Judge signs the commitment, C, and returns the signature to the Receiver.

After Authorize, the Receiver starts communicating with the Sender to compute the intersection.

1) The Receiver sends the Sender all blinded elements and the signed commitment.
2) The Sender computes $\mathsf{C} = \mathsf{Commit}_{\mathsf{vcs}}$ with the blinded elements. Then they check the signature of C. This way, the Sender ensures that the Judge authorized this set.
3) If the Sender approves the signature, the parties continue with a Diffie-Hellman-based PSI protocol. To this end, the Sender samples a random s and exponentiates both sets to the power s. The Receiver raises the Sender's set to r and can compute the intersection.

To include updates of $\mathcal{X}$, the Receiver must update the corresponding leaves of the tree, changing their authentication paths and the final commitment. Updates of the Senders' set $\mathcal{Y}$ can be handled by recomputing $\overline{X}$ and $\hat{Y}$ with fresh randomness s.

The following lemma shows that the bandwidth of the authorization phase primarily depends on the proof sizes.

Lemma 4.1. *The Receiver-Judge communication is:*

$$O(pn|\mathsf{Open}_{vcs}|). \tag{4.1}$$

Proof. (1) The protocol starts with the Receiver who sends integers r, n and commitment $\mathsf{C} = \mathsf{Commit}_{\mathsf{vcs}}$. For a concise VCS, this is of constant size. (2) The Judge samples pn indices and sends those to the Receiver; this is of order pn. (3) The Receiver responds with the pn requested elements in plaintext and their proof of commitment, resulting in an overall complexity of:

$$O(|\mathsf{Commit}_{vcs}| + |r| + |n| + pn(2 + |\mathsf{Open}_{vcs}|))$$

Omitting the constants gives the lemma. □

For $p < \frac{n \log_k(n)-1}{2n-1}$, we achieve better authorization bandwidth than Falzon and Markatou [22].

Lemma 4.2. *The Receiver-Sender communication is of order*

$$O(2n + m) \tag{4.2}$$

Proof. (1) The Receiver sends the signature of the commitment, which is of a constant size. Further, the Receiver sends the blinded set $\hat{X}$ of order n. (2) The Sender responds with the set $\hat{X}$ blinded again, and it own blined set $\hat{Y}$, of order $n + m$, this concludes the proof. □

4.2. Correctness of the protocol

Theorem 4.3. *The partial-APSI protocol in Fig. 3 outputs $\mathcal{X} \cap \mathcal{Y}$, where $\mathcal{X}$ is the Receiver $\mathcal{C}$'s input and $\mathcal{Y}$ is the Sender $\mathcal{S}$'s input if H is a collision-resistant hash function.*

Proof. The intersection is correct because for any $x_i = y_j$, where $x_i \in \mathcal{X}$ and $y_j \in \mathcal{Y}$, we see the following:

$$\overline{x}_j = (H(x_i \parallel ID_{\mathcal{C}}))^{rs} = (H(y_j \parallel ID_{\mathcal{C}}))^{sr} = \overline{y}_j,$$

this indicates that any element x_i in the intersection $\mathcal{X} \cap \mathcal{Y}$, is indeed in the final intersection. For any element x_i not in the intersection, it happens only with negligible probability that there exists a y_j such that $(H(x_i \parallel ID_{\mathcal{C}}))^{rs} = (H(y_j \parallel ID_{\mathcal{C}}))^{sr}$, due to the collision-resistant hash function. □

4.3. Security of the protocol

We assume the Judge to be semi-honest. The Judge is allowed to see a fraction of the Receiver's set, but the Judge may learn nothing beyond these values.

Theorem 4.4. *The Partial-APSI protocol in Fig. 3 is secure against a semi-honest Judge if the vector commitment scheme is hiding and H is a collision-resistant hash function.*

Proof. We prove this using a hybrid argument. Upon input of $1^\lambda, \{x_i\}_{i\in\mathcal{I}}, \mathcal{I}, |\mathcal{X}|, \mathcal{R}, p$, the simulator proceeds as follows:

- Sample element $r \leftarrow\!\$\, \mathbb{Z}_q^*$.
- Define a set $\hat{X}$ of size $|\mathcal{X}|$ such that the ith element is defined as $\hat{x}_i = H(x_i \| ID_{\mathcal{C}})^r$ if $i \in \mathcal{I}$ and $\hat{x}_i \leftarrow\!\$\, \mathbb{Z}_q^*$ otherwise.
- Compute $\mathsf{C} = \mathsf{Commit}_{\mathsf{vcs}}$. Forward C, r, and n to the Judge.
- Upon receiving $\mathcal{I}$, compute the commitment proof π_i of $\hat{x}_i$ and forward $(x_i, \pi_i)_{i\in\mathcal{I}}$ to the Judge.

We now consider the following sequence of hybrids.
Hybrid 0**:** The real honest interaction.
Hybrid 1**:** The same as the previous hybrid, except the output is replaced with the functionality output $f(\mathcal{X}, \mathcal{Y}, \mathcal{R})$. By correctness of the protocol (Theorem 4.3), this is computationally indistinguishable from hybrid 0.
Hybrid 2**:** The same as the previous hybrid, except sample $r \leftarrow\!\$\, \mathbb{Z}_q^*$; since r is distributed the same as in the previous hybrid, then the hybrids are indistinguishable.
Hybrid $(3, k)$ **for all** $k \le n - |\mathcal{I}|$**:** Let $j \in [n] \setminus \mathcal{I}$ be the smallest index in $\mathcal{X}$ such that $\hat{x}_j$ is computed as in the real world. This hybrid is the same as the previous hybrid, but $\hat{x}_j$ is replaced with a randomly sampled value $z_j \notin \mathcal{X} \cap \mathcal{Y}$. The hiding property of the commitment scheme (Definition 3.3) ensures that no polynomial time adversary can distinguish if the commitment contains $\hat{x}_j$ or z_j. The $(k-1)^{\text{th}}$ and k^{th} hybrids are thus computationally indistinguishable. Observe that the final hybrid is identical to the simulator, and the proof follows. □

Theorem 4.5. *The Partial-APSI protocol in Fig. 3 is secure against a semi-honest Sender if the hash function H is collision-resistant, the DDH assumption holds, and the vector commitment scheme is binding and hiding.*

Proof. We prove this using hybrid arguments. Our simulator has input $1^\lambda, \mathcal{Y}, |\mathcal{X}|$ and works as follows:

- The simulator samples without replacement $|X|$ elements of $\mathbb{Z}_q^*$ to simulate $\hat{X}$.
- Compute the commitment $\mathsf{C} = \mathsf{Commit}_{\mathsf{vcs}}(\hat{X})$.
- The simulator generates the key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and computes $\sigma(\mathsf{C}) \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{C})$.

Hybrid 0**:** The real interaction; the parties run the protocol honestly.

Hybrid 1**:** The same as the previous hybrid, except abort if there exists a pair $i, j \in [n]$, such that $\hat{x_i} = \hat{x_j}$. This means that there is a collision in the hash function, which happens with negligible probability because H is a collision-resistant hash function; thus, this hybrid is indistinguishable from hybrid 0.
Hybrid 2**:** The same as the previous hybrid, except that r is sampled from $\mathbb{Z}_q$. Since the sampling distribution is the same, the hybrids are indistinguishable from one another.
Hybrid $(3, k)$**:** Similar to the previous hybrid, but $\hat{x}_k$ is replaced by a value $x' \leftarrow\$\ \mathbb{Z}_q^*$, resampling as needed to avoid collisions within $\hat{X}$. Since $\mathbb{Z}_q^*$ is cyclic there exists a generator $g \in \mathbb{Z}_q^*$ and integer $a \in \mathbb{Z}_q$ such that for x_k we have $H(x_k \| ID_{\mathcal{C}}) = g^a$. Now, consider the tuple

$$(g, H(x_k \parallel ID_{\mathcal{C}}), g^r, H(x_k \parallel ID_{\mathcal{C}})^r) = (g, g^a, g^r, g^{ar})$$

and observe that, by the DDH assumption, this triple must be indistinguishable from a random group element g^b:

$$(g, H(x_k \parallel ID_{\mathcal{C}}), g^r, x') = (g, g^a, g^r, g^b).$$

Thus, by the DDH assumption, $H(x_k \parallel ID_{\mathcal{C}})^r$ is indistinguishable from a randomly sampled group element x'. Moreover, the VCS is hiding, so no adversary can detect the replacements. Furthermore, since $\mathbb{Z}_q^* \gg |\hat{X}|$, resampling can be done in polynomial time, and thus this hybrid runs in polynomial time and is indistinguishable from the previous hybrid.
Hybrid 4: The same as the previous hybrid, except the Judge's key pair is replaced with a key pair sampled from KeyGen. The key pairs are sampled from the same distribution, and thus the two hybrids are indistinguishable.
hybrid 4 is indistinguishable from our simulator. Since the simulator is indistinguishable from the real world, we conclude that the partial-APSI protocol is secure against a semi-honest Sender. □

Theorem 4.6. *The partial-APSI protocol in Fig. 3 is secure against an input-malicious Receiver $\mathcal{C}$ if the signature is EUF-CMA secure, the DDH assumption holds, the hash functions are modeled as a random oracle, and the vector commitment scheme is binding.*

Proof. We construct the simulator, which takes as input $1^\lambda, \mathcal{X}, |\mathcal{Y}|, \mathcal{R}, p$, and $O = f(\mathcal{X}, \mathcal{Y}, \mathcal{R})$, as follows:

- The simulator models the hash function H as a random oracle and stores the result in a table T. In particular, upon each query q to H, it checks if a hash value h exists such that $h = T[q]$. If so, return h, otherwise, it samples a random h, return h and record $T[q] = h$.
- To simulate the Judge:
  - The simulator generates a key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.
  - Upon an authorization request and receiving commitment C, the simulator samples a set $\mathcal{I}$ of size $|\mathcal{I}| = pn$ along a uniform distribution and forwards it to the Receiver.
  - Upon receiving the elements and proofs from the Receiver, the simulator checks the proofs and uses $\mathcal{R}$ to verify the elements.
  - If no malicious elements are detected and the proofs are correct, Sim computes the signature $\sigma(\mathsf{C}) \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{C})$. Otherwise, the simulator outputs `Reject`.
- To simulate the Sender $\mathcal{S}$, on input $\hat{X}$ and $\sigma(\mathsf{C})$ from the Receiver, Sim does the following:
  - Compute C' with $\mathsf{Commit}_{\mathsf{vcs}}$ and $\hat{X}$. Use knowledge of C from authentcation to check if $\mathsf{C}' \stackrel{?}{=} \mathsf{C}$. Abort if they are not equal. Then verify the signature $\sigma(\mathsf{C})$ and abort if and only if the verification fails.
  - The simulator samples a random $s \in \mathbb{Z}_q$
  - Initalizes empty set $\hat{Y}$. For each $y \in \mathcal{X} \cap \mathcal{Y}$, make hash query $H(y \parallel ID_{\mathcal{C}})$, raises it to the power s, and adds it to $\hat{Y}$. It then pads $\hat{Y}$ up to size $|\mathcal{Y}|$ by sampling random elements from $\mathbb{Z}_q^*$, shuffles $\hat{Y}$ and forwards $\hat{Y}$ and $\overline{X}$ to the Receiver.

