An Algebraic Approach to Maliciously Secure Private Set Intersection

Satrajit Ghosh[1⋆] and Tobias Nilges[2⋆⋆]

[1] Department of Computer Science, Aarhus University
[2] ITK Engineering GmbH

Abstract. Private set intersection (PSI) is an important area of research and has been the focus of many works over the past decades. It describes the problem of finding an intersection between the input sets of at least two parties without revealing anything about the input sets apart from their intersection.

In this paper, we present a new approach to compute the intersection between sets based on a primitive called Oblivious Linear Function Evaluation (OLE). On an abstract level, we use this primitive to efficiently add two polynomials in a randomized way while preserving the roots of the added polynomials. Setting the roots of the input polynomials to be the elements of the input sets, this directly yields an intersection protocol with optimal asymptotic communication complexity $O(m\kappa)$. We highlight that the protocol is information-theoretically secure against a malicious adversary assuming OLE.

We also present a natural generalization of the 2-party protocol for the fully malicious multi-party case. Our protocol does away with expensive (homomorphic) threshold encryption and zero-knowledge proofs. Instead, we use simple combinatorial techniques to ensure the security. As a result we get a UC-secure protocol with asymptotically optimal communication complexity $O((n^2+nm)\kappa)$, where n is the number of parties, m is the set size and κ is the security parameter. Apart from yielding an asymptotic improvement over previous works, our protocols are also conceptually simple and require only simple field arithmetic. Along the way we develop techniques that might be of independent interest.

1 Introduction

Private set intersection (PSI) has been the focus of research for decades and describes the following basic problem. Two parties, Alice and Bob, each have a

⋆ Supported by the European Union's Horizon 2020 research and innovation programme under grant agreement #669255 (MPCPRO), the European Union's Horizon 2020 research and innovation programme under grant agreement #731583 (SODA) and the Independent Research Fund Denmark project BETHE.

⋆⋆ Part of the research leading to these results was done while the author was at Aarhus University. Supported by the European Union's Horizon 2020 research and innovation programme under grant agreement #669255 (MPCPRO).

set S_{A} and S_{B}, respectively, and want to find the intersection $S_{\cap} = S_{\mathsf{A}} \cap S_{\mathsf{B}}$ of their sets. This problem is non-trivial if both parties must not learn anything but the intersection. There are numerous applications for PSI from auctions [29] over advertising [32] to proximity testing [30].

Over the years several techniques for two-party PSI have been proposed, which can be roughly placed in four categories: constructions built from specific number-theoretic assumptions [38,28,21,8,9,23], using garbled circuits [20,32], based on oblivious transfer (OT) [10,33,34,31,26,35,36] and based on oblivious polynomial evaluation (OPE) [13,7,18,17,12]. There also exists efficient PSI protocols in server-aided model [24].

Some of these techniques translate to the multi-party setting. The first (passively secure) multi-party PSI (MPSI) protocol was proposed by Freedman et al. [13] based on OPE and later improved in a series of works [25,37,5] to achieve full malicious security. Recently, Hazay and Venkitasubramaniam [19] proposed new protocols secure against semi-honest and fully malicious adversaries. They improve upon the communication efficiency of previous works by designating a central party that runs a version of the protocol of [13] with all other parties and aggregates the results.

Given the state of the art, it remains an open problem to construct a protocol with asymptotically optimal communication complexity in the fully malicious multi-party setting. The main reason for this is the use of zero-knowledge proofs and expensive checks in previous works, which incur an asymptotic overhead over passively secure solutions.

In a concurrent and independent work, Kolesnikov et al. [27] presented a new paradigm for solving the problem of MPSI from oblivious programmable pseudorandom functions (OPPRF). Their approach yields very efficient protocols for multi-party PSI, but the construction achieves only passive security against $n-1$ corruptions. However, their approach to aggregate the intermediate results uses ideas similar to our masking scheme in the multi-party case.

1.1 Our Contribution

We propose a new approach to (multi-party) private set intersection based on oblivious linear function evaluation (OLE). OLE allows two mutually distrusting parties to evaluate a linear function $ax + b$, where the sender knows a and b, and the receiver knows x. Nothing apart from the result $ax + b$ is learned by the receiver, and the sender learns nothing about x. OLE can be instantiated in the OT-hybrid model from a wide range of assumptions with varying communication efficiency, like LPN [1], Quadratic/Composite Residuosity [22] and Noisy Encodings [22,14], or even unconditionally [22].

Our techniques differ significantly from previous works and follow a new paradigm which leads to conceptually simple and very efficient protocols. Our approach is particularly efficient if all input sets are of similar size. To showcase the benefits of our overall approach, we also describe how our MPSI protocol can be modified into a threshold MPSI protocol.

Concretely, we achieve the following:

- Two-party PSI with communication complexity $O(m\kappa)$ and computational cost of $O(m \log m)$ multiplications. The protocol is information theoretically secure against a malicious adversary in the OLE-hybrid model.
- UC-secure Multi-party PSI in fully malicious setting with communication complexity $O((n^2 + nm)\kappa)$ and computational complexity dominated by $O(nm \log m)$ multiplications for the central party and $O(m \log m)$ multiplications for other parties.
- A simple extension of the multi-party PSI protocol to threshold PSI, with the same complexity. To the best of our knowledge, this is the first actively secure threshold multi-party PSI protocol.[3]

In comparison to previous works which rely heavily on exponentiations in fields or groups, our protocols require only field addition and multiplication (and OWF in the case of MPSI).

If we compare our result with the asymptotically optimal 2-party PSI protocols by [8,23], which have linear communication and computation, our first observation is that although they only have a linear number of modular exponentiations, the number of field operations is not linear but rather in the order of $O(m\kappa)$, and further they need a ZK proof in the ROM for each exponentiation, which is also expensive. Additionally, their result is achieved with specific number-theoretic assumptions, so the parameter sizes are probably not favourable compared to our protocol, and the construction is not black-box. We provide a detailed comparison of the concrete efficiency of our result with the recent protocol by Rindal and Rosulek [36], which has very good concrete efficiency.

In the setting of MPSI, our techniques result in asymptotic efficiency improvements over previous works in both communication and computational complexity (cf. Table 1).

We want to emphasize that our efficiency claims hold *including* the communication and computation cost for the OLE, if the recent instantiation by Ghosh et al. [14] is used, which is based on noisy Reed-Solomon codes. This OLE protocol has a constant communication overhead per OLE if instantiated with an efficient OT-extension protocol like [31] and therefore does not influence the asymptotic efficiency of our result.

Our results may seem surprising in light of the information-theoretic lower bound of $O(n^2 m\kappa)$ in the communication complexity for multi-party PSI in the fully malicious UC setting. We circumvent this lower bound by considering a slightly modified ideal functionality, resulting in a UC-secure solution for multi-party PSI with asymptotically optimal communication overhead. By asymptotically optimal, we mean that our construction matches the optimal bounds in the plain model for $m > n$, even for passive security, where n is the number of parties, m is the size of the sets and κ is the security parameter. All of our protocols work over fields $\mathbb{F}$ that are exponential in the size of the security parameter κ.

[3] Please see the full version of the paper [15].

[4] Our protocol is UC-secure in the fully malicious setting.

Protocol	Tools	Communication	Computation	Corruptions	Security
[27]	OPPRF	$O(nm\kappa)$	$O(n\kappa^2)$	$n-1$	semi-honest
[19]	THE	$O(nm\kappa)$	$O(nm \log m\kappa)$	$n-1$	semi-honest
[25]	THE, ZK	$O(n^2m^2\kappa)$	$O(n^2m + nm^2\kappa)$	$n-1$	malicious
[5]	THE, ZK	$O(n^2m\kappa)$	$O(n^2m + nm\kappa)$	$t < n/2$	malicious
[19]	CRS, THE	$O((n^2 + nm \log m)\kappa)$	$O(m^2\kappa)$	$n-1$	malicious
Ours+[14]	OLE	$O((n^2 + nm)\kappa)$	$O(nm \log m)$	$n-1$	malicious[4]

Table 1. Comparison of multi-party PSI protocols, where n is the number of parties, m the size of the input set and κ a security parameter. Here, THE denotes a threshold homomorphic encryption scheme, CRS a common reference string and OPPRF an oblivious programmable PRF. The computational cost is measured in terms of number of multiplications. Some of the protocols perform better if the sizes of the input sets differ significantly, or particular domains for inputs are used. The overhead described here assumes sets of similar size, with κ bit elements.

We believe that our approach provides an interesting alternative to existing solutions and that the techniques which we developed can find application in other settings as well.

1.2 Technical Overview

Abstractly, we let both parties encode their input set as a polynomial, such that the roots of the polynomials correspond to the inputs. This is a standard technique, but usually the parties then use OPE to obliviously evaluate the polynomials or some form of homomorphic encryption. Instead, we devise an OLE-based construction to add the two polynomials in an oblivious way, which results in an intersection polynomial. Kissner and Song [25] also create an intersection polynomial similar to ours, but encrypted under a layer of homomorphic encryption, whereas our technique results in a *plain* intersection polynomial. Since the intersection polynomial already hides everything but the intersection, one could argue that the layer of encryption in [25] incurs *additional overhead* in terms of expensive computations and complex checks.

In our case, both parties simply evaluate the intersection polynomial on their input sets and check if it evaluates to 0. This construction is information-theoretically secure in the OLE-hybrid model and requires only simple field operations. Conceptually, we compute the complete intersection in one step. In comparison to the naive OPE-based approach[5], our solution directly yields an asymptotic communication improvement in the input size. Another advantage is that our approach generalizes to the multi-party setting.

We start with a detailed overview of our constructions and technical challenges.

Oblivious polynomial addition from OLE. Intuitively, OLE is the generalization of OT to larger fields, i.e. it allows a sender and a receiver to compute

[5] Here we mean an OPE is used for each element of the receiver's input set. This can be circumvented by clever hashing strategies, e.g. [13,19].

a linear function $c(x) = ax + b$, where the sender holds a, b and the receiver inputs x and obtains c. OLE guarantees that the receiver learns nothing about a, b except for the result c, while the sender learns nothing about x.

Based on this primitive, we define and realize a functionality OPA that allows to add two polynomials in such a way that the receiver cannot learn the sender's input polynomial, while the sender learns nothing about the receiver's polynomial or the output. We first describe a passively secure protocol. Concretely, assume that the sender has an input polynomial $\mathbf{a}$ of degree $2d$, and the receiver has a polynomial $\mathbf{b}$ of degree d. The sender additionally draws a uniformly random polynomial $\mathbf{r}$ of degree d. Both parties point-wise add and multiply their polynomials, i.e. they evaluate their polynomials over a set of $2d + 1$ distinct points $\alpha_1, \ldots, \alpha_{2d+1}$, resulting in $a_i = \mathbf{a}(\alpha_i), b_i = \mathbf{b}(\alpha_i)$ and $r_i = \mathbf{r}(\alpha_i)$ for $i \in [2d + 1]$. Then, for each of $2d + 1$ OLEs, the sender inputs r_i, a_i, while the receiver inputs b_i and thereby obtains $c_i = r_i b_i + a_i$. The receiver interpolates the polynomial $\mathbf{c}$ from the $2d + 1$ (α_i, c_i) and outputs it. Since we assume semi-honest behaviour, the functionality is realized by this protocol.

The biggest hurdle in achieving active security for the above protocol lies in ensuring a non-zero $\mathbf{b}$ and $\mathbf{r}$ as input. Otherwise, e.g. if $\mathbf{b} = 0$, the receiver could learn $\mathbf{a}$. One might think that it is sufficient to perform a coin-toss and verify that the output satisfies the supposed relation, i.e. pick a random x, compute $\mathbf{a}(x), \mathbf{b}(x), \mathbf{r}(x)$ and $\mathbf{c}(x)$ and everyone checks if $\mathbf{b}(x)\mathbf{r}(x) + \mathbf{a}(x) = \mathbf{c}(x)$ and if $\mathbf{b}(x), \mathbf{r}(x)$ are non-zero [6]. For $\mathbf{r}(x) \neq 0$, the check is actually sufficient, because $\mathbf{r}$ must have degree at most d, otherwise the reconstruction fails, and only d points of $\mathbf{r}$ can be zero ($\mathbf{r} = 0$ would require $2d+1$ zero inputs). For $\mathbf{b} \neq 0$, however, just checking for $\mathbf{b}(x) \neq 0$ is not sufficient, because at this point, even if the input $\mathbf{b} \neq 0$, the receiver can input d zeroes in the OLE, which in combination with the check is sufficient to learn $\mathbf{a}$ completely. We resolve this issue by constructing an enhanced OLE functionality which ensures that the receiver input is non-zero. We believe that this primitive is of independent interest and describe it in more detail later in this section.

Two-party PSI from OLE. Let us first describe a straightforward two-party PSI protocol with one-sided output from the above primitive. Let S_A and S_B denote the inputs for Alice and Bob, respectively, where $|S_\mathsf{P}| = m$. Assuming that Bob is supposed to get the intersection, they pick random $\mathbf{p}_\mathsf{A}$ and $\mathbf{p}_\mathsf{B}$ with the restriction that $\mathbf{p}_\mathsf{P}(\gamma) = 0$ for $\gamma \in S_\mathsf{P}$. As they will use OPA, $\deg \mathbf{p}_\mathsf{A} = 2m$, while $\deg \mathbf{p}_\mathsf{B} = m$. Further, Alice picks a uniformly random polynomial $\mathbf{r}_\mathsf{A}$ of degree m and inputs $\mathbf{p}_\mathsf{A}, \mathbf{r}_\mathsf{A}$ into OPA. Bob inputs $\mathbf{p}_\mathsf{B}$, obtains $\mathbf{p}_\cap = \mathbf{p}_\mathsf{A} + \mathbf{p}_\mathsf{B}\mathbf{r}_\mathsf{A}$ and outputs all $\gamma_j \in S_\mathsf{B}$ for which $\mathbf{p}_\cap(\gamma_j) = 0$. Obviously, $\mathbf{r}_\mathsf{A}$ does not remove any of the roots of $\mathbf{p}_\mathsf{B}$, and therefore all points γ where $\mathbf{p}_\mathsf{B}(\gamma) = 0 = \mathbf{p}_\mathsf{A}(\gamma)$ remain in $\mathbf{p}_\cap$.

However, as a stepping stone for multi-party PSI, we are more interested in protocols that provide output to both parties. If we were to use the above protocol and simply announce $\mathbf{p}_\cap$ to Alice, then Alice could learn Bob's input.

[6] Since this check leaks some information about the inputs, we have to perform the check in a secure manner.

Therefore we have to take a slightly different approach. Let $\mathbf{u}_\mathsf{A}$ be an additional random polynomial chosen by Alice. Instead of using her own input in the OPA, Alice uses $\mathbf{r}_\mathsf{A}, \mathbf{u}_\mathsf{A}$, which gives $\mathbf{s}_\mathsf{B} = \mathbf{u}_\mathsf{A} + \mathbf{p}_\mathsf{B}\mathbf{r}_\mathsf{A}$ to Bob. Then they run another OPA in the other direction, i.e. Bob inputs $\mathbf{r}_\mathsf{B}, \mathbf{u}_\mathsf{B}$ and Alice $\mathbf{p}_\mathsf{A}$. Now, both Alice and Bob have a randomized "share" of the intersection, namely $\mathbf{s}_\mathsf{A}$ and $\mathbf{s}_\mathsf{B}$, respectively. Adding $\mathbf{s}_\mathsf{A}$ and $\mathbf{s}_\mathsf{B}$ yields a masked but correct intersection. We still run into the problem that sending either $\mathbf{s}_\mathsf{B}$ to Alice or $\mathbf{s}_\mathsf{A}$ to Bob allows the respective party to learn the other party's input. We also have to use additional randomization polynomials $\mathbf{r}'_\mathsf{A}, \mathbf{r}'_\mathsf{B}$ to ensure privacy of the final result.

