# Fast Secure Computations on Shared Polynomials and Applications to Private Set Operations

**Pascal Giorgi**
LIRMM, Univ. Montpellier, CNRS, France

**Fabien Laguillaumie**
LIRMM, Univ. Montpellier, CNRS,, France

**Lucas Ottow**
LIRMM, Univ. Montpellier, CNRS, France

**Damien Vergnaud**
LIP6, Sorbonne University, CNRS, France

## Abstract

Secure multi-party computation aims to allow a set of players to compute a given function on their secret inputs without revealing any other information than the result of the computation. In this work, we focus on the design of secure multi-party protocols for shared polynomial operations. We consider the classical model where the adversary is honest-but-curious, and where the coefficients (or any secret values) are either encrypted using an additively homomorphic encryption scheme or shared using a threshold linear secret-sharing scheme. Our protocols terminate after a constant number of rounds and minimize the number of secure multiplications.

In their seminal article at PKC 2006, Mohassel and Franklin proposed constant-rounds protocols for the main operations on (shared) polynomials. In this work, we improve the *fan-in multiplication of nonzero polynomials*, the *multi-point polynomial evaluation* and the *polynomial interpolation* (on secret points) to reach a quasi-linear complexity (instead of quadratic in Mohassel and Franklin's work) in the degree of shared input/output polynomials.

Computing with shared polynomials is a core component of several multi-party protocols for privacy-preserving operations on private sets, like the *private disjointness test* or the *private set intersection*. Using our new protocols, we are able to improve the complexity of such protocols and to design the first variants which always return a correct result.

**2012 ACM Subject Classification** Theory of computation → Cryptographic protocols; Security and privacy → Information-theoretic techniques

**Keywords and phrases** Multi-party computation, polynomial operations, privacy-preserving set operations

**Digital Object Identifier** 10.4230/LIPIcs.ITC.2024.11

**Related Version** *Full Version*: `https://eprint.iacr.org/2024/470`

**Funding** This work was supported by the France 2030 ANR project SecureCompute ANR-22-PECY-0003, the French ANR SANGRIA project ANR-21-CE39-0006, and the French project CRYPTANALYSE ANR-22-PECY-0010.

**Acknowledgements** We would like to thank the anonymous reviewers for their helpful comments.

## 1 Introduction

Secure multi-party computation (MPC), which dates back to fundamental works by Yao [36] and Goldreich, Micali, and Widgerson [19], is a family of cryptographic techniques that enables parties to jointly compute a function over their private inputs while keeping those inputs confidential. This approach ensures that none of the participating parties need to reveal any information on their data to one another, yet they can still obtain the desired


computation result. Unconditionally secure MPC protocols were first proposed by Ben-Or, Goldwasser, and Widgerson [2] and Chaum, Crépeau and Damgård [5]. On the other hand, several computationally secure protocols have been proposed, relying on many different techniques, like verifiable secret sharing [8, 15] or linearly homomorphic encryption [12, 9].

In the realm of computer algebra, polynomial evaluation and interpolation algorithms stand out as versatile tools with applications spanning numerous domains [14]. In 2006, Mohassel and Franklin [27] proposed several secure MPC protocols for various polynomial operations where the polynomial coefficients are private inputs of the parties. It is the main goal of this paper to improve some of their protocols and to illustrate their usability *via* several applications to private set operations protocols. All of the presented protocols are considered in the *honest-but-curious* model.

## 1.1 Secure computation on shared data

The computation on shared elements dates back to the works of Ben-Or, Goldwasser, and Widgerson [2] and Bar-Ilan and Beaver [1]. In the general setting, it involves $m$ players that want to compute a function (seen as an arithmetic circuit) over secret inputs (that we will consider as elements of $\mathbb{F}_q$). These secret inputs are assumed to be shared between the parties before any computation. This can done either via a secret-sharing scheme (such as an additive secret-sharing scheme or Shamir's secret-sharing scheme [35], in the information theoretical model) or via a threshold linearly homomorphic scheme (such as a threshold variant of Paillier's encryption [32] or CL encryption [3], in the computational model). Parties can compute any arithmetic circuit on the shared inputs by combining several basic operations on the shared data (additions and multiplications). While the work in [2, 1] addresses the general problem of computing any circuit, many other results have been provided to improve the efficiency on specific problems [7, 27, 11, 10, 28]. Our work follows the latter line of research without relying on any specific underlying secret-sharing scheme or any threshold homomorphic encryption scheme. One of our main objectives is to derive protocols in a general framework, encompassing both theoretical and computational models, that achieve a constant number of rounds of communication between each party while minimizing the total amount of exchanged data.

Let us denote by $[x]$ an element $x \in \mathbb{F}_q$ that is shared among the parties. In order to simplify the presentation, we will often assume that our protocols are implemented using a secret-sharing scheme. Therefore, we will note by $[x]_j$ the part of the secret belonging to the $j$-th player. Our framework assumes that the players are able to do elementary operations, such as *addition of two shares*, *scalar multiplication with public value*, and *multiplication of two shares*, in a constant number of rounds.

In the information-theoretic model, the first two operations do not require any communication as each player $j$ can just compute individually: $[a+b]_j = [a]_j + [b]_j$ and $\lambda[a]_j$ to get a share of the results $a+b$ and $\lambda a$. Computing the product $[ab]$ from the shares of $[a], [b]$ is trickier, and cannot be done locally. Several interactive solutions exist: the BGW protocol [2], or Beaver triples [1] offer alternatives which can be both done in a constant number of rounds.

In the case of threshold linearly homomorphic schemes, the first two operations can also be done locally on encrypted data, and solutions exist to compute the product of two encrypted field elements. Finally, our framework further requires that the sharing method allows for secure constant-round methods to share a constant element (for example to share the values 1 or 0) and to share a uniformly random (unknown) element.

The complexity measure of our multi-party protocols will be given as the number of "secure multiplication" in the base field $\mathbb{F}_q$, *i.e.* multiplication of two shared elements. Since it is the only operation requiring communication between players, this will express the communication complexity of our protocols. To guarantee that our protocols still run in constant-round, we will extensively use parallel executions of constant-round protocols.

We are considering the *honest-but-curious* model. Therefore, privacy is only a matter of checking that when a shared value is revealed, the revealed value is uniformly random and independent of the value of the secret inputs. We achieve security for any of the protocols presented in this paper as long as the elementary operations presented above (*i.e.* addition and multiplications on shared elements of $\mathbb{F}_q$) can be composed in parallel and remain secure. This follows a long line of work which are based on the same assumptions in secure linear algebra [1, 7, 31, 24, 10, 28] or in secure polynomial computation [27].

## 1.2 Toolbox for Secure Polynomial Computation

In their seminal paper, Mohassel and Franklin [27] proposed the very first protocols for secure polynomial multiplication, division with remainder, and polynomial interpolation. They considered the scenario in which the coefficients of the involved polynomials, evaluation points, or values are shared. For a polynomial $f = \sum_{k=0}^{d-1} f_k X^k$, we denote by $[f]$ a sharing of the polynomial, that is a collection of sharings of its coefficients $[f_0], \ldots, [f_{d-1}]$.

The goal of Mohassel and Franklin was to propose efficient protocols with good communication and round complexities. For example, in the case of polynomial interpolation, assuming that the $m$ parties hold shares (or ciphertexts) of $n \geq 1$ pairs $([x_i], [y_i]) \in \mathbb{F}_q^2$ for $i \in \{1, \ldots, n\}$ (with $x_i \neq x_j$ for $i \neq j$), they proposed a protocol with a constant number of rounds and communication complexity of $\mathcal{O}(n^2)$ multiplications. Note that this protocol has remained the most efficient since 2006.

As a first contribution, we present new protocols for efficient and secure operations on shared polynomials. Our model is identical to the one used by Mohassel and Franklin. In particular, our proposals can be implemented using a threshold linearly homomorphic encryption scheme (and in this case achieve semi-honest computational security) or using threshold linear secret sharing (and achieve then semi-honest information-theoretic security).

Our protocols are parameterized by an integer constant $\tau$. Our protocols have a number of rounds proportional to $\tau$ and they achieve a quasi-optimal communication complexity, i.e. exponential in $1 + 1/\tau$. More precisely:

- We present a first protocol (**FastPolyFanIn**) where the parties are given shares of $n$ non-zero polynomials $[f_1], \ldots, [f_n]$ in $\mathbb{F}_q[X]$ of degree less than $d$ and compute shares of the polynomial $[f_1 \times \cdots \times f_n]$ of degree at most $nd$. Mohassel and Franklin [27] proposed a constant-round protocol with communication complexity of $\mathcal{O}(n^2 d)$ multiplications. Our improved protocol has communication complexity of only $\mathcal{O}(\tau n^{1+1/\tau} d)$ multiplications and $\mathcal{O}(\tau)$ rounds.
- Our second protocol (**FastEval**) allows the parties sharing a polynomial $[f] \in \mathbb{F}_q[X]$ of degree at most $n$ and shared points $[\alpha_1], \ldots, [\alpha_n]$ in $\mathbb{F}_q$ to compute shares of the $n$ evaluations $[f(\alpha_1)], \ldots, [f(\alpha_n)]$. This protocol achieves communication complexity of $\mathcal{O}(\tau n^{1+1/\tau})$ multiplications and $\mathcal{O}(\tau)$ rounds. The previously best-known protocol has communication complexity $\mathcal{O}(n^2)$, see [10].
- Our third protocol (**FastInterpol**) performs polynomial interpolation on shared values (as in Mohassel-Franklin protocol), with communication complexity of $\mathcal{O}(\tau n^{1+1/\tau})$ multiplications and $\mathcal{O}(\tau)$ rounds.

**Table 1** Summary of our improvements on operations on shared polynomials ($n$ is the number of polynomials for unbounded fan-in multiplication, $d$ bounds the degree of the polynomials and $\tau$ is a predetermined constant).

| | **Our work** | **Mohassel-Franklin ([27])** |
|---|---|---|
| Unbounded fan-in mult. | $\mathcal{O}(\tau n^{1+1/\tau} d)$ | $\mathcal{O}(n^2 d)$ |
| Multi-point evaluation | $\mathcal{O}(\tau n^{1+1/\tau})$ | $\mathcal{O}(n^2)$ |
| Interpolation | $\mathcal{O}(\tau n^{1+1/\tau})$ | $\mathcal{O}(n^2)$ |

All of our protocols are perfectly correct, i.e. the result is always computed correctly. Furthermore, they are valid in the *semi-honest* (i.e. *honest-but-curious*) model. Table 1 summarizes the communication complexity of our protocol compared to existing ones in [27]. Our protocol for unbounded fan-in multiplication of shared polynomials can be used straightforwardly to compute the unbounded fan-in multiplication of $n$ shared elements $[x_1], \ldots, [x_n]$ of $\mathbb{F}_q$ that are not necessarily invertible (i.e. some elements might be zero). This is achieved by setting $[f_1] = [X - x_1], \ldots, [f_n] = [X - x_n]$ and extracting the constant coefficient of the product $[f_1 \ldots f_n]$. This protocol is already mentioned in [7], but it did not improve upon the more general approach of Bar-Ilan and Beaver [1] yielding a constant round protocol with a communication complexity of $\mathcal{O}(n^2)$ secure multiplications. Thanks to our new result, the latter operations can now be achieved in constant rounds but with $\mathcal{O}(\tau n^{1+1/\tau})$ secure multiplications.

## 1.3 Applications to Private Set Operations

As a second contribution, we show that our secure polynomials computation framework can serve to improve protocols on the so-called *privacy-preserving set operations* (*PSO*s), in particular for the general multi-party case involving more than two players.

The setting of PSOs is the following: each participant owns its private input set. The goal is to privately compute a predetermined function on these input sets while revealing no information about each set. We will focus here on some of the most classical functions on the intersection: *emptiness*, *cardinality*, *weighted sum*, or simply revealing the intersection set itself, eventually according to a certain threshold size. *Private Intersection Set* (*PSI*) is a crucial tool in privacy-preserving data analysis and collaborative applications where multiple parties want to discover shared interests, overlaps, or common elements in their datasets without revealing the specific items in their sets.