We now consider the following series of hybrids:
Hybrid 0**:** The real honest interaction.
Hybrid 1**:** The same as the previous hybrid, except we replace the Judges' key pair with a key pair sampled by $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$. This changes the signatures, and thus also the `Authentication_Check`. However, note that the keys for the signature scheme are generated along the same distribution, and thus they are indistinguishable from the previous hybrid. Additionally, the signature scheme is EUF-CMA secure. So, if the Receiver modified the signature or commitment, this would be detected, and we would abort. Note that if the set $\hat{X}$ for Intersect is not the same as $\hat{X}$ for Authorize, then the commitments are different with high probability (Definition 3.1). We can conclude that the two hybrids are indistinguishable.
Hybrid 2**:** The same as the previous hybrid, except $\mathcal{I}$ is sampled along a uniform distribution. It is generated along the same distribution as in the real world, but the `Verification` changes. We check all revealed elements using $\mathcal{R}$ and verify the commitment proofs. The probability of detecting a malicious element is $A = \binom{n-\mu n}{pn} / \binom{n}{np}$ (5.1), where μ is the fraction of malicious elements in the Receiver's input. This is the same probability for the Receiver being caught as in hybrid 1. Note, moreover, that the VCS is binding (Definition 3.2), and thus the Receiver has to reveal the elements they committed to. The hybrids are thus indistinguishable.
Hybrid 3**:** During this hybrid game, we sample s from $\mathbb{Z}_q$. It is generated along the same distribution; thus, this hybrid is indistinguishable from the previous hybrid.
Hybrid $(4, k)$ **for all** $k \leq m$**:** Same as the previous hybrid, but for all $y_k \notin \mathcal{X} \cap \mathcal{Y}$ a random value is sampled. Under the assumption that the hash function is modeled as a random oracle and the DDH assumption holds, this random value appears indistinguishable from $H(y \parallel ID_{\mathcal{C}})^s$ and the hybrids are indistinguishable.
Hybrid $(5, k)$ **for** $k \leq n$**:** The same as the previous hybrid, except abort if there is a collision, in particular if $H(y_k \| ID_{\mathcal{C}}) = H(y_j \| ID_{\mathcal{C}})$ for some $j \leq k$ or $H(y_k \| ID_{\mathcal{C}}) = H(x_i \| ID_{\mathcal{C}})$ for some $i \leq n$. This happens with probability $\frac{mn}{q}$ where $\mathbb{Z}_q^*$ is the field and m, n are the set sizes of $\mathcal{S}$ and $\mathcal{C}$. This is negligible in the group size and is indistinguishable from the previous hybrid.
We observe that hybrid $(5, n)$ is the same as the simulator, which concludes the proof. □

5. Game-Theoretic Evaluation

As noted in [45], modeling parties as semi-honest or malicious does not fully capture how adversaries may act in the real world; often, people are driven by self-gain or may try to cheat if they believe they will likely not be caught. Consider *rational actors* who act in their best interest and seek to maximize utility. Halpern and Teague [46] were the first to study rational adversaries in the context of secure multi-party computation, prompting a line of research (e.g., [47], [48] and [49] for an overview). Aumann and Lindell [45] proposed the concept of a *covert adversary* as an extension of the rational adversary, to capture adversaries who are willing to cheat as long as they are unlikely to be caught. This model sits between the semi-honest and malicious settings, providing a more realistic representation of adversarial behavior.

In this section, we analyze our PAPSI protocol through a game-theoretic lens. We apply the framework of George and Kamara [29] to model the behavior of a covert adversary. Their framework introduces *contracts* into protocols, which specify penalties that parties must pay if caught cheating. Covert adversaries act in their best interest, so the risk of penalties discourages them from deviating. George and Kamara show that such contracts can lead to favorable outcomes by incentivizing honest behavior.

We investigate how to augment a Partial-APSI protocol with contracts to incentivize parties to adhere to the protocol. We apply the adversarial level agreements (ALAs) framework of [29] and prove lower bounds on the necessary payouts to guarantee that the best strategy of a rational party is to adhere to the protocol. We note that this approach is protocol-agnostic and thus applies to *any* Partial-APSI protocol with a similar structure, which we elaborate on below.

Our approach. When a Receiver $\mathcal{C}$ initiates an intersection with a Sender $\mathcal{S}$, the parties must agree on the following:

- The Sender $\mathcal{S}$ specifies a minimum threshold $p \in (0, 1]$ that denotes the fraction of elements the Receiver must reveal to the Judge. This threshold must be met during the Authorize phase of the protocol.
- The Receiver $\mathcal{C}$ can accept or reject the threshold p by taking into account the following:
  1) The loss incurred by revealing a p fraction of its elements to the Judge.
  2) The utility gained by participating in the protocol.

As we will see, the payout for being caught cheating is inversely related to p. By setting a higher payout, we can tolerate a smaller fraction p while still ensuring that following the protocol would be in the Receiver's best interest.

5.1. Preliminaries

Adversarial-level agreements. We augment our Partial-APSI protocol with contracts that specify payouts for when a party is caught cheating. These contracts are referred to as *adversarial-level agreements* (ALAs).

Definition 5.1 ([29])**.** *Let* Π *be a cryptographic protocol with security parameter* λ *between parties* $\mathbf{P}_1, \ldots, \mathbf{P}_n$*. An* adversarial level agreement $\mathsf{ALA}_{\Pi,\kappa}(\mathbf{d})$ *consists of: (1) a specification of* Π*, defining each party's prescribed behavior and (2) damages* d_i *due from each party* $\mathbf{P}_i$ *if found deviating from* Π*, where* $\mathbf{d} = (d_1, \ldots, d_m)$ *is a vector that describes the damages for each party.*

Including ALAs introduces strategic interactions between rational parties. These interactions are formalized as a cryptographic inspection game (CIG).

Definition 5.2 ([29])**.** *A* cryptographic inspection game *(CIG) is a two-party game between an inspectee who wishes to deviate from a protocol and an inspector who wishes to enforce honest behavior. CIGs may have false negatives (but no false positives), and the inspector may also act dishonestly. These games are represented as trees, where nodes correspond to the players' possible actions and leaves to the potential outcomes.*

Any ALA-enhanced cryptographic protocol can be represented as a CIG. We describe the CIG for our concrete Partial-APSI protocol and then analyze the game to find parameters that guarantee the parties will follow the protocol. Please refer to [29] for a more in-depth discussion of adversarial level agreements and inspection games.

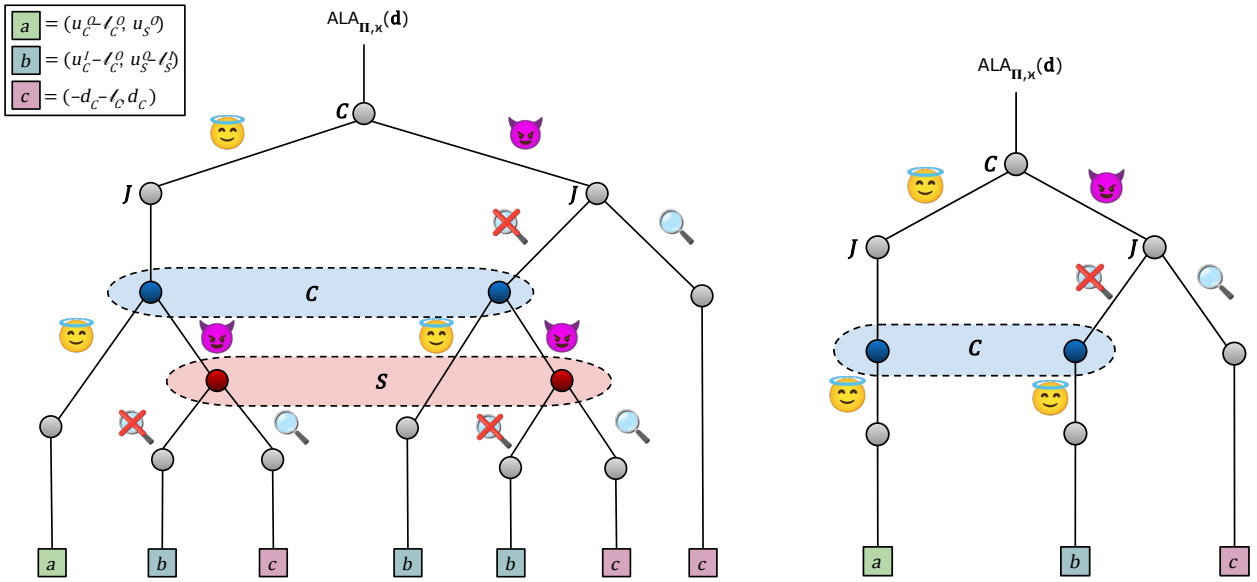
Notation. We use the notation convention of [29]. We denote the expected utility for a player $\mathbf{P}$ with strategy S as $\mathrm{U}_{\mathbf{P}}[S]$; this expectation is taken over the possible outcomes induced by the distribution of the other players' strategies and the game's randomness. A player's *best response* is the strategy that yields the highest expected utility given the other players' strategies. A *rational party* will act according to this best response.

Partial-APSI Structure. Our Partial-APSI protocol has the following structure, which also applies to the Partial-APSI protocol of [22] and can be generalized to future schemes. This structure will guide how we define the tree representing the Partial-APSI CIG.

- **Authorize:** The Receiver and the Judge first execute Authorize, during which the Judge $\mathcal{J}$ randomly selects a fraction p of the Receiver's element to be revealed. The Receiver $\mathcal{C}$ then sends the requested elements to the Judge and, if the elements are valid, the Judge provides a (set of) signature(s) to $\mathcal{C}$ authorizing all of their elements.
- **Intersect:** The Receiver and the Sender can then execute Intersect. The Receiver starts by sending $\hat{X}$ and the signature(s) (obtained during the authorization phase) to the Sender $\mathcal{S}$. The Sender then sends a response from which the Receiver can recover the set $(\mathcal{X}_{\text{AUTH}} \cap \mathcal{X}) \cap \mathcal{Y}$, where $\mathcal{X}_{\text{AUTH}}$ is the set of items authorized by the Judge.

We allow the Receiver to deviate from the protocol by changing their input, right before either Authorize or Intersect. Our CIG captures these two decisions.

Adversarial setting. We consider the setting of a rational Receiver and honest Judge/Sender. In this setting, both the Sender and the Judge adhere to the protocol, but the Receiver can deviate from the protocol by adding elements to its set to maximize its utility.



(a) The full extensive form game.

(b) The simplified game for our protocols.

Figure 4. The cryptographic inspection game for Partial-APSI between a rational Receiver and an honest Sender. The blue and red nodes contained in the dotted ovals form the information sets of the Receiver $\mathcal{C}$ and Sender $\mathcal{S}$, respectively. Here, the devil icon signifies deviation from the protocol, while the angel icon signifies adherence to the protocol. The magnifying glass denotes whether the deviation was detected.

