Multi-Party Threshold Private Set Intersection with Sublinear Communication

Saikrishna Badrinarayanan
Visa Research
sabadrin@visa.com

Peihan Miao
Visa Research
pemiao@visa.com

Peter Rindal
Visa Research
perindal@visa.com

Abstract

In multi-party threshold private set intersection (PSI), n parties each with a private set wish to compute the intersection of their sets if the intersection is sufficiently large. Previously, Ghosh and Simkin (CRYPTO 2019) studied this problem for the two-party case and demonstrated interesting lower and upper bounds on the communication complexity. In this work, we investigate the communication complexity of the multi-party setting ($n \geq 2$). We consider two functionalities for multi-party threshold PSI. In the first, parties learn the intersection if each of their sets and the intersection differ by at most T. In the second functionality, parties learn the intersection if the union of all their sets and the intersection differ by at most T.

For both functionalities, we show that any protocol must have communication complexity $\Omega(nT)$. We build protocols with a matching upper bound of $O(nT)$ communication complexity for both functionalities assuming threshold FHE. We also construct a computationally more efficient protocol for the second functionality with communication complexity $\widetilde{O}(nT^2)$ under a weaker assumption of threshold additive homomorphic encryption.

As a consequence, we achieve the first "regular" multi-party PSI protocol where the communication complexity only grows with the size of the set difference and does not depend on the size of the input sets.

1 Introduction

Private set intersection (PSI) protocols allow several mutually distrustful parties $P_1, P_2, \ldots, P_n$ each holding a private set $S_1, S_2, \ldots, S_n$ respectively to jointly compute the intersection $I = \bigcap_{i=1}^{n} S_i$ without revealing any other information. PSI has numerous privacy-preserving applications, e.g., DNA testing and pattern matching [TPKC07], remote diagnostics [BPSW07], botnet detection [NMH+10], online advertising [IKN+17]. Over the last years enormous progress has been made towards realizing this functionality efficiently [Mea86, HFH99, FNP04, KS05, HN10, DCT10, DCW13, PSZ14, PSSZ15, KKRT16, OOS16, RR17a, RR17b, KMP+17, HV17, PSWW18, PRTY19, GN19, GS19a, PRTY20] in the two-party, multiparty, and server-aided settings with both semi-honest and malicious security.

Threshold PSI. In certain scenarios, the standard PSI functionality is not sufficient. In particular, the parties may only be willing to reveal the intersection if they have a *large* intersection. For example, in privacy-preserving data mining and machine learning [MZ17] where the data is vertically partitioned among multiple parties (that is, each party holds different features of the same object), the parties may want to learn the intersection of their datasets and start their collaboration only if their common dataset is sufficiently large. If their common dataset is too small, in which case they are not interested in collaboration, it is undesirable to let them learn the intersection. In privacy-preserving ride sharing [HOS17], multiple users only want to share a ride if large parts of their trajectories on a map intersect. In this case, the users may be interested in the intersection of their routes, but only when the intersection is large. This problem can be formalized as *threshold private set intersection*, where, roughly speaking, the parties only learn the intersection if their sets differ by at most T elements.

Many works [FNP04, HOS17, PSWW18, ZC18, GN19] achieve this functionality by first computing the cardinality of the intersection and then checking if this is sufficiently large. The communication complexity of these protocols scales at least linearly in the size of the smallest input set. Notice that Freedman et al. [FNP04] proved a lower bound of $\Omega(m)$ on the communication complexity of any private set intersection protocol, where m is the size of the smallest input set. This lower bound directly extends to protocols that only compute the cardinality of the intersection, which constitutes a fundamental barrier to the efficiency of the above protocols.

Recently, the beautiful work of Ghosh and Simkin [GS19a] revisited the communication complexity of *two-party* threshold PSI and demonstrated that the $\Omega(m)$ lower bound can be circumvented by performing a private intersection cardinality testing (i.e., testing whether the intersection is sufficiently large) instead of computing the actual cardinality. After passing the cardinality testing, their protocol allows each party to learn the set difference, where the communication complexity only grows with T, which could be sublinear in m. Specifically, [GS19a] proved a communication lower bound of $\Omega(T)$ for two-party threshold PSI and presented a protocol achieving a matching upper bound $O(T)$ based on fully homomorphic encryption (FHE). They also showed a computationally more efficient protocol with communication complexity of $\widetilde{O}(T^2)$ based on weaker assumptions, namely additively homomorphic encryption (AHE).

In this work, we investigate the communication complexity of *multi-party* threshold PSI. In particular, we ask the question of whether sublinear lower and upper bounds can also be achieved in the multi-party setting.

1.1 Our Contributions

We first identify and formalize the definition of multi-party threshold private set intersection. While the definition for "regular" multi-party PSI[1] follows naturally from the definition of two-party PSI since the goal remains the same (to compute the intersection), the same does not hold in the threshold case. In particular, we put forth and study two functionalities for multi-party threshold PSI that are in fact both equivalent in the two-party case but are vastly different in the multi-party scenario. Assume there are n parties $P_1, P_2, \ldots, P_n$, and each party P_i holds a private set S_i of size m. The first functionality allows the parties to learn the intersection $I = \bigcap_{i=1}^n S_i$ only if $|S_i \setminus I| \leq T$, or equivalently, $|I| \geq m - T$. In the second functionality, the parties can learn the intersection I only if $|\left(\bigcup_{i=1}^n S_i\right) \setminus I| \leq T$.