Many algebraic approaches, mostly based on cryptographic assumptions, allow to provide efficient solutions for this problem. In particular, one can either rely on homomorphic encryption [13, 25] or on oblivious linear evaluation [17, 16] to achieve many of PSO functionalities. Some of the results, notably on *PSI*, have been also proposed without any cryptographic assumptions, achieving security under the information-theoretic model. Note that all these methods rely on the natural representation of a set $\{\alpha_1, \ldots, \alpha_n\} \subset \mathbb{F}_q$ by the degree-$n$ polynomial $f(X) = (X - \alpha_1) \times \cdots \times (X - \alpha_n)$ which allow for instance to recover the intersection as the *greatest common divisor* of many polynomials. It is then very natural to rely on our protocol for distributed secure polynomial computation for dealing efficiently with *PSOs*.

We shall mention that *PSI* is an intensively studied topic and many other methods, not only algebraic, have been designed. We refer the reader to [29] for a nice survey on the numerous approaches to the *PSI* problem. While we are only interested in specific function on the intersection set, the general circuit-*PSI* problem [33, 34] allows to evaluate

**Table 2** Summary of the communication complexities, in terms of secure multiplication, for *Private Disjointness Test* protocols. It is assumed that each player hold a set of $n$ entries, and that $\tau$ is a pre-determined constant.

| | **Comm.** | **Rounds** | **# players** | **Uncond.** |
|---|---|---|---|---|
| Ye et al. [37] | $\mathcal{O}(n^2)$ | $\mathcal{O}(1)$ | 2 | Yes |
| Couteau et al. [6] | $\mathcal{O}(n)$ | $\mathcal{O}(1)$ | 2 | No |
| Chandran et al.[4] | $\mathcal{O}(mn\log^2(m))$ | $\mathcal{O}(\log(m\log n))$ | $m$ | No |
| Sathya Narayanan et al. [30] | $\mathcal{O}(mn^2)$ | $\mathcal{O}(\log(mn))$ | $m$ | Yes |
| Ours (section 4.1) | $\mathcal{O}(mn+\tau n^{1+1/\tau})$ | $\mathcal{O}(\tau)$ | $m$ | Yes |

any function given as a circuit on the intersection set. This generic problem received also a lot of attention, and some protocols might yield the best solution for a particular computation, e.g. testing the emptiness of the intersection set in the multi-party setting [4].

### 1.3.1 Constant-round multi-party protocol for *Private Disjointness Test*

Testing the emptiness of the intersection set is called *Private Disjointness Test* (*PDT*) in the literature and many secure protocols have been proposed to deal efficiently with *PDT*. The seminal work of Freedman, Nissim and Pinkas [13] proposed the first efficient solutions to both the two-party and the multi-party setting. Their protocols are secure against semi-honest adversaries or even malicious in the random oracle model. Their work has been further improved in [23, 21] notably to remove the random oracle hypothesis in the two-party setting. While the proposed protocols are efficient in terms of communication between the parties, the high depth of the protocol makes the need for a linear number of rounds of communication in the input sets' sizes.

The first constant round protocol for *PDT* is due to Ye et al. [37] for both honest-but-curious and malicious adversary cases. Unlike the previous two-party protocols, their protocols are unconditionally secure and only require communication complexity that is quadratic in the input set size. This result is improved under computational assumption by Couteau, Peters, and Pointcheval [6] who provide a two-party protocol for *PDT* achieving constant round, linear communication complexity and being secure against malicious adversaries.

While many works have studied *PDT* in the two-party setting, only very few works have been done in the multi-party setting, notably after the seminal work of Ye et al. [37]. Only the work of Sathya Narayanan *et al* [30] mentioned a dedicated protocol for *PDT* for more than two players. The given protocol is unconditionally secure against a semi-honest adversary, and it requires a logarithmic number of rounds and a quadratic number of communications in the input sets' size. One can achieve a similar result under a computational assumption (from oblivious transfer and cuckoo hashing) using the *circuit-PSI* protocol from [4], but with a number of rounds that is only double-logarithmic in the input sets' size.

While the challenge of designing constant round protocol having only a linear number of communication in the input sets' size is almost done in the two-party setting, the question remains open for the multi-party setting. By using our new multi-point evaluation protocol, we are able to propose an unconditionally secure constant-round protocol, against a semi-honest adversary, that achieves quasi-linear communication complexity in the input sets' size. More precisely, assuming the $m$ players all hold a set of $n$ elements, our protocol achieves a communication complexity of $\mathcal{O}(mn+\tau n^{1+1/\tau})$ multiplications and $\mathcal{O}(\tau)$ communication rounds, where $\tau$ is any non-zero integer constant given to the protocol. Table 2 summarizes the different complexity, security, and number of players for the best-known protocol for *PDT* in this setting.

One can see that our protocol is the first to achieve a constant number of rounds in the multi-party setting. Furthermore, it achieves the lowest number of communications under unconditional security. While our communication cost might be more important than the *circuit-PSI* approach [4] in some cases, the size of the exchanged data will remain in a small $\mathbb{F}_q$, say of size $n$, which should make a difference in practice compared to the larger fields of [4] required to reach the desired computational security.

We shall mention that our solution as well as all the previous works on *PDT* may produce an incorrect result, depending on the chosen implementation. In our case, this probability is negligible if the size of each set $n$ is negligible compared to the domain size $q$. Moreover, this probability is one-sided, and having a wrong result does not reveal information about the inputs. Therefore, the protocol can be repeated to exponentially decrease the error probability.

### 1.3.2 A general framework to solve *PSO*s without any error probability

Among the *Private Set Operations*, *Private Set Intersection* has been the most studied since its introduction by Freedman, Nissim, and Pinkas in [13]. As for *PDT* the authors propose efficient protocols for both the two-party and the multi-party *PSI*, achieving security under the semi-honest adversary model or the malicious one. However, their protocol does not achieve a constant number of rounds of communication. Li and Wu provide in [26] the first constant-round protocol for *PSI*, and their protocol is unconditionally secure both with semi-honest and malicious adversaries, but the communication complexity is not yet optimal. Following a similar idea as [13], Hazay and Venkitasubramaniam propose the first protocol achieving constant-round and a linear communication complexity, under the computational model only.

Kissner and Song [25] extend multi-party protocols on sets beyond *PSI* notably by computing some other functions on the input sets.: e.g. union, element reduction, and intersection cardinality. In the *Cardinality Set Intersection* problem, the goal is to compute the number of elements in the intersection while not revealing other information on the intersection set. The protocol in [25] requires some cryptographic assumption and it involves a linear number of rounds of communication in the input sets size. The article [30] mentioned a first solution for *Cardinality Set Intersection* with unconditional security under passive and active adversaries, achieving a logarithmic number of communication rounds.

*T-PSI* (*Threshold Private Set Intersection*) is a variant of *PSI* in which the intersection is revealed only if its size surpasses a given threshold $t$. Gosh and Nilges present in [16] a first solution for *Threshold-PSI* in the multi-party setting. Their protocol is based on *Oblivious Linear Evaluations* and achieves unconditional security against malicious adversaries. The communication complexity of their protocol remains however quadratic in the size of each input set. In the two-party case, by using cryptographic assumption (namely the existence of fully homomorphic encryption), Ghosh and Simkin [17] managed to achieve a sub-linear communication complexity, i.e. the number of communications is quasi-linear in the number of elements that differ between the two sets. This result was extended to the multi-party setting in [18] using other cryptographic assumptions (the existence of linear homomorphic encryption and obvious transfer).

One of the most recent problems with private sets is the *Private Intersection Sum* problem. The setting, introduced in [22], is that one party has a set of elements together with some weight for each element, and he wants to compute the sum of the weight of all elements in common with a set of another party, of course without learning which elements are in common. The protocol proposed in [22] achieves security only in the computational model and for the two-party case. To our knowledge, no specific construction for this problem exists neither in the information-theoretic setting nor for more than two parties.

Unlike *PDT*, all of these contributions achieve a constant number of rounds and an optimal number of communications. However, the proposed protocols are not always secure under the information-theoretic setting, and all of them may fail to produce a correct result. Our work aims at bridging the gap to always achieve security without any cryptographic assumptions and to provide protocols designed for more than two parties, and that are perfectly correct (i.e. no incorrect result can be computed).

Here again, we show that by re-using our multi-point evaluation protocol together with some techniques that enable us to securely deal with boolean formula [11] we can achieve a general framework for *PSO*s that yields constant-round protocol without any incorrect result. As our solution embraces our generic MPC framework, the results are valid for the computational as well as the information-theoretic model. More precisely, we achieve a communication complexity of $\mathcal{O}(\tau m n^{1+1/\tau} + mn \log n \log\log n)$ secure multiplications for *PSI*, *PDT*, *Cardinality Set Intersection*, *Threshold-PSI* and *Private Intersection Sum*.

## 2 Technical overview

In this section, we present a brief overview of the techniques used in our protocols. This covers protocols on shared polynomials and for private set operations. We only focus on the main ideas of how the protocols work, and we give the full technical details in later sections. Without further assumption, the number of parties in the protocols will always be $m$.

### 2.1 Fast operations on shared polynomials

In section 3, we present new constant-round protocols for the unbounded fan-in multiplication of polynomials, multi-point evaluation, and interpolation involving only shared data. Our approach has some similarities with the work of Mohassel and Weinreb from [28] to lower the number of communications for secure linear algebra operations. More precisely, our approach allows us to choose any constant parameter $\tau \in \mathbb{N}^*$ and provides a quasi-optimal communication complexity, *i.e.* exponential in $1 + 1/\tau$, while achieving a constant number of rounds of $\mathcal{O}(\tau)$. Our improvement is based on the observation that the studied operations are strongly regular and thus can be decomposed into several instances of the same problem with smaller entries. Therefore, applying a generalized divide and conquer approach, *i.e.* splitting the problem of size accordingly to $\tau$, and using existing protocols on sub-instances suffices to improve the communication complexity. For the sake of clarity, we will only present the idea behind our protocol for the case $\tau = 2$ in this technical overview. All the technical details and the more general approach for any $\tau$ is postponed to section 3.

#### 2.1.1 Unbounded fan-in multiplication of polynomials

Let $[f_1], \ldots, [f_n]$ be $n$ shared non-zero polynomials in $\mathbb{F}_q[X]$ of degree $< d$. Parties want to compute shares of the polynomial $[f_1 \times \cdots \times f_n]$ of degree $< nd$. Mohassel and Franklin proposed in [27] a constant round protocol to compute such a product with $\mathcal{O}(n^2 d)$ secure multiplications in $\mathbb{F}_q$. We must mention that the complexity is quadratic in $n$ because the protocol uses $\mathcal{O}(n)$ products in an extension field of $\mathbb{F}_q$ of degree $nd$. Our goal here is to perform most of the computation in smaller extension fields to reduce the complexity.

Assuming that $n$ is a perfect square, one may remark that dividing the computation in $\sqrt{n}$ sub-products of $\sqrt{n}$ polynomials allow us to reach a better complexity. Indeed, parties can compute in parallel each sub-product of $\sqrt{n}$ polynomials with $\sqrt{n}$ calls to the protocol of Mohassel and Franklin [27] for a total cost of $\mathcal{O}(n^{1.5} d)$ secure multiplication in $\mathbb{F}_q$. To finish

the computation, parties have to multiply $\sqrt{n}$ shared polynomials of degree less than $\sqrt{n}d$. Again this can be achieved by one call to the protocol from [27] for a cost of $\mathcal{O}(n^{1.5}d)$ secure multiplication in $\mathbb{F}_q$. We thus reduce the number of secured multiplication by $\sqrt{n}$ while we only double the number of rounds. We can generalize this idea to any fixed parameter $\tau \in \mathbb{N}^*$ in order to replace $\sqrt{n}$ with $n^{1/\tau}$ and then achieve a number of secure multiplication of $\mathcal{O}(n^{1+\frac{1}{\tau}})$ and $\mathcal{O}(\tau)$ rounds. Indeed, the explanation corresponds to the particular case of $\tau = 2$, but grouping the products by chunks of size $n^{\frac{1}{\tau}}$ at each step would only require $\tau$ steps to get the result.