5.2. Rational Receiver and Honest Judge / Sender

In Fig. 4, we depict, as a tree, the cryptographic inspection game (CIG) for our PAPSI protocol. Even if Definition 5.2 allows for dishonest behavior from the Judge, we assume the Judge to be semi-honest to align with our protocol. Each node in the tree represents a point in time when a party must make a decision, with the outgoing edges below that node corresponding to the possible actions the party can take.

For example, in Fig. 4(a), the child of the root labeled $\mathcal{C}$ represents the point during the Authorize phase when the Receiver must decide whether or not to inject malicious elements into its set. The left downward edge corresponds to honest behavior, while the right edge represents the decision to inject malicious elements. The child nodes labeled with J represent the detection of malicious behavior. The dark blue children of this node correspond to the Receiver adding / not adding malicious elements before the Intersect phase. Next, consider the children labeled by J: the left child corresponds to the case where the Receiver behaved honestly. In this case, any subset of elements sampled by the Judge will be valid, so the Judge approves the input with 100% probability. As such, this node has only one downward edge, representing successful approval. The right child corresponds to the case where the Receiver injected malicious elements. Here, when the Judge samples a p-fraction of the Receiver's elements, there's a chance a malicious one is selected. The left edge represents the Judge approving the input (no malicious elements detected), while the right edge corresponds to detecting a malicious element and rejecting the input set. Finally, let's examine the children of the Judge nodes that correspond to the Receiver. Consider, specifically, the two nodes depicted in dark blue. Once again, the Receiver faces a decision: it may either act honestly (leftward downward edges) or inject malicious elements into the input of the Intersect phase (rightward downward edges).

At this point, we make an important observation. In our Partial-APSI protocol (Fig. 3), the Sender will *always* detect if the Receiver modifies their input set between the Authorize and Intersect phases, because the signature verification will then fail. Since we are considering a covert adversary (i.e., one that only deviates when it can avoid detection), we can conclude that a covert Receiver will *never* choose to inject malicious elements at this point. As a result, we can simplify the tree on the left, resulting in the tree in Fig. 4(b).

We now turn our attention to the leaves of the CIG tree, which are labeled with payoff pairs for the Receiver, $\mathcal{C}$, and the Sender, $\mathcal{S}$. When the Receiver behaves honestly, they learn the intersection and receive a payoff of $u_{\mathcal{C}}^{O} - \ell_{\mathcal{C}}^{O}$, where $u_{\mathcal{C}}^{O}$ is the utility of learning the intersection, and $\ell_{\mathcal{C}}^{O}$ is the loss from revealing a p-fraction of their elements. The Sender, who learns nothing, receives a payoff of $u_{\mathcal{S}}^{O}$, representing the utility gained from participating in the protocol.

When the Receiver injects malicious elements into their set before the Authorize phase, and these are undetected, they potentially gain additional information about the Sender's input beyond the intersection. In this case, the Receiver earns a payoff of $u_{\mathcal{C}}^{I} - \ell_{\mathcal{C}}^{O}$, where $u_{\mathcal{C}}^{I}$ denotes the utility of learning both the intersection and any extra elements leaked from the Sender's set, and $\ell_{\mathcal{C}}^{O}$ is the cost of revealing a p-fraction of their own elements. The Sender, who is unaware of the deviation, learns nothing and receives a payoff of $u_{\mathcal{S}}^{O} - \ell_{\mathcal{S}}^{I}$, where $\ell_{\mathcal{S}}^{I}$ reflects the loss incurred from unintended leakage of their input.

If the Receiver deviates and is caught deviating, the protocol aborts. It is clear that the Receiver thus does not learn the intersection. According to the ALA, the Receiver must pay the payoff, $d_{\mathcal{C}}$, to the Sender. To ensure this

happens, the payoff might take the form of a deposit. Additionally, the Receiver revealed the p-fraction of their elements and loses $\ell_{\mathcal{C}}^{O}$, a total loss of: $\ell_{\mathcal{C}}^{O} + d_{\mathcal{C}}$.

In Fig. 4, the blue and red nodes denote an information set for $\mathcal{C}$ and $\mathcal{S}$, respectively. An *information set* for a player is the nodes that are indistinguishable to the player. Therefore, the subtrees under those nodes are identical. See, for example, the red nodes; whether or not the Receiver had malicious input in the authorization phase, the Sender's point of view remains the same, and the subtrees are identical.

We now analyze the game in Fig. 4(b) to find the relationship of the above parameters, utilities, and losses that guarantee that a rational adversary is incentivized to follow the protocol. Let μ be the fraction of malicious elements that a malicious Receiver would inject, and let p denote the fraction of elements revealed to the Judge. If the Receiver deviates and has μn malicious elements before the authorization, the probability A that the Judge does not detected this (ie, the pn elements are from the $n - \mu n$ honest elements) is:

$$A = \frac{\binom{n-\mu n}{pn}}{\binom{n}{pn}} \tag{5.1}$$

Given this probability, we obtain the following lemma.

Lemma 5.3. *Let Π be a two-party Partial-APSI protocol between a rational Receiver and an honest Sender augmented by $\mathsf{ALA}_{\Pi,\kappa}(\mathbf{d})$ where $\mathbf{d} = (d_{\mathcal{C}}, d_{\mathcal{S}})$ specifies the damages paid by $\mathcal{C}$ and $\mathcal{S}$, respectively. If*

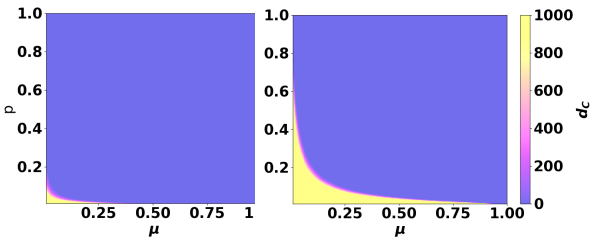
$$d_{\mathcal{C}} > \frac{Au_{\mathcal{C}}^{I} - u_{\mathcal{C}}^{O}}{1 - A} \tag{5.2}$$

where A is defined in (5.1), then the dominant strategy for a Receiver is to follow the protocol.

Proof. First, observe that the Receiver's expected utility for not deviating is $\mathrm{U}_{\mathcal{C}}[\mathsf{NoDeviation}] = u_{\mathcal{C}}^{O} - \ell_{\mathcal{C}}^{O}$. We now compute the Receiver's expected utility for deviating:

$$\begin{aligned}
&\mathrm{U}_{\mathcal{C}}[\mathsf{Deviation}] \\
&= A(u_{\mathcal{C}}^{I} - \ell_{\mathcal{C}}^{O}) - (1-A)(\ell_{\mathcal{C}}^{O} + d_{\mathcal{C}}) \\
&= Au_{\mathcal{C}}^{I} - \ell_{\mathcal{C}}^{O} - (1-A)d_{\mathcal{C}} \\
&< Au_{\mathcal{C}}^{I} - \ell_{\mathcal{C}}^{O} - (1-A)\frac{Au_{\mathcal{C}}^{I} - u_{\mathcal{C}}^{O}}{1-A} \\
&= Au_{\mathcal{C}}^{I} - \ell_{\mathcal{C}}^{O} - Au_{\mathcal{C}}^{I} + u_{\mathcal{C}}^{O} \\
&= u_{\mathcal{C}}^{O} - \ell_{\mathcal{C}}^{O}
\end{aligned}$$

where A is defined as above. The term highlighted in blue is the expected utility when the Receiver deviates, and the Judge does not detect the deviation. The term in pink is the expected utility when the Receiver deviates, and the Judge detects the deviation. The inequality on the third line follows from substituting inequality (5.2) Since the Receiver's expected utility from deviating is upper-bounded by its expected utility from following the protocol, a rational Receiver will act honestly, and the lemma follows. □



(a) Shoe store. (b) Real estate.

Figure 5. We show the damages d_C needed to incentivize a rational Receiver to follow the protocol. (right) Shoe store, Section 5.3.1. (left) Real estate, Section 5.3.2.

5.3. Example Scenarios

5.3.1. Shoe Store. Let us consider an e-commerce $\mathcal{C}$ that sells shoes. They want to promote a new type of shoe. $\mathcal{C}$ buys a shopping advertisement on Google ($\mathcal{S}$). The cost-per-click of this advertisement is \$0.69 [50]. The online store wishes to get the intersection between Google users who saw the ad and clients who purchased the shoes. To this end, $\mathcal{C}$ has a set $\mathcal{X}$ of $n = 1024$ IP addresses of clients and $\mathcal{S}$ a set $\mathcal{Y}$ of $m = 65536$ users.
Intersection: By learning the intersection, the store can determine if ad viewers are new to the store, have purchased additional items, etc. They can adjust their ad and advertising strategy accordingly, e.g., by using free channels. The utility of learning 1 element in the intersection is \$0.69. If the intersection size is $0.1n$, $u_{\mathcal{C}}^{O} = \$0.69 \cdot 0.1n = \$70.66$
Cheating: All those $y \in \mathcal{Y} \backslash \{\mathcal{X} \cap \mathcal{Y}\}$ have seen the ad, but did not buy the shoes of $\mathcal{C}$. Instead, they might have bought different shoes from a competitor. Knowing this provides the store with information on which shoes to sell. We assume that one extra customer results in one extra purchase of \$100. Moreover, $\mathcal{C}$ could guess elements of $\mathcal{Y}$ by adding IP addresses of their competitors' customers. Concluding $u_{\mathcal{C}}^{I} = u_{\mathcal{C}}^{O} + \$100\mu n$.
Authorization: Finally, revealing elements for authorization might be harmful. If the Judge sells this information to other online stores, they lose their customers with the \$100 profit. This results in $l_{\mathcal{C}}^{O} = \$100 \cdot pn$.

In Fig. 5(a), we can see that for $p \geq 20\%$, no damages are needed to incentivize a rational Receiver (shoe store) to adhere to the protocol. For lower values of p, serious damages of around \$1000 are preferred.

5.3.2. Real Estate. With similar reasoning, we consider the example of a website offering legal help for bankruptcy. For them, the cost-per-click for a search ad is \$6.75[50], and the number of customers is likely much lower, $n = 256$. With an intersection size of $0.1n$, this results in $u_{\mathcal{C}}^{O} = 6.75 \cdot 0.1n = \$172.80$. Assume they earn $\$4,015$ per customer [51]. With a similar cheating evaluation as the online store, we obtain $u_{\mathcal{C}}^{I} = u_{\mathcal{C}}^{O} + \$172.80\mu$. A similar loss of revealing clients gives $l_{\mathcal{C}}^{O} = \$4,015pn$

In contrast to the shoe store (Fig. 5(a)), we can see from Fig. 5(b) that the bankruptcy lawyer does need damages for the vast majority of p. Namely, the lawyer has much more to gain by deviating. Only for $p \geq 60\%$,

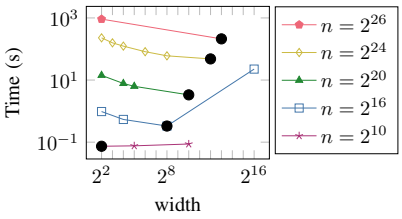


Figure 6. The total time of the protocol for different widths, $m = 2^{10}$, and $p = 20\%$. The black dots indicate the optimal width.