We briefly discuss the difference between the two functionalities. The first functionality focuses on whether the intersection is sufficiently large, hence we call it $\mathcal{F}_{\mathsf{TPSI\text{-}int}}$. The second functionality focuses on whether the set difference is sufficiently small, thus we call it $\mathcal{F}_{\mathsf{TPSI\text{-}diff}}$. In the two-party case, we have the guarantee that $|\left(\bigcup_{i=1}^n S_i\right) \setminus I| = 2 \cdot |S_i \setminus I|$, so we don't have to differentiate

[1]By "regular" PSI, we refer to the standard notion of PSI without threshold.

between these two functionalities. However, in the multi-party case, we only know that $2 \cdot |S_i \setminus I| \leq |(\bigcup_{i=1}^{n} S_i) \setminus I| \leq n \cdot |S_i \setminus I|$, hence the two functionalities could lead to very different outcomes. Which functionality to choose and what threshold to set in practice highly depend on the actual application.

Sublinear Communication. The core contribution of this work is demonstrating sublinear (in the set sizes) communication lower and upper bounds for both functionalities of multi-party threshold PSI. We summarize our results in Table 1. For lower bound, we prove that both functionalities require at least $\Omega(nT)$ bits of communication. For upper bound, we present protocols for both functionalities achieving a matching upper bound of $O(nT)$ based on n-out-of-n threshold fully homomorphic encryption (TFHE) [BGG+18]. We also give a computationally more efficient protocol based on weaker assumptions, namely n-out-of-n threshold additively homomorphic encryption (TAHE) [Ben94, Pai99], with communication complexity of $\widetilde{O}(nT^2)$.[2] All these protocols achieve semi-honest security where up to $(n-1)$ parties could be corrupted.

Functionality	Communication Lower Bound	TFHE-based Upper Bound	TAHE-based Upper Bound
$\mathcal{F}_{\mathsf{TPSI\text{-}int}}$	$\Omega(nT)$	$O(nT)$	unknown
$\mathcal{F}_{\mathsf{TPSI\text{-}diff}}$	$\Omega(nT)$	$O(nT)$	$\widetilde{O}(nT^2)$

Table 1: Communication lower and upper bounds for multi-party threshold PSI.

Our Protocols. As summarized in Table 1, we present three protocols for upper bounds, one for $\mathcal{F}_{\mathsf{TPSI\text{-}int}}$ and two for $\mathcal{F}_{\mathsf{TPSI\text{-}diff}}$. At a high level, all three protocols compute their functionality in two phases. In the first phase, they perform a multi-party private intersection cardinality testing where the parties jointly decide whether their intersection is sufficiently large. In particular, for $\mathcal{F}_{\mathsf{TPSI\text{-}int}}$, the cardinality testing, which we call $\mathcal{F}_{\mathsf{CTest\text{-}int}}$, allows all the parties to learn whether $|I| \geq (m - T)$. For $\mathcal{F}_{\mathsf{TPSI\text{-}diff}}$, the cardinality testing, which we call $\mathcal{F}_{\mathsf{CTest\text{-}diff}}$, allows all the parties to learn whether $|(\bigcup_{i=1}^{n} S_i) \setminus I| \leq T$. The communication complexity of our multi-party private intersection cardinality testing protocols is summarized in Table 2. In particular, for $\mathcal{F}_{\mathsf{CTest\text{-}int}}$, we present a protocol with communication complexity $O(nT)$ based on TFHE. For $\mathcal{F}_{\mathsf{CTest\text{-}diff}}$, we show a TFHE-based construction with communication complexity $O(nT)$ and a TAHE-based construction with communication complexity $\widetilde{O}(nT^2)$.

Functionality	TFHE-based Protocol	TAHE-based Protocol
$\mathcal{F}_{\mathsf{CTest\text{-}int}}$	$O(nT)$	unknown
$\mathcal{F}_{\mathsf{CTest\text{-}diff}}$	$O(nT)$	$\widetilde{O}(nT^2)$

Table 2: Communication complexity of our protocols for multi-party private cardinality testing.

If the intersection is sufficiently large, namely it passes the cardinality testing, then the parties start the second phase of our protocols, which allows each party P_i to learn their set difference

[2] $\widetilde{O}(\cdot)$ hides polylog factors. All the upper bounds omit a $\mathsf{poly}(\lambda)$ factor where λ is the security parameter.

$S_i \setminus I$. We present a singe protocol for the second phase, which works for both $\mathcal{F}_{\mathsf{TPSI\text{-}int}}$ and $\mathcal{F}_{\mathsf{TPSI\text{-}diff}}$. The second-phase protocol is based on TAHE and has communication complexity of $O(nT)$. Thus, to construct a protocol for multi-party threshold PSI, we combine the first-phase protocols summarized in Table 2 with the second-phase one described above. Doing so, we achieve the communication upper bounds in Table 1.

This modular design enables our constructions to minimize the use of TFHE as it is not needed in the second phase. Moreover, it allows future work to focus on improving Table 2. In particular, since the second-phase protocol is already tight in communication ($O(nT)$) based on the weaker assumption of TAHE, future work could focus on building protocols for private intersection cardinality testing with better communication complexity and/or from weaker assumptions.