### 2.1.2 Multi-point evaluation

Let $[f]$ be a shared polynomial in $\mathbb{F}_q[X]$ of degree $< n$ and $[\alpha_1], \ldots, [\alpha_n]$ be shared of points in $\mathbb{F}_q$. Parties want to compute the shares of the $n$ polynomial evaluations $[f(\alpha_1)], \ldots, [f(\alpha_n)]$.

In the case that parties want to evaluate $[f]$ at a single shared point $[\alpha]$, they can rely on [10] to have a secure protocol that is constant-round and that has a linear communication complexity in the degree of $f$. Note that the protocol heavily relies on unbounded fan-in multiplication of $2 \times 2$ matrices in order to guarantee that no leakage occurs when $[\alpha] = [0]$. No previous works have considered the simultaneous evaluation on a shared set of points, and to the best of our knowledge, the single points approach from [10] remains the most efficient for that case. In particular, this yields a constant-round protocol with a communication complexity of $\mathcal{O}(n^2)$ for evaluating $f$ on a set of $n$ shared points.

Instead of relying on computing powers of the $[\alpha_i]$ as in [10], our approach relies on the polynomial division of $f$ by well-chosen polynomials. It is fairly classical that $[f(\alpha_i)] = [f \bmod (X - \alpha_i)]$ (see [14]). Here, we exploit the constant-round protocol for the polynomial division of Franklin and Mohassel [27] that only requires a linear number of secure multiplications according to the dividend size. Unfortunately, applying straightforwardly this protocol for the division of $f$ by each $(X-\alpha_i)$ would also require $\mathcal{O}(n^2)$ secure multiplications. Instead, we will re-use our unbounded fan-in multiplication protocol to compute the product of the $n$ polynomials $(X - \alpha_i)$. Together with this product, we can obtain for free the $\sqrt{n}$ intermediate sub-products computed by the protocol (corresponding to the splitting of the result into $\sqrt{n}$ products of degree $\sqrt{n}$). There, we can reduce the polynomial $[f]$ modulo all these intermediate polynomials at a cost of $\sqrt{n}$ call to the division protocol of [27], yielding a communication complexity of $\mathcal{O}(n^{1.5})$ secure multiplications. Finally, we can further reduce these $\sqrt{n}$ distinct polynomials of degree less than $\sqrt{n}$ modulo the corresponding $(X - \alpha_i)$. Here again this can be done with only $\mathcal{O}(n^{1.5})$ secure multiplications since each reduction modulo one $(X - \alpha_i)$ costs $\mathcal{O}(n^{0.5})$.

To generalize this approach for any value of $\tau$, we will exploit the general divide-and-conquer strategy of our unbounded fan-in multiplication protocol. In particular, we will reduce the polynomial $[f]$ modulo all the intermediate polynomials computed in our unbounded fan-in multiplication protocol, using a breadth-first browsing of the $\tau$-ary tree.

### 2.1.3 Interpolation

Given $2n$ shared elements $[\alpha_1], \ldots, [\alpha_n]$ and $[y_1], \ldots, [y_n]$ in $\mathbb{F}_q$ such that $\alpha_1, \ldots, \alpha_n$ are distinct, parties want to compute the shares of $[f]$ such that $f$ is the unique polynomial in $\mathbb{F}_q[X]_{<n}$ such that $y_i = f(\alpha_i)$ for $1 \leq i \leq n$. Franklin and Mohassel [27] proposed a constant-round protocol for this operation that requires $\mathcal{O}(n^2)$ secure multiplications. Their idea is to use Lagrange interpolation to compute $f = \sum_{i=1}^{n} y_i L_i(X)/L_i(\alpha_i)$, where $L = \prod_i^n (X - \alpha_i)$ and $L_i = L/(X - \alpha_i)$. Therefore, using constant-round protocols for

unbounded fan-in polynomial multiplication, euclidean division, and field element inversion suffices to reconstruct $f$. Unfortunately, this approach requires $\mathcal{O}(n^2)$ secure multiplication since most of the tasks boil down to $n$ calls to a protocol that requires a linear number of multiplications, *i.e.* $n$ divisions involving the polynomial $L$ or the $n$ polynomial evaluations $L_i(\alpha_i)$.

To remove the need to compute the terms depending on the $L_i$s, we use the classical remark that $f/L = \sum_{i=1}^{n} c_i/(X-\alpha_i)$ where $c_i = y_i/L'(\alpha_i)$ and $L'$ is the derivative of $L$. First, we compute $L$ and then the $c_i$'s using our previous protocols for unbounded fan-in multiplication of polynomials and multi-point evaluation. This is done in constant round with $\mathcal{O}(n^{1.5})$ secure multiplications (using $\tau = 2$). Then, we reconstruct the numerator of the fraction $f/L$ in two steps, using a similar splitting strategy as in our previous protocols. First, we compute $\sqrt{n}$ different sums of $\sqrt{n}$ fractions of the form $c_i/(X-\alpha_i)$.

Let us define the polynomial $P_{1,1} = \prod_{l=1}^{\sqrt{n}}(X-\alpha_l)$. We can remark that the following equality holds $\sum_{i=1}^{\sqrt{n}} c_i/(X-\alpha_i) = (1/P_{1,1})\sum_{i=1}^{\sqrt{n}} c_i P_{1,1}/(X-\alpha_i)$ and that $G_{1,1} = \sum_{i=1}^{\sqrt{n}} c_i P_{1,1}/(X-\alpha_i)$ is a polynomial of degree $< \sqrt{n}$.

This equality extends naturally to all the $\sqrt{n}$ sums, and we thus can define the resulting fractions as $G_{1,1}/P_{1,1}, \ldots, G_{1,\sqrt{n}}/P_{1,\sqrt{n}}$. One may remark that computing numerators of these fractions amounts to taking linear combinations of the quotient of $P_{1,i}/(X-\alpha_j)$, which are exactly the same quotients as in the multi-point evaluation of $L$ at the $\alpha_i$'s (second step). This step costs exactly $\mathcal{O}(n^{1.5})$ secure multiplication remarking that there is a total of $n$ linear combinations of polynomial of degree less than $\sqrt{n}$.

As a second step, we write $f = \sum_{j=1}^{\sqrt{n}} G_{1,j}\frac{L}{P_{1,j}}$ where $L/P_{1,j}$ are polynomials of degree exactly $n-\sqrt{n}$. It thus remains to perform $\sqrt{n}$ products of polynomials of degree at most $n$ and sum the results. Using the division and multiplication protocols of Mohassel and Franklin [27], both steps can be done in constant rounds with $\mathcal{O}(n^{1.5})$ secure multiplications. Altogether, we obtain a constant round protocol with $\mathcal{O}(n^{1.5})$ secure multiplications.

As before, splitting every computations in chunks of size $n^{\frac{1}{\tau}}$ implies $\mathcal{O}(\tau)$ rounds, yielding a protocol with communication complexity $\mathcal{O}(\tau n^{1+\frac{1}{\tau}})$.

## 2.2 Private set operations

We present in section 4 our solutions to many variants of the *PSI* problem using our fast protocols on shared polynomials. In these problems, $m$ parties have the respective sets $\mathcal{A}_1, \ldots, \mathcal{A}_m \subseteq \mathbb{F}_q$ of size $n$ each and wish to compute some function of the intersection, i.e. $f(\bigcap_{i=1}^{m}\mathcal{A}_i)$ for a predetermined function $f$. Following the seminal work of [13], the main algebraic approaches in the literature rely on encoding the parties' sets as polynomials. Let a set $\mathcal{A} \subseteq \mathbb{F}_q$, one can define $P_{\mathcal{A}}(X) = \prod_{\alpha\in\mathcal{A}}(X-\alpha)$ to be an encoding of $A$. Each party can then compute locally its own polynomial $P_j = P_{\mathcal{A}_j}$ for all $1 \le j \le m$, and engage in a constant round protocol to distribute shares of all these polynomials. From there, parties would have to compute the gcd of these shared polynomials to get a representation of the intersection, and then apply some computation on this gcd to get the desired result.

### 2.2.1 Constant-round protocol for Private Disjointness Test

Our first contribution concerns the *PDT* problem where the $m$ parties want to know whether the intersection set is empty or not. We revamp a technique from [25] and [26] that uses a property on the gcd of many polynomials [14, Section 6.9]. More precisely, let $G = \gcd(P_{\mathcal{A}_1}, \ldots, P_{\mathcal{A}_m})$, and $R = \sum_j R_j P_{\mathcal{A}_j}$ where the $R_j$'s are random polynomials of degree at most $n$ in $\mathbb{F}_q[X]$. Therefore, we have that $G = \gcd(R, P_{\mathcal{A}_j})$ for $1 \le j \le m$ with

high probability. The protocols of [25] and [26] aim at computing the polynomial $R$ and making it public so that any party can compute locally the intersection set. Since the $R_j$ are random polynomials, the security relies on the fact that $R$ will not be distinguishable from any degree-$n$ polynomial in the polynomial ideal $\langle G \rangle$.

For $PDT$ we cannot afford to make the polynomial $R$ public. Indeed, every party would then learn the intersection and this leaks more information than the emptiness of the intersection. However, we can keep this polynomial $R$ private and use our fast protocol on shared polynomials to perform the computation. More precisely, let us define $[R] = \sum_{j=2}^{m} [r_j][P_{\mathcal{A}_j}]$ where $r_j$ are nonzero random elements from $\mathbb{F}_q$. This polynomial $R$ is an encoding of the intersection set between the parties $(2, \ldots, m)$ with high probability. It is sufficient to ask the first party to share its elements $[\alpha_j]$ such that $\alpha_j \in \mathcal{A}_1$ among all the participants. Our protocol for $PDT$ consists of evaluating the polynomial $[R]$ on all the $[\alpha_j]$, multiplying these evaluations, and checking whether the product is zero or not. All the steps are constant round and only need $\mathcal{O}(mn + \tau n^{1+1/\tau})$ secure multiplications using our multi-point evaluation protocol from Section 3.

As for many of the algebraic solutions to $PDT$, our approach may fail to produce a correct answer. Indeed, the polynomial $R$ may contain roots that are not in the intersection while being an element of the set $\mathcal{A}_1$. This could happen with probability at most $n/q$. By taking a domain size $q$ that is way larger than the size of the sets $n$, this probability is negligible. Moreover, as the error is one-sided, parties can decide to repeat the protocol several times to further lower the probability of an incorrect result.

### 2.2.2 Perfectly correct protocol for Private Set Operations

As seen in Section 1.3.2 most of the known protocols for $PSO$ are constant round with an optimal communication complexity of $\mathcal{O}(mn)$ secure multiplications. However, similarly to our previous approach for $PDT$, the result may be incorrect due to the use of randomization: e.g. either because of sampling polynomial in the polynomial ideal defined by the intersection polynomial [26, 17] or because hashing technique may have collisions [13, 20]. One may ask whether it would be possible to have a protocol with similar complexities, i.e. constant round and linear communication, that always returns a correct output.

Our protocol in section 4 is a first step toward achieving such a result. In particular, we achieve most of the PSO functionalities without any errors, using a constant number of rounds and a sub-linear communication complexity of $\mathcal{O}(mn^{1+\frac{1}{\tau}} + mn \log n \log\log n)$ for any integer $\tau > 0$. Our method somehow generalizes the idea in [13, 20] to the multi-party case without having to call any two-party protocol. Here again, we ask the first party to share its set of elements with the other parties, and each of the other parties shares their elements encoded as a polynomial. From there, we can use our multi-point evaluation protocol to evaluate all parties' polynomials (except party one) over all the elements of the first party (or any designated party by the protocol).