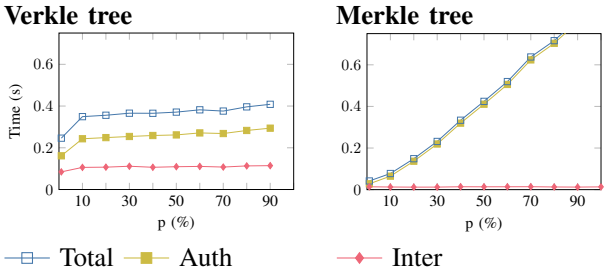


Figure 7. Runtime for different p, $m = 2^{10}$, and $n = 2^{16}$ with $k = 2^8$.

no damages are needed. If a lower value of p is required and introducing damages does not pose problems, a lower value of p can be chosen, and serious damages can be introduced. We conclude that the optimal damage depends on the use case and the percentage p.

6. Experimental results

To assess our Partial-APSI protocol, we implement it in Rust and evaluate it using sets of IP addresses generated with Faker [52]. First and foremost, we should choose an appropriate vector commitment scheme. In Section 3 we discussed several candidates. For the Merkle tree, we use the Rust crate [53] and an additional blinding mechanism (see Appendix E). For the Pointproofs, we use Algorand's implementation [54]. These are limited to sets of size 2^{16}; thus, we use Pointproofs for the Verkle tree. We benchmarked our protocol on an AMD Ryzen Threadripper 7970X 32-core, 64 CPUs. Our implementation is available on https://github.com/TjitskeKoster/Bandwidth_Efficient_Partially_Authorized_PSI.git.

6.1. Different vector commitment schemes

Here, we evaluate which vector commitment scheme best suits our protocol. We experimentally compare the use of a Merkle tree versus a Pointproof-Verkle tree.

Width of the tree. For each set size n, the optimal width to minimize the runtime of our protocol differs. We thus start by determining the width k of the Verkle tree (see Fig. 6). We highlight the optimal width with a black dot. We used $m = 2^{10}$ and $p = 20\%$, and the results are the averages of 10 runs. Unfortunately, we could not run the pointproof protocol for widths larger than 2^{16} due to limitations in the library. In Appendix C, we show that the percentage p does not influence the width of the tree a lot. We thus choose $p = 20\%$ because it aligns with the use case in Fig. 5, and no additional damages are needed to incentivize the rational store to adhere to the protocol when $p = 20\%$.

Evaluation across different p. In Fig. 7, we examine our protocol's performance across different values of p. We observe that when our protocol is instantiated with a Verkle tree, the total runtime stabilizes at around $p > 4\%$. In contrast, with a Merkle tree, this happens at around $p > 80\%$. We expect this convergence once all inner nodes appear in a proof of commitment. For $n = 2^{16}$, $k = 2^8$, and $p = 4\%$, we ask: how likely is it to sample at least one leaf per parent node? For uniform sampling, the expected value is 99%. For the Verkle tree, we can thus expect that an increase in p above 4% does not result in a huge increase in time. In the Merkle tree with 2 children per parent, this convergence occurs much later.

We choose $p = 20\%$ for the remaining comparisons, as this aligns with our use case and allows us to run the Merkle tree in a reasonable amount of time.

Bandwidth. We investigate the bandwidth difference between our protocol when using the two trees, with $m = 2^{10}$ and $p = 20\%$. Fig. 8 plots the bandwidths for various n. We conclude that the Verkle tree uses less bandwidth than the Merkle tree. This is to be expected because the proof length of the Verkle tree is $\log_k(n)$ and for the Merkle tree it is $\log_2(n)$. To illustrate, for $n = 2^{16}$, and $k = 2^8$, $\log_k(n) = 2$, whereas $\log_2(n) = 16$.

Runtime Verkle and Merkle tree. To determine the best tree for our protocol, we compare the runtime of the protocol for both trees in the Appendix; Table 3, Appendix E. In total time, the Merkle tree outperforms the Verkle tree for $n \leq 2^{16}$ and $m \leq 2^{20}$. This has several reasons. Firstly, the Commit algorithm of the Merkle tree is faster than that of the Verkle tree, reducing the Sender's time. Secondly, the verification V is faster for the Merkle tree. However, the Merkle tree needs an additional blinding and verification from the Judge. Slowing down the Judge for larger n. The extra blinding results slow down the Server for larger m. Additionally, the entire protocol performs better with the Verkle tree for increasing p (Fig. 7). Taking this into account, along with the fact that the Verkle tree requires less bandwidth, we continue with the Pointproofs-Verkle tree.

6.2. Comparison with Related Work

In Table 2, we experimentally compare our protocol against Falzon and Markatou's [22] implementation [55], written in C++. Both protocols are parallelized and run on the same AMD Ryzen Threadripper machine.

Bandwidth. We observe that our protocol requires less bandwidth than Falzon and Markatou [22] for all n and m.

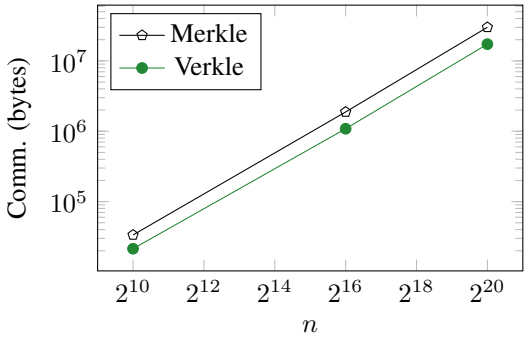


Figure 8. Plots of total bandwidth in bytes for $m = 2^{20}$, $p = 20\%$ and different n.

TABLE 2. COMPARISON OF OUR PAPSI PROTOCOL WITH [22] ON AN AMD RYZEN THREADRIPPER 7970X 32-CORE CPU. WE USE DATASETS WITH IP ADDRESSES AND $p = 20$. ALL NUMBERS ARE ROUNDED TO THE NEAREST INTEGER. WE HIGHLIGHT CASES WHERE WE OUTPERFORM [22]. WE TAKE THE AVERAGE OF 10 RUNS, APART FORM THE ROW INDICATED WITH * TO AVOID ABORTION OF THE RUN.

Set sizes		Scheme	Communcation (KB)		Runtime (ms)			Total Runtime			
m	n		Auth	Inter	$\mathcal{J}$	$\mathcal{C}$	$\mathcal{S}$	LAN	1Gbps	200Mbps	50Mbps
2^{10}	2^{10}	FM [22]	146	295	1	8	6	15	575	575	575
		Verkle	9	12	21	31	20	73	73	73	73
	2^{16}	FM [22]	9507	6489	76	646	6	728	1288	1288	1288
		Verkle	557	528	35	219	95	349	349	349	349
	2^{20}	FM [22]	152082	100860	1116	9807	6	10930	11490	11490	11490
		Verkle	8907	8393	165	5722	1028	6915	6916	6917	6919
	2^{24}	FM [22]	2432504	1610810	19400	172531	12	191945	192505	192505	192505
		Verkle	142499	134222	1374	35243	10942	47558	47561	47570	47603
	2^{26}	FM [22]	9739449	6442648	79619	730126	13	809759	810319	810319	810321
		Verkle	569990	536875	5026	166448	40181	211654	211663	211699	211831
2^{16}	2^{10}	FM [22]	146	12746	1	51	481	533	1093	1093	1093
		Verkle	9	270	20	42	35	96	97	97	97
	2^{16}	FM [22]	9516	18939	77	616	486	1181	1741	1741	1741
		Verkle	557	786	41	219	104	364	364	365	365
	2^{20}	FM [22]	152255	113311	1172	9873	488	11534	12094	12094	12094
		Verkle	8907	8651	169	2232	907	3308	3309	3309	3311
	2^{24}	FM [22]	2435286	1623261	18635	171663	495	190794	191354	191354	191354
		Verkle	142499	134480	1416	35427	10972	47816	47818	47827	47860
	2^{26}	FM [22]	9737824	6455099	76906	724382	525	801815	802375	802375	802377
		Verkle	569990	537133	4796	166218	40779	211793	211802	211838	211970
2^{20}	2^{10}	FM [22]	147	202473	1	750	8332	9083	9643	9643	9643
		Verkle	9	4203	21	58	124	202	203	203	203
	2^{16}	FM [22]	9520	208666	78	1301	8361	9741	10301	10301	10301
		Verkle	557	4719	37	239	162	438	438	438	439
	2^{20}	FM [22]	152256	303038	1196	10558	8390	20145	20705	20705	20705
		Verkle	8907	12583	168	2263	979	3410	3411	3412	3414
	2^{24}	FM [22]	2436770	1812987	19454	171482	8357	199295	199855	199855	199855
		Verkle	142499	138412	1382	35356	11076	47814	47817	47826	47859
	2^{26}	FM [22]	9738788	6644826	75754	727536	8399	811690	812250	812250	812252
		Verkle	569989	541065	4832	166082	40176	211090	211099	211135	211268
2^{24}	2^{10}	FM [22]	146	3238100	1	14280	148125	162406	162966	162966	162966
		Verkle	9	67117	17	1302	1181	2500	2501	2503	2511
	2^{16}	FM [22]	9510	3244294	77	14138	147893	162109	162669	162669	162669
		Verkle	557	67633	37	1454	1254	2745	2746	2749	2757
	2^{20}	FM [22]	152136	3338665	1168	23451	147832	172452	173012	173012	173012
		Verkle	8907	75498	166	3542	2060	5769	5770	5772	5783
	2^{24}	FM [22]	2436698	4848615	19420	186636	148426	354483	355043	355043	355044
		Verkle	142499	201327	1388	37061	12130	50579	50582	50593	50635
	2^{26}	FM [22]*	9761513	9680453	78776	782378	152182	1013336	1013896	1013896	1013899
		Verkle	569981	603980	4959	169767	41720	216446	216456	216493	216634

In fact, for $n, m = 2^{20}$, we require approximately $21.2\times$ less bandwidth. There are two primary reasons for this: first, the elements of our blinded sets are smaller, around 10 bytes instead of 32. Because the elements of [22] must be defined over a field that supports bilinear pairings. Secondly, in the authorization phase of [22], the number of commitments and signatures is linear in n. In contrast, in our protocol, we send only one commitment and one signature. Thus, we achieve lower asymptotic complexity for the authorization phase.

Runtime. Regarding general runtime, we tend to outperform [22], as m increases. This is primarily because the intersection of Falzon and Markatou [22] requires computing and evaluating m OPRFs. In our protocol, the Sender needs to build the verkle tree, which grows with n. Additionally, for the authorization phase, the Judge in the protocol of [22] must check n EEA proofs (proving all elements are raised to the same power r, without revealing r) and compute n signatures. In our protocol, no EEA proofs are needed, and the Judge signs a single token; these two improvements reduce the Judge's runtime. For set sizes $n = m = 2^{20}$, and $p = 20\%$, we are about $5.9\times$ faster over a LAN network.