Communication Topology. All our protocols are designed in the so-called *star network* topology, where a designated party communicates with every other party. An added benefit of this topology is that not all parties must be online at the same time. Our communication lower bounds are proved in *point-to-point* fully connected networks, which are a generalization of the star network.

For networks with broadcast channels, we prove another communication lower bound of $\Omega(T \log n + n)$ for $\mathcal{F}_{\mathsf{TPSI\text{-}int}}$ in Section 9 and leave further exploration in the broadcast model for future work.

1.2 Other Implications

Sublinear Communication PSI. Our multi-party threshold PSI protocols for both $\mathcal{F}_{\mathsf{TPSI\text{-}int}}$ and $\mathcal{F}_{\mathsf{TPSI\text{-}diff}}$ can also be used to achieve multi-party "regular" PSI where the communication complexity only grows with the size of the set difference and independent of the input set sizes.

In particular, if we run a sequence of multi-party threshold PSI protocols on $T = 2^0, 2^1, 2^2, \ldots$ until hitting the smallest $T = 2^k$ where the protocol outputs the intersection, then we can achieve multi-party PSI. The communication complexity of the resulting protocol is a factor $\log T$ times that of a single instance but still independent of the input set sizes. Therefore, when the intersection is very large, namely the set difference is significantly smaller than the set sizes, this new approach achieves the *first* multi-party PSI with sublinear (in the set sizes) communication complexity.

This approach can also be applied to our multi-party private intersection cardinality testing protocols to achieve the *first* multi-party private intersection cardinality protocol with sublinear communication where the parties want to jointly just learn the cardinality of their set intersection.

Compact MPC. It is an open problem to construct a *compact* MPC protocol in the plain model where the communication complexity does not grow with the output length of the function. Prior works [HW15, BFK$^+$19] construct compact MPC for general functions in the presence of a trusted setup (CRS, random oracle) from strong computational assumptions such as obfuscation. Our multi-party threshold PSI protocols have communication complexity independent of the output size (the set intersection). To the best of our knowledge, ours are the *first* compact MPC protocols for even a specific function in the plain model. The only prior compact protocol in the plain model we are aware of is the two-party threshold PSI protocol [GS19a].

1.3 Concurrent and Future Work

Concurrent and Independent Work. In a very recent update to the full version of the paper by Ghosh and Simkin [GS19b], they extend the two-party threshold PSI protocol to the multi-party setting and consider the functionality $\mathcal{F}_{\mathsf{TPSI\text{-}int}}$. In particular, they do not consider the functionality $\mathcal{F}_{\mathsf{TPSI\text{-}diff}}$ that we additionally consider in our work.

For functionality $\mathcal{F}_{\mathsf{TPSI\text{-}int}}$, Ghosh and Simkin [GS19b] also first construct a TFHE-based protocol for the multi-party private intersection cardinality testing functionality $\mathcal{F}_{\mathsf{CTest\text{-}int}}$. They then have a second-phase protocol for computing the intersection in which they use generic MPC to compute the evaluations of a random polynomial. Our protocol for that step of is computationally more efficient based on threshold additive homomorphic encryption. Furthermore, our communication complexity is linear in the number of parties, which was not achieved in [GS19b].

Open Problems. We briefly mention a collection of interesting open problems for future work.

- Build private intersection cardinality testing protocols for $\mathcal{F}_{\mathsf{CTest\text{-}int}}$ from weaker computational assumptions than TFHE.
- Construct practically more efficient threshold PSI protocols with sublinear communication.
- Explore multi-party threshold PSI in the broadcast model.
- Design more efficient threshold PSI protocols in the setting where all parties have different sized sets (rather than padding all of them to have the same size and using our protocol).
- Construct threshold PSI protocols with malicious security.
- Optimize round complexity of threshold PSI with sublinear communication.

2 Technical Overview

We now give an overview of the techniques used in our work. As mentioned previously, we consider the *star network* topology. We denote P_1 to be the designated party that can communicate with all the other parties.

2.1 TFHE-Based Protocol for $\mathcal{F}_{\mathsf{CTest\text{-}int}}$

To construct multi-party private intersection cardinality testing from TFHE, our starting point is the FHE-based protocol for private intersection cardinality testing in the two-party setting [GS19a]. We first briefly recall the idea. For two parties Alice and Bob with sets $S_A = \{a_1, \ldots, a_m\}$ and $S_B = \{b_1, \ldots, b_m\}$ respectively, we encode the elements into two polynomials $\mathsf{p}_A(x) = \prod_{i=1}^{m}(x - a_i)$ and $\mathsf{p}_B(x) = \prod_{i=1}^{m}(x - b_i)$. Let $I := S_A \cap S_B$ be the intersection. A key observation in [MTZ03,GS19a] for such an encoding procedure is that $\mathsf{p}(x) := \frac{\mathsf{p}_B(x)}{\mathsf{p}_A(x)} = \frac{\mathsf{p}_{B\setminus I}(x)}{\mathsf{p}_{A\setminus I}(x)}$. Both the numerator and denominator of p have degree $|S_A \setminus I|$. If $|S_A \setminus I| \leq T$, then $\mathsf{p}(x)$ has degree at most $2T$ and can be recovered from $2T + 1$ evaluations by rational function interpolation.[3] Given $\mathsf{p}(x)$, the elements in $S_A \setminus I$ are simply the roots of the polynomial in the denominator.