Then, we convert all these evaluations into shared booleans using protocols from [11]. These booleans indicate if $P_j$ evaluates to 0 on the $i$-th element of the first party (for $1 \leq i \leq n$ and $1 \leq j \leq m$). Since manipulations of these shared booleans are not too difficult (as explained in [11]), this allows us to solve many problems related to $PSI$ in the claimed complexities. From this, it is straightforward to compute the logical AND of these booleans (see [11, Section 5.1]). The result is the booleans $[b_j]$, which indicate if the $j$th element of the first player is in the intersection. From this, it is easy to compute the solution to many problems, in a complexity that is smaller or equal to the complexity of the previous steps. We give here a few examples. The solution to $PSI$ can be computed as $\prod[Q_j]$ where

$[Q_j] = [b_j(X - \alpha_j) + (1 - b_j)]$ ($Q_j = X - \alpha_j$ if the $j$th element $\alpha_j$ of the first player is in the intersection and $Q_j = 1$ otherwise). The solution for *PDT* can be computed as $\bigwedge b_j$. The solution to *Cardinality Set Intersection* can be computed as $\sum[b_j]$. The solution to *Private Intersection Sum* can be computed as $\sum[b_j][y_j]$. See table 3 for a summary of how to solve these problems using the shared booleans $[b_j]$.

# 3 Fast operations on shared polynomials

This section is devoted to new protocols for classical operations on shared polynomials that require a constant number of rounds and an almost optimal communication complexity. This follows the work of Mohassel and Franklin in PKC'06 [27] that first proposed such optimal protocols for the multiplication or the Euclidean division of shared polynomials. While their work also improved on the generic constant-round approach of Bar-Ilan and Beaver [1] for the interpolation, unbounded fan-in multiplication, and gcd on shared polynomials, the obtained communication complexity is not yet optimal. In the next sections, we will provide new constant-round protocols with almost optimal communication complexity, *i.e.* quasi-linear in the degree of shared input/output polynomials. This concerns the unbounded fan-in multiplication of shared polynomials and the multi-evaluation or the interpolation for polynomials on sets of points that are all shared.

## 3.1 Technical background

We begin by presenting some existing techniques that we need in our results. All of the following ideas are presented in either [1], [7] [27] or [10]. In particular, the protocols specifically designed for polynomials are results from [27]. All of these techniques are secure and in a constant number of rounds.

**Generating a random invertible field element.** This protocol generates a shared *non-zero* element $[y]$ of $\mathbb{F}_q$ (whose value is hidden from the players). This is done by generating two (unknown) shared elements of $\mathbb{F}_q$, multiplying them together securely, and revealing the result. If it is invertible, then one of the elements is taken as the result, otherwise, parties rerun the protocol. This protocol fails with probability at most $2/q$, so it only takes a constant number of secure multiplications (except with negligible probability). In our protocol, we denote this operation as "$[y] \xleftarrow{\$} \mathbb{F}_q^*$". Note that this can also be used for generating invertible matrices of constant size.

**Inversion of an invertible field element.** To invert an invertible shared element $[x] \in \mathbb{F}_q^*$, parties can use the previous technique to generate $[y] \xleftarrow{\$} \mathbb{F}_q^*$, compute $[z] = [x][y]$ and reveal its value. Lastly, each participant can locally compute their share of $[x^{-1}] = z^{-1}[y]$. It only takes a constant number of secure multiplication. Note that this can also be used for inverting invertible matrices of constant size.

**Unbounded fan-in multiplication of invertible field elements.** Multiplying $n$ shared *invertible* elements $[x_1], \ldots, [x_n]$ cannot be done in a naive way in a constant number of rounds. To compute the product in MPC, parties generate $n$ random invertible elements $[r_1], \ldots, [r_n]$ in parallel, as well as their inverse. Then, they compute in parallel $[p_1] = [x_1][r_1]$ and $[p_j] = [r_{j-1}^{-1}][x_j][r_j]$ for $2 \leq j \leq n$. The values of the $p_j$'s are then revealed and everyone can compute their share of $\left[\prod x_j\right] = \left(\prod p_j\right)[r_n]$. This requires $2n - 1$ secure multiplications

in total. Note that this protocol can also be used for computing the shared product of invertible matrices and that every prefix of the total product can be computed as a sharing, by locally computing $\left[\prod^k x_j\right] = \left(\prod^k p_j\right)[r_k]$. We denote this protocol as **FanInMul**.

**Powers of any field element.** This cannot be done directly using **FanInMul** with all multiplicand being the same shared element $[x]$ if $[x]$ might be zero. Indeed, when computing powers of $[0]$, the protocol would leak that information when revealing the products of the $p_j$'s. To overcome this problem, Cramer, Kiltz and Padró in [10, Section 3] replaced $x$ by an invertible $2 \times 2$ polynomial matrix $M(x)$ whose powers are related to Chebyshev polynomials of the first kind. Their method to compute shared powers consists then essentially in applying **FanInMul** to $M(x)$ and then carrying out public linear operations corresponding to the change of basis between the Chebyshev basis and the monomial one. Note that this latter step does not require any communication. The resulting protocol requires $\mathcal{O}(n)$ secure multiplications. From there, they can evaluate a publicly known or shared polynomial of degree $n$ in a shared point $[x]$ in the same complexity.

**Product of two polynomials.** Multiplying two shared polynomials of degree $n$ in $\mathbb{F}_q[X]$ naively would require $\mathcal{O}(n^2)$ secure multiplications in $\mathbb{F}_q$. Mohassel and Franklin [27, Section 3] proposed a protocol in $\mathcal{O}(n)$ instead, by noting that evaluating and interpolating a shared polynomial on public and pre-agreed points does not require any communication. It therefore suffices to evaluate both polynomials on $2n+1$ public points, securely multiply the evaluations on each point in parallel, and interpolate the resulting polynomial. Overall, only $2n+1 = \mathcal{O}(n)$ secure multiplications are performed. Using this protocol makes it possible to compute secure multiplication on shared elements of an extension field of $\mathbb{F}_q$ of degree $n$ in $\mathcal{O}(n)$ secure multiplications in $\mathbb{F}_q$. This protocol needs that parties first agree on a public quotient polynomial, ensuring that modular reduction can be done with linear communication. We denote the protocol for polynomial multiplication over $\mathbb{F}_q[X]$ by **Poly2Mult**.

**Euclidian division of polynomials.** Mohassel and Franklin described in [27, Section 4] a protocol to compute shares of the quotient and remainder of two shared polynomials $[f]$ and $[g]$ with $d = \deg f \geq \deg g$. The main idea is to adapt the classical fast division approach [14, Section 9] to the shared setting. Notably, this is achieved by re-using the protocol for inverting group element of [1] to the case of the multiplicative subgroup $\mathbb{F}_q[X]/X^t$, *i.e.* when computing shared reverse polynomial of the quotient, and to use their subsequent protocol **Poly2Mult**. The division protocol requires a total $\mathcal{O}(d)$ secure multiplications in $\mathbb{F}_q$, and we denote it by **PolyDiv**.

**Unbounded fan-in multiplication of non-zero polynomials.** Given $n$ non-zero polynomials $[f_1], \ldots, [f_n]$ of degree less than $d$, Mohassel and Franklin proposed in [27, Section 5] a protocol to compute $\left[\prod f_j\right]$ in constant-round, achieving the best-known complexity to date. This method boils down to using protocol **FanInMul** in an extension field of degree at least $n \times d$ and considering the polynomials as elements of this larger field. Since the protocol **FanInMul** requires $\mathcal{O}(n)$ secure multiplications in this extension field, each of them requires $\mathcal{O}(nd)$ secure multiplications in $\mathbb{F}_q$, thus the whole protocol requires a total of $\mathcal{O}(n^2 d)$ secure multiplications in $\mathbb{F}_q$. We note that this protocol imposes all parties to agree on a predetermined irreducible polynomial of degree $nd$. We suppose that such a polynomial can be predetermined outside any call of this protocol and thus no communication complexity will be counted for its computation. We denote this protocol by **PolyMult**.

## 3.2 Unbounded fan-in multiplication of polynomials

Let $[f_1], \ldots, [f_n]$ be $n$ shared non-zero polynomials in $\mathbb{F}_q[X]$ of degree $< d$. Parties want to compute shares of the polynomial $[f_1 \times \cdots \times f_n]$ of degree $< nd$. Without loss of generality, we assume that $n = \lambda^\tau$ for an integer $\lambda$ (the extra padding required to achieve this only adds a size negligible in $n$). Our approach consists in computing in parallel sub-products of exactly $\lambda$ polynomials of increasing degree $< \lambda^i d$ for $i \in [0, \ldots, \tau - 1]$. For this, we define in definition 1 the polynomial products $P_{i,j}$ that we need to compute and that follow the following recursive definition. Lemma 2 bounds the degree of those polynomials. See the appendix for a proof of lemma 2.

▶ **Definition 1.** *Let $P_{i,j}$ be a polynomial of $\mathbb{F}_q[X]$ defined such that :*
$P_{0,j} = f_j \quad$ *for* $1 \leq j \leq n$, *and* $P_{i,j} = \prod_{l=1}^{\lambda} P_{i-1,(j-1)\lambda+l} \quad$ *for* $1 \leq i \leq \tau, 1 \leq j \leq \lambda^{\tau-i}$.

▶ **Lemma 2.** *For $0 \leq i \leq \tau$ and $1 \leq j \leq \lambda^{\tau-i}$, the polynomial $P_{i,j}$ is of degree less than $\lambda^i d$. Moreover, we have $\prod_{j=1}^{\lambda^{\tau-i}} P_{i,j} = \prod_{j=1}^{n} f_j$ for $0 \leq i \leq \tau$.*

From this lemma we notice that $P_{\tau,1} = \prod_{j=1}^{n} f_j$. We are then able to define a protocol that computes the shared polynomial $[P_{\tau,1}]$ in $\mathcal{O}(\tau)$ rounds. At each step, starting from step $i = 1$, parties compute in parallel all the shares of $[P_{i,j}]$ from the shares of $[P_{i-1,j}]$ obtained in the previous steps. The protocol FastPolyMult is described below as well as theorem 3 ensuring the correctness, security, and complexity of the protocol. The proof is available in the appendix.

**Algorithm 1** FastPolyMult.

**Input:** $n$ shared non-zero polynomial $[f_1], \ldots, [f_n]$ in $\mathbb{F}_q[X]_{<d}$, and $\tau \in \mathbb{N}^*$
**Output:** shares of polynomial $\left[\prod_{i=1}^{n} f_i\right]$

1 **for** $i$ **from** 1 **to** $\tau$ **do**
  In parallel, players compute for $1 \leq j \leq \lambda^{\tau-i}$:
  $[P_{i,j}] = \prod_{l=1}^{\lambda} [P_{i-1,(j-1)\lambda+l}]$ ▷ **PolyMult**
**end**
2 **return** $[P_{\tau,1}]$

▶ **Theorem 3.** *FastPolyMult is correct, secure and requires $\mathcal{O}(\tau)$ rounds of communications and $\mathcal{O}(\tau n^{1+\frac{1}{\tau}} d)$ secure multiplications in $\mathbb{F}_q$.*

Note that in this protocol, the shared polynomials $[P_{i,j}]$ can be also returned as output. Notably, we will use these polynomials in the next section to improve the multi-evaluation of a shared polynomial on a shared set of points.

We note that the protocol FastPolyMult can be used to achieve faster unbounded fan-in multiplication of $n$ shared (possibly zero) elements $[x_1], \ldots, [x_n]$ of $\mathbb{F}_q$. One can use the technique from [10] to construct a constant-round unbounded fan-in multiplication protocol having a linear communication complexity. However, such protocol only works with elements from the subgroup $\mathbb{F}_q^*$. In order to allow elements from $\mathbb{F}_q$ one must rely on the generic framework of Bar-Ilan and Beaver [1], but this comes with an expense of $\mathcal{O}(n^2)$ communications. In [7, Section 6.4], the authors suggest an alternative that is to compute shares of the polynomial $\prod(X - x_i)$ and get its constant term which is $(-1)^n \prod x_i$. Parties then simply need to multiply this shared coefficient by a publicly known constant to obtain the desired product. Thanks to our new protocol for unbounded fan-in multiplication of polynomials, we are now able to tackle this task with an almost linear communication complexity, improving on any previously known methods.