Our solution is to simply use the masks $\mathbf{u}$ to enforce the addition of the two shares. Let us fix Alice as the party that combines the result. Bob computes $\mathbf{s}'_\mathsf{B} = \mathbf{s}_\mathsf{B} - \mathbf{u}_\mathsf{B} + \mathbf{p}_\mathsf{B}\mathbf{r}'_\mathsf{B}$ and sends it to Alice. Alice computes $\mathbf{p}_\cap = \mathbf{s}'_\mathsf{B} + \mathbf{s}_\mathsf{A} - \mathbf{u}_\mathsf{A} + \mathbf{p}_\mathsf{A}\mathbf{r}'_\mathsf{A}$. This way, the only chance to get rid of the blinding polynomial $\mathbf{u}_\mathsf{B}$ is to add both values. But since each input is additionally randomized via the $\mathbf{r}$ polynomials, Alice cannot subtract her own input from the sum. Since the same also holds for Bob, Alice simply sends the result to Bob.

The last step is to check if the values that are sent and the intersection polynomial are consistent. We do this via a simple coin-toss for a random x, and the parties evaluate their inputs on x and can abort if the relation $\mathbf{p}_\cap = \mathbf{p}_\mathsf{B}(\mathbf{r}_\mathsf{A} + \mathbf{r}'_\mathsf{B}) + \mathbf{p}_\mathsf{A}(\mathbf{r}'_\mathsf{A} + \mathbf{r}_\mathsf{B})$ does not hold, i.e. $\mathbf{p}_\cap$ is computed incorrectly. This type of check enforces semi-honest behaviour, and was used previously e.g. in [2].

A note on the MPSI functionality. We show that by slightly modifying the ideal functionality for multi-party PSI we get better communication efficiency, without compromising the security at all. A formal definition is given in Section 6.1. Typically, it is necessary for the simulator to extract *all* inputs from the malicious parties, input them into the ideal functionality, and then continue the simulation with the obtained ideal intersection. In a fully malicious setting, however, this requires every party to communicate in $O(m\kappa)$ with every other party—otherwise the input is information-theoretically undetermined and cannot be extracted—which results in $O(n^2 m\kappa)$ communication complexity.

The crucial observation here is that in the setting of multi-party PSI, an intermediate intersection between a single malicious party and *all* honest parties is sufficient for simulation. This is due to the fact that inputs by additional malicious parties can only reduce the size of the intersection, and as long as we observe the additional inputs at some point, we can correctly reduce the intersection in the ideal setting before outputting it. On a technical level, we no longer need to extract all malicious inputs right away to provide a correct simulation of the intersection. Therefore, it is not necessary for every party to communicate in $O(m\kappa)$ with every other party. Intuitively, the intermediate intersection corresponds to the case where all malicious parties have the same input. We therefore argue that the security of this modified setting is identical to standard MPSI up to input substitution of the adversary.[7]

[7] Our multi-party PSI functionality uses similar idea as augmented semi-honest multi-party PSI as in previous works [27].

Multi-party PSI. The multi-party protocol is a direct generalization of the two-party protocol, with some small adjustments. We consider a network with a star topology, similar to the recent result of [19]. One party is set to be the central party, and all other parties (mainly) interact with this central party to compute the result. The main idea here is to delegate most of the work to the central party, which in turn allows to reduce the communication complexity. Since no party is supposed to get any intermediate intersections, we let each party create an additive sharing of their intersection with the central party.

First, consider the following (incorrect) toy example. Let each party P_i execute the two-party PSI as described above with P_0, up to the point where both parties have shares $\mathbf{s}^i_{P_0}, \mathbf{s}'_{P_i}$. All parties P_i send their shares $\mathbf{s}'_{P_i}$ to P_0, who adds all polynomials and broadcasts the output. By design of the protocols and the inputs, this yields the intersection of all parties. Further, the communication complexity is in $O(nm\kappa)$, which is optimal. However, this protocol also allows P_0 to learn all intermediate intersections with the other parties, which is not allowed. Previously, all maliciously secure multi-party PSI protocols used threshold encryption to solve this problem, and indeed it might be possible to use a similar approach to ensure active security for the above protocol. For example, a homomorphic threshold encryption would allow to add all these shares homomorphically, without leaking the intermediate intersections. But threshold encryption incurs a significant computational overhead (and increases the complexity of the protocol and its analysis) which we are unwilling to pay.

Instead, we propose a new solution which is conceptually very simple. We add another layer of masking on the shares $\mathbf{s}_{P_i}$, which forces P_0 to add *all* intermediate shares—at least those of the honest parties. For this we have to ensure that the communication complexity does not increase, so all parties exchange seeds (instead of sending random polynomials directly), which are used in a PRG to mask the intermediate intersections. This technique is somewhat reminiscent of the pseudorandom secret-sharing technique by Cramer et al. [6]. We emphasize that we do not need any public key operations.

Concretely, all parties exchange a random seed and use it to compute a random polynomial in such a way that every pair of parties P_i, P_j holds two polynomials $\mathbf{v}_{ij}, \mathbf{v}_{ji}$ with $\mathbf{v}_{ij} + \mathbf{v}_{ji} = 0$. Then, instead of sending $\mathbf{s}'_{P_i}$, each party P_i computes $\mathbf{s}''_{P_i} = \mathbf{s}'_{P_i} + \sum \mathbf{v}_{ij}$ and sends this value. If P_0 obtains this value, it has to add the values $\mathbf{s}''_{P_i}$ of all parties to remove the masks, otherwise $\mathbf{s}''_{P_i}$ will be uniformly random.

Finally, to ensure that the central party actually computed the aggregation, we add a check similar to two-party PSI, where the relation, i.e. computing the sum, is verified by evaluating the inputs on a random value x which is obtained by a multi-party coin-toss.

Threshold (M)PSI. We defer the threshold extensions to the full version of this paper [15] and only give a very brief technical overview.

First of all, we clarify the term *threshold PSI*. We consider the setting where all parties have m elements as their input, and the output is only revealed if the intersection of the inputs among all parties is bigger than a certain threshold

ℓ. In [16] Hallgren et al. defined this notion for two party setting, and finds application whenever two entities are supposed to be matched once a certain threshold is reached, e.g. for dating websites or ride sharing.

We naturally extend the idea of threshold PSI from [16] to the multi-party setting and propose the first actively secure threshold multi-party PSI protocol. On a high level, our solution uses a similar idea to [16], but we use completely different techniques and achieve stronger security and better efficiency. The main idea is to use a robust secret sharing scheme, and the execution of the protocol basically transfers a subset of these shares to the other parties, one share for each element in the intersection. If the intersection is large enough, the parties can reconstruct the shared value.

Specifically, the trick is to modify the input polynomials of each party P_i for the MPSI protocol and add an additional check. Instead of simply setting $\mathbf{p}_i$ such that $\mathbf{p}_i(\gamma_j) = 0$ for all $\gamma_j \in S_i$, we set $\tilde{\mathbf{p}}_i(\gamma_j) = 1$. Further, for each of the random polynomials $\tilde{\mathbf{r}}_i, \tilde{\mathbf{r}}'_i$ we set $\tilde{\mathbf{r}}_i(\gamma_j) = \rho_j$ and $\tilde{\mathbf{r}}'_i(\gamma_j) = \rho'_j$, where $\rho_1, \ldots, \rho_n$, $\rho'_1, \ldots, \rho'_n$ are the shares of two robust (ℓ, n)-secret sharings of random values s_i^0 and s_i^1, respectively. Now, by computing the modified intersection polynomial $\tilde{\mathbf{p}}_\cap$ as before, each party obtains exactly $m_\cap = |S_\cap|$ shares, one for each $\gamma_j \in S_i$.

Now if $m_\cap \geq \ell$ then each party can reconstruct $r_\cap = \sum_{i=1}^{n} (s_i^0 + s_i^1)$. otherwise the intersection remains hidden completely. We omitted some of the details due to the space constraints and refer to the full version [15].

A New Flavour of OLE. One of the main technical challenges in constructing our protocols is to ensure a non-zero input into the OLE functionality by the receiver. Recall that an OLE computes a linear function $ax + b$. We define an enhanced OLE functionality (cf. Section 3) which ensures that $x \neq 0$, otherwise the output is uniformly random. Our protocol which realises this functionality makes two black-box calls to a normal OLE and is otherwise purely algebraic.

Before we describe the solution, let us start with a simple observation. If the receiver inputs $x = 0$, an OLE returns the value b. Therefore, it is critical that the receiver cannot force the protocol to output b. One way to achieve this is by forcing the receiver to multiply b with some correlated value via an OLE, let's call it $\hat{x}$. Concretely, we can use an OLE where the receiver inputs $\hat{x}$ and a random s, while the sender inputs b and obtains $\hat{x}b + s$. Now if the sender uses $a + b\hat{x} + s, 0$ as input for another OLE, where the receiver inputs x, the receiver obtains $ax + b\hat{x}x + sx$. Which means that if $\hat{x} = x^{-1}$ then the receiver can extract the correct output. This looks like a step in the right direction, since for $x = 0$ or $\hat{x} = 0$, the output would not be b. On the other hand, the receiver can now force the OLE to output a by choosing $\hat{x} = 0$ and $x = 1$, so maybe we only shifted the problem.

The final trick lies in masking the output such that it is uniform for inconsistent inputs $x, \hat{x}$. We do this by splitting b into two shares that only add to b if $x \cdot \hat{x} = 1$. The complete protocol looks like this: the receiver plays the sender for one OLE with input x^{-1}, s, and the sender inputs a random u to obtain $t = x^{-1}u + s$. Then the sender plays the sender for the second OLE and inputs $t + a, b - u$, while the receiver inputs x and obtains $c' = (t + a)x + b - u =$

$ux^{-1}x + sx + ax + b - u = ax + b + sx$, from which the receiver can subtract sx to get the result. A cheating receiver with inconsistent input $x^*, \hat{x}^*$ will get $ax + b + u(x^*\hat{x}^* - 1)$ as an output, which is uniform over the choice of u.

2 Preliminaries

We assume $|\mathbb{F}| \in \theta(2^\kappa)$, where κ is a statistical security parameter. Typically, $x \in \mathbb{F}$ denotes a field element, while $\mathbf{p} \in \mathbb{F}[X]$ denotes a polynomial. Let $\mathcal{M}_0(\mathbf{p})$ denote the zero-set for $\mathbf{p} \in \mathbb{F}[X]$, i.e. $\forall x \in \mathcal{M}_0(\mathbf{p}), \mathbf{p}(x) = 0$.

In the proofs, $\hat{x}$ denotes an element either extracted or simulated by the simulator, while x^* denotes an element sent by the adversary.

We slightly abuse notation and denote by $\mathbf{v} = \mathsf{PRG}_d(s)$ the deterministic pseudorandom polynomial of degree d derived from evaluating PRG on seed s.

2.1 Security Model

We prove our protocol in the Universal Composability (UC) framework [4]. In the framework, security of a protocol is shown by comparing a real protocol π in the real world with an ideal functionality $\mathcal{F}$ in the ideal world. $\mathcal{F}$ is supposed to accurately describe the security requirements of the protocol and is secure per definition. An environment $\mathcal{Z}$ is plugged either to the real protocol or the ideal protocol and has to distinguish the two cases. For this, the environment can corrupt parties. To ensure security, there has to exist a simulator in the ideal world that produces a protocol transcript indistinguishable from the real protocol, even if the environment corrupts a party. We say π UC-realises $\mathcal{F}$ if for all adversaries $\mathcal{A}$ in the real world there exists a simulator $\mathcal{S}$ in the ideal world such that all environments $\mathcal{Z}$ cannot distinguish the transcripts of the parties' outputs.

Oblivious Linear Function Evaluation. Oblivious Linear Function Evaluation (OLE) is the generalized version of OT over larger fields. The sender has as input two values $a, b \in \mathbb{F}$ that determine a linear function $f(x) = a \cdot x + b$ over $\mathbb{F}$, and the receiver gets to obliviously evaluate the linear function on input $x \in \mathbb{F}$. The receiver will learn only $f(x)$, and the sender learns nothing at all. The ideal functionality is shown in Figure 1.

2.2 Technical Lemmas

We state several lemmas which are used to show the correctness of our PSI protocols later on.

Lemma 2.1. *Let $\mathbf{p}, \mathbf{q} \in \mathbb{F}[X]$ be non-trivial polynomials. Then,*

$$\mathcal{M}_0(\mathbf{p}) \cap \mathcal{M}_0(\mathbf{p} + \mathbf{q}) = \mathcal{M}_0(\mathbf{p}) \cap \mathcal{M}_0(\mathbf{q}) = \mathcal{M}_0(\mathbf{q}) \cap \mathcal{M}_0(\mathbf{p} + \mathbf{q}).$$

This lemma shows that the sum of two polynomials contains the intersection with respect to the zero-sets of both polynomials.

Functionality $\mathcal{F}_{\text{OLE}}$

1. Upon receiving a message $(\texttt{inputS}, (a, b))$ from the sender with $a, b \in \mathbb{F}$, verify that there is no stored tuple, else ignore that message. Store a and b and send a message $(\texttt{input})$ to $\mathcal{A}$.
2. Upon receiving a message $(\texttt{inputR}, x)$ from the receiver with $x \in \mathbb{F}$, verify that there is no stored tuple, else ignore that message. Store x and send a message $(\texttt{input})$ to $\mathcal{A}$.
3. Upon receiving a message $(\texttt{deliver}, \mathsf{S})$ from $\mathcal{A}$, check if both (a, b) and x are stored, else ignore that message. Send $(\texttt{delivered})$ to the sender.
4. Upon receiving a message $(\texttt{deliver}, \mathsf{R})$ from $\mathcal{A}$, check if both (a, b) and x are stored, else ignore that message. Set $y = a \cdot x + b$ and send $(\texttt{output}, y)$ to the receiver.

Fig. 1. Ideal functionality for oblivious linear function evaluation.

Proof. Let $\mathcal{M}_\cap = \mathcal{M}_0(\mathbf{p}) \cap \mathcal{M}_0(\mathbf{q})$.

"$\supseteq$": $\forall x \in \mathcal{M}_\cap$: $\mathbf{p}(x) = \mathbf{q}(x) = 0$. But $\mathbf{p}(x) + \mathbf{q}(x) = 0$, so $x \in \mathcal{M}_0(\mathbf{p} + \mathbf{q})$.

"$\subseteq$": It remains to show that there is no x such that $x \in \mathcal{M}_0(\mathbf{p}) \cap \mathcal{M}_0(\mathbf{p}+\mathbf{q})$ but $x \notin \mathcal{M}_\cap$, i.e. $\mathcal{M}_0(\mathbf{p}) \cap (\mathcal{M}_0(\mathbf{p}+\mathbf{q}) \setminus \mathcal{M}_\cap) = \emptyset$. Similarly, $\mathcal{M}_0(\mathbf{q}) \cap (\mathcal{M}_0(\mathbf{p}+\mathbf{q}) \setminus \mathcal{M}_\cap) = \emptyset$.