Two-party protocol. At a high level, the two-party protocol works as follows. First, Alice and Bob evaluate their own polynomials on $2T + 1$ publicly known distinct points $\{\alpha_1, \ldots, \alpha_{2T+1}\}$

[3]A rational function is a fraction of two polynomials. We refer to Minskey et al. [MTZ03] for details on rational function interpolation over a field. Also, we note that monic polynomials can be interpolated with $2T$ evaluation but we use $2T + 1$ for consistency with our other protocols.

to obtain $\{\mathsf{p}_A(\alpha_1), \ldots, \mathsf{p}_A(\alpha_{2T+1})\}$ and $\{\mathsf{p}_B(\alpha_1), \ldots, \mathsf{p}_B(\alpha_{2T+1})\}$, respectively. Then, Alice generates a public-secret key pair of the FHE scheme and sends Bob the FHE public key, encrypted evaluations $\{\mathsf{p}_A(\alpha_1), \ldots, \mathsf{p}_A(\alpha_{2T+1})\}$, a uniformly random z and encrypted evaluation $\mathsf{p}_A(z)$. Bob can then homomorphically interpolate the rational function $\mathsf{p}(x)$ from $\{\mathsf{p}_A(\alpha_1), \ldots, \mathsf{p}_A(\alpha_{2T+1})\}$ and $\{\mathsf{p}_B(\alpha_1), \ldots, \mathsf{p}_B(\alpha_{2T+1})\}$ and homomorphically compute encrypted $\mathsf{p}(z)$. Bob then computes $\mathsf{p}_B(z)$ and homomorphically computes encrypted $\frac{\mathsf{p}_B(z)}{\mathsf{p}_A(z)}$. We know that $\mathsf{p}(z) = \frac{\mathsf{p}_B(z)}{\mathsf{p}_A(z)}$ if and only if the degree of $\mathsf{p}(x)$ is $\leq 2T$, namely $|S_A \setminus I| \leq T$. Therefore Bob homomorphically computes an encryption of the predicate $b = \left(\mathsf{p}(z) \overset{?}{=} \frac{\mathsf{p}_B(z)}{\mathsf{p}_A(z)}\right)$ and sends the encryption of b back to Alice. Finally Alice decrypts and learns whether $|S_A \setminus I| \leq T$.

Generalize to multi-party. To generalize this protocol to n parties, a natural idea is to consider

$$\mathsf{p}(x) := \frac{\mathsf{p}_2(x) + \cdots + \mathsf{p}_n(x)}{\mathsf{p}_1(x)} = \frac{\mathsf{p}_{2\setminus I}(x) + \cdots + \mathsf{p}_{n\setminus I}(x)}{\mathsf{p}_{1\setminus I}(x)}, \tag{1}$$

where $\mathsf{p}_i(x)$ encodes the set $S_i = \{a_1^i, \ldots, a_m^i\}$ as $\mathsf{p}_i(x) := \prod_{j=1}^m (x - a_j^i)$. Instead of FHE, we make use of n-out-of-n threshold fully homomorphic encryption (TFHE) with distributed setup, which enables n parties to jointly generate an FHE public key, where the secret key is shared among the n parties. The protocol for $\mathcal{F}_{\mathsf{CTest\text{-}int}}$ roughly works as follows. The n parties first jointly generate the TFHE keys. Each party P_i then evaluates its own polynomial $\mathsf{p}_i(\cdot)$ on $\{\alpha_1, \ldots, \alpha_{2T+1}, z\}$ and sends encrypted evaluations $\{\mathsf{p}_i(\alpha_1), \ldots, \mathsf{p}_i(\alpha_{2T+1}), \mathsf{p}_i(z)\}$ to P_1. Now P_1 can homomorphically interpolate $\mathsf{p}(x)$ from $2T+1$ evaluations and homomorphically compute an encryption of the predicate $b = \left(\mathsf{p}(z) \overset{?}{=} \frac{\mathsf{p}_2(z)+\cdots+\mathsf{p}_n(z)}{\mathsf{p}_1(z)}\right)$. Finally the n parties jointly decrypt the encryption of b.

Unexpected degree reduction. This seemingly correct protocol has a subtle issue.[4] Intuitively, we want to argue that $\mathsf{p}(x)$ in Equation 1 has degree $\leq 2T$ if and only if $|S_1 \setminus I| \leq T$. However, this is not true because Equation 1 is not accurate. In particular, in addition to the elements in intersection I that will be canceled out in the numerator and denominator of $\mathsf{p}(x)$, elements not in the intersection might also be canceled out in certain scenarios. This is undesirable as it unexpectedly reduces the degree of the numerator and denominator of $\mathsf{p}(x)$! Here is a concrete example. Consider three sets with distinct elements $S_1 = \{a\}$, $S_2 = \{b\}$, $S_3 = \{c\}$, where $b+c = 2\cdot a$. The intersection $I = \emptyset$. Ideally we want the rational polynomial $\mathsf{p}(x)$ to have degree 1 in both the numerator and denominator because $|S_1 \setminus I| = 1$. However,

$$\mathsf{p}(x) = \frac{(x-b)+(x-c)}{x-a} = \frac{2x-(b+c)}{x-a} = \frac{2x-2a}{x-a} = 2.$$

The key issue in the above example is that the term $(x-a)$ is unexpectedly canceled out. More generally, this approach does not work because the polynomial in the numerator $\mathsf{p}_2(x) + \cdots + \mathsf{p}_n(x)$ may accidentally create roots that are not in the intersection, but coincide with elements in S_1 thereby reducing the degree of the numerator and denominator.