We shall mention that our strategy which consists to use a generic divide-and-conquer strategy with a depth of $\tau$ can also be applied to the approach of Bar-Ilan and Beaver. Their idea consists of replacing the multiplication of two field elements by a constant number of $3 \times 3$ non-zero matrix products. This yields a new unbounded fan-in multiplication problem with a similar number of terms but with constant-size matrices instead of field elements. Those products can of course be gathered similarly to our polynomials $P_{i,j}$, hence obtaining also a communication complexity of $\mathcal{O}(\tau n^{1+\frac{1}{\tau}})$.

Computing the same operations with potentially zero polynomials in the same multiplication complexity remains an open question. To our knowledge, the best currently known method is to consider the input polynomials as elements of a field extension of degree $nd$, and then to either use FastPolyMult on polynomials whose coefficients are in the field extension or to use the approach of Bar-Ilan and Beaver coupled with our generic divide and conquer strategy in the field extension. Both methods require $\mathcal{O}(\tau n^{2+\frac{1}{\tau}} d)$ secure multiplications to compute the product of $n$ potentially zero polynomials of degree less than $d$.

## 3.3 Polynomial evaluation on shared set of points

Let $[f]$ be a shared polynomial in $\mathbb{F}_q[X]$ of degree $< n$ and $[\alpha_1], \ldots, [\alpha_n]$ be shared of points in $\mathbb{F}_q$. Parties want to compute the shares of the $n$ polynomial evaluations $[f(\alpha_1)], \ldots, [f(\alpha_n)]$.

Assuming that $n$ is a perfect square, we can replace the $n$ evaluations of $f$ with $n$ evaluations of polynomials of degrees less than $\sqrt{n}$. Indeed, let $P_{1,1} = \prod_{l=1}^{\sqrt{n}}(X - \alpha_l)$ and $R_{1,1} = f \bmod P_{1,1}$ we have that $f(\alpha_l) = R_{1,1}(\alpha_l)$ for $1 \leq l \leq \sqrt{n}$. The same kind of relation holds for all polynomials $R_{1,j} = f \bmod P_{1,j}$ where $P_{1,j}$ follows Definition 1 with $\tau = \sqrt{n}$ and $f_j = (X - \alpha_j)$, *i.e.* $P_{1,j} = \prod_{l=(j-1)\sqrt{n}+1}^{j\sqrt{n}}(X - \alpha_l)$ for $1 \leq j \leq \sqrt{n}$. Using our protocol FastPolyMult we can compute the shared polynomials $[P_{1,1}], \ldots, [P_{1,\sqrt{n}}]$ in constant-round with only $\mathcal{O}(n^{1.5})$ secure multiplications in $\mathbb{F}_q$. Computing the shared polynomials $[f \bmod P_{1,1}], \ldots, [f \bmod P_{1,\sqrt{n}}]$ amounts to the same complexity and rounds by using $\sqrt{n}$ calls to the protocol **PolyDiv**, *i.e.* each division involves $f$ and a polynomial of degree $\sqrt{n}$. To conclude the computation, parties have now to take every shared polynomial $[f \bmod P_{1,j}]$ of degree $< \sqrt{n}$ and to reduce each of them modulo the corresponding linear polynomials $(X - \alpha_k)$. This amounts to exactly $n$ calls to protocol **PolyDiv** with a dividend of degree $< \sqrt{n}$ and a divisor of degree 1. This final step also costs $\mathcal{O}(n^{1.5})$ secure multiplications in $\mathbb{F}_q$, and it is also constant-round. This idea generalizes by assuming $n = \lambda^\tau$ for $\tau \in \mathbb{N}^*$. We can define the polynomial $P_{i.j}$ as in Definition 1 where $f_j = (X - \alpha_j)$. Let us also define the polynomials $R_{i,j}$ as follows, which satisfy a recurrence relationas stated in lemma 4. A proof for lemma 4 is avaible in the appendix.

▶ **Lemma 4.** *Let $R_{i,j} = f \bmod P_{i,j}$ be a polynomial of $\mathbb{F}_q[X]$. These polynomials satisfy the following recurrence relation:*
*$R_{\tau,1} = f$, and $R_{i,j} = R_{i+1,\lceil j/\lambda \rceil} \bmod P_{i,j}$ for $0 \leq i \leq \tau - 1$ and $1 \leq j \leq \lambda^{\tau-i}$.*

One may remark from the definitions of the polynomials $R_{i,j}$ that $R_{0,j} = f \bmod (X - \alpha_j) = f(\alpha_j)$ for $1 \leq j \leq n$. We can thus obtain a protocol in $\mathcal{O}(\tau)$ rounds by first computing the shares of the polynomials $[P_{i,j}]$ and then apply the recursive property of the polynomials $R_{i,j}$ to compute the shares of $[R_{0,j}]$ from $[f]$ and the $[P_{i,j}]$'s. Protocol FastEval is described below as well as theorem 5 ensuring the expected properties for this protocol. The proof for theorem 5 is avaible in the appendix. In the protocol, we let $[P_{0,1}], \ldots, [P_{0,n}] = [(X - \alpha_0)], \ldots, [(X - \alpha_n)]$.

**Algorithm 2** FastEval.

**Input:** A shared polynomial $[f]$ of $\mathbb{F}_q[X]_{<n}$, a set $[\alpha_1], \ldots, [\alpha_n]$ of shared points in $\mathbb{F}_q$ and $\tau \in \mathbb{N}^*$

**Output:** $[f(\alpha_1)], \ldots, [f(\alpha_n)]$

Players compute $[P_{i,j}]$ for $1 \leq i \leq \tau$, $1 \leq j \leq \lambda^{\tau-i}$ ▷ FastPolyMult

**for** $i$ **from** $\tau - 1$ **down to** $0$ **do**

  In parallel, players compute for $1 \leq j \leq \lambda^i$:

  $[R_{i,j}] = [R_{i+1,\lceil j/\lambda \rceil}] \mod [P_{i,j}]$ ▷ **PolyDiv**

**end**

**return** $[R_{0,1}], \ldots, [R_{0,n}]$ ;

▶ **Theorem 5.** *FastEval is correct, secure and requires $\mathcal{O}(\tau)$ rounds of communications and $\mathcal{O}(\tau n^{1+\frac{1}{\tau}})$ secure multiplications in $\mathbb{F}_q$.*

## 3.4 Polynomial interpolation

Given $2n$ shared elements $[\alpha_1], \ldots, [\alpha_n]$ and $[y_1], \ldots, [y_n]$ in $\mathbb{F}_q$ such that the $\alpha_i$s are distinct, parties want to compute the shares of $[f]$ such that $f$ is the unique polynomial in $\mathbb{F}_q[X]_{<n}$ such that $y_i = f(\alpha_i)$ for $1 \leq i \leq n$. The Lagrange interpolation states that $f = \sum_{i=1}^{n} y_i L_i(X)/L_i(\alpha_i)$ where $L = \prod_i^n (X - \alpha_i)$ and $L_i = L/(X - \alpha_i)$. It is well known, see [14], that the computation of $L_i(\alpha_i)$ can be replaced by $L'(\alpha_i)$ where $L'$ is the derivative of $L$. To further remove the need to compute the polynomial $L_i$, one can use the also classical remark that $f/L = \sum_{i=1}^{n} c_i/(X - \alpha_i)$ where $c_i = y_i/L'(\alpha_i)$. Since our constant-round protocols FastPolyMult and FastEval allow us to compute efficiently the shares of $[L]$ and $[L'(\alpha_i)]$ for $1 \leq i \leq n$, the only remaining difficulty is the shares of $\sum_{i=1}^{n} c_i/(X - \alpha_i)$. Note that the derivative of $L'$ can be done without any communication and the last multiplication by $L(X)$ is not needed as the $\alpha_i$'s are all distinct hence the denominator of $\sum_{i=1}^{n} c_i/(X - \alpha_i)$ will be indeed $L(X)$. Assuming that $n$ is a perfect square, as we already did before, we can define the polynomial $P_{1,1} = \prod_{l=1}^{\sqrt{n}}(X - \alpha_l)$. We can remark that the following equality holds for the first $\sqrt{n}$ sumands: $\sum_{i=1}^{\sqrt{n}} \frac{c_i}{(X-\alpha_i)} = \frac{1}{P_{1,1}} \sum_{i=1}^{\sqrt{n}} \frac{c_i P_{1,1}}{(X-\alpha_i)}$, where $G_{1,1} = \sum_{i=1}^{\sqrt{n}} c_i P_{1,1}/(X - \alpha_i)$ is a polynomial of degree $< \sqrt{n}$ by definition of $P_{1,1}$. Doing similarly for the $\sqrt{n}$ chunks of the equation, each involving $\sqrt{n}$ summands, we will get $f/L = \sum_{j=1}^{\sqrt{n}} \frac{G_{1,j}}{P_{1,j}}$ where all denominators are of degree exactly $\sqrt{n}$. Therefore, we can write $f = \sum_{j=1}^{\sqrt{n}} G_{1,j} \frac{L}{P_{1,j}}$ where $L/P_{1,j}$ are polynomials of degree $n - \sqrt{n}$.

Assuming that shares of the $\sqrt{n}$ polynomial $[P_{1,j}]$ are known, this is the case since they are needed to compute shares of $[L]$ to get $[L']$ using protocol FastPolyMult. We also assume that the shares of $[c_i]$ has been computed efficiently using our protocol FastEval. Parties will have to perform $\sqrt{n}$ divisions for each $P_{1,j}$ by the adequate linear forms $(X - \alpha_i)$. Since $P_{1,j}$ is of degree $\sqrt{n}$ this amounts to $\sqrt{n} \times \mathcal{O}(\sqrt{n})$ secure multiplications in $\mathbb{F}_q$ using **PolyDiv** for each $P_{i,j}$. Computing all the shares $[G_{1,i}]$ thus requires $\mathcal{O}(n^{1.5})$ secure multiplications in $\mathbb{F}_q$. Parties then need to compute shares of $[L/P_{1,1}], \ldots, [L/P_{1,\sqrt{n}}]$. This is achieved with $\sqrt{n}$ call to **PolyDiv** with polynomials of degree at most $n$ and it thus requires $\mathcal{O}(n^{1.5})$ secure multiplications in $\mathbb{F}_q$. Lastly, parties compute the shares of $[f] = \sum_{j=1}^{\sqrt{n}} [G_{1,j}] \times [L/P_{1,j}]$ which is done with $\mathcal{O}(n^{1.5})$ secure multiplications in $\mathbb{F}_q$ using **Poly2Mult**, *i.e.* all polynomials have degree at most $\sqrt{n}$. Altogether, the whole interpolation protocol is constant-round and has a communication complexity of $\mathcal{O}(n^{1.5})$.

To further generalize this approach, let us assume that $n = \lambda^\tau$ for $\tau \in \mathbb{N}^*$ and the polynomial $P_{i.j}$ are defined by Definition 1 where $f_j = (X - \alpha_j)$. We now define the general form for the polynomials $G_{i,j}$ in definition 6, and state lemma 7 which ensures that the polynomials $G_{i,j}$ have the expected property. The proof of lemma 7 is avaible in the appendix.

▶ **Definition 6.** *Let $G_{i,j}$ be polynomials defined by the following relations, for $0 \leq i \leq \tau$ and $1 \leq j \leq \lambda^{\tau-i}$: $G_{0,j} = c_j$, and $G_{i,j} = \sum_{l=1}^{\lambda} G_{i-1,(j-1)\lambda+l} P_{i,j} / P_{i-1,(j-1)\lambda+l}$.*

▶ **Lemma 7.** *For $0 \leq i \leq \tau$ and $1 \leq j \leq \lambda^{\tau-i}$, $G_{i,j}$ has degree $< \lambda^i$. Moreover, $\sum_{j=1}^{\lambda^{\tau-i}} G_{i,j}/P_{i,j} = f/L$ for $1 \leq i \leq \tau$.*

We shall mention that thanks to the definition of the $G_{i,j}$ we have $G_{\tau,1} = f$. Protocol FastInterpol as well as the related theorem 8 follow. Its proof is avaible in the appendix.