Assume for the sake of contradiction that $\mathcal{M}_0(\mathbf{p}) \cap (\mathcal{M}_0(\mathbf{p}+\mathbf{q}) \setminus \mathcal{M}_\cap) \neq \emptyset$. Let $x \in \mathcal{M}_0(\mathbf{p}) \cap (\mathcal{M}_0(\mathbf{p}+\mathbf{q}) \setminus \mathcal{M}_\cap)$. Then, $\mathbf{p}(x) = 0$, but $\mathbf{q}(x) \neq 0$, otherwise $x \in \mathcal{M}_\cap$. But this means that $\mathbf{p}(x) + \mathbf{q}(x) \neq 0$, i.e. $x \notin \mathcal{M}_0(\mathbf{p}+\mathbf{q})$. This contradicts our assumption, and we get that $\mathcal{M}_0(\mathbf{p}) \cap (\mathcal{M}_0(\mathbf{p}+\mathbf{q}) \setminus \mathcal{M}_\cap) = \emptyset$.

Symmetrically, we get that $\mathcal{M}_0(\mathbf{q}) \cap (\mathcal{M}_0(\mathbf{p}+\mathbf{q}) \setminus \mathcal{M}_\cap) = \emptyset$. The claim follows. □

Lemma 2.2. *Let $d \in \mathsf{poly}(\log|\mathbb{F}|)$. Let $\mathbf{p} \in \mathbb{F}[X]$, $\deg(\mathbf{p}) = d$ be a non-trivial random polynomial with $\Pr[x \in \mathcal{M}_0(\mathbf{p})] \leq \mathsf{negl}(|\mathbb{F}|)$ for all x. Then, for all $\mathbf{q}_1, \ldots, \mathbf{q}_l \in \mathbb{F}[X]$ with $\deg(\mathbf{q}_i) \leq d$,*

$$\Pr[(\mathcal{M}_0(\mathbf{p}) \cap \mathcal{M}_0(\sum_{i=1}^{l} \mathbf{q}_i + \mathbf{p})) \neq (\mathcal{M}_0(\mathbf{p}) \cap \bigcap_{i=1}^{l} \mathcal{M}_0(\mathbf{q}_i))] \leq \mathsf{negl}(|\mathbb{F}|).$$

This lemma is basically an extension of Lemma 2.1 and shows that the sum of several polynomials does not create new elements in the intersection unless the supposedly unknown zero-set of $\mathbf{p}$ can be guessed with non-negligible probability.

Proof. "$\subseteq$": We first observe that $\bigcap_{i=1}^{l} \mathcal{M}_0(\mathbf{q}_i) \subseteq \mathcal{M}_0(\sum_{i=1}^{l} \mathbf{q}_i)$: it holds that for all $x \in \bigcap_{i=1}^{l} \mathcal{M}_0(\mathbf{q}_i)$, $\mathbf{q}_i(x) = 0$ for $i \in [l]$. It follows that $\sum_{i=1}^{l} \mathbf{q}_i(x) = 0$, i.e. $x \in \mathcal{M}_0(\sum_{i=1}^{l} \mathbf{q}_i)$.

"$\supseteq$": Assume for the sake of contradiction that

$$(\mathcal{M}_0(\mathbf{p}) \cap \mathcal{M}_0(\sum_{i=1}^{l} \mathbf{q}_i) + \mathbf{p}) \neq (\mathcal{M}_0(\mathbf{p}) \cap \bigcap_{i=1}^{l} \mathcal{M}_0(\mathbf{q}_i))$$

with non-negligible probability ϵ. Let $\mathcal{M} = \mathcal{M}_0(\sum_{i=1}^{l} \mathbf{q}_i + \mathbf{p}) \setminus \bigcap_{i=1}^{l} \mathcal{M}_0(\mathbf{q}_i)$.

Then with probability at least ϵ, the set $\mathcal{M}$ is not empty. Further, we can bound $|\mathcal{M}| \leq d$. Pick a random $x \in \mathcal{M}$. It now holds that $\Pr[x \in \mathcal{M}_0(\mathbf{p})] \geq \epsilon/d$, which directly contradicts our assumption that for an unknown $\mathbf{p}$ the probability of guessing $x \in \mathcal{M}_0(\mathbf{p})$ is negligible over choice of $\mathbf{p}$. The claim follows. □

Lemma 2.3. *Let $d, d' \in \mathsf{poly}(\log|\mathbb{F}|)$. Let $\mathbf{r} \in \mathbb{F}[X]$, $\deg(\mathbf{r}) = d$ be a uniformly random polynomial. For all non-trivial $\mathbf{p} \in \mathbb{F}[X]$, $\deg(\mathbf{p}) = d'$,*

$$\Pr_{\mathbf{r}\in\mathbb{F}[X]}[(\mathcal{M}_0(\mathbf{r}) \cap \mathcal{M}_0(\mathbf{p})) \neq \emptyset] \leq \mathsf{negl}(|\mathbb{F}|).$$

This lemma establishes that the intersection of a random polynomial with another polynomial is empty except with negligible probability.

Proof. This follows from the fundamental theorem of algebra, which states that a polynomial of degree d evaluates to 0 in a random point only with probability $d/|\mathbb{F}|$.

Since $\mathbf{r}$ (and therefore all $x \in \mathcal{M}_0(\mathbf{r})$) is uniformly random and $|\mathcal{M}_0(\mathbf{r})| = d$, while $|\mathcal{M}_0(\mathbf{p})| = d'$, we get that

$$\Pr[(\mathcal{M}_0(\mathbf{r}) \cap \mathcal{M}_0(\mathbf{p})) \neq \emptyset] \leq dd'/|\mathbb{F}|.$$

□

Lemma 2.4. *Let $d \in \mathsf{poly}(\log|\mathbb{F}|)$. Let $\mathbf{p} \in \mathbb{F}[X]$, $\deg(\mathbf{p}) = d$ be a fixed but unknown non-trivial polynomial. Further let $\mathbf{r} \in \mathbb{F}[X]$, $\deg(\mathbf{r}) = d$ be a uniformly random polynomial. For all non-trivial $\mathbf{q}, \mathbf{s} \in \mathbb{F}[X]$ with $\deg(\mathbf{q}) \leq d$ and $\deg(\mathbf{s}) \leq d$,*

$$\Pr_{\mathbf{r}\in\mathbb{F}[X]}[(\mathcal{M}_0(\mathbf{p}) \cap \mathcal{M}_0(\mathbf{ps} + \mathbf{rq})) \neq (\mathcal{M}_0(\mathbf{p}) \cap \mathcal{M}_0(\mathbf{q}))] \leq \mathsf{negl}(|\mathbb{F}|).$$

This lemma shows that the multiplication of (possibly maliciously chosen) polynomials does not affect the intersection except with negligible probability, if one random polynomial is used.

Proof.

$$\begin{aligned}
\mathcal{M}_0(\mathbf{p}) \cap \mathcal{M}_0(\mathbf{ps} + \mathbf{rq}) &\overset{Lemma\ 2.1}{=} \mathcal{M}_0(\mathbf{p}) \cap (\mathcal{M}_0(\mathbf{ps}) \cap \mathcal{M}_0(\mathbf{qr})) \\
&= \mathcal{M}_0(\mathbf{p}) \cap \big((\mathcal{M}_0(\mathbf{p}) \cup \mathcal{M}_0(\mathbf{s})) \cap (\mathcal{M}_0(\mathbf{q}) \cup \mathcal{M}_0(\mathbf{r}))\big) \\
&= \mathcal{M}_0(\mathbf{p}) \cap \big((\mathcal{M}_0(\mathbf{p}) \cap \mathcal{M}_0(\mathbf{q})) \cup \underbrace{(\mathcal{M}_0(\mathbf{p}) \cap \mathcal{M}_0(\mathbf{r}))}_{\mathcal{T}_1} \\
&\quad \cup \underbrace{(\mathcal{M}_0(\mathbf{s}) \cap \mathcal{M}_0(\mathbf{q}))}_{\subseteq \mathcal{M}_0(\mathbf{q})} \cup \underbrace{(\mathcal{M}_0(\mathbf{s}) \cap \mathcal{M}_0(\mathbf{r}))}_{\mathcal{T}_2}\big)
\end{aligned}$$

From Lemma 2.3 it follows that $\Pr[\mathcal{T}_1 \neq \emptyset] \leq d^2/|\mathbb{F}|$, and also $\Pr[\mathcal{T}_2 \neq \emptyset] \leq d^2/|\mathbb{F}|$. Since

$$\mathcal{M}_0(\mathbf{p}) \cap \big((\mathcal{M}_0(\mathbf{p}) \cap \mathcal{M}_0(\mathbf{q})) \cup \mathcal{M}_0(\mathbf{q})\big) = \mathcal{M}_0(\mathbf{p}) \cap \mathcal{M}_0(\mathbf{q}),$$

we get

$$\Pr_{\mathbf{r}\in\mathbb{F}[X]}[(\mathcal{M}_0(\mathbf{p}) \cap \mathcal{M}_0(\mathbf{ps}+\mathbf{rq})) \neq (\mathcal{M}_0(\mathbf{p}) \cap \mathcal{M}_0(\mathbf{q}))] \leq 2d^2/|\mathbb{F}|.$$

□

3 Enhanced Oblivious Linear Function Evaluation $\mathcal{F}_{\text{OLE}^+}$

In this section we present an enhanced version of the OLE functionality. The standard OLE functionality allows the sender to input a, b, while the receiver inputs x and obtains $ax + b$. For our applications, we do not want the receiver to be able to learn b, i.e. it has to hold that $x \neq 0$. Our approach is therefore to modify the OLE functionality in such a way that it outputs a random field element upon receiving an input $x = 0$ (cf. Figure 2). A different approach might be to output a special abort symbol or 0, but crucially the output must *not* satisfy the relation $ax + b$. This is a particularly useful feature, as we will show in the next section.

$\mathcal{F}_{\text{OLE}^+}$

1. Upon receiving a message $(\texttt{inputS}, (a, b))$ from the sender with $a, b \in \mathbb{F}$, verify that there is no stored tuple, else ignore that message. Otherwise, store (a, b) and send $(\texttt{input})$ to $\mathcal{A}$.
2. Upon receiving a message $(\texttt{inputR}, x)$ from the receiver with $x \in \mathbb{F}$, verify that there is no stored value, else ignore that message. Otherwise, store x and send $(\texttt{input})$ to $\mathcal{A}$.
3. Upon receiving a message $(\texttt{deliver})$ from $\mathcal{A}$, check if both (a, b) and x are stored, else ignore that message. If $x \neq 0$, set $c = ax + b$, otherwise pick a uniformly random $c \in \mathbb{F}$ and send $(\texttt{output}, c)$ to the receiver. Ignore all further messages.

Fig. 2. Ideal functionality for the enhanced oblivious linear function evaluation.

While it might be possible to modify existing OLE protocols in such a way that a non-zero input is guaranteed, we instead opt to build a protocol black-box from the standard OLE functionality $\mathcal{F}_{\text{OLE}}$.

We refer to the introduction for an abstract overview and a description of the ideas of our construction. The formal description of the protocol is given in Figure 3.

Lemma 3.1. *Π_{OLE^+} unconditionally UC-realizes $\mathcal{F}_{\text{OLE}^+}$ in the $\mathcal{F}_{\text{OLE}}$-hybrid model.*

Proof. The simulator against a corrupted sender simulates both instances of $\mathcal{F}_{\text{OLE}}$. Let α_1 be the sender's input in the first OLE, and (α_2, α_3) be the inputs into the second OLE. The simulator sets $\hat{b} = \alpha_1 + \alpha_3$ and $\hat{a} = \alpha_2 - \hat{t}$,

Π_{OLE^+}

1. Receiver (Input $x \in \mathbb{F}$): Pick $s \in \mathbb{F}$ and send $(\texttt{inputS}, (x^{-1}, s))$ to the first $\mathcal{F}_{\mathrm{OLE}}$.
2. Sender (Input $a, b \in \mathbb{F}$): Pick $u \in \mathbb{F}$ uniformly at random and send $(\texttt{inputR}, u)$ to the first $\mathcal{F}_{\mathrm{OLE}}$ to learn $t = ux^{-1} + s$. Send $(\texttt{inputS}, (t + a, b - u))$ to the second $\mathcal{F}_{\mathrm{OLE}}$.
3. Receiver: Send $(\texttt{inputR}, x)$ to the second $\mathcal{F}_{\mathrm{OLE}}$ and obtain $c = ax+b+sx$. Output $c - sx$.

Fig. 3. Protocol that realizes $\mathcal{F}_{\mathrm{OLE}^+}$ in the $\mathcal{F}_{\mathrm{OLE}}$-hybrid model.

where $\hat{t}$ is chosen as the uniformly random output to $\mathcal{A}_{\mathsf{S}}$ of the first OLE. The simulator simply inputs $(\texttt{inputS}, (\hat{a}, \hat{b}))$ into $\mathcal{F}_{\mathrm{OLE}^+}$. Let us briefly argue that this simulation is indistinguishable from a real protocol run. The value $\hat{t}$ is indistinguishable from a valid t, since the receiver basically uses a one-time-pad s to mask the multiplication. Therefore, the sender can only change his inputs into the OLEs. Since his inputs uniquely determine both $\hat{a}$ and $\hat{b}$, the extraction by the simulator is correct and the simulation is indistinguishable from a real protocol run.

Against a corrupted receiver, the simulator simulates the two instance of $\mathcal{F}_{\mathrm{OLE}}$ and obtains the receiver's inputs (ξ_1, ξ_3) and ξ_2. If $\xi_1 \cdot \xi_2 = 1$, the simulator sets $\hat{x} = \xi_2$, sends $(\texttt{inputR}, \hat{x})$ to $\mathcal{F}_{\mathrm{OLE}^+}$ and receives $(\texttt{output}, c)$. It forwards $c' = c + \xi_2\xi_3$ to $\mathcal{A}_{\mathsf{R}}$. If $\xi_1 \cdot \xi_2 \neq 1$, the simulator sends $(\texttt{inputR}, 0)$ to $\mathcal{F}_{\mathrm{OLE}^+}$ and forwards the output c to the receiver. It remains to argue that this simulation is indistinguishable from the real protocol. From $\mathcal{A}$'s view, the output c is determined as

$$c = u\xi_1\xi_2 + a\xi_2 + b - u + \xi_2\xi_3 = a\xi_2 + b + u(\xi_1\xi_2 - 1) + \xi_2\xi_3.$$

We can ignore the last term, since it is known to $\mathcal{A}$. If $\xi_1\xi_2 \neq 1$, then $u(\xi_1\xi_2 - 1)$ does not vanish and the result will be uniform over the choice of u. Thus, by using ξ_2 as the correct input otherwise, we extract the correct value and the simulation is indistinguishable from the real protocol. □

4 Randomized Polynomial Addition from OLE

Concretely, we have two parties, the sender with a polynomial of degree $2d$ as input and the receiver with a polynomial of degree d as input. The goal is that the receiver obtains the sum of these two polynomials such that it cannot learn the sender's polynomial fully. We want to achieve this privacy property by using a randomization polynomial that prevents the receiving party from simply subtracting its input from the result. This functionality is defined in Figure 4.

Notice that we have some additional requirements regarding the inputs of the parties. First, the degree of the inputs has to be checked, but the functionality also makes sure that the receiver does not input a 0 polynomial, because otherwise he might learn the input of the sender. Also note that the functionality

leaks some information about the sender's polynomial. Looking ahead in the PSI protocol, where the input of the sender is always a uniformly random $2d$ degree polynomial, this leakage of the ideal functionality will not leak any non-trivial information in the PSI protocol.