Cost. We compare the costs of the protocols using the c4d-standard-64 machine on Google Cloud [56], which

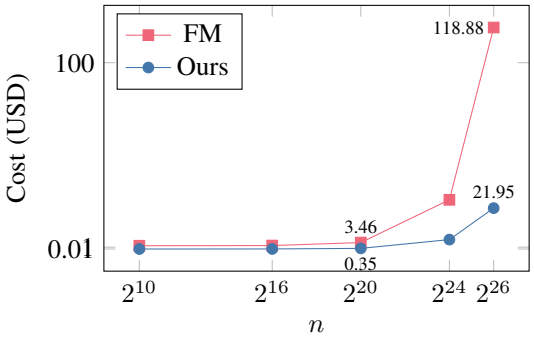


Figure 9. Costs of running the protocol 100 times for $m = 2^{20}$, $p = 20\%$ and different n. The cost scales linearly with n.

is most similar to our machine[3]. We assume the parties are in different parts of the world: the Judge at the International Criminal Court in the Netherlands, the Receiver in Belgium, and the Sender at Google headquarters in Mountain View, California (US), with the closest machine in Oregon. The different costs of the virtual machines are; $2.96 per hour in Oregon, $3.11 per hour in the Netherlands and $3.26 per hour in Belgium [56]. The network costs are $0.02 for EU-EU communication and $0.05 for EU-US communication [58]. Fig. 9 plots the cost of running the protocol. We see that we massively reduce the costs of [22]. In fact, for $n, m = 2^{20}$, our protocol is around 10 times cheaper and for $n = 2^{26}$ around 5 times. Despite the fact that Figure 9 has a logarithmic x-axis, the increase in the cost is linear in n.

7. Conclusion

We present a Partial-APSI protocol with significantly improved bandwidth compared to previous work [22]. We improve on bandwidth because the field elements in our implementation require less bandwidth than those in [22]. In the authorization phase, we also require less bandwidth for increasing n, achieving lower asymptotic bandwidth. During the intersection phase, our bandwidth is linear in $2n + m$, whereas the protocol of Falzon and Markatou is linear in $3m$, leading to different trade-offs. We experimentally compare our protocol with Falzon and Markatou [22]. We concluded that for $n = m = 2^{20}$, we require approximately $21.2\times$ less bandwidth and are about $5.9\times$ faster over a LAN network. We also provide a game-theoretic analysis, demonstrating that by incorporating appropriate payouts, a rational Receiver can be incentivized to adhere to the protocol.

Several interesting research directions stem from this work. First, it would be interesting to find a way to generalize the partial authorization phase to apply to any (A)PSI protocol. Secondly, there exist settings where we would like to prevent maliciously chosen inputs to PSI protocols, but no appropriate Judge can be found. This leads to the question of whether we can develop such protocols without a trusted third party. Finally, our preliminary game-theoretic analysis on Partial-APSI could be extended to other MPC protocols.

3. Both machines have an AMD kernel, an x86 architecture, 64 CPUs, and approximately 248 GiB of memory [57].

References

[1] P. Baldi, R. Baronio, E. D. Cristofaro, P. Gasti, and G. Tsudik, "Countering GATTACA: efficient and secure testing of fully-sequenced human genomes," in *Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS 2011, October 17-21, 2011*, Y. Chen, G. Danezis, and V. Shmatikov, Eds., Chicago, Illinois, USA: ACM, 2011, pp. 691–702. DOI: 10.1145/2046707.2046785. [Online]. Available: https://doi.org/10.1145/2046707.2046785.

[2] L. Shen, X. Chen, D. Wang, B. Fang, and Y. Dong, "Efficient and private set intersection of human genomes," in *IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2018, Madrid, Spain, December 3-6, 2018*, H. J. Zheng et al., Eds., New York: IEEE Computer Society, 2018, pp. 761–764. DOI: 10.1109/BIBM.2018.8621291. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/BIBM.2018.8621291.

[3] M. Li, S. Yu, N. Cao, and W. Lou, "Privacy-preserving distributed profile matching in proximity-based mobile social networks," *IEEE Transactions on Wireless Communications*, pp. 2024–2033, 2013. DOI: 10.1109/TWC.2013.032513.120149.

[4] S. Nagaraja, P. Mittal, C.-Y. Hong, M. Caesar, and N. Borisov, "Botgrep: Finding p2p bots with structured graph analysis," in *Proceedings of the 19th USENIX Conference on Security*, ser. USENIX Security'10, Washington, DC: USENIX Association, 2010, p. 7.

[5] H. Chen, K. Laine, and P. Rindal, "Fast private set intersection from homomorphic encryption," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, B. Thuraisingham, D. Evans, T. Malkin, and D. Xu, Eds., ACM, 2017, pp. 1243–1255. DOI: 10.1145/3133956.3134061. [Online]. Available: https://doi.org/10.1145/3133956.3134061.

[6] L. Hetz, T. Schneider, and C. Weinert, "Scaling mobile private contact discovery to billions of users," in *Computer Security - ESORICS 2023 - 28th European Symposium on Research in Computer Security, The Hague, The Netherlands, September 25-29, 2023, Proceedings, Part I*, G. Tsudik, M. Conti, K. Liang, and G. Smaragdakis, Eds., ser. Lecture Notes in Computer Science, Springer, 2023, pp. 455–476. DOI: 10.1007/978-3-031-50594-2_23. [Online]. Available: https://doi.org/10.1007/978-3-031-50594-2_23.

[7] D. Kales, C. Rechberger, T. Schneider, M. Senker, and C. Weinert, "Mobile private contact discovery at scale," in *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019*, N. Heninger and P. Traynor, Eds., USENIX Association, 2019, pp. 1447–1464. [Online]. Available: https://www.usenix.org/conference/usenixsecurity19/presentation/kales.

[8] M. Ion et al., "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. DOI: 10.1109/EUROSP48549.2020.00031. [Online]. Available: https://doi.org/10.1109/EuroSP48549.2020.00031.

[9] M. Ion et al., "Private intersection-sum protocol with applications to attributing aggregate ad conversions," *IACR Cryptol. ePrint Arch.*, p. 738, 2017. [Online]. Available: http://eprint.iacr.org/2017/738.

[10] T. Lepoint, S. Patel, M. Raykova, K. Seth, and N. Trieu, "Private join and compute from PIR with default," in *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part II*, M. Tibouchi and H. Wang, Eds., ser. Lecture Notes in Computer Science, Springer, 2021, pp. 605–634. DOI: 10.1007/978-3-030-92075-3_21. [Online]. Available: https://doi.org/10.1007/978-3-030-92075-3_21.

[11] B. A. Huberman, M. Franklin, and T. Hogg, "Enhancing privacy and trust in electronic communities," in *Proceedings of the 1st ACM Conference on Electronic Commerce*, ser. EC '99, Denver, Colorado, USA: Association for Computing Machinery, 1999, pp. 78–86, ISBN: 1581131763. DOI: 10.1145/336992.337012. [Online]. Available: https://doi.org/10.1145/336992.337012.

[12] 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*, Oakland, California, USA: IEEE Computer Society, 1986, pp. 134–137. DOI: 10.1109/SP.1986.10022. [Online]. Available: https://doi.org/10.1109/SP.1986.10022.

[13] B. Pinkas, T. Schneider, C. Weinert, and U. Wieder, "Efficient circuit-based PSI via cuckoo hashing," in *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part III*, J. B. Nielsen and V. Rijmen, Eds., ser. Lecture Notes in Computer Science, Springer, 2018, pp. 125–157. DOI: 10.1007/978-3-319-78372-7_5. [Online]. Available: https://doi.org/10.1007/978-3-319-78372-7_5.

[14] M. Chase and P. Miao, "Private set intersection in the internet setting from lightweight oblivious prf," in *Advances in Cryptology – CRYPTO 2020: 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17–21, 2020, Proceedings, Part III*, Santa Barbara, CA, USA: Springer-Verlag, 2020, pp. 34–63, ISBN: 978-3-030-56876-4. DOI: 10.1007/978-3-030-56877-1_2. [Online]. Available: https://doi.org/10.1007/978-3-030-56877-1_2.

[15] B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai, "Spot-light: Lightweight private set intersection from sparse ot extension," in *Advances in Cryptology – CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part III*, Santa Barbara, CA, USA: Springer-Verlag, 2019, pp. 401–431, ISBN: 978-3-030-26953-1. DOI: 10.1007/978-3-030-26954-8_13. [Online]. Available: https://doi.org/10.1007/978-3-030-26954-8_13.

[16] F. Falzon and T. Tang, "Learning from functionality outputs: Private join and compute in the real world," in *34th USENIX Security Symposium, USENIX Security 2025, Seattle, WA, USA, August 13-15, 2025*, USENIX Association, 2025. [Online]. Available: https://www.usenix.org/system/files/conference/usenixsecurity25/sec25cycle1-prepub-994-falzon.pdf.

[17] X. Guo, Y. Han, Z. Liu, D. Wang, Y. Jia, and J. Li, "Birds of a feather flock together: How set bias helps to deanonymize you via revealed intersection sizes," in *31st USENIX Security Symposium (USENIX Security 22)*, Boston, MA: USENIX Association, Aug. 2022, pp. 1487–1504, ISBN: 978-1-939133-31-1. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/guo.

[18] B. Jiang, J. Du, and Q. Yan, "Anonpsi: An anonymity assessment framework for PSI," in *31st Annual Network and Distributed System Security Symposium, NDSS 2024, San Diego, California, USA, February 26 - March 1, 2024*, San Diego, California, USA: The Internet Society, 2024. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/anonpsi-an-anonymity-assessment-framework-for-psi/.

[19] J. Vos, J. van Assen, T. Koster, E. A. Markatou, and Z. Erkin, *On the insecurity of bloom filter-based private set intersections*, Cryptology ePrint Archive, Paper 2024/1901, 2024. [Online]. Available: https://eprint.iacr.org/2024/1901.

[20] J. Camenisch and G. M. Zaverucha, "Private intersection of certified sets," in *Financial Cryptography and Data Security*, R. Dingledine and P. Golle, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 108–127, ISBN: 978-3-642-03549-4.

[21] E. De Cristofaro, S. Jarecki, J. Kim, and G. Tsudik, "Privacy-preserving policy-based information transfer," in *Privacy Enhancing Technologies*, I. Goldberg and M. J. Atallah, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 164–184, ISBN: 978-3-642-03168-7.

[22] F. Falzon and E. A. Markatou, "Re-visiting authorized private set intersection: A new privacy-preserving variant and two protocols," *Proc. Priv. Enhancing Technol.*, pp. 792–807, 2025. DOI: 10.56553/POPETS-2025-0041. [Online]. Available: https://doi.org/10.56553/popets-2025-0041.