[4]In fact, this subtle issue was initially overlooked by [GS19b] in their recent update to the multi-party setting causing their protocol to produce incorrect output on certain inputs. We pointed this out to the authors and they have subsequently fixed it. This was confirmed in personal communication.

Randomness to the rescue. On first thought, it seems that this approach is fundamentally flawed as such additional roots can always be created when we add polynomials in the numerator. Our key insight to solve the problem is to add "enough" randomness to each polynomial in the numerator to prevent this issue from arising except with negligible probability. In more detail, we add a single random multiplicative term $(x - r_i)$ to each polynomial p_i and set a new polynomial $\mathsf{p}'_i(x) := \mathsf{p}_i(x) \cdot (x - r_i)$ for a random r_i chosen by party P_i. Now, consider the rational polynomial

$$\mathsf{p}'(x) := \frac{\mathsf{p}'_2(x) + \cdots + \mathsf{p}'_n(x)}{\mathsf{p}'_1(x)} = \frac{\mathsf{p}'_{2\setminus I}(x) + \cdots + \mathsf{p}'_{n\setminus I}(x)}{\mathsf{p}'_{1\setminus I}(x)}.$$

At a high level, the term $(x - r_i)$ will randomize the roots of the numerator sufficiently to ensure that these roots are unlikely to coincide with the roots of the denominator. We formally prove that this unexpected degree reduction (that is, elements not in the intersection canceling out in the numerator and denominator) only happens with negligible probability by just adding this one random term to each polynomial.

Alternative protocol. There is another way to compute the functionality $\mathcal{F}_{\mathsf{CTest\text{-}int}}$ from TFHE. As we will see, the second construction is in fact simpler and cleaner than the first one. However, we still present the first construction above to illustrate the subtle issue that we discussed which will also later arise when computing the concrete intersection in the second phase of our protocols.

We describe the alternative protocol as follows. The n parties first jointly generate the TFHE keys as before. Then, as in the two-party protocol, party P_1 homomorphically interpolates

$$\widetilde{\mathsf{p}}_i(x) = \frac{\mathsf{p}_i(x)}{\mathsf{p}_1(x)} = \frac{\mathsf{p}_{i\setminus 1}(x)}{\mathsf{p}_{1\setminus i}(x)}$$

from $2T + 1$ evaluations and homomorphically computes an encryption of $D_{1,i} = S_1 \setminus S_i$ for every other party P_i. Note that if $|S_1 \setminus I| \leq T$, then the degree of each $\widetilde{\mathsf{p}}_i(x)$ is at most $2T$, hence P_1 can interpolate it using $2T+1$ evaluations. Since $S_1 \setminus I = \bigcup_{i=2}^{m}(S_1 \setminus S_i)$, party P_1 can homomorphically compute an encryption of $(S_1 \setminus I)$ and an encryption of predicate $b = \left(|S_1 \setminus I| \overset{?}{\leq} T\right)$. Finally, the n parties jointly decrypt the encryption of b to learn the output.

2.2 TFHE-Based Protocol for $\mathcal{F}_{\mathsf{CTest\text{-}diff}}$

The above alternative construction can be slightly modified to compute $\mathcal{F}_{\mathsf{CTest\text{-}diff}}$ from TFHE. In particular, party P_1 homomorphically interpolates $\widetilde{\mathsf{p}}_i(x) = \frac{\mathsf{p}_i(x)}{\mathsf{p}_1(x)} = \frac{\mathsf{p}_{i\setminus 1}(x)}{\mathsf{p}_{1\setminus i}(x)}$ from $(2T+1)$ evaluations and computes encrypted $D_{1,i} = S_1 \setminus S_i$ as well as $D_{i,1} = S_i \setminus S_1$ for every other party P_i. Note that if $|\left(\bigcup_{i=1}^{m} S_i\right) \setminus I| \leq T$, then $|S_i \setminus I| \leq T$ for all i and the degree of each $\widetilde{\mathsf{p}}_i(x)$ is at most $2T$, hence P_1 can interpolate it using $(2T + 1)$ evaluations. Observe that $\left(\bigcup_{i=1}^{m} S_i\right) \setminus I = \bigcup_{i=2}^{m} (D_{1,i} \cup D_{i,1})$, because each element $a \in \left(\bigcup_{i=1}^{m} S_i\right) \setminus I$ must be one of the two cases: (1) $a \in S_1$ and $a \notin S_i$ for some i (i.e., $a \in D_{1,i}$), or (2) $a \notin S_1$ and $a \in S_i$ for some i (i.e., $a \in D_{i,1}$). Therefore, party P_1 can homomorphically compute an encryption of $\left(\bigcup_{i=1}^{m} S_i\right) \setminus I$ and an encryption of predicate $b = \left(|(\bigcup_{i=1}^{m} S_i) \setminus I| \overset{?}{\leq} T\right)$. Finally, as before, the n parties jointly decrypt the encryption of b to learn the output.