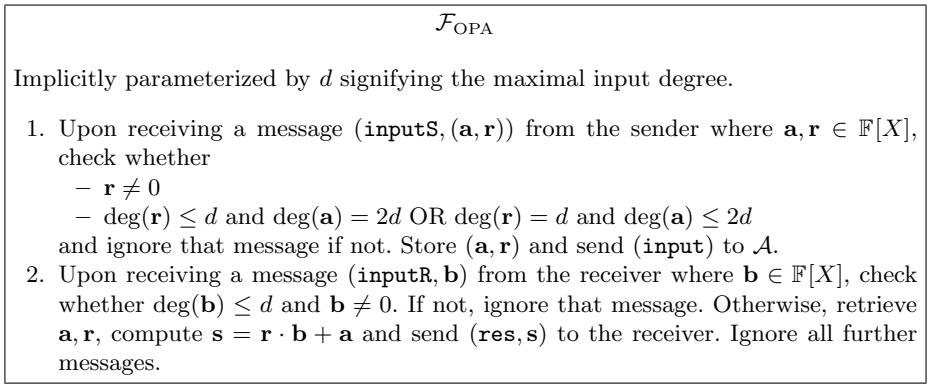

Fig. 4. Ideal functionality that allows to obliviously compute an addition of polynomials.

It is instructive to first consider a passively secure protocol. In the semi-honest case, both sender and receiver evaluate their input polynomials on a set of distinct points $\mathcal{P} = \{\alpha_1, \ldots, \alpha_{2d+1}\}$, where d is the degree of the input polynomials. The sender additionally picks a random polynomial $\mathbf{r} \in \mathbb{F}[X]$ of degree d and also evaluates it on $\mathcal{P}$.

Instead of using OLE in the "traditional" sense, i.e. instead of computing $\mathbf{ab}+\mathbf{r}$ where $\mathbf{r}$ blinds the multiplication of the polynomials, we basically compute $\mathbf{rb}+\mathbf{a}$. This means that the sender randomizes the polynomial of the receiver, and then adds his own polynomial. This prevents the receiver from simply subtracting his input polynomial and learning $\mathbf{a}$. In a little more detail, sender and receiver use $2d+1$ OLEs to add the polynomials as follows: for each $i \in [2d+1]$, the sender inputs (r_i, a_i) in OLE i, while the receiver inputs b_i and obtains $s_i = r_i b_i + a_i$. He then interpolates the resulting polynomial $\mathbf{s}$ of degree $2d$ using the $2d + 1$ values s_i.

In going from passive to active security, we have to ensure that the inputs of the parties are correct. Here, the main difficulty obviously lies in checking for $\mathbf{b} = 0$. In fact, since $\mathcal{F}_{\text{OPA}}$ does not even leak a single point a_i we have to make sure that all $b_i \neq 0$. However, this can easily be achieved by using $\mathcal{F}_{\text{OLE}^+}$ instead of $\mathcal{F}_{\text{OLE}}$. We also have to verify that the inputs are well-formed via a simple polynomial check. For a more detailed overview we refer the reader to the introduction.

The complete actively secure protocol is shown in Figure 5. Here, we use two instances of $\mathcal{F}_{\text{OLE}}$ that implement a commitment and a check. We named the first OLE that is used for a commitment to a blinding value u $\mathcal{F}_{\text{OLE}}^{\mathsf{com}}$. The check

is performed by comparing the blinded reconstructed polynomial $\mathbf{s}$ evaluated in x_{S} with the inputs in this location using the second OLE denoted by $\mathcal{F}_{\mathrm{OLE}}^{\mathsf{check}}$.[8]

Π_{OPA}

Let $\mathcal{P} = \{\alpha_1, \ldots, \alpha_{2d+1}\}, \alpha_i \in \mathbb{F}$ be a set of distinct points and let $\mathcal{F}_{\mathrm{OLE}}^{\mathsf{check}}, \mathcal{F}_{\mathrm{OLE}}^{\mathsf{com}}$ be instances of $\mathcal{F}_{\mathrm{OLE}}$.

1. Sender (Input $\mathbf{a}, \mathbf{r} \in \mathbb{F}[X]$, $\deg(\mathbf{a}) \leq 2d, \deg(\mathbf{r}) = d$): Evaluate $\mathbf{a}, \mathbf{r}$ on $\mathcal{P}$ to obtain $(a_i, r_i), i \in [2d+1]$. Input (r_i, a_i) into $\mathcal{F}_{\mathrm{OLE}^+}^{(\mathrm{i})}$.
2. Receiver (Input $\mathbf{b} \in \mathbb{F}[X]$, $\deg(\mathbf{b}) \leq d$): Evaluate $\mathbf{b}$ on $\mathcal{P}$ and obtain $b_i, i \in [2d+1]$. Input b_i into $\mathcal{F}_{\mathrm{OLE}^+}^{(\mathrm{i})}$ and receive $s_i = a_i + b_i \cdot r_i$. Reconstruct $\mathbf{s}$ from the s_i and check if $\deg(\mathbf{s}) \leq 2d$, otherwise abort.
3. Consistency check: Sender picks a random $x_s \in \mathbb{F}$ and send it to the receiver.
   - Receiver: Pick random $u, v \in \mathbb{F}$ and input them into $\mathcal{F}_{\mathrm{OLE}}^{\mathsf{com}}$. Further input $(\mathbf{b}(x_s), -\mathbf{s}(x_s) + u)$ into $\mathcal{F}_{\mathrm{OLE}}^{\mathsf{check}}$.
   - Sender: Pick a random $t \in \mathbb{F}$ and input it into $\mathcal{F}_{\mathrm{OLE}}^{\mathsf{com}}$ and obtain $c = ut + v$. Input $\mathbf{r}(\mathbf{x_s})$ into $\mathcal{F}_{\mathrm{OLE}}^{\mathsf{check}}$ and obtain $\mathbf{r}(x_s)\mathbf{b}(\mathbf{x_s}) - \mathbf{s}(x_s) + u + \mathbf{a}(x_s) = u'$. Send u' to the receiver.
   - Receiver: If $u' \neq u$ abort, otherwise send v to the sender.
   - Sender: If $u't + v \neq c$ abort.
4. Receiver picks $x_r \in \mathbb{F}$ and runs similar consistency check with the sender.

Fig. 5. Protocol that realizes $\mathcal{F}_{\mathrm{OPA}}$ in the ($\mathcal{F}_{\mathrm{OLE}^+}$,$\mathcal{F}_{\mathrm{OLE}}$)-hybrid model.

Lemma 4.1. *Π_{OPA} unconditionally UC-realizes $\mathcal{F}_{\mathrm{OPA}}$ in the $\mathcal{F}_{\mathrm{OLE}^+}$-hybrid model.*

Proof (Sketch.). **Corrupted Sender.** The simulator $\mathcal{S}_{\mathsf{S}}$ against a corrupted sender proceeds as follows. It simulates $\mathcal{F}_{\mathrm{OLE}^+}^{(\mathrm{i})}$ and thereby obtains (r_i^*, a_i^*) for all $i \in [2d+1]$. From these values, the simulator reconstructs $\hat{\mathbf{r}}$ and $\hat{\mathbf{a}}$. It aborts in Step 3 if $\deg(\hat{\mathbf{r}}) > d$ or $\deg(\hat{\mathbf{a}}) > 2d$. It also aborts if $\hat{\mathbf{a}}$ or $\hat{\mathbf{r}}$ are zero, and otherwise sends $(\mathtt{inputS}, (\hat{\mathbf{a}}, \hat{\mathbf{r}}))$ to $\mathcal{F}_{\mathrm{OPA}}$.

The extraction of the corrupted sender's inputs is correct if his inputs $\mathbf{r}^*$ corresponds to a polynomial of degree d and $\mathbf{a}^*$ corresponds to a polynomial of degree $2d$. Thus, the only possibility for an environment to distinguish between the simulation and the real protocol is by succeeding in answering the check while using a malformed input, i.e. a polynomial of incorrect degree or 0-polynomials. If the polynomials have degree greater than d and $2d$, respectively, the resulting polynomial $\mathbf{s}$ has degree $2d+1$ instead of $2d$, i.e. the receiver cannot reconstruct the result from $2d+1$ points. Since the sender learns nothing about the receiver's inputs, the thus incorrectly reconstructed polynomial will be uniformly random from his point of view and the probability that his response to the challenge is

[8] The commitment we implicitly use has been used previously in [11], as has the check sub-protocol.

correct is $1/|\mathbb{F}|$. Also, both $\hat{\mathbf{a}}$ and $\hat{\mathbf{r}}$ have to be non-zero, because in each case the polynomials are evaluated in $2d+1$ points, and it requires $2d+1$ zeros as a_i, r_i to get a 0 polynomial. But since both $\mathbf{a}, \mathbf{r}$ have degree at most $2d$, there are at most $2d$ roots of these polynomials. Therefore, in order to pass the check, $\mathbf{a}(x)$ and $\mathbf{b}(x)$ would need to be 0, which is also checked for.

Corrupted Receiver. The simulator $\mathcal{S}_{\mathsf{R}}$ against a corrupted receiver simulates $\mathcal{F}_{\mathrm{OLE}^+}^{(\mathrm{i})}$ and obtains $\mathbf{b}_i^*$ for all $i \in [2d+1]$. It reconstructs $\hat{\mathbf{b}}$ and aborts the check in Step 3 if $\deg(\hat{\mathbf{b}}) > d$. The simulator sends $(\texttt{inputR}, \hat{\mathbf{b}})$ to $\mathcal{F}_{\mathrm{OPA}}$ and receives $(\texttt{res}, \hat{\mathbf{s}})$. It evaluates $\hat{\mathbf{s}}$ on $\mathcal{P}$ and returns s_i for the corresponding OLEs. $\mathcal{S}_{\mathsf{R}}$ simulates the rest according to the protocol.

Clearly, if the corrupted receiver $\mathcal{A}_{\mathsf{R}}$ inputs a degree d polynomial, the simulator will extract the correct polynomial. In order to distinguish the simulation from the real protocol, the adversary can either input 0 in an OLE or has to input a polynomial of higher degree, while still passing the check. In the first case, assume w.l.o.g. that $\mathcal{A}_{\mathsf{R}}$ cheats in $\mathcal{F}_{\mathrm{OLE}^+}^{(\mathrm{j})}$ for some j. This means $\mathcal{A}_{\mathsf{R}}$ receives a value $\hat{s}_i$, which is uniformly random. This means that only with probability $1/|\mathbb{F}|$ will $\hat{s}_i$ satisfy the relation $\mathbf{rb} + \mathbf{a}$ and the check will fail, i.e. he can lie about u, but the commitment to u cannot be opened without knowing t. In the second case, the resulting polynomial would be of degree $2d+1$, while the receiver only gets $2d+1$ points of the polynomial. Therefore the real polynomial is underdetermined and $\mathcal{A}$ can only guess the correct value $\hat{\mathbf{s}}(x)$, i.e. the check will fail with overwhelming probability. □

5 Maliciously Secure Two-party PSI

In this section we provide a maliciously secure two-party PSI protocol with output for *both* parties, i.e. we realize $\mathcal{F}_{\mathrm{PSI}}$ as described in Figure 6.

$\mathcal{F}_{\mathrm{PSI}}$

1. Upon receiving a message $(\texttt{input}, P, S_P)$ from party $P \in \{\mathsf{A}, \mathsf{B}\}$, store the set S_P. Once all inputs are given, set $S_\cap = S_\mathsf{A} \cap S_\mathsf{B}$ and send $(\texttt{output}, S_\cap)$ to $\mathcal{A}$.
2. Upon receiving a message $(\texttt{deliver})$ from $\mathcal{A}$, send $(\texttt{output}, S_\cap)$ to the honest party.

Fig. 6. Ideal functionality $\mathcal{F}_{\mathrm{PSI}}$ for two-party PSI.

We briefly sketch the protocol in the following; a more detailed overview can be found in the introduction. First, Alice and Bob simply transform their input sets into polynomials. Then, both compute a randomized share of the intersection via our previously defined OPA in such a way that Alice can send her share to Bob without him being able to learn her input. This can be achieved by adding a simple mask to the intermediate share. Bob adds both shares and

sends the output to Alice. The protocol only requires two OPA and a simple check which ensures semi-honest behaviour, and no computational primitives. A formal description is given in Figure 7.

Π_{2PSI}

Let $m = \max_i |\mathcal{S}_i| + 1$.

Computation of Intersection

1. Alice (Input S_A): Pick a random polynomial $\mathbf{p}_\mathsf{A}$ of degree m such that $\mathbf{p}_\mathsf{A}(\gamma_j) = 0$ for all $\gamma_j \in S_\mathsf{A}$. Generate two random polynomials $\mathbf{r}_\mathsf{A}, \mathbf{r}'_\mathsf{A}$ of degree m and a random polynomial $\mathbf{u}_\mathsf{A}$ of degree $2m$.
   - Input $\mathbf{r}_\mathsf{A}, \mathbf{u}_\mathsf{A}$ into $\mathcal{F}^{(1)}_{\text{OPA}}$.
   - Input $\mathbf{p}_\mathsf{A}$ into $\mathcal{F}^{(2)}_{\text{OPA}}$ and obtain $\mathbf{s}_\mathsf{A} = \mathbf{p}_\mathsf{A}\mathbf{r}_\mathsf{B} + \mathbf{u}_\mathsf{B}$.
   - Set $\mathbf{s}'_\mathsf{A} = \mathbf{s}_\mathsf{A} - \mathbf{u}_\mathsf{A} + \mathbf{p}_\mathsf{A}\mathbf{r}'_\mathsf{A}$ and send it to Bob.
2. P_B (Input S_B): Pick a random polynomial $\mathbf{p}_\mathsf{B}$ of degree m such that $\mathbf{p}_\mathsf{B}(\gamma_j) = 0$ for all $\gamma_j \in S_\mathsf{B}$. Generate two random polynomials $\mathbf{r}_\mathsf{B}, \mathbf{r}'_\mathsf{B}$ of degree m and a random polynomial $\mathbf{u}_\mathsf{B}$ of degree $2m$.
   - Input $\mathbf{r}_\mathsf{B}, \mathbf{u}_\mathsf{B}$ into $\mathcal{F}^{(2)}_{\text{OPA}}$.
   - Input $\mathbf{p}_\mathsf{B}$ into $\mathcal{F}^{(1)}_{\text{OPA}}$ and obtain $\mathbf{s}_\mathsf{B} = \mathbf{p}_\mathsf{B}\mathbf{r}_\mathsf{A} + \mathbf{u}_\mathsf{A}$.
   - Upon receiving $\mathbf{s}'_\mathsf{A}$, compute $\mathbf{p}_\cap = \mathbf{s}'_\mathsf{A} + \mathbf{s}_\mathsf{B} + \mathbf{p}_\mathsf{B}\mathbf{r}'_\mathsf{B} - \mathbf{u}_\mathsf{B}$ and send it to Alice.