**Algorithm 3** FastInterpol.

**Input:** $2n$ shared elements $[\alpha_1], \ldots, [\alpha_n]$ and $[y_1], \ldots, [y_n]$ in $\mathbb{F}_q$ such that $\alpha_1, \ldots, \alpha_n$ are distinct, $\tau \in \mathbb{N}^*$

**Output:** $[f]$ such that $f$ is the unique polynomial of degree $< n$ such that $y_i = f(\alpha_i)$.

1 Players compute $[P_{i,j}]$ for $1 \leq i \leq \tau$, $1 \leq j \leq \lambda^{\tau-i}$ ▷ FastPolyMult

2 Players compute locally $[L'] = [P'_{\tau,1}]$ ▷ no communication

3 Players compute $[L'(\alpha_1)], \ldots, [L'(\alpha_n)]$ ▷ FastEval

4 Players compute $[G_{0,1}, \ldots, G_{0,n}] = [y_1][L'(\alpha_1)^{-1}], \ldots, [y_n][L'(\alpha_n)^{-1}]$ ▷ only secure multiplications and inversions of field elements

5 **for** $i$ **from** 1 **to** $\tau$ **do**

a. Players compute (in parallel) for $1 \leq j \leq \lambda^{\tau-i}$, $1 \leq l \leq \lambda$:
$[\gamma_{i,j,l}] = [P_{i,j}/P_{i-1,(j-1)\lambda+l}]$ ▷ **PolyDiv**

b. Players compute (in parallel) for $1 \leq j \leq \lambda^{\tau-i}$, $1 \leq l \leq \lambda$:
$[\beta_{i,j,l}] = [G_{i-1,(j-1)\lambda+l}][\gamma_{i,j,l}]$ ▷ **Poly2Mult**

c. Players compute for $1 \leq j \leq \lambda^{\tau-i}$:
$[G_{i,j}] = \sum_{l=1}^{\lambda} [\beta_{i,j,l}]$ ▷ no communication

**end**

6 **return** $[G_{\tau,1}]$

▶ **Theorem 8.** *FastInterpol is correct, secure and requires $\mathcal{O}(\tau)$ rounds of communications and $\mathcal{O}(\tau n^{1+\frac{1}{\tau}})$ secure multiplications in $\mathbb{F}_q$.*

## 4 Application to private set operations

In this section, we show how our protocols can be used to design several multi-party protocols for operations on private sets. We suppose that $m$ players participate in the protocol and each of them has a set of $n$ elements of $\mathbb{F}_q$. They want to compute a predetermined function of the intersection of their input sets. We denote by $\mathcal{A}_1, \ldots, \mathcal{A}_m$ their respective sets. In the *Private Set Intersection* problem, first introduced in [13], the participants want to compute the intersection $\bigcap_{i=1}^m \mathcal{A}_i$. In the *Private Disjointness Test* problem, also introduced in [13], the player wants to know whether the intersection $\bigcap_{i=1}^m \mathcal{A}_i$ is empty or not. In the *Cardinality Set Intersection* (*CSI*) problem, participants want to know the number of elements in the intersection. In the *Threshold Private Set Intersection* (*T-PSI*) problem, parties want to obtain the intersection if and only if the number of elements inside the intersection exceeds a public threshold $t$. In the *Private Intersection Sum* (*PIS*) problem, introduced in [22], the first participant has also a set $\mathcal{Y}$ of $n$ integers such that each element is associated with an element of $\mathcal{A}_1$. The goal is to compute the sum of elements of $\mathcal{Y}$ such that the associated elements of $\mathcal{A}_1$ are in the intersection $\bigcap_{i=1}^m \mathcal{A}_i$.

All the multi-party protocols that we present to solve these problems manipulate shared polynomials in order to compute the solution without revealing information about a secret input set. For a set $\mathcal{A} \subseteq \mathbb{F}_q$, we define the encoding polynomial $P_{\mathcal{A}}(X) = \prod_{\alpha \in \mathcal{A}}(X - \alpha)$, and we let $P_j = P_{\mathcal{A}_j}$ for all $1 \leq j \leq m$. The obvious thing to note is that for $x \in \mathbb{F}_q$, $x$ is in $\mathcal{A}$ if and only if $P_{\mathcal{A}}(x) = 0$. Moreover, in our methods, we use polynomial evaluations on a designated player's inputs. For the sake of clarity, we suppose that this player is the first and we let $\mathcal{A}_1 = \{\alpha_1, \ldots, \alpha_m\}$. In section 4.2, we also manipulate shared booleans, which are equal to 1 when the predicate that the boolean represents is true and 0 otherwise.

## 4.1 A probabilistic solution for *Private Disjointness Test*

In this section, we present a protocol for solving the *PDT* problem using our evaluation protocol from section 3.3. Our approach is inspired by techniques used for instance in [25] and [26]. Essentially, it consists in privately generating random polynomials $R_1, \ldots, R_m$ and privately computing $F = \sum R_i F_i$ before revealing $F$ to all the participants. They then evaluate $F$ locally on their input elements. This method doesn't leak any information other than the intersection because $F$ is in fact a uniformly random multiple of $\gcd(P_1, \ldots, P_m)$ with a given degree bound. Moreover, with high probability $F$ evaluates to 0 only on the elements in the intersection among all the input elements. This solution for *PSI* requires $\mathcal{O}(mn)$ secure multiplications. Our solution for *PDT* uses the same idea of privately computing a polynomial $G = \sum r_i P_i$. But in this case, $G$ is *not* revealed so that the $r_i$'s only need to be random elements in $\mathbb{F}_q$. Then, $G$ is privately evaluated using our protocol FastEval on the elements of the designated participant, which are denoted by $\alpha_1, \ldots, \alpha_n$. With probability $n/q$, the only $\alpha_j$s on which $G$ evaluates to zero are the points in the intersection. In order for this probability to be negligible, $q$ needs to be overwhelmingly large compared to $n$ when using this protocol. Then, since the remark at the end section 3.2 allows participants to use FastPolyMult to multiply $n$ potentially zero shared field elements, parties can compute the product of the evaluations. Knowing if the product is zero is enough to know whether the intersection is empty or not. A formal description of this protocol FastPDT and theorem 9 follow. The proof of theorem 9 is avaible in the appendix.

**Algorithm 4** FastPDT.

**Input:** Player $i \geq 2$ knows $\mathcal{A}_i \subseteq \mathbb{F}_q$ of size $n$, Player 1 knows $\mathcal{A}_1 = \{\alpha_1, \ldots, \alpha_n\} \subseteq \mathbb{F}_q$, everyone knows $\tau \in \mathbb{N}$.

**Output:** All players know whether $\mathcal{A}_1 \bigcap \cdots \bigcap \mathcal{A}_m = \emptyset$ or not.

1. For $2 \leq i \leq m$, each player $i$ computes $P_i = \prod_{\alpha \in \mathcal{A}_i}(X - \alpha)$ locally. All these polynomials as well as $\alpha_1, \ldots, \alpha_n$ are shared between all the participants.
2. In parallel, players generate $[r_i] \xleftarrow{\$} \mathbb{F}_q$ for $2 \leq i \leq m$.
3. In parallel, players compute for $2 \leq i \leq m$: $[r_i P_i] = [r_i][P_i]$. ▷ multiplications
4. Without communication, players compute $[G] = \sum_{i=2}^{m} [r_i P_i]$.
5. Players compute $[G(\alpha_1)], \ldots, [G(\alpha_n)]$. ▷ FastEval
6. Players compute $[b] = [G(\alpha_1) \ldots G(\alpha_n)]$. ▷ FastPolyMult
7. Players generate the sharing $[r] \xleftarrow{\$} \mathbb{F}_q^*$.
8. Players compute $[br] = [b][r]$ and reveal the value $br$. If $br = 0$, players return "not empty". Else, players return "empty".

▶ **Theorem 9.** *FastPDT is secure and requires $\mathcal{O}(\tau)$ rounds of communications and $\mathcal{O}(mn + \tau n^{1+1/\tau})$ secure multiplications. Moreover, parties always deduce the correct result if the intersection is non-empty. If it is empty, then parties deduce the correct result with a probability larger than $1 - n/q$.*

## 4.2 A New Generic Technique for Perfectly Correct Private Set Operations

We present a general approach to designing secure protocols for *PSOs* on the intersection of sets. This solution is slower than our method to solve *PDT* but can be used as a general framework to solve multiple problems related to set intersection, with no probability of returning an incorrect result. We will need two MPC techniques presented in [11]. The first protocol (*cf.* [11, Section 7.1]) computes, from a share $[x]$ of an element of $\mathbb{F}_q$, a shared boolean denoted $[x = 0]$ which is equal to one if and only if $x = 0$. It is constant-round and requires $\mathcal{O}(\log q \log\log q)$ secure multiplications. The authors explain that with negligible probability, this protocol can leak information. However, it is possible for the participant to detect when it is the case, and to abort the protocol and retry it before information is leaked. The second one is explained in [11, Section 5.1]. It aims to compute a symmetrical logical operation (in our case we only need to compute the logical "and" and the logical "or") from $n$ sharing in constant-round and $\mathcal{O}(n)$ secure multiplications, and is secure. For these protocols to work, it is required that the field $F_q$ is a prime field. Moreover, for this protocol we can take a prime $q$ such that $q = \mathcal{O}(n)$, so that the secure multiplication complexity of computing $[x = 0]$ from $[x]$ is $\mathcal{O}(\log q \log\log q) = \mathcal{O}(\log n \log\log n)$.

Our generic method is then as follows: players privately evaluate every polynomial $P_i$ on the input elements of the first player (using our protocol from 3.3). Then, they convert the evaluations into booleans using the method from [11], *i.e.*, to get shares of $[P_i(\alpha_j) \neq 0]$ (these shared booleans are 0 if the evaluation is also 0 and 1 otherwise). These booleans can be used to privately compute shares of the booleans $b_j = (\alpha_j \notin \mathcal{A}_1 \bigcap \cdots \bigcap \mathcal{A}_m)$, which indicate if each element is in the intersection. This can be done using the second method from [11], since $b_j = \bigvee_{i=2}^{m} (P_i(\alpha_j) \neq 0)$. Once the parties have shares of the $b_j$, it is almost straightforward to get the output of the desired problem. We first present the generic protocol BoolIntersection and then give a few examples of how it can be used to solve *PSI*-related problems. The proof of theorem 10 can be found in the appendix.

**Algorithm 5** BoolIntersection.

**Input:** Player $i \geq 2$ knows $\mathcal{A}_i \subseteq \mathbb{F}_q$ of size $n$, Player 1 knows $\mathcal{A}_1 = \{\alpha_1, \ldots, \alpha_n\} \subseteq \mathbb{F}_q$, everyone knows $\tau \in \mathbb{N}$.
**Output:** $[b_1] = [\alpha_1 \notin \mathcal{A}_1 \bigcap \cdots \bigcap \mathcal{A}_m], \ldots, [b_n] = [\alpha_n \notin \mathcal{A}_1 \bigcap \cdots \bigcap \mathcal{A}_m]$.

1 For $2 \leq i \leq m$, each player $i$ computes $P_i = \prod_{\alpha \in \mathcal{A}_i} (X - \alpha)$ locally. All these polynomials as well as $\alpha_1, \ldots, \alpha_n$ are shared between all the players.
2 In parallel, players compute for $2 \leq i \leq m$ and $1 \leq j \leq n$: $[P_i(\alpha_j)]$. ▷ FastEval
3 In parallel, players compute for $2 \leq i \leq m$ and $1 \leq j \leq n$: $[P_i(\alpha_j) \neq 0]$. ▷ [11, Section 7.1]
4 In parallel, players compute for $1 \leq j \leq n$: $[b_j] = \bigvee_{i=2}^{m} [P_i(\alpha_j) \neq 0]$. ▷ [11, Section 5.1]
5 **return** $[b_1], \ldots, [b_n]$.