[23] *General data protection regulation (GDPR)*, https://gdpr-info.eu/, Visited 8 July 2025, 2016.

[24] C. P. P. A. E. Division, *Applying data minimzation to consumer requests*, https://cppa.ca.gov/pdf/enfadvisory202401.pdf, Visited 8 July 2025, Jan. 2024.

[25] E. De Cristofaro and G. Tsudik, "Practical private set intersection protocols with linear complexity," in *Financial Cryptography and Data Security*, R. Sion, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 143–159, ISBN: 978-3-642-14577-3. DOI: 10.1007/978-3-642-14577-3_13.

[26] E. De Cristofaro, J. Kim, and G. Tsudik, "Linear-complexity private set intersection protocols secure in malicious model," in *Advances in Cryptology - ASIACRYPT 2010*, M. Abe, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 213–231, ISBN: 978-3-642-17373-8. DOI: 10.1007/978-3-642-17373-8_13.

[27] S. K. Debnath and R. Dutta, "Secure and efficient private set intersection cardinality using bloom filter," in *Information Security*, J. Lopez and C. J. Mitchell, Eds., Cham: Springer International Publishing, 2015, pp. 209–226, ISBN: 978-3-319-23318-5. DOI: 10.1007/978-3-319-23318-5_12.

[28] C. M. Westhoff, "Blood group genotyping," *Blood, The Journal of the American Society of Hematology*, pp. 1814–1820, 2019.

[29] M. George and S. Kamara, "Adversarial level agreements for two-party protocols," in *ASIA CCS '22: ACM Asia Conference on Computer and Communications Security, Nagasaki, Japan, 30 May 2022 - 3 June 2022*, Y. Suga, K. Sakurai, X. Ding, and K. Sako, Eds., Nagasaki, Japan: ACM, 2022, pp. 816–830. DOI: 10.1145/3488932.3517385. [Online]. Available: https://doi.org/10.1145/3488932.3517385.

[30] F. Kerschbaum, "Outsourced private set intersection using homomorphic encryption," in *7th ACM Symposium on Information, Compuer and Communications Security, ASIACCS '12, Seoul, Korea, May 2-4, 2012*, H. Y. Youm and Y. Won, Eds., Seoul, Korea: ACM, 2012, pp. 85–86. DOI: 10.1145/2414456.2414506. [Online]. Available: https://doi.org/10.1145/2414456.2414506.

[31] 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., Denver, Colorado, USA: ACM, 2015, pp. 121–131. DOI: 10.1145/2808138.2808149. [Online]. Available: https://doi.org/10.1145/2808138.2808149.

[32] E. Stefanov, E. Shi, and D. Song, "Policy-enhanced private set intersection: Sharing information while enforcing privacy policies," in *Public Key Cryptography – PKC 2012*, M. Fischlin, J. Buchmann, and M. Manulis, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 413–430, ISBN: 978-3-642-30057-8. DOI: 10.1007/978-3-642-30057-8_25.

[33] M. Nagy, E. D. Cristofaro, A. Dmitrienko, N. Asokan, and A. Sadeghi, "Do I know you?: Efficient and privacy-preserving common friend-finder protocols and applications," in *Annual Computer Security Applications Conference, ACSAC '13, New Orleans, LA, USA, December 9-13, 2013*, C. N. P. Jr., Ed., Orleans, LA, USA: ACM, 2013, pp. 159–168. DOI: 10.1145/2523649.2523668. [Online]. Available: https://doi.org/10.1145/2523649.2523668.

[34] P. D'Arco, M. I. González Vasco, A. L. Pérez del Pozo, and C. Soriente, "Size-hiding in private set intersection: Existential results and constructions," in *Progress in Cryptology - AFRICACRYPT 2012*, A. Mitrokotsa and S. Vaudenay, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 378–394, ISBN: 978-3-642-31410-0. DOI: 10.1007/978-3-642-31410-0_23.

[35] 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*, ser. CCS '24, Salt Lake City, UT, USA: Association for Computing Machinery, 2024, pp. 1478–1492, ISBN: 9798400706363. DOI: 10.1145/3658644.3690349. [Online]. Available: https://doi.org/10.1145/3658644.3690349.

[36] B. C. Ghosh, S. Patranabis, D. Vinayagamurthy, V. Ramakrishna, K. Narayanam, and S. Chakraborty, "Private certifier intersection," *Cryptology ePrint Archive*, 2022.

[37] A. Goel, P. Miao, P. V. L. Pham, and S. Singh, *PICS: Private intersection over committed (and reusable) sets*, Cryptology ePrint Archive, Paper 2025/1071, 2025. [Online]. Available: https://eprint.iacr.org/2025/1071.

[38] Y. Yang et al., "A blind signature-based authorization scheme for enhancing the privacy of cloud-assisted private set intersection," *Computer Networks*, p. 111 844, 2025, ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2025.111844. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128625008102.

[39] D. Boneh, "The decision diffie-hellman problem," in *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings*, J. Buhler, Ed., ser. Lecture Notes in Computer Science, Portland, Oregon, USA: Springer, 1998, pp. 48–63. DOI: 10.1007/BFB0054851. [Online]. Available: https://doi.org/10.1007/BFb0054851.

[40] D. Catalano and D. Fiore, "Vector commitments and their applications," in *Public-Key Cryptography - PKC 2013 - 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, February 26 - March 1, 2013. Proceedings*, K. Kurosawa and G. Hanaoka, Eds., ser. Lecture Notes in Computer Science, Nara, Japan: Springer, 2013, pp. 55–72. DOI: 10.1007/978-3-642-36362-7_5. [Online]. Available: https://doi.org/10.1007/978-3-642-36362-7_5.

[41] R. C. Merkle, "A certified digital signature," in *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, G. Brassard, Ed., ser. Lecture Notes in Computer Science, Santa Barbara, California, USA: Springer, 1989, pp. 218–238. DOI: 10.1007/0-387-34805-0_

21. [Online]. Available: https://doi.org/10.1007/0-387-34805-0_21.

[42] S. Gorbunov, L. Reyzin, H. Wee, and Z. Zhang, "Pointproofs: Aggregating proofs for multiple vector commitments," in *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*, J. Ligatti, X. Ou, J. Katz, and G. Vigna, Eds., Virtual Event, USA: ACM, 2020, pp. 2007–2023. DOI: 10.1145/3372297.3417244. [Online]. Available: https://doi.org/10.1145/3372297.3417244.

[43] B. Libert and M. Yung, "Concise mercurial vector commitments and independent zero-knowledge sets with short proofs," in *Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings*, D. Micciancio, Ed., ser. Lecture Notes in Computer Science, Zurich, Switzerland: Springer, 2010, pp. 499–517. DOI: 10.1007/978-3-642-11799-2_30. [Online]. Available: https://doi.org/10.1007/978-3-642-11799-2_30.

[44] J. Kuszmaul, *Verkle trees*, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:218475793.

[45] Y. Aumann and Y. Lindell, "Security against covert adversaries: Efficient protocols for realistic adversaries," in *Proceedings of the 4th Conference on Theory of Cryptography*, ser. TCC'07, Amsterdam, The Netherlands: Springer-Verlag, 2007, pp. 137–156, ISBN: 9783540709350.

[46] J. Halpern and V. Teague, "Rational secret sharing and multiparty computation: Extended abstract," in *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing*, ser. STOC '04, Chicago, IL, USA: Association for Computing Machinery, 2004, pp. 623–632, ISBN: 1581138520. DOI: 10.1145/1007352.1007447. [Online]. Available: https://doi.org/10.1145/1007352.1007447.

[47] P. D. Azar and S. Micali, "Rational proofs," in *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing*, ser. STOC '12, New York, New York, USA: Association for Computing Machinery, 2012, pp. 1017–1028, ISBN: 9781450312455. DOI: 10.1145/2213977.2214069. [Online]. Available: https://doi.org/10.1145/2213977.2214069.

[48] S. D. Gordon and J. Katz, "Rational secret sharing, revisited," in *Proceedings of the 5th International Conference on Security and Cryptography for Networks*, ser. SCN'06, Maiori, Italy: Springer-Verlag, 2006, pp. 229–241, ISBN: 3540380809. DOI: 10.1007/11832072_16. [Online]. Available: https://doi.org/10.1007/11832072_16.

[49] J. Katz, "Bridging game theory and cryptography: Recent results and future directions," in *Proceedings of the 5th Conference on Theory of Cryptography*, ser. TCC'08, New York, USA: Springer-Verlag, 2008, pp. 251–272, ISBN: 354078523X.

[50] D. Moons, *27 google ads benchmarks (2025)*, Average Google Shopping CPC - Clothing & Apparel and Average Google Search Ads Cost Per Click - Legal. [Online]. Available: https://www.storegrowers.com/google-ads-benchmarks/.

[51] C. Brock, What is the average lawyer retainer fee? Bankruptcy. [Online]. Available: https://www.lawpay.com/about/blog/lawyer-hourly-rate-by-state/.

[52] *Faker*. [Online]. Available: https://faker.readthedocs.io/en/master/providers/faker.providers.internet.html.

[53] A. Suprunchuk, *Rs_merkle*. [Online]. Available: https://crates.io/crates/rs_merkle.

[54] Algorand, *Pointproofs vector commitment parameter generation*. [Online]. Available: https://github.com/algorand/pointproofs-paramgen/commits/master/.

[55] F. Falzon and E. A. Markatou, *Re-visiting authorized private set intersection: A new privacy-preserving variant and two protocols*, https://github.com/markatou/Partial-APSI, [Online; accessed 27-May-2025], 2025.

[56] Google, *Compute engine pricing*. [Online]. Available: https://cloud.google.com/compute/all-pricing.

[57] Google, *Machine families resource and comparison guide*. [Online]. Available: https://cloud.google.com/compute/docs/machine-resource#machine_type_comparison.

[58] Google, *All networking pricing*. [Online]. Available: https://cloud.google.com/vpc/network-pricing.

[59] A. Kate, G. M. Zaverucha, and I. Goldberg, "Constant-size commitments to polynomials and their applications," in *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, December 5-9, 2010. Proceedings*, M. Abe, Ed., ser. Lecture Notes in Computer Science, Singapore: Springer, 2010, pp. 177–194. DOI: 10.1007/978-3-642-17373-8_11. [Online]. Available: https://doi.org/10.1007/978-3-642-17373-8_11.

[60] D. Feist, *Pcs multiproofs using random evaluation*, Jun. 2021. [Online]. Available: https://dankradfeist.de/ethereum/2021/06/18/pcs-multiproofs.html.

[61] A. Tomescu, I. Abraham, V. Buterin, J. Drake, D. Feist, and D. Khovratovich, "Aggregatable subvector commitments for stateless cryptocurrencies," in *Security and Cryptography for Networks - 12th International Conference, SCN 2020, Amalfi, Italy, September 14-16, 2020, Proceedings*, C. Galdi and V. Kolesnikov, Eds., ser. Lecture Notes in Computer Science, Amalfi, Italy: Springer, 2020, pp. 45–64. DOI: 10.1007/978-3-030-57990-6_3. [Online]. Available: https://doi.org/10.1007/978-3-030-57990-6_3.