2.3 TAHE-Based Protocol for $\mathcal{F}_{\mathsf{CTest\text{-}diff}}$

To construct multi-party private intersection cardinality testing from threshold additive homomorphic encryption (TAHE), we start with the two-party protocol [GS19a] based on additive homomorphic encryption (AHE).

Two-party protocol. For two parties Alice and Bob with private sets S_A and S_B, if we encode their elements into two polynomials $\mathsf{p}_A(x) = \sum_{i=1}^{m} x^{a_i}$ and $\mathsf{p}_B(x) = \sum_{i=1}^{m} x^{b_i}$, then the number of monomials in the polynomial $\mathsf{p}(x) := \mathsf{p}_A(x) - \mathsf{p}_B(x)$ is exactly $|(S_A \setminus S_B) \cup (S_B \setminus S_A)|$. Now the problem of cardinality testing (i.e., determining if $|(S_A \setminus S_B) \cup (S_B \setminus S_A)| \leq 2T$) has be reduced to determining whether the number of monomials in $\mathsf{p}(x)$ is $\leq 2T$. Using the polynomial sparsity test of Grigorescu et al. [GJR10], we can further reduce the problem to determining whether the Hankel matrix below is singular or not:

$$H = \begin{bmatrix} \mathsf{p}(u^0) & \mathsf{p}(u^1) & \dots & \mathsf{p}(u^{2T}) \\ \mathsf{p}(u^1) & \mathsf{p}(u^2) & \dots & \mathsf{p}(u^{2T+1}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathsf{p}(u^{2T}) & \mathsf{p}(u^{2T+1}) & \dots & \mathsf{p}(u^{4T}) \end{bmatrix},$$

where u is chosen uniformly at random. In the two-party protocol, Alice generates a public-secret key pair of the AHE scheme and sends Bob the public key, a uniformly random u along with encrypted Hankel matrix for p_A. Then Bob can homomorphically compute an encryption of Hankel matrix for p. Now Alice holds the secret key and Bob holds an encryption of matrix H. They need to jointly perform a matrix singularity testing to determine if the matrix is singular or not without revealing any other information, which can be done using the protocol of Kiltz et al. [KMWF07].

Our approach. In our multi-party protocol, we will make use of n-out-of-n threshold additive homomorphic encryption (TAHE) with distributed setup. Our strategy is to first find a polynomial where the number of monomials equals the size of the set difference $|(\bigcup_{i=1}^{m} S_i) \setminus I|$. Furthermore, the polynomial should only involve linear operations among the parties to allow for additive homomorphic evaluation. Then, we construct a multi-party protocol for matrix singularity testing.

Identifying the right polynomial. At first sight, if we want to compute the total set difference $|(\bigcup_{i=1}^{m} S_i) \setminus I|$, we should take into account the set difference between every pair of parties. However, from our experience in the TFHE-based constructions, it suffices to consider the set difference between P_1 and every other party. In particular, we encode the set $S_i = \{a_1^i, \dots, a_m^i\}$ as $\mathsf{p}_i(x) := \sum_{j=1}^{m} x^{a_j^i}$, and let $\mathsf{p}(x) := (\mathsf{p}_1(x) - \mathsf{p}_2(x)) + \dots + (\mathsf{p}_1(x) - \mathsf{p}_n(x)) = (n-1)\mathsf{p}_1(x) - \sum_{i=2}^{n} \mathsf{p}_i(x)$. We can show that the number of terms in $\mathsf{p}(x)$ is exactly $|(\bigcup_{i=1}^{m} S_i) \setminus I|$. If the parties first jointly generate public key and secret key shares of TAHE, then every party P_i (except P_1) can send an encryption of Hankel matrix for p_i to party P_1 and P_1 can homomorphically compute an encryption of Hankel matrix for p. Now, party P_1 holds an encryption of matrix H, where the secret key is shared among all the parties.

Our next goal is to construct a new multi-party protocol that allows the n parties to jointly decide whether the matrix is singular or not generalizing the result in the work of Kiltz et al. [KMWF07].

Multi-party homomorphic matrix multiplication. A key building block in the two-party protocol of Kiltz et al. [KMWF07] is a two-party protocol for homomorphic matrix multiplication. Here, we design a multi-party protocol for this task. In particular, P_1 holds two encrypted matrices $\mathsf{Enc}(A)$ and $\mathsf{Enc}(B)$ where the secret key is shared among all the parties and they want to jointly enable P_1 to learn $\mathsf{Enc}(A \cdot B)$ without leaking any other information. We briefly describe our multi-party homomorphic matrix multiplication protocol inspired by [Bea91]. First, each party P_i sends encryptions of two matrices R_i^A and R_i^B to party P_1, where R_i^A and R_i^B are uniformly random matrices sampled by P_i. Then P_1 homomorphically computes encryptions of $R^A = \sum_{i=1}^n R_i^A$, $M^A = A + R^A$, $M^B = B + \sum_{i=1}^n R_i^B$ and sends to all the parties. The n parties can jointly decrypt two matrices M^A and M^B to P_1. Then, each party P_i homomorphically computes encryptions of three matrices $M^A R_i^B, R_i^A M^B, R^A R_i^B$ and sends them to P_1. Finally, P_1 can homomorphically compute an encryption of $A \cdot B$ by performing a linear combination of these ciphertexts. In particular, $A \cdot B = M^A M^B - \sum_{i=1}^n M^A R_i^B - \sum_{i=1}^n R_i^A M^B + \sum_{i=1}^n R^A R_i^B$. We refer the reader to Section 6 for more details on the other components of our multi-party protocol for singularity testing.