▶ **Theorem 10.** *BoolIntersection is correct, secure and requires $\mathcal{O}(\tau)$ rounds of communications and $\mathcal{O}(mn \log n \log\log n + \tau m n^{1+1/\tau})$ secure multiplications.*

Once parties execute protocol BoolIntersection, it is easy to solve a multitude of problems without any error and in a secure manner. We give below a few examples for which the extra steps require less communication than the protocol to generate the booleans. Table 3 will summarize the computations that are described below.

**Table 3** Algebraization of variants of the *PSI* primitive for $n$ parties with private sets $\mathcal{A}_1, \ldots, \mathcal{A}_n$ and the intersection set $\mathcal{I} = \mathcal{A}_1 \cap \cdots \cap \mathcal{A}_n$ where $b_j$ denotes the Boolean $b_j = (\alpha_j \notin \mathcal{A}_1 \bigcap \cdots \bigcap \mathcal{A}_m)$ and $Q_j = b_j(X - \alpha_j) + (1 - b_j)$ for $j \in \{1, \ldots, n\}$.

| Name | Aim | Algebraization |
|---|---|---|
| *PSI* | $\mathcal{I}$ | $\prod_{j=1}^{n}[Q_j]$ |
| *PDT* | $\mathcal{I} \stackrel{?}{=} \emptyset$ | $[b'] = \bigwedge_{j=1}^{m}[b_j]$ |
| *CSI* | $\#\mathcal{I}$ | $\sum_{j=1}^{n}[b_j]$ |
| *T-PSI* | $\mathcal{I}$ only if $\#\mathcal{I} \geq t$ | $\left[t \geq \sum_{j=1}^{n} b_j\right] \prod_{j=1}^{n}[Q_j]$ |
| *PIS* | $\sum_{j \mid \alpha_j \in \mathcal{I}} y_j$ | $\sum_{j=1}^{n}[b_j][y_j]$ |

**Private Disjointness Test (PDT).** Parties can simply compute $[b'] = \bigwedge_{j=1}^{m}[b_j]$ and reveal its value.

**Private Set Intersection (PSI).** By letting $Q_j = b_j(X - \alpha_j) + (1 - b_j)$, we note that $Q_j = 1$ if $\alpha_j$ is not in the intersection and $Q_j = X - \alpha_j$ otherwise. Therefore $P_{\mathcal{A}_1 \bigcap \cdots \bigcap \mathcal{A}_m} = \prod Q_j$. To solve *PSI*, parties can compute in parallel $[Q_j]$ for $1 \leq j \leq n$, and then use protocol FastPolyMult to compute $[P_{\mathcal{A}_1 \bigcap \cdots \bigcap \mathcal{A}_m}]$ and then reveal it. Parties can then evaluate locally this polynomial on their input points in order to know the intersection.

**Cardinality Set Intersection (CSI).** Parties can simply compute and reveal $\sum[b_j]$ without secure multiplications to know the number of elements in the intersection.

**Threshold Private Set Intersection (T-PSI).** Given the threshold $t$, parties can compute $[l] = \sum[b_j]$ without communication and then compute $[t \geq l]$ using techniques presented in [11]. Then, using the same method as for solving *PSI*, parties compute $[P_{\mathcal{A}_1 \bigcap \cdots \bigcap \mathcal{A}_m}]$ and they reveal $[t \geq l][P_{\mathcal{A}_1 \bigcap \cdots \bigcap \mathcal{A}_m}]$. If it is the zero polynomial, then it means that the threshold is not reached and nothing is revealed. If it is a non-zero polynomial, then it means the threshold is reached and the intersection can be computed locally by every participant.

**Private Intersection Sum (PIS).** Given a payload $\mathcal{Y} = \{y_1, \ldots, y_n\}$ known by the first player, its elements are shared between everyone. Then, players can simply compute $[b_j][y_j]$ for $1 \leq j \leq n$ in parallel. Lastly, players compute $\sum[b_j y_j]$ and reveal it to obtain the result.

All these constant-round protocols are secure, with no occurrence of an incorrect result, and require $\mathcal{O}(\tau m n^{1+1/\tau} + mn \log q \log\log q)$ secure multiplications.

## References

1 Judit Bar-Ilan and Donald Beaver. Non-cryptographic fault-tolerant computing in constant number of rounds of interaction. In Piotr Rudnicki, editor, *8th ACM PODC*, pages 201–209. ACM, August 1989. doi:10.1145/72981.72995.

2 Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th ACM STOC*, pages 1–10. ACM Press, May 1988. doi:10.1145/62212.62213.

3 Guilhem Castagnos and Fabien Laguillaumie. Linearly homomorphic encryption from DDH. In Kaisa Nyberg, editor, *CT-RSA 2015*, volume 9048 of *LNCS*, pages 487–505. Springer, Heidelberg, April 2015. doi:10.1007/978-3-319-16715-2_26.

4 Nishanth Chandran, Nishka Dasgupta, Divya Gupta, Sai Lakshmi Bhavana Obbattu, Sruthi Sekar, and Akash Shah. Efficient linear multiparty PSI and extensions to circuit/quorum PSI. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 1182–1204. ACM Press, November 2021. doi:10.1145/3460120.3484591.

5 David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *20th ACM STOC*, pages 11–19. ACM Press, May 1988. doi:10.1145/62212.62214.

6 Geoffroy Couteau, Thomas Peters, and David Pointcheval. Encryption switching protocols. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 308–338. Springer, Heidelberg, August 2016. doi:10.1007/978-3-662-53018-4_12.

7 Ronald Cramer and Ivan Damgård. Secure distributed linear algebra in a constant number of rounds. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 119–136. Springer, Heidelberg, August 2001. doi:10.1007/3-540-44647-8_7.

8 Ronald Cramer, Ivan Damgård, and Ueli M. Maurer. General secure multi-party computation from any linear secret-sharing scheme. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 316–334. Springer, Heidelberg, May 2000. doi:10.1007/3-540-45539-6_22.

9 Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. Multiparty computation from threshold homomorphic encryption. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 280–299. Springer, Heidelberg, May 2001. doi:10.1007/3-540-44987-6_18.

10 Ronald Cramer, Eike Kiltz, and Carles Padró. A note on secure computation of the Moore-Penrose pseudoinverse and its application to secure linear algebra. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 613–630. Springer, Heidelberg, August 2007. doi:10.1007/978-3-540-74143-5_34.

11 Ivan Damgård, Matthias Fitzi, Eike Kiltz, Jesper Buus Nielsen, and Tomas Toft. Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 285–304. Springer, Heidelberg, March 2006. doi:10.1007/11681878_15.

12 Matthew K. Franklin and Stuart Haber. Joint encryption and message-efficient secure computation. *Journal of Cryptology*, 9(4):217–232, September 1996. doi:10.1007/BF00189261.

13 Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 1–19. Springer, Heidelberg, May 2004. doi:10.1007/978-3-540-24676-3_1.

14 Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra (third edition)*. Cambridge University Press, 2013. doi:10.1017/CBO9781139856065.

15 Rosario Gennaro, Michael O. Rabin, and Tal Rabin. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In Brian A. Coan and Yehuda Afek, editors, *17th ACM PODC*, pages 101–111. ACM, June / July 1998. doi:10.1145/277697.277716.

16 Satrajit Ghosh and Tobias Nilges. An algebraic approach to maliciously secure private set intersection. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 154–185. Springer, Heidelberg, May 2019. doi:10.1007/978-3-030-17659-4_6.

17 Satrajit Ghosh and Mark Simkin. The communication complexity of threshold private set intersection. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019,*

*Part II*, volume 11693 of *LNCS*, pages 3–29. Springer, Heidelberg, August 2019. doi:10.1007/978-3-030-26951-7_1.

18 Satrajit Ghosh and Mark Simkin. Threshold private set intersection with better communication complexity. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023, Part II*, volume 13941 of *LNCS*, pages 251–272. Springer, Heidelberg, May 2023. doi:10.1007/978-3-031-31371-4_9.

19 Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *14th ACM STOC*, pages 365–377. ACM Press, May 1982. doi:10.1145/800070.802212.

20 Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. Scalable multi-party private set-intersection. In Serge Fehr, editor, *PKC 2017, Part I*, volume 10174 of *LNCS*, pages 175–203. Springer, Heidelberg, March 2017. doi:10.1007/978-3-662-54365-8_8.

21 Susan Hohenberger and Stephen A. Weis. Honest-verifier private disjointness testing without random oracles. In George Danezis and Philippe Golle, editors, *PET 2006*, volume 4258 of *LNCS*, pages 277–294. Springer, Heidelberg, June 2006. doi:10.1007/11957454_16.

22 Mihaela Ion, Ben Kreuter, Ahmet Erhan Nergiz, Sarvar Patel, Shobhit Saxena, Karn Seth, Mariana Raykova, David Shanahan, and Moti Yung. On deploying secure computing: Private intersection-sum-with-cardinality. In *IEEE European Symposium on Security and Privacy, EuroS&P 2020, Genoa, Italy, September 7-11, 2020*, pages 370–389. IEEE, 2020. doi:10.1109/EuroSP48549.2020.00031.

23 Aggelos Kiayias and Antonina Mitrofanova. Testing disjointness of private datasets. In Andrew Patrick and Moti Yung, editors, *FC 2005*, volume 3570 of *LNCS*, pages 109–124. Springer, Heidelberg, February / March 2005.

24 Eike Kiltz, Payman Mohassel, Enav Weinreb, and Matthew K. Franklin. Secure linear algebra using linearly recurrent sequences. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 291–310. Springer, Heidelberg, February 2007. doi:10.1007/978-3-540-70936-7_16.

25 Lea Kissner and Dawn Xiaodong Song. Privacy-preserving set operations. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 241–257. Springer, Heidelberg, August 2005. doi:10.1007/11535218_15.

26 Ronghua Li and Chuankun Wu. An unconditionally secure protocol for multi-party set intersection. In Jonathan Katz and Moti Yung, editors, *ACNS 07*, volume 4521 of *LNCS*, pages 226–236. Springer, Heidelberg, June 2007. doi:10.1007/978-3-540-72738-5_15.

27 Payman Mohassel and Matthew Franklin. Efficient polynomial operations in the shared-coefficients setting. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *PKC 2006*, volume 3958 of *LNCS*, pages 44–57. Springer, Heidelberg, April 2006. doi:10.1007/11745853_4.

28 Payman Mohassel and Enav Weinreb. Efficient secure linear algebra in the presence of covert or computationally unbounded adversaries. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 481–496. Springer, Heidelberg, August 2008. doi:10.1007/978-3-540-85174-5_27.

29 Daniel Morales, Isaac Agudo, and Javier Lopez. Private set intersection: A systematic literature review. *Computer Science Review*, 49:100567, 2023. doi:10.1016/j.cosrev.2023.100567.

30 G. Sathya Narayanan, T. Aishwarya, Anugrah Agrawal, Arpita Patra, Ashish Choudhary, and C. Pandu Rangan. Multi party distributed private matching, set disjointness and cardinality of set intersection with information theoretic security. In Juan A. Garay, Atsuko Miyaji, and Akira Otsuka, editors, *CANS 09*, volume 5888 of *LNCS*, pages 21–40. Springer, Heidelberg, December 2009.

31 Kobbi Nissim and Enav Weinreb. Communication efficient secure linear algebra. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 522–541. Springer, Heidelberg, March 2006. doi:10.1007/11681878_27.

32 Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238. Springer, Heidelberg, May 1999. doi:10.1007/3-540-48910-X_16.

33 Benny Pinkas, Thomas Schneider, Oleksandr Tkachenko, and Avishay Yanai. Efficient circuit-based PSI with linear communication. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 122–153. Springer, Heidelberg, May 2019. doi:10.1007/978-3-030-17659-4_5.