Output Verification

3. Alice: Pick a random $x_\mathsf{A} \in \mathbb{F}$ and send it to Bob.
4. Bob: Set $\alpha_\mathsf{B} = \mathbf{p}_\mathsf{B}(x_\mathsf{A})$, $\beta_\mathsf{B} = \mathbf{r}_\mathsf{B}(x_\mathsf{A})$ and $\delta_\mathsf{B} = \mathbf{r}'_\mathsf{B}(x_\mathsf{A})$. Pick a random $x_\mathsf{B} \in \mathbb{F}$ and send $(x_\mathsf{B}, \alpha_\mathsf{B}, \beta_\mathsf{B}, \delta_\mathsf{B})$ to Alice.
5. Alice: Check if $\mathbf{p}_\mathsf{A}(x_\mathsf{A})(\beta_\mathsf{B} + \mathbf{r}'_\mathsf{A}(x_\mathsf{A})) + \alpha_\mathsf{B}(\mathbf{r}_\mathsf{A}(x_\mathsf{A}) + \delta_\mathsf{B}) = \mathbf{p}_\cap(x_\mathsf{A})$, otherwise abort. For each $\gamma_j \in S_\mathsf{A}$: If $\mathbf{p}_\cap(\gamma_j) = 0$, add γ_j to $S_\cap$. Send $\alpha_\mathsf{A} = \mathbf{p}_\mathsf{A}(x_\mathsf{B})$, $\beta_\mathsf{A} = \mathbf{r}_\mathsf{A}(x_\mathsf{B})$ and $\delta_\mathsf{A} = \mathbf{r}'_\mathsf{A}(x_\mathsf{B})$ to Bob. Output $S_\cap$.
6. Bob: Check if $\alpha_\mathsf{A}(\mathbf{r}_\mathsf{B}(x_\mathsf{B}) + \delta_\mathsf{A}) + \mathbf{p}_\mathsf{B}(x_\mathsf{B})(\beta_\mathsf{A} + \mathbf{r}'_\mathsf{B}(x_\mathsf{B})) = \mathbf{p}_\cap(x_\mathsf{B})$, otherwise abort. For each $\gamma_j \in S_\mathsf{B}$: If $\mathbf{p}_\cap(\gamma_j) = 0$, add γ_j to $S_\cap$. Output $S_\cap$.

Fig. 7. Protocol Π_{2PSI} UC-realises $\mathcal{F}_{\text{PSI}}$ in the $\mathcal{F}_{\text{OPA}}$-hybrid model.

Theorem 5.1. *The protocol Π_{2PSI} UC-realises $\mathcal{F}_{\text{PSI}}$ in the $\mathcal{F}_{\text{OPA}}$-hybrid model with communication complexity $O(m\kappa)$.*

Proof. Let us argue that $\mathbf{p}_\cap = \mathbf{p}_\mathsf{A}(\mathbf{r}'_\mathsf{A} + \mathbf{r}_\mathsf{B}) + \mathbf{p}_\mathsf{B}(\mathbf{r}_\mathsf{A} + \mathbf{r}'_\mathsf{B})$ actually hides the inputs. The main observation here is that $\mathbf{r}'_\mathsf{P} + \mathbf{r}_{\bar{\mathsf{P}}}$ is uniformly random as long as one party is honest. Since $\mathbf{p}_\mathsf{A} + \mathbf{p}_\mathsf{B}$ validly encodes the intersection (see Lemma 2.1), $\mathbf{p}_\cap$ is uniformly random over the choice of the randomization polynomials $\mathbf{r}_\mathsf{A}, \mathbf{r}'_\mathsf{A}, \mathbf{r}_\mathsf{B}$ and $\mathbf{r}'_\mathsf{B}$, except for the roots denoting the intersection.

Corrupted Alice. We show the indistinguishability of the simulation of $\mathcal{S}_\mathsf{A}$ (cf. Figure 8). The simulator extracts Alice's inputs and then checks for any

deviating behaviour. If such behaviour is detected, it aborts, even if the protocol would succeed. Proving indistinguishability of the simulation shows that the check in the protocol basically enforces semi-honest behaviour by Alice, up to input substitution.

Simulator $\mathcal{S}_\mathsf{A}$

1. Extract the inputs $\hat{\mathbf{p}}_\mathsf{A}, \hat{\mathbf{r}}_\mathsf{A}, \hat{\mathbf{u}}_\mathsf{A}$ by simulating $\mathcal{F}_{\text{OPA}}$.
2. Find the roots $\hat{\gamma}_1, \ldots, \hat{\gamma}_m$ of $\hat{\mathbf{p}}_\mathsf{A}$ and thereby the set $\hat{S}_\mathsf{A} = \{\hat{\gamma}_1, \ldots, \hat{\gamma}_m\}$.
3. Send $(\texttt{input}, \mathsf{A}, \hat{S}_\mathsf{A})$ to $\mathcal{F}_{\text{PSI}}$.
4. Upon receiving $(\texttt{output}, \hat{S}_\cap)$ from $\mathcal{F}_{\text{PSI}}$, pick a random degree m polynomial $\hat{\mathbf{p}}_\mathsf{B}$ such that $\hat{\mathbf{p}}_\mathsf{B}(\gamma) = 0$ for all $\gamma \in \hat{S}_\cap$.
5. Use $\hat{\mathbf{p}}_\mathsf{B}$ as input for simulating the $\mathcal{F}_{\text{OPA}}$ together with random polynomials $\hat{\mathbf{r}}_\mathsf{B}$ and $\hat{\mathbf{u}}_\mathsf{B}$, i.e. keep $\hat{\mathbf{s}}_\mathsf{B} = \hat{\mathbf{p}}_\mathsf{B} \cdot \hat{\mathbf{r}}_\mathsf{A} + \hat{\mathbf{u}}_\mathsf{A}$ and send $\hat{\mathbf{s}}_\mathsf{A} = \hat{\mathbf{p}}_\mathsf{A} \cdot \hat{\mathbf{r}}_\mathsf{B} + \hat{\mathbf{u}}_\mathsf{B}$ to $\mathcal{A}$.
6. Simulate the rest according to Π_{2PSI}, but abort in Step 6, if after setting

$$\hat{\mathbf{r}}'_\mathsf{A} = (\mathbf{s}'^*_\mathsf{A} + \hat{\mathbf{u}}_\mathsf{A} - \hat{\mathbf{u}}_\mathsf{B} - \hat{\mathbf{p}}_\mathsf{A}\hat{\mathbf{r}}_\mathsf{B})/\hat{\mathbf{p}}_\mathsf{A},$$

$\alpha^*_\mathsf{A} \neq \hat{\mathbf{p}}_\mathsf{A}(x)$, $\beta^*_\mathsf{A} \neq \hat{\mathbf{r}}_\mathsf{A}(x)$ or $\delta^*_\mathsf{A} \neq \hat{\mathbf{r}}'_\mathsf{A}(x)$, even though the check in Step 6 would pass.

Fig. 8. Simulator $\mathcal{S}_\mathsf{A}$ against a corrupted Alice.

Consider the following series of hybrid games.

Hybrid 0: $\mathsf{Real}^{\mathcal{A}_\mathsf{A}}_{\Pi_{\text{2PSI}}}$.

Hybrid 1: Identical to Hybrid 0, except that $\mathcal{S}_1$ simulates $\mathcal{F}_{\text{OPA}}$, learns all inputs and aborts if $\alpha^*_\mathsf{A} \neq \hat{\mathbf{p}}_\mathsf{A}(x)$ or $\beta^*_\mathsf{A} \neq \hat{\mathbf{r}}_\mathsf{A}(x)$, but the check is passed.
Let $\alpha^*_\mathsf{A} = \alpha_\mathsf{A} + e$ be $\mathcal{A}_\mathsf{A}$'s check value. Then the check in Step 6 will fail with overwhelming probability. Let σ denote the outcome of the check. If $\mathcal{A}_\mathsf{A}$ behaves honestly, then

$$\sigma = \alpha^*_\mathsf{A}(\mathbf{r}_\mathsf{B}(x) + \delta^*_\mathsf{A}) + \mathbf{p}_\mathsf{B}(x)(\beta^*_\mathsf{A} + \mathbf{r}'_\mathsf{B}(x)) - \mathbf{p}_\cap(x) = 0.$$

Using $\alpha^*_\mathsf{A} = \alpha_\mathsf{A} + e$, however, we get

$$\sigma' = (\alpha_\mathsf{A}+e)(\mathbf{r}_\mathsf{B}(x)+\delta^*_\mathsf{A})+\mathbf{p}_\mathsf{B}(x)(\beta^*_\mathsf{A}+\mathbf{r}'_\mathsf{B}(x))-\mathbf{p}_\cap(x) = e\cdot(\mathbf{r}_\mathsf{B}(x)+\delta^*_\mathsf{A}) \neq \mathsf{const}.$$

This means that the outcome of the check is uniformly random from $\mathcal{A}_\mathsf{A}$'s view over the choice of $\mathbf{r}_\mathsf{B}$ (or $\mathbf{p}_\mathsf{B}$ for $\beta^*_\mathsf{A} \neq \mathbf{r}_\mathsf{A}(x)$). Therefore, the check will fail except with probability $2/|\mathbb{F}|$ and Hybrids 0 and 1 are statistically close.

Hybrid 2: Identical to Hybrid 1, except that $\mathcal{S}_2$ aborts according to Step 6 in Figure 8.
An environment distinguishing Hybrids 1 and 2 must manage to send $\mathbf{s}'^*_\mathsf{A}$ such that

$$\mathbf{s}'^*_\mathsf{A} + \hat{\mathbf{u}}_\mathsf{A} - \hat{\mathbf{u}}_\mathsf{B} \neq \hat{\mathbf{p}}_\mathsf{A} \cdot (\hat{\mathbf{r}}_\mathsf{B} + \hat{\mathbf{r}}'_\mathsf{A})$$

while passing the check in Step 6 with non-negligible probability.

Let $\mathbf{f} = \mathbf{s}'^{*}_{\mathsf{A}} + \hat{\mathbf{u}}_{\mathsf{A}} - \hat{\mathbf{u}}_{\mathsf{B}} - \hat{\mathbf{p}}_{\mathsf{A}} \cdot (\hat{\mathbf{r}}_{\mathsf{B}} + \hat{\mathbf{r}}'_{\mathsf{A}})$. We already know that $\mathbf{f}(x) = 0$, otherwise we have $\alpha^{*}_{\mathsf{A}} = \alpha_{\mathsf{A}} + \mathbf{f}(x) \neq \alpha_{\mathsf{A}}$ (or an invalid β^{*}_{A}), and the check fails. But since x is uniformly random, the case that $\mathbf{f}(x) = 0$ happens only with probability $m/|\mathbb{F}|$, which is negligible. Therefore, Hybrid 1 and Hybrid 2 are statistically close.

Hybrid 3: Identical to Hybrid 2, except that $\mathcal{S}_3$ generates the inputs $\hat{\mathbf{s}}_{\mathsf{A}}, \hat{\mathbf{s}}_{\mathsf{B}}$ according to Step 5 in Figure 8 and adjusts the output. This corresponds to $\mathsf{Ideal}^{\mathcal{S}_{\mathsf{A}}}_{\mathcal{F}_{\mathrm{PSI}}}$.

The previous hybrids established that the inputs $\hat{\mathbf{p}}_{\mathsf{A}}, \hat{\mathbf{r}}_{\mathsf{A}}$ are extracted correctly. Therefore, by definition, $\hat{S}_{\mathsf{A}} = \mathcal{M}_0(\hat{\mathbf{p}}_{\mathsf{A}})$. It remains to argue that the simulated outputs are indistinguishable. First, note that the received intersection $\hat{S}_{\cap} = \mathcal{M}_0(\hat{\mathbf{p}}_{\mathsf{B}})$ defines $\hat{\mathbf{p}}_{\mathsf{B}}$. From Lemma 2.4 it follows that $\mathcal{M}_0(\mathbf{p}_{\cap}) = \mathcal{M}_0(\hat{\mathbf{p}}_{\mathsf{A}}) \cap \mathcal{M}_0(\hat{\mathbf{p}}_{\mathsf{B}}) = \hat{S}_{\cap}$ w.r.t. $\mathcal{M}_0(\hat{\mathbf{p}}_{\mathsf{B}})$, even for a maliciously chosen $\hat{\mathbf{r}}_{\mathsf{A}}$, i.e. the $\mathcal{A}_{\mathsf{A}}$ cannot increase the intersection even by a single element except with negligible probability.

Further, note that $\hat{\mathbf{s}}_{\mathsf{A}} = \hat{\mathbf{p}}_{\mathsf{A}} \cdot \hat{\mathbf{r}}_{\mathsf{B}} + \hat{\mathbf{u}}_{\mathsf{B}}$ is uniformly distributed over the choice of $\hat{\mathbf{u}}_{\mathsf{B}}$, and $\hat{\mathbf{p}}_{\cap}$ is uniform over the choice of $\hat{\mathbf{r}}_{\mathsf{B}}, \hat{\mathbf{r}}'_{\mathsf{B}}$.

Finally, since $\hat{\mathbf{r}}_{\mathsf{B}}, \hat{\mathbf{r}}'_{\mathsf{B}}$ are uniformly random and the degree of $\hat{\mathbf{p}}_{\mathsf{B}}$ is m, i.e. $\max_i |\mathcal{S}_i| + 1$, the values $\hat{\alpha}_{\mathsf{B}}$, $\hat{\beta}_{\mathsf{B}}$ and $\hat{\delta}_{\mathsf{B}}$ are uniformly distributed as well. In conclusion, the Hybrids 2 and 3 are statistically close.

As a result we get that for all environments $\mathcal{Z}$,

$$\mathsf{Real}^{\mathcal{A}_{\mathsf{A}}}_{\Pi_{\mathrm{2PSI}}}(\mathcal{Z}) \approx_s \mathsf{Ideal}^{\mathcal{S}_{\mathsf{A}}}_{\mathcal{F}_{\mathrm{PSI}}}(\mathcal{Z}).$$

Corrupted Bob. The simulator against a corrupted Bob is essentially the same as the one against a corrupted Alice, except for a different way to check his inputs. For the full proof we refer the reader to the full version [15] of the paper.

Efficiency. The protocol makes two calls to OPA, which in turn is based on OLE. Overall, $2m$ calls to OLE are necessary in OPA. Given the recent constant overhead OLE of Ghosh et al. [14], the communication complexity of Π_{2PSI} lies in $O(m\kappa)$.

The computational cost of the protocol is dominated by multi-point evaluation of polynomials of degree m, which requires $O(m \log m)$ multiplications using fast modular transform [3]. Note that this cost includes computational cost of the OLE instantiation from [14]. This concludes the proof.

□

6 Maliciously Secure Multi-party PSI

6.1 Ideal Functionality

The ideal functionality for multi-party private set intersection $\mathcal{F}^{*}_{\mathrm{MPSI}}$ simply takes the inputs from all parties and computes the intersection of these inputs. Our functionality $\mathcal{F}^{*}_{\mathrm{MPSI}}$ in Figure 9 additionally allows an adversary to learn

$\mathcal{F}^*_{\text{MPSI}}$

Let A denote the set of malicious parties, and H the set of honest parties.

1. Upon receiving a message $(\texttt{input}, P_i, S_i)$ from party P_i, store the set S_i. Once all inputs $i \in [n]$ are input, set $S_\cap = \bigcap_{i=1}^{n} S_i$ and send $(\texttt{output}, S_\cap)$ to $\mathcal{A}$.
2. Upon receiving a message $(\texttt{deliver}, S'_\cap)$ from $\mathcal{A}$, check if $S'_\cap \subseteq S_\cap$. If not, set $S'_\cap = \perp$. Send $(\texttt{output}, S'_\cap)$ to H.

Fig. 9. Ideal functionality $\mathcal{F}^*_{\text{MPSI}}$ for multi-party PSI.

the intersection and then possibly update the result to be only a subset of the original result.

Let us briefly elaborate on why we chose to use this modified functionality. In the UC setting, in order to extract the inputs of all malicious parties, any honest party has to communicate with all malicious parties. In particular, since the simulator has to extract the complete input, this requires at least $O(nm)$ communication per party for the classical MPSI functionality. In turn, for the complete protocol, this means that the communication complexity lies in $O(n^2m)$.