[62] CleanPegasus, *Kzg-commitment*. [Online]. Available: https://crates.io/crates/kzg-commitment.

Appendix A.

In the appendix, we justify our choice of using Pointproofs for the Verkle tree and provide details on how our protocol scales better with a Verkle tree than with a Merkle tree. We provide in Appendix B additional computational assumptions needed for the Pointproofs. We also add the computational assumptions needed for the so-called KZG

vector commitment scheme. In Appendix C we evaluate the best width of the verkle tree with both KZG and Pointproofs. In Appendix D we explain why we chose the Pointproofs as the better commitment scheme for the Verkle tree. The last section of the appendix, Appendix E compares and evaluates the differences between the Merkle and Verkle tree.

Appendix B. Computational assumptions

For completeness, we define the computational assumptions needed for the KZG and pointproof commitment scheme in this section.

Definition B.1. Discrete Logarithm problem *(DL): For a generator g of group $\mathbb{G}$ of prime order p and $a \in \mathbb{Z}_p$ it holds that for all probabilistic polynomial time algorithms $\mathcal{A}$,*

$$\Pr[\mathcal{A}(g, g^a) = a] \leq \mathsf{negl}(\lambda),$$

for security parameter $\mathsf{negl}(\lambda)$.

A variant of the DDH assumption is the t-Strong Diffie-Hellman (t-SDH) assumption [59].

Definition B.2. t-Strong Diffie-Hellman (t-SDH) assumption*: Given a group $\mathbb{G}$ of prime order q and a tuple $\left\langle g, g^{\alpha}, g^{\alpha^2}, ..., g^{\alpha^t} \right\rangle$ for $g \in \mathbb{G}$, it holds that for all probabilistic polynomial time algorithms $\mathcal{A}$,*

$$\Pr\left[\mathcal{A}(g, g^{\alpha}, g^{\alpha^2}, ..., g^{\alpha^t}) = \left\langle c, g^{\frac{1}{\alpha+c}} \right\rangle\right] \leq \mathsf{negl}(\lambda),$$

for any value $c \in \mathbb{Z}_{\shortparallel} \setminus \{-\alpha\}$ and security parameter $\mathsf{negl}(\lambda)$.

A variant of the t-SDH assumption is the t-Bilinear Strong Diffie-Hellman (t-BSDH) assumption [59].

Definition B.3. t-Bilinear Strong Diffie-Hellman (t-BSDH) assumption *Consider a similar setting as for the t-SDH assumption with an additional bilinear pairing e, it holds that for all probabilistic polynomial time algorithms $\mathcal{A}$,*

$$\Pr\left[\mathcal{A}(g, g^{\alpha}, g^{\alpha^2}, ..., g^{\alpha^t}) = \left\langle c, e(g,g)^{\frac{1}{\alpha+c}} \right\rangle\right] \leq \mathsf{negl}(\lambda),$$

for any value $c \in \mathbb{Z}_{\shortparallel} \setminus \{-\alpha\}$ and security parameter $\mathsf{negl}(\lambda)$.

A variant of the t-BSDH assumption is the t-weak bilinear Diffie-Hellman assumption [42]

Definition B.4. t-weak Bilinear Strong Diffie-Hellman (t-wBSDH) assumption *Consider a similar setting as for the t-BSDH assumption, it holds that for all probabilistic polynomial time algorithms $\mathcal{A}$,*

$$\Pr\left[\mathcal{A}(g, g^{\alpha}, ..., g^{\alpha^t}, g^{\alpha^{t+1}}, ..., g^{\alpha^n}) = \left\langle c, e(g,g)^{\alpha+1} \right\rangle\right] \leq \mathsf{negl}(\lambda),$$

for security parameter $\mathsf{negl}(\lambda)$.

Definition B.5. The Algebraic Group Model (AGM) *Given a group $\mathbb{G}$ and elements $x_1, ..., x_n \in \mathbb{G}$ for all probabilistic polynomial time algorithms $\mathcal{A}$,*

$$\Pr[Z = \prod_{i=1}^{n} x_i^{z_i} \mid z_1, .., z_n, Z \leftarrow \mathcal{A}] \leq \mathsf{negl}(\lambda),$$

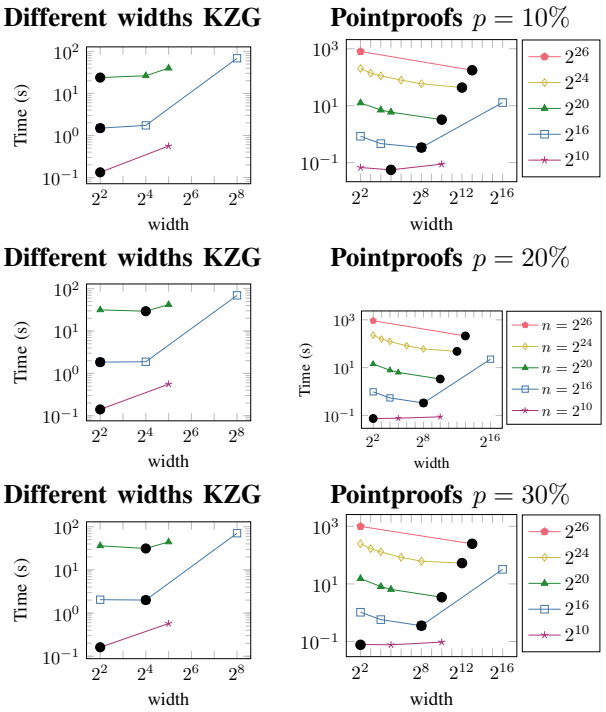


Figure 10. The total time of the protocol for $m = 2^{10}$. Each color corresponds to a different n. The black dots indicate the optimal width.

for security parameter $\mathsf{negl}(\lambda)$.

Appendix C. Widths of the tree for different percentages

In the paper, we determined the optimal width of the tree by evaluating the runtime of our protocol for $m = 2^{20}$, $p = 20\%$, and various values of n. It might be clear that the value of m is irrelevant for the width of the tree, but p might influence the optimal width. To assess this influence, Fig. 10 plots the runtime of the protocol for different p and widths. We conclude that width depends somewhat on p, but the dependence is not significant.

Appendix D. Why not KZG-based Verkle tree?

We begin this section with an explanation of the KZG commitment scheme and then evaluate its performance.

D.1. KZG vector commitment

Kate et al. [59] proposed a commitment scheme based on polynomials, which can be adapted into a vector commitment scheme as Feist [60] shows. On a high level, the scheme takes several values m_i for $i \leq n$ and interpolates a polynomial for the set of points $\{(x_i, m_i)\}_{i \leq n}$. The commitment is the polynomial evaluated at a secret value. Fortunately, this commitment is concise.

The notion of polynomial binding is introduced to guarantee the security of a polynomial commitment scheme. This notion captures that it is unlikely to find two polynomials with the same commitment. It can be proven that the KZG commitment scheme is polynomial binding, hiding, and binding if the discrete logarithm and

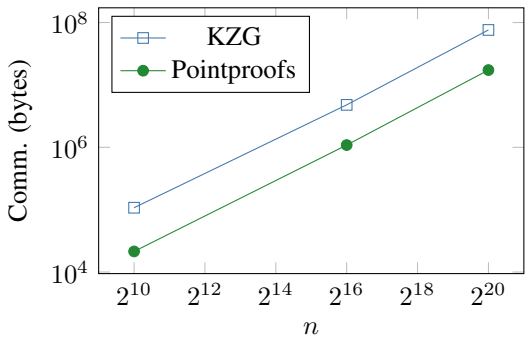


Figure 11. Plots of total bandwidth in bytes for $m = 2^{20}$, $p = 20\%$ and different n.

the t-(bilinear) strong Diffie-Hellman assumptions hold, for a definition of the assumptions see Appendix B. For polynomial binding and hiding, we refer the reader to [59], for binding, we refer the reader to [61]. This vector commitment scheme supports batch openings and is concise.

D.2. Implementation

For the KZG commitments, we used the Rust crates [62]. We instantiate our protocol with a KZG Verkle tree and evaluate the optimal branching factor of the tree.

From Fig. 6, we see that the optimal width (announced with a black dot) is different for both the KZG and the Pointproofs commitment scheme. We highlight that the KZG proofs are about a factor of 10 slower than the Pointproofs.

In Fig. 11, we see that our protocol with the Pointproof-Verkle tree uses less bandwidth than the KZG-Verkle tree. This is mainly because the KZG library did not allow for serialization of the elements.

We concluded that a pointproofs-based verkle tree outperforms a KZG-verkle tree in both bandwidth and runtime. We therefore proceed with the Pointproofs-verkle tree.

Appendix E. Comparison Merkle and Verkle tree

E.1. Merkle tree

In this section, we will explore instantiating our protocol using a Merkle tree. This instantiation is not trivial; we see that our protocol depicted in Figure 3 is secure against a semi-honest Judge if and only if the VCS is hiding (Theorem 4.4). However, the Merkle tree is not a hiding commitment scheme. We therefore need an additional round of blinding. Note that the blinding mechanism we introduced in Figure 3 is insufficient because the Receiver provides the blinding parameter r to the Judge.

To securely instantiate the protocol using a Merkle tree, we thus introduce a round of blinding before computing the leaves. The Receiver samples a random r' and blinds their elements as

$$\hat{\mathcal{X}} = \{H(x_i \| ID_{\mathcal{C}})^{r'} \mid x \in \mathcal{X}\}.$$

The Receiver provides r' to the Sender, such that the Sender can blind their elements in the same way. The Sender has to blind their elements to ensure the correctness of the scheme.

However, it is not trivial how to open the double-blinded elements to the Judge. The Receiver cannot provide the blinding factor r'. The Receiver will instead give a Zero-Knowledge proof that all elements are raised to the same exponent r'. Inspired by Falzon and Markatou [22], we will introduce an Exponent Equality Argument (EEA) proof. The full protocol with the EEA proofs is depicted in Figure 12. The additions for the Merkle tree are depicted in pink.

E.2. EEA proofs

An Exponent Equality Argument proof is a mechanism to prove for a set of elements $\{H(x_i)^{r'}\}_{i\in\mathcal{I}}$ that they are all raised to the same exponent r', without revealing r'. The EEA proof is a zero-knowledge proof, and as usual, it requires a challenge from the verifier. However, we follow Falzon and Markatou [22] and use the Fiat-Shamir heuristic to compute the challenge from the protocol transcript. This allows us to make the EEA proofs non-interactive. For convience we write $h_i = H(x_i \| ID_{\mathcal{C}})$. We need an additional hash function $H_1 \colon \mathbb{Z}_q^{*n} \times \mathbb{Z}_q^{*n} \to \mathbb{Z}_q^*$. Then the EEA proofs are as follows:

The prover: input: (x_i, r', h_i, $ID_{\mathcal{C}}$, H, H_1)

1) The prover samples a random element $z \leftarrow\$\ \mathbb{Z}_q^*$
2) Then they compute $t_i = h_i^z$ for all $i \in \mathcal{I}$.
3) The challenge c is $c = H_1(\{t_i, h_i\})$
4) Lastly, they compute $s = z + cr'$.
5) The prover provides for all $i \in \mathcal{I}$: s, t_i together with $\tilde{h}_i = h_i^{r'}$, x_i and $ID_{\mathcal{C}}$ to the verifier.