34 Peter Rindal and Phillipp Schoppmann. VOLE-PSI: Fast OPRF and circuit-PSI from vector-OLE. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 901–930. Springer, Heidelberg, October 2021. doi:10.1007/978-3-030-77886-6_31.

35 Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.

36 Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pages 160–164. IEEE Computer Society Press, November 1982. doi:10.1109/SFCS.1982.38.

37 Qingsong Ye, Huaxiong Wang, Josef Pieprzyk, and Xian-Mo Zhang. Efficient disjointness tests for private datasets. In Yi Mu, Willy Susilo, and Jennifer Seberry, editors, *ACISP 08*, volume 5107 of *LNCS*, pages 155–169. Springer, Heidelberg, July 2008.


# A Appendix

**Proof of Lemma 2.** Let us prove the lemma by induction on $0 \leq i \leq \tau$. When $i = 0$, for $1 \leq j \leq n$, $P_{0,j} = f_j$ is of degree $< d$ by hypothesis, and $\prod_{i=1}^{n} P_{0,i} = \prod_{i=1}^{n} f_i$. When $1 \leq i \leq \tau$, by definition $P_{i,j} = \prod_{l=1}^{\lambda} P_{i-1,(j-1)\lambda+l}$ for $1 \leq j \leq \lambda^{\tau-i}$. Therefore, $P_{i,j}$ is the product of $\lambda$ polynomials of degree $< \lambda^{i-1}d$ by induction hypothesis, thus it is a polynomial of degree $< \lambda^i d$. Moreover, we notice that $[\![1, \lambda^{\tau-i+1}]\!] = \{(j-1)\lambda+l \mid 1 \leq j \leq \lambda^{\tau-i},\ 1 \leq l \leq \lambda\}$. Therefore, by induction:

$$\prod_{j=1}^{\lambda^{\tau-i}} P_{i,j} = \prod_{j=1}^{\lambda^{\tau-i}} \prod_{l=1}^{\lambda} P_{i-1,(j-1)\lambda+l} = \prod_{j=1}^{\lambda^{\tau-i+1}} P_{i-1,j} = \prod_{j=1}^{n} f_j,$$

which concludes the proof. ◀

**Proof of Lemma 3.** Correctness of the protocol is ensured by Lemma 2 while its security is guaranteed by the use of protocol **PolyMult** and that the $f_i$'s are non-zero. Since the latter protocol is constant-round and there is $\tau$ sequential steps, the protocol requires $\mathcal{O}(\tau)$ rounds of communications. Moreover, **PolyMult** is used $\lambda^{\tau-i}$ times at step $i$ and each call requires $\mathcal{O}(\lambda^2\lambda^{i-1}d)$ secure multiplication,*i.e.* each call computes the product of $\lambda$ shared polynomials of degree $< \lambda^{i-1}d$. Hence, step $i$ requires $\mathcal{O}(\lambda^{\tau+1}d)$ secure multiplications. Summing over all the steps leads to $\mathcal{O}(\tau\lambda^{\tau+1}d) = \mathcal{O}(\tau n^{1+\frac{1}{\tau}}d)$ secure multiplications in $\mathbb{F}_q$. ◀

**Proof of Lemma 4.** $R_{i+1,\lceil j/\lambda \rceil}$ is well-defined since $1 \leq j \leq \lambda^{\tau-i}$, therefore $1 \leq \lceil j/\lambda \rceil \leq \lambda^{\tau-i-1}$. It now suffices to prove that $P_{i,j} | P_{i+1,\lceil j/\lambda \rceil}$, because then:

$$R_{i+1,\lceil j/\lambda \rceil} \bmod P_{i,j} = (f \mod P_{i+1,\lceil j/\lambda \rceil}) \mod P_{i,j} = f \mod P_{i,j} = R_{i,j}.$$

By letting $l = j - (\lceil j/\lambda \rceil - 1)\lambda$, we have that $1 \leq l \leq \lambda$ and $j = (\lceil j/\lambda \rceil - 1)\lambda + l$. By definition of $P_{i+1,\lceil j/\lambda \rceil}$ (see Def. 1), we have that $P_{i,j}$ divides $P_{i+1,\lceil j/\lambda \rceil}$. This concludes the proof. ◀

**Proof of Theorem 5.** Correctness is ensured by the definition of the $R_{i,j}$ and that the protocol **PolyDiv** correctly computes Euclidean division. Moreover, it is secure since the only used protocols, *i.e.* FastPolyMult and **PolyDiv**, are secure protocols. Protocol FastPolyMult requires $\mathcal{O}(\tau)$ rounds and **PolyDiv** is constant-round and is used in $\tau$ sequential step, so the total number of round is $\mathcal{O}(\tau)$. Lastly, using Lemma 2 one may remark that at step $i$ of the loop we perform $\lambda^{\tau-i}$ divisions with a dividend of degree $\lambda^{i+1}$ and a divisor of degree $\lambda^i$. Therefore, step $i$ requires at most $\mathcal{O}(\lambda^{\tau+1})$ secure multiplications. Summed over all steps, this leads to protocol FastPolyMult requiring at most $\mathcal{O}(\tau n^{1+\frac{1}{\tau}})$ secure multiplications. ◀

**Proof of Lemma 7.** The statements are proven by induction for $0 \leq i \leq \tau$. When $i = 0$, $\deg G_{0,j} = \deg c_j = 0 < 1$. Moreover, Lagrange formula implies that $f/L = \sum_{i=1}^{n} c_i/(X - \alpha_i) = \sum_{j=1}^{\lambda^\tau} G_{0,j}/P_{0,j}$. When $0 \leq i \leq \tau$, we notice that $P_{i,j}$ has degree $\lambda^i$ and $P_{i-1,(j-1)\lambda+l}$ has degree $\lambda^{i-1}$ for $1 \leq j \leq \lambda^{\tau-i}$ and $1 \leq l \leq \lambda$. Therefore using Definition 6:

$$\begin{aligned}\deg(G_{i,j}) &\leq \max_{1\leq j\leq \lambda^{\tau-i}} \left(\deg(G_{i-1,(j-1)\lambda+l}) + \deg(P_{i,j}) - \deg(P_{i-1,(j-1)\lambda+l})\right)\\ &< \lambda^{i-1} + \lambda^i - \lambda^{i-1}\\ &< \lambda^i.\end{aligned}$$

Moreover, $[\![0, \lambda^{\tau-i+1}]\!] = \{(j-1)\lambda + l \mid 1 \leq j \leq \lambda^{\tau-i},\ 1 \leq l \leq \lambda\}$ thus by using both Definitions 1 and 6 :

$$\begin{aligned}\sum_{j=1}^{\lambda^{\tau-i+1}} \frac{G_{i-1,j}}{P_{i-1,j}} &= \sum_{j=1}^{\lambda^{\tau-i}} \sum_{l=1}^{\lambda} \frac{G_{i-1,(j-1)\lambda+l}}{P_{i-1,(j-1)\lambda+l}}\\ &= \sum_{j=1}^{\lambda^{\tau-i}} \frac{\sum_{l=1}^{\lambda} G_{i-1,(j-1)\lambda+l} P_{i,j}/P_{i-1,(j-1)\lambda+l}}{P_{i,j}}\\ &= \sum_{j=1}^{\lambda^{\tau-i}} \frac{G_{i,j}}{P_{i,j}}.\end{aligned}$$

Therefore, $f/L = \sum_{j=1}^{\lambda^{\tau-i+1}} G_{i-1,j}/P_{i-1,j} = \sum_{j=1}^{\lambda^{\tau-i}} G_{i,j}/P_{i,j}$. ◀

**Proof of Theorem 8.** Correctness is ensured by the definition of the polynomials $P_{i.j}$ and $G_{i,j}$ and Lemma 7 and that all the underlying protocols, *i.e.* FastPolyMult, FastEval, **PolyDiv** and **Poly2Mult**, are correct.

Moreover, the protocol is secure since protocols FastPolyMult, FastEval, **PolyDiv** and **Poly2Mult** are all secure. Protocols FastPolyMult and FastEval require both $\mathcal{O}(\tau)$ rounds. Since we call in sequence $\tau$ times protocols **PolyDiv** and **Poly2Mult** that are constant-round, our protocol require a total of $\mathcal{O}(\tau)$ rounds of communications.

For the complexity analysis, steps 1 and 3 require $\mathcal{O}(\tau n^{1+\frac{1}{\tau}})$ secure multiplications in $\mathbb{F}_q$ according to theorems 3 and 5. The other steps except step 5 are negligible for the communication complexity. At step $i$ of the loop, **PolyDiv** is called $\lambda^{\tau-i+1}$ times on shared dividends of the form $[P_{i,j}]$, and by Lemma 2 these polynomials are all of degree $< \lambda^i$, thus requiring $\mathcal{O}(\lambda^{\tau+1})$ secure multiplications in $\mathbb{F}_q$. At the same step, **Poly2Mult** is called $\lambda^{\tau-i+1}$ times on shared polynomials of degrees $< \lambda^i$, which requires also $\mathcal{O}(\lambda^{\tau+1})$ secure multiplications in $\mathbb{F}_q$. Overall, summing over the $\tau$ loops, step 5 requires at most $\mathcal{O}(\tau\lambda^{\tau+1}) = \mathcal{O}(\tau n^{1+\frac{1}{\tau}})$ secure multiplications in $K$, which concludes the proof. ◀

**Proof of Theorem 9.** FastPolyMult, FastEval are secure according to theorem 3 and 5. Moreover, the only information revealed is $br$, and since $r$ is a uniformly random element of $\mathbb{F}_q^*$ it only indicates if $G$ was evaluated to zero on at least one $\alpha_j$, which is the intended output. Thus FastPDT is secure. Moreover, each step requires at most $\mathcal{O}(1)$ rounds of communication except for steps 5 and 6 which require $\mathcal{O}(\tau)$ rounds. Therefore, the protocol requires $\mathcal{O}(\tau)$ rounds. For the multiplication complexity, step 3 requires $\mathcal{O}(mn)$ secure multiplications since the $P_i$'s are of degree $n$ and there is $m-1$ polynomials. Steps 5 and 6 require $\mathcal{O}(\tau n^{1+1/\tau})$ secure multiplications. Other steps require at most a constant number of secure multiplication. In total, the protocol requires $\mathcal{O}(mn+\tau n^{1+1/\tau})$ secure multiplications. We now compute the probability of an incorrect result. If the intersection is not empty, at least one $\alpha_j$ will be such that $G(\alpha_j)=0$, so $b=0$, $br=0$, and parties will deduce the correct result. If the intersection is empty, then for each $\alpha_j$, at least one $P_i$ will be such that $P_i(\alpha_j)\neq 0$. Therefore $r_iP_i(\alpha_j)$ and $G(\alpha_j)$ are uniformly random over $\mathbb{F}_q$. The overall probability of an incorrect result is therefore:

$$\Pr(b=0)=\Pr\big(\bigcup_{j=1}^{n} G(\alpha_j)=0\big)\leq\sum_{j=1}^{n}\Pr(G(\alpha_j)=0)\leq n/q,$$

which is the desired result. ◀

**Proof of Theorem 10.** FastEval and protocols from [11] are secure and correct, thus BoolIntersection is also secure and correct. Every step is constant-round except for the calls to FastEval which requires $\mathcal{O}(\tau)$ rounds, thus the protocol requires $\mathcal{O}(\tau)$ rounds. Lastly, Step 2 requires $\mathcal{O}(\tau mn^{1+1/\tau})$ secure multiplication since it consists of calling FastEval $m-1$ times. Step 3 requires $\mathcal{O}(nm\cdot\log n\log\log n)$ secure multiplications since it consists of calling the conversion protocol from [11] $n(m-1)$ times. Step 4 requires $\mathcal{O}(nm)$ secure multiplications since it consists of calling the protocol from [11, Section 5.1] $n$ times. Overall, the protocol requires $\mathcal{O}(mn\log n\log\log n+\tau mn^{1+1/\tau})$ secure multiplications. ◀