Instead, we want to take an approach similar to the recent work of Hazay et al. [19], i.e. we have one central party, and some of the work is delegated to this party. This removes the need for the other parties to extensively communicate with each other and potentially allows communication complexity $O(mn)$, which is asymptotically optimal in any setting. However, if we assume that the central party and at least one additional party are corrupted, the honest party does not (extensively) interact with this additional party and does not learn its inputs; it can only learn the input of the central party. If the input set of the other malicious party is the same as the one of the central party, the output remains the same. If this input is different, however, the actual intersection might be smaller. One might argue that this case simply corresponds to input substitution by the malicious party, but for any type of UC simulation this poses a problem, since the output of the honest party in the protocol might be different from the intersection in the ideal world. Thus, $\mathcal{F}^*_{\text{MPSI}}$ allows a malicious party to modify the output. Crucially, the updated intersection can only be smaller and may not changed arbitrarily by the adversary. We believe that this weaker multiparty PSI functionality is sufficient for most scenarios.

6.2 Multi-party PSI from OLE

Our multi-party PSI protocol uses the same techniques that we previously employed to achieve two-party PSI. This is similar in spirit to the approach taken in [19], who employ techniques from the two-party PSI of [13] and apply them in the multi-party setting. We also adopt the idea of a designated central party that performs a two-party PSI with the remaining parties, because this allows to delegate most of the computation to this party and saves communication.

Apart from that, our techniques differ completely from [19]. Abstractly, they run the two-party PSI with each party and then use threshold encryption and zero-knowledge proofs to ensure the correctness of the computation. These tools inflict a significant communication and computation penalty.

In our protocol (cf. Figure 10) we run our two-party PSI between the central party and every other party, but we ensure privacy of the aggregation not via threshold encryption and zero-knowledge proofs, but instead by a simple masking of the intermediate values and a polynomial check. This masking is created in a setup phase, where every pair of parties exchanges a random seed that is used to create two random blinding polynomials which cancel out when added.

Once the central party receives all shares of the computation, it simply add these shares, thereby removing the random masks. The central party broadcasts the result to all parties. Then, all parties engage in a multi-party coin-toss and obtain a random value x. Since all operations in the protocol are linear operations on polynomials, the parties evaluate their input polynomials on x and broadcast the result. This allows every party to locally verify the relation and as a consequence also the result. Here we have to ensure that a rushing adversary cannot cheat by waiting for all answers before providing its own answer. We solve this issue by simply committing to the values first, and the unveiling them in the next step. This leads to malleability problems, i.e. we have to use non-malleable commitments[9].

Theorem 6.1. *The protocol Π_{MPSI} computationally UC-realises $\mathcal{F}^{*}_{\text{MPSI}}$ in the $\mathcal{F}_{\text{OPA}}$-hybrid model with communication complexity in $O((n^2 + nm)\kappa)$.*

Proof. We have to distinguish between the case where the central party is malicious and the case where it is honest. We show UC-security of Π_{MPSI} by defining a simulator $\mathcal{S}$ for each case which produces an indistinguishable simulation of the protocol to any environment $\mathcal{Z}$ trying to distinguish the ideal world from the real world. The approach of the simulation is straightforward: the simulator extracts the input polynomials into $\mathcal{F}_{\text{OPA}}$ and thus obtains an intersection of the adversary's inputs.

In the case of an honest central party, all parties communicate with this party, i.e. the simulator can extract all inputs of all malicious parties. In the case where P_0 is malicious, however, the simulator can at most learn the central party's input at the beginning. He inputs this result into the ideal functionality and uses the intermediate result for the simulation. The malicious central party can later "simulate" the other malicious parties and thereby possibly change the intersection for the honest parties. We show that $\mathcal{A}$ can only reduce the intersection unless it already knows $x \in S_j$ for at least one $j \in \mathsf{H}$, i.e. we assume that $\mathcal{A}$ cannot predict a single element of the set of an honest party except with negligible probability. This reduced intersection can be passed by the simulator to the ideal functionality.

[9] In order to achieve our claimed efficiency we actually use UC commitments, but non-malleable commitments are sufficient for the security of the protocol.

Π_{MPSI}

Let $m = \max_i \{|\mathcal{S}_i|\} + 1$ and NMCOM be a bounded-concurrent non-malleable commitment scheme against synchronized adversaries. $\mathcal{F}_{\text{OPA}}^{(\text{i,j})}$ denotes the jth instance for parties P_0 and P_i.

Setup

1. All parties P_i and P_j for $i, j \in \{1, \ldots, n-1\}$ exchange a random polynomial as follows. For all $j \neq i$, if $\mathbf{v}_{ij} = \perp$, P_i picks seed_{ij} uniformly at random and sets $\mathbf{v}_{ij} = \mathsf{PRG}_{2m}(\mathsf{seed}_{ij})$. It sends seed_{ij} to P_j, who sets $\mathbf{v}_{ji} = -\mathsf{PRG}_{2m}(\mathsf{seed}_{ij})$.

Share Computation

2. P_0 (Input S_0): Compute a polynomial $\mathbf{p}_0$ of degree m s.t. $\mathbf{p}_0(\gamma_j) = 0$ for all $\gamma_j \in S_0$. Generate n random polynomials $\mathbf{r}_0^i \in \mathbb{F}[X], i \in \{1, \ldots, n-1\}$ and $\mathbf{r}_0' \in \mathbb{F}[X]$ of degree m each and $n-1$ random polynomials $\mathbf{u}_0^i \in \mathbb{F}[X], i \in \{1, \ldots, n-1\}$ of degree $2m$. For $i \in [n-1]$
   - Input $\mathbf{r}_0^i, \mathbf{u}_0^i$ into $\mathcal{F}_{\text{OPA}}^{(\text{i,1})}$ for each $i \in \{1, \ldots, n-1\}$.
   - Input $\mathbf{p}_0$ into $\mathcal{F}_{\text{OPA}}^{(\text{i,2})}$ and obtain $\mathbf{s}_0^i = \mathbf{p}_0 \cdot \mathbf{r}_i + \mathbf{u}_i$.
3. P_i (Input S_i): Compute a polynomial $\mathbf{p}_i$ of degree m s.t. $\mathbf{p}_i(\gamma_j) = 0$ for all $\gamma_j \in S_i$. Additionally, pick $\mathbf{r}_i, \mathbf{r}_i' \in \mathbb{F}[X]$ uniformly of degree m and $\mathbf{u}_i \in \mathbb{F}[X]$ uniformly of degree $2m$.
   - Input $\mathbf{p}_i$ into $\mathcal{F}_{\text{OPA}}^{(\text{i,1})}$ and obtain $\mathbf{s}_i = \mathbf{p}_i \cdot \mathbf{r}_0^i + \mathbf{u}_0^i$.
   - Input $\mathbf{r}_i, \mathbf{u}_i$ into $\mathcal{F}_{\text{OPA}}^{(\text{i,2})}$.
   - Set $\mathbf{s}_i' = \mathbf{s}_i - \mathbf{u}_i + \mathbf{p}_i \mathbf{r}_i' + \sum_{i \neq j} \mathbf{v}_{ij}$ and send it to P_0.

Output Aggregation and Verification

4. P_0: Compute $\mathbf{p}_\cap = \sum_{i=1}^{n-1} (\mathbf{s}_i' + \mathbf{s}_0^i - \mathbf{u}_0^i + \mathbf{p}_0 \mathbf{r}_0')$ and broadcast $\mathbf{p}_\cap$.
5. *All parties*:
   - Run a multiparty coin-toss protocol Π_{CT} to obtain a random $x \in \mathbb{F}$.
   - Evaluate $\alpha_i = \mathbf{p}_i(x)$, $\beta_i = \mathbf{r}_i(x)$, $\delta_i = \mathbf{r}_i'(x)$ and compute $(\mathsf{com}_i, \mathsf{unv}_i) = \mathsf{NMCOM.Commit}(\alpha_i, \beta_i, \delta_i)$. Broadcast com_i.
   - Once all commitments are received, broadcast unv_i and $(\alpha_i, \beta_i, \delta_i)$. Abort if $\sum \alpha_0 \cdot (\beta_i + \delta_0) + \alpha_i \cdot (\beta_0 + \delta_i) \neq \mathbf{p}_\cap(x)$ or $\mathsf{NMCOM.Open}(\mathsf{com}_i, \mathsf{unv}_i, (\alpha_i, \beta_i, \delta_i)) \neq 1$.
   - For each $\gamma_j \in S_i$: if $\mathbf{p}_\cap(\gamma_j) = 0$, add γ_j to $S_\cap$. Output $S_\cap$.

Fig. 10. Protocol Π_{MPSI} UC-realises $\mathcal{F}_{\text{MPSI}}^*$ in the $\mathcal{F}_{\text{OPA}}$-hybrid model.

P_0 is malicious: Consider the simulator in Figure 11.

We show the indistinguishability of the simulation and the real protocol through the following hybrid games. In the following, let $\mathcal{A}$ denote the dummy adversary controlled by $\mathcal{Z}$.

Hybrid 0: $\mathsf{Real}_{\Pi_{\text{MPSI}}}^{\mathcal{A}}$.

Simulator $\mathcal{S}_{P_0}$

Let $\mathsf{A} = \{i | P_i \text{ is malicious}\}$ denote the index set of corrupted parties, where $|\mathsf{A}| = t \leq n-1$. Further let H denote the index set of honest parties.

1. Simulate the setup and obtain all $\mathbf{v}_{ij}^*$ for $i \in \mathsf{A}$ and $j \in \mathsf{H}$. Pick uniformly random $\hat{\mathbf{v}}_{jl} \in \mathbb{F}[X]$ of degree $2m$ for $j, l \in \mathsf{H}$ and set $\hat{\mathbf{v}}_{lj} = -\hat{\mathbf{v}}_{jl}$.
2. Extract the inputs $(\hat{\mathbf{p}}_0^j, \hat{\mathbf{r}}_0^j, \hat{\mathbf{u}}_0^j)$ of P_0 for all $j \in \mathsf{H}$ by simulating $\mathcal{F}_{\text{OPA}}$.
3. Abort in Step 5 of Π_{MPSI} if the $\hat{\mathbf{p}}_0^j$ are not all identical. Set $\hat{\mathbf{p}}_{\mathsf{A}} = \hat{\mathbf{p}}_0^j$ for a random $j \in \mathsf{H}$, and find the roots $\hat{\gamma}_1, \ldots, \hat{\gamma}_{2m}$ of $\hat{\mathbf{p}}_{\mathsf{A}}$ and thereby the set $\hat{S}_{\mathsf{A}} = \{\hat{\gamma}_1, \ldots, \hat{\gamma}_{2m}\}$.
4. Send $(\texttt{input}, P_i, \hat{S}_{\mathsf{A}})$ to $\mathcal{F}_{\text{MPSI}}^*$ for all parties $i \in \mathsf{A}$.
5. Upon receiving $(\texttt{output}, \hat{S}_\cap)$ from $\mathcal{F}_{\text{MPSI}}^*$, pick $n-t$ random degree m polynomials $\hat{\mathbf{p}}_j$ such that $\hat{\mathbf{p}}_j(\gamma) = 0$ for all $\gamma \in \hat{S}_\cap, j \in \mathsf{H}$.
6. Use the $\hat{\mathbf{p}}_j$ as input for each instance of $\mathcal{F}_{\text{OPA}}$ together with random polynomials $\hat{\mathbf{r}}_j$ and $\hat{\mathbf{u}}_j$ for $j \in \mathsf{H}$, i.e. send $\hat{\mathbf{s}}_0^j = \hat{\mathbf{p}}_0^j \cdot \hat{\mathbf{r}}_j + \hat{\mathbf{u}}_j$ to $\mathcal{A}$ and keep $\hat{\mathbf{s}}_j = \hat{\mathbf{p}}_j \cdot \hat{\mathbf{r}}_0^j + \hat{\mathbf{u}}_0^j$.
7. Simulate the rest according to Π_{MPSI}, but abort in Step 5, if the extracted $\hat{\mathbf{p}}_{\mathsf{A}}(x) \neq \alpha_0$ or $\hat{\mathbf{r}}_0^j(x) \neq \beta_0^j$ for any $j \in \mathsf{H}$, even if the check passes otherwise.
8. Upon receiving $\mathbf{p}_\cap^*$ from $\mathcal{A}$, if the check in Step 5 of Π_{MPSI} passes, test for all $s \in \hat{S}_\cap$ if $\mathbf{p}_\cap^*(s) = 0$. If yes, set $\hat{S}'_\cap = \hat{S}'_\cap \cup s$. Send $(\texttt{deliver}, \hat{S}'_\cap)$ to $\mathcal{F}_{\text{MPSI}}^*$.

Fig. 11. Simulator $\mathcal{S}_{P_0}$ for $P_0 \in \mathsf{A}$.

Hybrid 1: Identical to Hybrid 0, except that $\mathcal{S}_1$ simulates $\mathcal{F}_{\text{OPA}}$ and learns all inputs.

Hybrid 2: Identical to Hybrid 1, except that $\mathcal{S}_2$ aborts according to Step 7 in Figure 11.

Hybrid 3: Identical to Hybrid 2, except that $\mathcal{S}_3$ aborts if the extracted $\hat{\mathbf{p}}_0$ are not identical, but the check is passed.

Hybrid 4: Identical to Hybrid 3, except that $\mathcal{S}_4$ replaces the $\mathbf{v}_{jl}$ between honest parties j, l by uniformly random polynomials.

Hybrid 5: Identical to Hybrid 4, except that $\mathcal{S}_5$ generates the inputs $\hat{\mathbf{s}}_0^j, \hat{\mathbf{s}}_j$ according to Step 6 in Figure 11 and adjusts the output. This corresponds to $\mathsf{Ideal}_{\mathcal{F}_{\text{MPSI}}^*}^{\mathcal{S}_{P_0}}$.

Hybrids 0 and 1 are trivially indistinguishable. We show that Hybrid 1 and Hybrid 2 are computationally indistinguishable in Lemma 6.1.1. This step ensures that the correct $\hat{p}_0$ was extracted, and that all the intermediate values of the honest parties are added up. Hybrids 2 and 3 are indistinguishable due to the security of the coin-toss. This is formalized in Lemma 6.1.2. As an intermediate step to complete the full simulation, we replace all pseudorandom polynomials $\mathbf{v}_{jl}$ between honest parties j, l by uniformly random ones. Computational indistinguishability of Hybrid 3 and Hybrid 4 follows from a straightforward reduction to the pseudorandomness of PRG. We establish the statistical indistinguishability of Hybrids 4 and 5 in Lemma 6.1.3. As a result we get that for

all PPT environments $\mathcal{Z}$,

$$\mathsf{Real}^{\mathcal{A}}_{\Pi_{\text{MPSI}}}(\mathcal{Z}) \approx_c \mathsf{Ideal}^{\mathcal{S}_{P_0}}_{\mathcal{F}^*_{\text{MPSI}}}(\mathcal{Z}).$$

Lemma 6.1.1 *Assume that* NMCOM *is a bounded-concurrent non-malleable commitment scheme against synchronizing adversaries. Then Hybrid 1 and Hybrid 2 are computationally indistinguishable.*

Proof. The only difference between Hybrid 1 and Hybrid 2 lies in the fact that $\mathcal{S}_2$ aborts if the extracted $\hat{\mathbf{p}}_{\mathsf{A}}$ evaluated on x does not match the check value α_0, but the check is still passed. Therefore, in order for $\mathcal{Z}$ to distinguish both hybrids, it has to be able to produce a value $\alpha_0^* \neq \hat{\mathbf{p}}_{\mathsf{A}}(x)$ and pass the check with non-negligible probability ϵ. W.l.o.g. it is sufficient that α_0^* is incorrect for only one $\hat{\mathbf{p}}_0$. We show that such a $\mathcal{Z}$ breaks the non-malleability property of NMCOM.