Subtle issue with modularization. There is a subtle issue when incorporating the multi-party matrix singularity testing protocol into the private intersection cardinality testing. Ideally we would like to prove security separately for the multi-party matrix singularity testing protocol and then use it as a black box. However, it turns out formally proving the security of our multi-party matrix singularity testing protocol is hard. In particular, if we formalize the ideal functionality for the multi-party matrix singularity testing, one party holds an encrypted matrix and the secret key is shared among all the parties. In a simulation-based security proof, we would hope to argue that nothing is leaked about the underlying matrix by relying on the security of the underlying encryption scheme, intuitively, based on the fact that any subset of parties cannot recover the secret key of the encryption scheme. On the other hand, since the distinguisher can in fact choose the inputs for all the parties, it knows all the secret key shares of the encryption scheme. Hence, we can not rely on the security of the encryption scheme. The same issue arises when we try to modularize multi-party homomorphic matrix multiplication and formally prove it separately. The fundamental problem in both scenarios is that when formalizing these ideal functionalities, the natural design would result in involving cryptographic primitives as part of the input/output behaviour, which makes it hard to argue security based on these primitives. We get around this problem by including all the sub-protocols in our final multi-party private intersection cardinality testing protocol and directly providing a single security proof for our cardinality testing protocol.

Challenges for functionality $\mathcal{F}_{\mathsf{CTest\text{-}int}}$. We briefly discuss the hardness of applying similar techniques to obtain a TAHE-based protocol for the other functionality $\mathcal{F}_{\mathsf{CTest\text{-}int}}$. The main challenge is to find a polynomial that can be linearly computed by the parties and the number of monomials in the polynomial equals $|S_1 \setminus I|$. If such a polynomial can be found, then the second step of multi-party matrix singularity testing follows the same way. We leave it as an open problem to identify such a polynomial.

2.4 Computing Set Intersection

After the first phase of our protocols (i.e., cardinality testing), we present a single construction for the second phase that computes the concrete set intersection for both $\mathcal{F}_{\mathsf{TPSI\text{-}int}}$ and $\mathcal{F}_{\mathsf{TPSI\text{-}diff}}$. Again, we start from the two-party protocol [GS19a] based on AHE.

Two-party protocol. For two parties Alice and Bob, we use the first encoding method to encode the elements into two polynomials $\mathsf{p}_A(x) = \prod_{i=1}^{m}(x - a_i)$ and $\mathsf{p}_B(x) = \prod_{i=1}^{m}(x - b_i)$. After the cardinality testing, we already know that the rational polynomial $\mathsf{p}(x) := \frac{\mathsf{p}_B(x)}{\mathsf{p}_A(x)} = \frac{\mathsf{p}_{B\setminus I}(x)}{\mathsf{p}_{A\setminus I}(x)}$ has degree at most $2T$. If Alice learns the evaluation of $\mathsf{p}_B(\cdot)$ on $2T+1$ distinct points $\{\alpha_1, \ldots, \alpha_{2T+1}\}$, then she can evaluate p_A on those points by herself and compute $\{\mathsf{p}(\alpha_1), \ldots, \mathsf{p}(\alpha_{2T+1})\}$. Using these evaluations of $\mathsf{p}(\cdot)$, Alice can recover $\mathsf{p}(x)$ by rational polynomial interpolation, and then learn the set difference $S_A \setminus I$ from the denominator of $\mathsf{p}(x)$. However, $\mathsf{p}(x)$ also allows Alice to learn $S_B \setminus I$, which breaks security. Instead of letting Alice learn the evaluations of $\mathsf{p}_B(\cdot)$, the two-party protocol of [GS19a] enables Alice to learn the evaluations of a "noisy" polynomial $\mathsf{V}(x) := \mathsf{p}_A(x) \cdot \mathsf{R}_1(x) + \mathsf{p}_B(x) \cdot \mathsf{R}_2(x)$, where R_1 and R_2 are uniformly random polynomials of degree T. Note that

$$\mathsf{p}'(x) := \frac{\mathsf{V}(x)}{\mathsf{p}_A(x)} = \frac{\mathsf{p}_{A\setminus I}(x) \cdot \mathsf{R}_1(x) + \mathsf{p}_{B\setminus I}(x) \cdot \mathsf{R}_2(x)}{\mathsf{p}_{A\setminus I}(x)}$$

has degree at most $3T$. Given $3T+1$ evaluations of $\mathsf{V}(\cdot)$, Alice can interpolate $\mathsf{p}'(x)$ and figure out the denominator, but now the numerator is sufficiently random and does not leak any other information about S_B. It is crucial that Alice doesn't pick R_1 on her own and similarly, Bob doesn't pick R_2 on his own. The protocol of [GS19a] uses an oblivious linear function evaluation (OLE) protocol to let Alice and Bob jointly generate and evaluate the random polynomials R_1 and R_2.