The verifier: input: (s, t_i, $\tilde{h}_i$, x_i, $ID_{\mathcal{C}}$, H, H_1)

1) First computes $h_i = H(x_i \| ID_{\mathcal{C}})$.
2) Then they compute $c = H_1(\{t_i, h_i\})$
3) Then the verifier checks for all $i \in \mathcal{I}$ whether $h_i^s = t_i \cdot \tilde{h}_i^c$

The verifier will approve the EEA proofs if all checks validate to true. The correctness of the EEA holds because

$$\begin{aligned} h_i^s &= t_i \cdot \tilde{h}_i^c \\ h_i^{z+cr'} &= h_i^z \cdot (h^{r'})_i^c \\ h_i^{z+cr'} &= h_i^z \cdot h_i^{cr'}. \end{aligned}$$

For a formal proof that this is Honest-Verifier Zero Knowledge and Knowledge Sound, we refer to Appendix A.1 of Falzon and Markatou [22]. Honest-Verifier Zero Knowledge implies that a verifier executing the protocol as prescribed cannot learn anything about the secret (in our case, r'). Knowledge soundness guarantees that the prover can only prove true statements.

In our setting, we required the EEA proofs to prove that all elements of the set $\{H(x_i)^{r'}\}_{i\in\mathcal{I}}$ are all raised to the same r'. For a semi-honest Receiver and Judge, the EEA proofs guarantee that this is true and do not leak any information beyond the truth of the statement.

E.3. Hiding Merkle tree

Using the extra binding explained before and the EEA proofs guarantees that the Merkle tree is hiding and binding. We will not provide formal proof of this statement, but

rather an intuition. The Honest-Verifier Zero-Knowledge property of the EEA proofs guarantees that the Merkle tree is hiding. The Knowledge soundness guarantees that the Merkle tree is binding.

With this extra layer of security, the Merkle tree is a hiding and binding vector commitment scheme. We can thus use it to instantiate the protocol of Figure 3. There are some slight additions and extra messages; we color these pink, and we provide the full protocol in Figure 12.

E.4. Comparison

In this section, we compare the runtime and bandwidth of our protocol using a Verkle and Merkle tree. The results are visible in Table 3. We conclude that for small values of n, ie, $n \leq 2^{16}$, and for $m \leq 2^{24}$ the Merkle tree outperforms the Verkle tree. This is because the Commit_{Merkle} is faster than the Commit_{Verkle}, thus the `Authentication_Check` the Sender has to perform is faster for small values of n. However, as m grows, the Sender needs to blind their elements with r', significantly slowing down their overall computation time.

As an additional note, the evaluation in Table 3 uses a percentage $p = 20\%$. For increasing p, we expect the Verkle tree to be faster, as shown in Fig. 7. We can conclude that for growing n, m, and p, the Verkle tree will outperform the Merkle tree.

We aim to provide a Partial-APSI protocol with reduced bandwidth compared to previous work, and we want it to scale well with increasing n and p. We therefore chose the Verkle tree as most suitable for our goal. However, for smaller n, m, and p, we suggest using the Merkle tree.

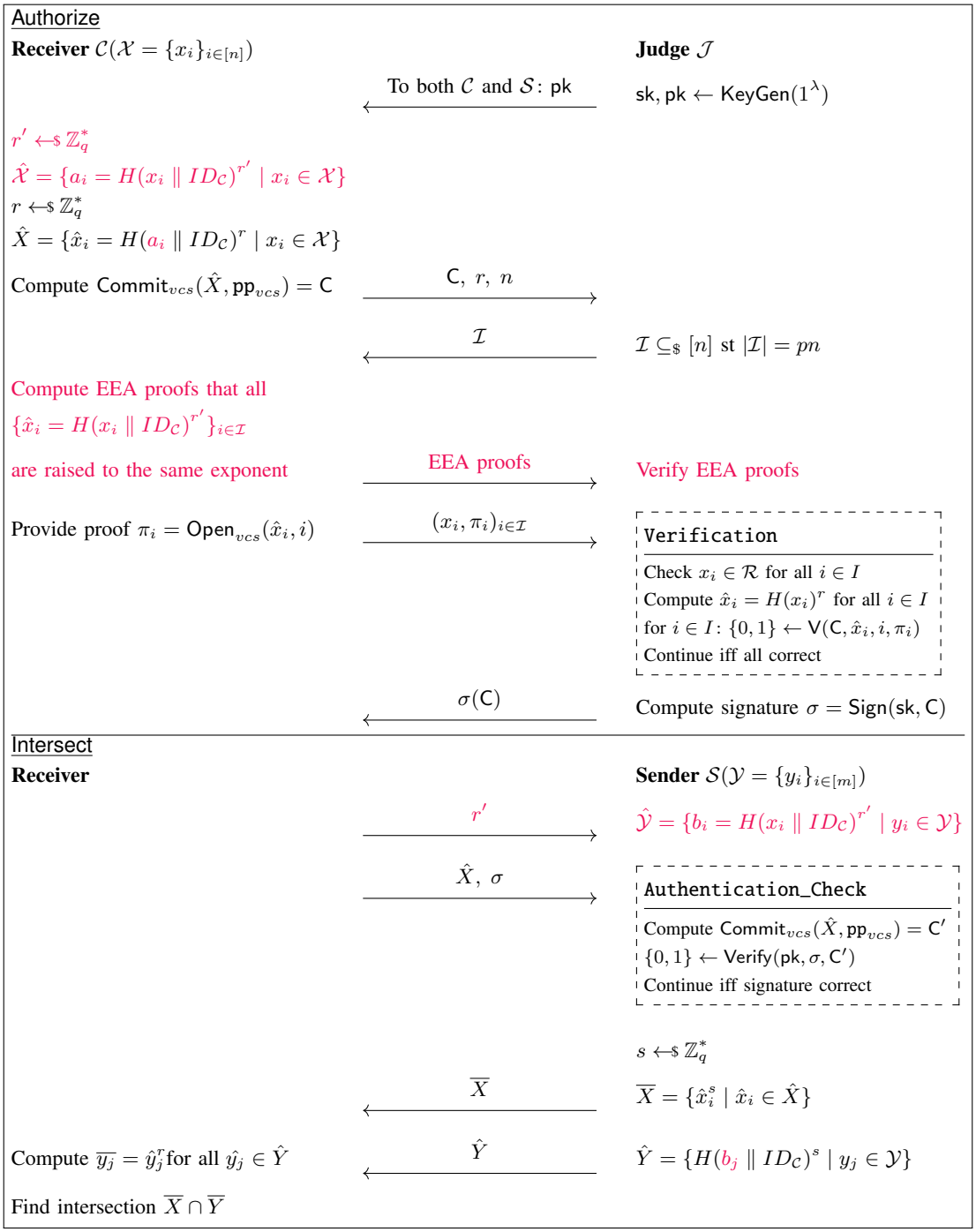


Figure 12. The Partially Authorized PSI protocol of Figure 3 instantiated with an extra blinding factor. This additional blinding is needed for non-hiding vector commitment schemes, such as a Merkle tree.

TABLE 3. COMPARISON OF OUR PAPSI PROTOCOL WITH A MERKLE AND VERKLE TREE ON AN AMD RYZEN THREADRIPPER 7970X 32-CORE CPU USING DATASETS OF IP ADDRESSES. THE PERCENTAGE $p = 20$. ALL NUMBERS ARE ROUNDED TO THE NEAREST INTEGER. WE HIGHLIGHT CASES WHERE THE VERKLE TREE OUTPERFORMS THE MERKLE TREE IN BLUE. WE TAKE THE AVERAGE OF 2 RUNS; SLIGHT DIFFERENCES CAN THEREFORE OCCUR WITH THOSE IN TABLE 2.

Set sizes		Scheme	Communcation (KB)		Runtime (ms)			Total Runtime			
m	n		Auth	Inter	$\mathcal{J}$	$\mathcal{C}$	$\mathcal{S}$	LAN	1Gbps	200Mbps	50Mbps
2^{10}	2^{10}	**Merkle**	21	12	0	1	0	1	1	1	1
		Verkle	9	12	13	21	15	49	49	49	49
	2^{16}	**Merkle**	1355	528	57	76	11	143	144	144	144
		Verkle	557	528	54	261	110	425	426	426	426
	2^{20}	**Merkle**	21652	8393	14503	12603	196	27302	27302	27303	27307
		Verkle	8907	8393	170	2283	908	3361	3361	3362	3364
	2^{24}	**Merkle**	346412	134222	3471833	3543796	3265	7018895	7018899	7018915	7018972
		Verkle	142498	134222	1493	35616	10890	47999	48001	48010	48044
2^{16}	2^{10}	**Merkle**	21	270	0	2	8	10	11	11	11
		Verkle	9	270	20	28	28	75	76	76	76
	2^{16}	**Merkle**	1352	786	66	75	18	159	159	159	159
		Verkle	557	786	37	241	95	373	373	373	373
	2^{20}	**Merkle**	21658	8651	13382	12502	193	26077	26078	26079	26083
		Verkle	8907	8651	166	2260	891	3317	3318	3318	3320
	2^{24}	**Merkle**	346436	134480	3114965	3092401	3219	6210586	6210590	6210605	6210663
		Verkle	142498	134480	1418	35541	10889	47848	47851	47860	47893
2^{20}	2^{10}	**Merkle**	22	4203	0	33	124	157	157	157	158
		Verkle	9	4203	14	57	93	164	165	165	165
	2^{16}	**Merkle**	1354	4719	66	112	134	312	312	313	313
		Verkle	557	4719	37	270	168	475	476	476	477
	2^{20}	**Merkle**	21646	12583	12183	14770	297	27250	27251	27252	27256
		Verkle	8907	12583	167	2294	969	3431	3431	3432	3435
	2^{24}	**Merkle**	346368	138412	3380588	3090520	3349	6474457	6474461	6474476	6474534
		Verkle	142499	138412	1439	35665	11116	48219	48222	48231	48265
2^{24}	2^{10}	**Merkle**	22	67117	1798	1260	2080	5138	5139	5141	5149
		Verkle	9	67117	14	1423	1248	2685	2685	2688	2696
	2^{16}	**Merkle**	1352	67633	65	1347	2039	3451	3452	3455	3463
		Verkle	557	67633	36	1629	1308	2973	2974	2977	2985
	2^{20}	**Merkle**	21649	75498	13339	14896	2208	30443	30445	30448	30459
		Verkle	8907	75498	166	3673	2090	5928	5929	5932	5942
	2^{24}	**Merkle**	346385	201327	3374741	3214353	5292	6594386	6594390	6594408	6594474
		Verkle	142498	201327	1486	37481	12098	51064	51067	51078	51120