Let σ denote the outcome of the check. If $\mathcal{A}$ is honest, i.e. $\alpha_0 = \hat{\mathbf{p}}_0(x)$ and $\beta_0^i = \hat{\mathbf{r}}_0^i(x)$, then

$$\sigma = \sum_{i=0}^{n} (\alpha_0(\beta_i + \delta_0) + \alpha_i(\beta_0^i + \delta_i)) - \mathbf{p}_{\cap}(x) = 0, \tag{1}$$

where

$$\mathbf{p}_{\cap} = \sum_{i \in \mathsf{A}} (\mathbf{s}_i + \mathbf{s}_0^i) + \sum_{j \in \mathsf{H}} (\mathbf{s}_j + \mathbf{s}_0^j).$$

We first observe that $\sum_{j\in\mathsf{H}} (\mathbf{s}_j + \mathbf{s}_0^j) = \sum_{j\in\mathsf{H}} \hat{\mathbf{p}}_j(\hat{\mathbf{r}}_0^j + \hat{\mathbf{r}}_j') + \hat{\mathbf{p}}_0(\hat{\mathbf{r}}_0' + \hat{\mathbf{r}}_j)$ is uniform over the choice of the $\hat{\mathbf{r}}_j, \hat{\mathbf{r}}_j'$. Therefore, if $\mathcal{A}$ uses $\mathbf{p}_{\cap}^*$ without adding $\sum_{j\in\mathsf{H}} (\mathbf{s}_j + \mathbf{s}_0^j)$, the check will fail with overwhelming probability.

Since $\mathcal{A}$ controls the inputs of the malicious parties $i \in \mathsf{A}$, in order to pass the check it is sufficient for $\mathcal{A}$ to satisfy the following simplification of Equation (1).

$$\sigma' = \sum_{j \in \mathsf{H}} (\alpha_0(\beta_j + \delta_0) + \alpha_j(\beta_0^j + \delta_j)) - \sum_{j \in \mathsf{H}} (\mathbf{s}_j(x) + \mathbf{s}_0^j(x)) = \mathsf{const}$$

Here const is a fixed constant known to $\mathcal{A}$ (0 if $\mathcal{A}$ is honest) determined by setting the inputs α_i, β_i for $i \in \mathsf{A}$ accordingly. But if $\alpha_0^* \neq \hat{\mathbf{p}}_0(x)$, i.e. $\alpha_0^* = \alpha_0 + e$, then we get that

$$\begin{aligned}
\sigma' &= \sum_{j \in \mathsf{H}} ((\alpha_0 + e)(\beta_j + \delta_0) + \alpha_j(\beta_0^j + \delta_j)) - \sum_{j \in \mathsf{H}} (\mathbf{s}_j(x) + \mathbf{s}_0^j(x)) \\
&= \sum_{j \in \mathsf{H}} (\alpha_0(\beta_j + \delta_0) + \alpha_j(\beta_0^j + \delta_j)) - \sum_{j \in \mathsf{H}} (\mathbf{s}_j(x) + \mathbf{s}_0^j(x)) + e \sum_{j \in \mathsf{H}} (\beta_j + \delta_0) \\
&= e \sum_{j \in \mathsf{H}} (\beta_j + \delta_0) \neq \mathsf{const}
\end{aligned}$$

Similarly for $\beta_0^j \neq \hat{\mathbf{r}}_0^j(x)$ for any $j \in \mathsf{H}$. Thus, except for the case of $\alpha_0^* = \alpha_0 + e/\sum_{j\in\mathsf{H}} \beta_j$, the check will fail for $\alpha_0^* \neq \hat{\mathbf{p}}_0(x)$. But since we assumed that

$\mathcal{A}$ passes the check with non-negligible probability, and NMCOM is statistically binding, $\mathcal{A}$ has to produce a valid commitment to $\tilde{\alpha}_0 = \alpha_0 + e/\sum_{j\in\mathsf{H}}(\beta_j + \delta_0)$ with the same probability.

Note, that $\mathcal{A}$ interacts in both the left and right session of NMCOM with the same party (actually all parties simultaneously, since everything is broadcast). But this means that $\mathcal{A}$ cannot let the left session finish before starting the right session, i.e. $\mathcal{A}$ is a synchronizing adversary against NMCOM. Concretely, in the left session, $\mathcal{S}_2$ commits to $(\hat{\mathbf{p}}_j(x), \hat{\mathbf{r}}_j(x), \hat{\mathbf{r}}'_j(x)) = (\alpha_j, \beta_j, \delta_j)$ for $j \in \mathsf{H}$, while $\mathcal{A}$ commits in the right session to $(\alpha_0, \{\beta_0^i\}_{i\in[n]}, \delta_0)$ and $(\alpha_i, \beta_i, \delta_i)$ for $i \in \mathsf{A}$ to $\mathcal{S}_2$. Further, the number of sessions that $\mathcal{A}$ can start is bounded in advance at $n-1$, i.e. it is sufficient to consider bounded-concurrency.

Consider the two views

$$\mathsf{Real} = \{\hat{\mathbf{s}}_j, \{\mathsf{com}_j\}\}_{j\in\mathsf{H}}, \qquad \mathsf{Rand} = \{\hat{\mathbf{s}}_j, \{\widehat{\mathsf{com}}_j\}\}_{j\in\mathsf{H}},$$

where $\mathsf{com}_j \leftarrow \mathsf{NMCOM.Commit}(\alpha_j, \beta_j)$ and $\widehat{\mathsf{com}}_j \leftarrow \mathsf{NMCOM.Commit}(0)$. Real corresponds to a real protocol view of $\mathcal{A}$ before committing itself[10].

Obviously, $\mathsf{Real} \approx_c \mathsf{Rand}$ if NMCOM is non-malleable. However, we will argue that $\mathcal{A}$ cannot output a valid commitment on $\tilde{\alpha}_0$ except with negligible probability, i.e.

$$\Pr[(\mathsf{com}_0^*, \mathsf{unv}_0^*, (\tilde{\alpha}_0, \{\tilde{\beta}_0^i\}_{i\in[n]}, \tilde{\delta}_0) \leftarrow \mathcal{A}(\mathsf{Rand}) \wedge \mathsf{valid}] \leq \mathsf{negl}(\kappa),$$

where valid is the event that $\mathsf{NMCOM.Open}(\mathsf{com}_0^*, \mathsf{unv}_0^*, (\tilde{\alpha}_0, \{\tilde{\beta}_0^i\}_{i\in[n]}, \delta_0) = 1$. We first observe that $\hat{\mathbf{p}}_j$ and $\hat{\mathbf{r}}_j$ for $j \in \mathsf{H}$ cannot be obtained by $\mathcal{A}$ via $\hat{\mathbf{s}}_j = \hat{\mathbf{p}}_j \cdot \hat{\mathbf{r}}_0^j - \hat{\mathbf{u}}_j$. The polynomial $\hat{\mathbf{s}}_j$ itself is uniformly random over the choice of $\hat{\mathbf{u}}_j$, and the only equation that $\mathcal{A}$ has is $\hat{\mathbf{p}}_\cap = \sum_{i\in\mathsf{A}}(\mathbf{s}_i + \mathbf{s}_0^i) + \sum_{j\in\mathsf{H}}(\mathbf{s}_j + \mathbf{s}_0^j) = \sum_{i\in\mathsf{A}}(\hat{\mathbf{p}}_0 \cdot (\hat{\mathbf{r}}_i + \hat{\mathbf{r}}'_0) + \hat{\mathbf{p}}_i \cdot (\hat{\mathbf{r}}_0^i + \hat{\mathbf{r}}'_i)) + \sum_{j\in\mathsf{H}}(\hat{\mathbf{p}}_0 \cdot (\hat{\mathbf{r}}_j + \hat{\mathbf{r}}'_0) + \hat{\mathbf{p}}_j \cdot (\hat{\mathbf{r}}_0^j + \hat{\mathbf{r}}'_j))$. Note, that the honest $\hat{\mathbf{r}}_j, \hat{\mathbf{r}}'_j$ have degree d and therefore hide $\hat{\mathbf{p}}_j$. Further, the commitments com_j contain the value 0 and are therefore independent of $\hat{\mathbf{p}}_j$ and $\hat{\mathbf{r}}_j$. Thus, the probability that $\mathcal{A}$ obtains a commitment on $\tilde{\alpha}_0$ is negligible.

But since $\mathsf{Real} \approx_c \mathsf{Rand}$, we also get that

$$\Pr[(\mathsf{com}_0^*, \mathsf{unv}_0^*, (\tilde{\alpha}_0, \{\tilde{\beta}_0^i\}_{i\in[n]}, \tilde{\delta}_0) \leftarrow \mathcal{A}(\mathsf{Real}) \wedge \mathsf{valid}] \leq \mathsf{negl}(\kappa),$$

which contradicts our assumption that $\mathcal{A}$ produces the commitment with non-negligible probability ϵ.

In conclusion, Hybrid 1 and Hybrid 2 are computationally indistinguishable. □

Lemma 6.1.2 *Assume that Π_{CT} provides a uniformly random x with computational security. Then Hybrid 2 and Hybrid 3 are computationally indistinguishable.*

[10] For ease of notation, here we assume that the commitments are completely sent before $\mathcal{A}$ commits himself. The very same argument also holds if $\mathcal{A}$ only received synchronized messages of com_j and has to start committing concurrently.

Proof. Assume that there exists an environment $\mathcal{Z}$ that distinguishes Hybrids 2 and 3 with non-negligible probability ϵ. In order to distinguish Hybrid 2 and Hybrid 3 $\mathcal{Z}$ has to provide two distinct polynomials for a malicious P_0 and still pass the check in the protocol. Then we can construct from $\mathcal{Z}$ an adversary $\mathcal{B}$ that predicts the outcome of Π_{CT} with non-negligible probability.

Let $\mathcal{A}$ input w.l.o.g. two polynomials $\hat{\mathbf{p}}_0^1 \neq \hat{\mathbf{p}}_0^2$. The check with the random challenge x allows $\mathcal{A}$ to send only one value α_0^*, but from Lemma 6.1.1 we know that it has to hold that $\alpha_0^* = \hat{\mathbf{p}}_0^1(x) = \hat{\mathbf{p}}_0^2(x)$, or the check will fail. First note that two polynomials of degree m agree in a random point x over $\mathbb{F}$ only with probability $m/|\mathbb{F}|$, which is negligible in our case.

Our adversary $\mathcal{B}$ proceeds as follows. It simulates the protocol for $\mathcal{Z}$ according to $\mathcal{S}_1$ up to the point where $\mathcal{S}_1$ learns the polynomials $\hat{\mathbf{p}}_0^1 \neq \hat{\mathbf{p}}_0^2$. $\mathcal{B}$ sets $\mathbf{f} = \hat{\mathbf{p}}_0^1 - \hat{\mathbf{p}}_0^2$ and computes the roots $\gamma_1, \ldots, \gamma_m$ of $\mathbf{f}$. One of these roots has to be the random point x, otherwise $\hat{\mathbf{p}}_0^1(x) - \hat{\mathbf{p}}_0^2(x) \neq 0$ and the check in Π_{MPSI} fails (since there is only one α_0^*). $\mathcal{B}$ picks a random index $l \in [m]$ and predicts the output of the coin-flip as γ_l. Thus, $\mathcal{B}$ predicts the outcome of the coin-toss correctly with probability ϵ/m, which is non-negligible. This contradicts the security of Π_{CT}.

This establishes the indistinguishability of Hybrid 2 and Hybrid 3. □

Lemma 6.1.3 *Hybrid 4 and Hybrid 5 are statistically close.*

Proof. A malicious environment $\mathcal{Z}$ can distinguish Hybrid 4 and Hybrid 5 if (a) the extracted inputs are incorrect or if (b) the simulated messages can be distinguished from real ones.

Concerning (a), if the inputs were not correctly extracted, $\mathcal{Z}$ would receive different outputs in the two hybrids. We already established that the extracted polynomial $\hat{\mathbf{p}}_0$ is correct. Similarly, the extracted $\hat{\mathbf{r}}_0^j$ are also correct. By implication this also ensures that the intermediate intersection is computed correctly.

We argue that the correction of the intersection is also correct, i.e. the set $\hat{S}'_\cap$ is computed correctly and in particular it holds that $(\mathcal{M}_0(\mathbf{p}^*_\cap) \cap \mathcal{M}_0(\hat{\mathbf{p}}_j)) \subseteq \hat{S}_\cap$. First of all, we have to show that the intermediate intersection polynomial $\hat{\mathbf{p}}_{\mathsf{int}}$ actually provides the intersection for all parties. For all P_j it holds with overwhelming probability:

$$\begin{aligned}
\mathcal{M}_0(\hat{\mathbf{p}}_j) \cap \mathcal{M}_0(\hat{\mathbf{p}}_{\mathsf{int}}) &= \mathcal{M}_0(\hat{\mathbf{p}}_j) \cap \mathcal{M}_0\Big(\sum_{j \in \mathsf{H}} (\hat{\mathbf{p}}_0 \cdot (\hat{\mathbf{r}}_j + \hat{\mathbf{r}}'_0) + \hat{\mathbf{p}}_j \cdot (\hat{\mathbf{r}}_0^j + \hat{\mathbf{r}}'_j))\Big) \\
&\overset{Lemma\ 2.2}{=} \mathcal{M}_0(\hat{\mathbf{p}}_j) \cap \Big(\bigcap_{j \in \mathsf{H}} \mathcal{M}_0((\hat{\mathbf{p}}_0 \cdot (\hat{\mathbf{r}}_j + \hat{\mathbf{r}}'_0) + \hat{\mathbf{p}}_j \cdot (\hat{\mathbf{r}}_0^j + \hat{\mathbf{r}}'_j)))\Big) \\
&\overset{Lemma\ 2.4}{=} \mathcal{M}_0(\hat{\mathbf{p}}_j) \cap \Big(\bigcap_{j \in \mathsf{H}} \mathcal{M}_0(\hat{\mathbf{p}}_0) \cap \mathcal{M}_0(\hat{\mathbf{p}}_j)\Big) \\
&= \hat{S}_\cap
\end{aligned}$$

Once the intermediate intersection is computed, the adversary can only add an update polynomial $\hat{\mathbf{p}}_{\mathsf{upt}}$ to get the final intersection polynomial $\mathbf{p}^*_\cap$. It remains to show that this final intersection does not include any points that were not already in the intermediate intersection for any of the parties' polynomials $\hat{\mathbf{p}}_j$.

For this, we consider the intersection of every honest party's (unknown) input $\mathbf{p}_j$ with the intersection. It has to hold that $\hat{S}'_\cap \subseteq \hat{S}_\cap$ for all P_j except with negligible probability. Here we require that $\Pr[x \in \mathcal{M}_0(\mathbf{p}_j)] \leq \mathsf{negl}(|\mathbb{F}|)$ for all x, i.e. the adversary can only guess an element of P_j's input set.

$$\begin{aligned}
\mathcal{M}_0(\hat{\mathbf{p}}_j) \cap \mathcal{M}_0(\mathbf{p}^*_\cap) &= \mathcal{M}_0(\hat{\mathbf{p}}_j) \cap (\mathcal{M}_0(\hat{\mathbf{p}}_{\mathsf{int}} + \hat{\mathbf{p}}_{\mathsf{upt}})) \\
&\overset{Lemma\ 2.2}{=} \mathcal{M}_0(\hat{\mathbf{p}}_j) \cap (\mathcal{M}_0(\hat{\mathbf{p}}_{\mathsf{int}}) \cap \mathcal{M}_0(\hat{\mathbf{p}}_{\mathsf{upt}})) \\
&= \mathcal{M}_0(\hat{\mathbf{p}}_j) \cap (\hat{S_\cap} \cap \mathcal{M}_0(\hat{\mathbf{p}}_{\mathsf{upt}})) \\
&\subseteq \mathcal{M}_0(\hat{\mathbf{p}}_j) \cap \hat{S_\cap} = \hat{S_\cap}
\end{aligned}$$

Therefore, $\hat{S}'_\cap \subseteq \hat{S}_\cap$, and the output in both hybrids is identical.

Regarding (b), we make the following observations. Since $\mathcal{S}_4$ sends $\hat{\mathbf{s}}'_j = \hat{\mathbf{s}}_j - \mathbf{u}_j + \sum_{i \neq j} \mathbf{v}_{ij}$, the value $\hat{\mathbf{s}}'_j$ is uniformly random over the choice of $\mathbf{u}_j$ (and over $\sum \mathbf{v}_{ij}$, if $t \leq n-2$). Therefore, the simulation of $\hat{\mathbf{s}}'_j$ is identically distributed to Hybrid 4.

Similarly, we have:

$$\sum_{j \in \mathsf{H}} (\hat{\mathbf{s}}'_j + \hat{\mathbf{s}}^j_0) = \sum_{j \in \mathsf{H}} (\hat{\mathbf{p}}_0 \cdot (\hat{\mathbf{r}}_j + \hat{\mathbf{r}}'_0) + \hat{\mathbf{p}}_j \cdot (\hat{\mathbf{r}}^j_0 + \hat{\mathbf{r}}'_j)) \quad [+ \sum_{i \in \mathsf{A}, j \in \mathsf{H}} \mathbf{v}_{ij}]$$

We can ignore the $\mathbf{v}_{ij}$ values, since these are known to $\mathcal{A}$. The sum is uniform over the choice of the $\hat{\mathbf{r}}_j, \hat{\mathbf{r}}'_j$ apart from the points $\gamma \in \hat{S}_\cap$ (since $\mathcal{F}_{\mathrm{OPA}}$ guarantees that $\hat{p}_0 \neq 0$) and therefore identically distributed to Hybrid 5, since the extraction in correct. □

P_0 **is honest:** The proof itself is very similar to the proof of a corrupted P_0. It is actually easier to simulate in the sense that $\mathcal{S}_{\bar{P}_0}$ observes the inputs of all malicious parties. In this sense, Π_{MPSI} actually realises $\mathcal{F}_{\mathrm{MPSI}}$ if P_0 is honest, since no adjustment of the output is necessary. We refer to the full version [15] of the paper for the proof.

Efficiency. The setup, i.e. the distribution of seeds, has communication complexity $O(n^2\kappa)$. The oblivious addition of the polynomials has communication overhead of $O(nm\kappa)$. The check phase first requires a multi-party coin-toss.

In the full version of this paper [15], we sketch a coin-tossing protocol in combination with an OLE-based commitment scheme (replacing the non-malleable commitment for better efficiency) that results in an asymptotic communication overhead of $O(n^2\kappa)$ for the check and the coin-toss phase. Combining this with the above observations, Π_{MPSI} has communication complexity $O((n^2 + nm)\kappa)$ in the $\mathcal{F}_{\mathrm{OLE}}$-hybrid model.

For concrete instantiations of $\mathcal{F}_{\mathrm{OLE}}$, the OLE protocol of Ghosh et al. [14] has a constant communication overhead per OLE. In summary, the complete protocol has communication complexity $O((n^2+nm)\kappa)$, which is asymptotically optimal for $m \geq n$.

Similar to the two-party case, the computational cost is dominated by the cost of polynomial interpolation. In particular, the central party has to run the two-party protocol n times, which leads to a computational overhead of $O(nm \log m)$ multiplications. The other parties basically have the same computational overhead as in the two-party case. □

7 Performance analysis

In this section, we give an estimation of the communication efficiency with concrete parameters and provide a comparison with existing results. For this, we simply count the number of field elements that have to be sent for the protocols. We first look at the communication overhead of the OLE primitive. Instantiated with the result by Ghosh et al. [14], each OLE has an overhead of 64 field elements including OT extension (32 without), which translates to 256 field elements per item per OPA. The factor 4 stems from the fact that OPA needs $2d$ OLE to compute a degree d output, and OLE+ requires two OLE per instance.

Protocol	Communication cost	
	$m = 2^{16}$	$m = 2^{20}$
[36](EC-ROM)	79 MB ($\kappa = 40$)	1.32 GB ($\kappa = 40$)
[36](DE-ROM)	61 MB ($\kappa = 40$)	1.07 GB ($\kappa = 40$)
[36](SM, $\sigma = 40$)	451 MB ($\kappa = 40$)	$>$ 7.7 GB ($\kappa = 40$)
[36](SM, $\sigma = 64$)	1.29 GB ($\kappa = 40$)	22.18 GB ($\kappa = 40$)
Ours ($\sigma = 40$)	80 MB ($\kappa = 40$)	1.25 GB ($\kappa = 40$)
Ours ($\sigma = 64$)	128 MB ($\kappa = 64$)	2 GB ($\kappa = 64$)

Table 2. Comparison of two-party PSI protocols from [36] for input-size $m = \{2^{16}, 2^{20}\}$, where κ denotes statistical security parameter, σ denotes size of each item in bits, SM denotes standard model, ROM denotes random oracle model.

2-party PSI. To get a feeling for the concrete communication efficiency of our two-party protocol, we compare it with the recent maliciously UC-secure protocols from [36]. These protocols give only one-sided output, whereas our protocol gives two-sided output. However, OPA is sufficient for one-sided PSI, consequently a one-sided PSI would cost 256 field elements per item in our case.

Table 2 clearly shows that the communication overhead of our protocol is significantly less than the standard model (SM) protocol from [36]. Note that our instantiation is also secure in SM, given $O(\kappa)$ base OTs. Like [36] we use the OT-extension protocol from [31] for the instantiation. Even if we compare our result to the ROM approach of [36], we achieve fairly competitive communication efficiency.

One should consider that in the ROM there exist other PSI protocols with linear communication complexity [8,23]. The concrete bandwidth of those protocols are much less than our specific instantiation, for example for sets of 2^{20}

elements the total communication cost of [8] is about 213 MB[11]. Further [23] has lower bandwidth than [8]. However, in both the cases communication efficiency comes at the cost of huge computational expenses due to lots of public key operations. We believe that the simple field arithmetic of our protocols (including the cost of the OLE of [14]) does not incur such a drawback in practice.

Protocol	Parties	Corr.	Comm. 2^{20} elements
[27] (passive)	n	$n-1$	$(n-1) \cdot 467$ MB
Ours (active)	n	$n-1$	≈ 2.5 GB
Ours (passive)	n	$n-1$	≈ 1.25 GB

Table 3. Comparison of communication overhead per party of MPSI protocol with [27] for 2^{20} elements with 40 bit statistical security, without the cost for OT extension.

Multi-party PSI. To the best of our knowledge, there are currently no maliciously secure MPSI implementations with which we could compare our result. A direct comparison with the *passively* secure MPSI from [27], however, directly shows the difference in asymptotic behaviour to our result. Their communication costs per party increase with the number of parties, whereas it remains constant in our case (except for the central party). If we average over all parties, the central party's overhead can be distributed over all parties, which at most doubles the average communication cost per party (cf. Table 3). We can upper bound the communication cost per party by 2.5 GB for 2^{20} elements (excluding the cost for OT extension in order to get comparable results to [27]). From the table we can deduce that with only 6 parties, our actively secure protocol is more efficient than their passive one. Replacing the actively secure OPA in our MPSI protocol with the passively secure one yields a passively secure MPSI protocol. We gain another factor of 2 in communication efficiency and our construction is more efficient than [27] starting from 4 parties.

References

1. Applebaum, B., Damgård, I., Ishai, Y., Nielsen, M., Zichron, L.: Secure arithmetic computation with constant computational overhead. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 223–254. Springer, Heidelberg (Aug 2017)
2. Ben-Sasson, E., Fehr, S., Ostrovsky, R.: Near-linear unconditionally-secure multiparty computation with a dishonest minority. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 663–680. Springer, Heidelberg (Aug 2012)
3. Borodin, A., Moenck, R.: Fast modular transforms. J. Comput. Syst. Sci. 8(3), 366–386 (Jun 1974), http://dx.doi.org/10.1016/S0022-0000(74)80029-2
4. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd FOCS. pp. 136–145. IEEE Computer Society Press (Oct 2001)

[11] For reference see Figure 8 of [36]

5. Cheon, J.H., Jarecki, S., Seo, J.H.: Multi-party privacy-preserving set intersection with quasi-linear complexity. IEICE Transactions 95-A(8), 1366–1378 (2012)
6. Cramer, R., Damgård, I., Ishai, Y.: Share conversion, pseudorandom secret-sharing and applications to secure computation. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 342–362. Springer, Heidelberg (Feb 2005)
7. Dachman-Soled, D., Malkin, T., Raykova, M., Yung, M.: Efficient robust private set intersection. In: Abdalla, M., Pointcheval, D., Fouque, P.A., Vergnaud, D. (eds.) ACNS 09. LNCS, vol. 5536, pp. 125–142. Springer, Heidelberg (Jun 2009)
8. De Cristofaro, E., Kim, J., Tsudik, G.: Linear-complexity private set intersection protocols secure in malicious model. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 213–231. Springer, Heidelberg (Dec 2010)
9. De Cristofaro, E., Tsudik, G.: Practical private set intersection protocols with linear complexity. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, pp. 143–159. Springer, Heidelberg (Jan 2010)
10. Dong, C., Chen, L., Wen, Z.: When private set intersection meets big data: an efficient and scalable protocol. In: Sadeghi, A.R., Gligor, V.D., Yung, M. (eds.) ACM CCS 13. pp. 789–800. ACM Press (Nov 2013)
11. Döttling, N., Ghosh, S., Nielsen, J.B., Nilges, T., Trifiletti, R.: TinyOLE: Efficient actively secure two-party computation from oblivious linear function evaluation. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 17. pp. 2263–2276. ACM Press (Oct / Nov 2017)
12. Freedman, M.J., Hazay, C., Nissim, K., Pinkas, B.: Efficient set intersection with simulation-based security. Journal of Cryptology 29(1), 115–155 (Jan 2016)
13. Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 1–19. Springer, Heidelberg (May 2004)
14. Ghosh, S., Nielsen, J.B., Nilges, T.: Maliciously secure oblivious linear function evaluation with constant overhead. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part I. LNCS, vol. 10624, pp. 629–659. Springer, Heidelberg (Dec 2017)
15. Ghosh, S., Nilges, T.: An algebraic approach to maliciously secure private set intersection. Cryptology ePrint Archive, Report 2017/1064 (2017), `https://eprint.iacr.org/2017/1064`
16. Hallgren, P.A., Orlandi, C., Sabelfeld, A.: Privatepool: Privacy-preserving ridesharing. In: 30th IEEE Computer Security Foundations Symposium, CSF 2017, Santa Barbara, CA, USA, August 21-25, 2017. pp. 276–291 (2017)
17. Hazay, C.: Oblivious polynomial evaluation and secure set-intersection from algebraic PRFs. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 90–120. Springer, Heidelberg (Mar 2015)
18. Hazay, C., Nissim, K.: Efficient set operations in the presence of malicious adversaries. Journal of Cryptology 25(3), 383–433 (Jul 2012)
19. Hazay, C., Venkitasubramaniam, M.: Scalable multi-party private set-intersection. In: Fehr, S. (ed.) PKC 2017, Part I. LNCS, vol. 10174, pp. 175–203. Springer, Heidelberg (Mar 2017)
20. Huang, Y., Evans, D., Katz, J.: Private set intersection: Are garbled circuits better than custom protocols? In: NDSS 2012. The Internet Society (Feb 2012)
21. Huberman, B.A., Franklin, M., Hogg, T.: Enhancing privacy and trust in electronic communities. In: Proceedings of the 1st ACM Conference on Electronic Commerce. pp. 78–86. EC '99 (1999)

22. Ishai, Y., Prabhakaran, M., Sahai, A.: Secure arithmetic computation with no honest majority. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 294–314. Springer, Heidelberg (Mar 2009)
23. Jarecki, S., Liu, X.: Fast secure computation of set intersection. In: Garay, J.A., Prisco, R.D. (eds.) SCN 10. LNCS, vol. 6280, pp. 418–435. Springer, Heidelberg (Sep 2010)
24. Kamara, S., Mohassel, P., Raykova, M., Sadeghian, S.S.: Scaling private set intersection to billion-element sets. In: Christin, N., Safavi-Naini, R. (eds.) FC 2014. LNCS, vol. 8437, pp. 195–215. Springer, Heidelberg (Mar 2014)
25. Kissner, L., Song, D.X.: Privacy-preserving set operations. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 241–257. Springer, Heidelberg (Aug 2005)
26. Kolesnikov, V., Kumaresan, R., Rosulek, M., Trieu, N.: Efficient batched oblivious PRF with applications to private set intersection. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 16. pp. 818–829. ACM Press (Oct 2016)
27. Kolesnikov, V., Matania, N., Pinkas, B., Rosulek, M., Trieu, N.: Practical multi-party private set intersection from symmetric-key techniques. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 17. pp. 1257–1272. ACM Press (Oct / Nov 2017)
28. Meadows, C., Mutchler, D.: Matching secrets in the absence of a continuously available trusted authority. IEEE Transactions on Software Engineering SE-13(2), 289–292 (Feb 1987)
29. Naor, M., Pinkas, B., Sumner, R.: Privacy preserving auctions and mechanism design. In: EC. pp. 129–139 (1999)
30. Narayanan, A., Thiagarajan, N., Lakhani, M., Hamburg, M., Boneh, D.: Location privacy via private proximity testing. In: NDSS 2011. The Internet Society (Feb 2011)
31. Orrù, M., Orsini, E., Scholl, P.: Actively secure 1-out-of-N OT extension with application to private set intersection. In: Handschuh, H. (ed.) CT-RSA 2017. LNCS, vol. 10159, pp. 381–396. Springer, Heidelberg (Feb 2017)
32. Pinkas, B., Schneider, T., Segev, G., Zohner, M.: Phasing: Private set intersection using permutation-based hashing. In: 24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015. pp. 515–530 (2015)
33. Pinkas, B., Schneider, T., Zohner, M.: Faster private set intersection based on OT extension. In: Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014. pp. 797–812 (2014)
34. Pinkas, B., Schneider, T., Zohner, M.: Scalable private set intersection based on OT extension. ACM Trans. Priv. Secur. 21(2), 7:1–7:35 (2018)
35. Rindal, P., Rosulek, M.: Improved private set intersection against malicious adversaries. In: Coron, J., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part I. LNCS, vol. 10210, pp. 235–259. Springer, Heidelberg (Apr / May 2017)
36. Rindal, P., Rosulek, M.: Malicious-secure private set intersection via dual execution. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 17. pp. 1229–1242. ACM Press (Oct / Nov 2017)
37. Sang, Y., Shen, H.: Privacy preserving set intersection based on bilinear groups. In: Proceedings of the Thirty-first Australasian Conference on Computer Science - Volume 74. pp. 47–54. ACSC '08 (2008)
38. Shamir, A.: On the power of commutativity in cryptography, pp. 582–595. Springer Berlin Heidelberg, Berlin, Heidelberg (1980)