Alternative approach using TAHE. We take a different approach to let Alice learn the evaluations of $\mathsf{V}(x)$ in the two-party setting using 2-out-of-2 TAHE, which is arguably simpler than the approach in [GS19a] and can be generalized to the multi-party setting. We first define $\mathsf{V}(x) := \Big(\mathsf{p}_A(x) \cdot \big(\mathsf{R}_1^A(x) + \mathsf{R}_1^B(x)\big) + \mathsf{p}_B(x) \cdot \big(\mathsf{R}_2^A(x) + \mathsf{R}_2^B(x)\big)\Big)$, where $(\mathsf{R}_1^A, \mathsf{R}_2^A)$ and $(\mathsf{R}_1^B, \mathsf{R}_2^B)$ are uniformly random polynomials of degree T generated by Alice and Bob, respectively. Instead of relying on OLE to jointly generate the random polynomials R_1 and R_2, now we let each party generate an additive share of the polynomials. To obtain an evaluation of $\mathsf{V}(\alpha)$, Alice first sends an encryption of $\mathsf{p}_A(\alpha)$ and $\mathsf{R}_2^A(\alpha)$ to Bob. Then Bob homomorphically computes an encryption of $r = \mathsf{p}_A(\alpha) \cdot \mathsf{R}_1^B(\alpha) + \mathsf{p}_B(\alpha) \cdot \big(\mathsf{R}_2^A(\alpha) + \mathsf{R}_2^B(\alpha)\big)$ and sends it back to Alice. Alice can homomorphically compute an encryption of $\mathsf{V}(\alpha) = \mathsf{p}_A(\alpha) \cdot \mathsf{R}_1^A(\alpha) + r$. Finally, both parties jointly decrypt it.

Generalization. To generalize to multiple parties, we first encode each set $S_i = \{a_1^i, \ldots, a_m^i\}$ as a polynomial $\mathsf{p}_i(x) := \prod_{j=1}^{m}(x - a_j^i)$, and then define

$$\begin{aligned}\mathsf{V}(x) :=& \mathsf{p}_1(x) \cdot \mathsf{R}_1(x) + \cdots + \mathsf{p}_n(x) \cdot \mathsf{R}_n(x) \\ :=& \mathsf{p}_1(x) \cdot (\mathsf{R}_{1,1}(x) + \cdots + \mathsf{R}_{n,1}(x)) + \cdots + \mathsf{p}_n(x) \cdot (\mathsf{R}_{1,n}(x) + \cdots + \mathsf{R}_{n,n}(x)),\end{aligned}$$

where $(\mathsf{R}_{i,1}, \ldots, \mathsf{R}_{i,n})$ are uniformly random polynomials of degree T generated by party P_i. For both functionalities $\mathcal{F}_{\mathsf{TPSI\text{-}int}}$ and $\mathcal{F}_{\mathsf{TPSI\text{-}diff}}$, if the protocol passes the cardinality testing, then

$$\mathsf{p}'(x) := \frac{\mathsf{V}(x)}{\mathsf{p}_1(x)} = \frac{\mathsf{p}_{1\setminus I}(x) \cdot \mathsf{R}_1(x) + \cdots + \mathsf{p}_{n\setminus I}(x) \cdot \mathsf{R}_n(x)}{\mathsf{p}_{1\setminus I}(x)}$$

has degree at most $3T$. If P_1 learns $3T+1$ evaluations of $\mathsf{V}(\cdot)$, then it can interpolate $\mathsf{p}'(x)$ and recover $S_1 \setminus I$ from the denominator while the numerator does not leak any other information. To enable party P_1 to learn an evaluation of $\mathsf{V}(\alpha)$ after the TAHE setup, every party P_i first sends an encryption of $\{\mathsf{R}_{i,1}(\alpha), \ldots, \mathsf{R}_{i,n}(\alpha)\}$ to P_1. Then, P_1 can homomorphically compute an encryption of $\{\mathsf{R}_1(\alpha), \ldots, \mathsf{R}_n(\alpha)\}$ and send an encryption of $\mathsf{R}_i(\alpha)$ to party P_i. Each party P_i then homomorphically computes an encryption of $\mathsf{p}_i(\alpha_i) \cdot \mathsf{R}_i(\alpha)$ and sends it back to P_1. Finally, P_1 homomorphically computes an encryption of $\mathsf{V}(\alpha) = \sum_{i=1}^{n} (\mathsf{p}_i(\alpha_i) \cdot \mathsf{R}_i(\alpha))$ and jointly decrypts it with all the other parties.

Communication blow-up. This protocol requires $O(n^2)$ communication complexity per evaluation, and the total communication complexity is $O(n^2T)$ for $(3T+1)$ evaluations. However, recall that our goal was to run this phase of the protocol with just $O(nT)$ communication to match the lower bound $\Omega(nT)$. Observe that the bottleneck of the communication in this approach is in the first step, where every party P_i sends n encrypted evaluations $\{\mathsf{R}_{i,1}(\alpha), \ldots, \mathsf{R}_{i,n}(\alpha)\}$ to P_1. We now focus on reducing the communication complexity of the protocol.

Idea 1: Mask your own polynomials. Our first idea is to reduce the number of random polynomials used in defining $\mathsf{V}(\cdot)$. Intuitively, suppose the central party of the network P_1 is honest, then for every other party P_i, it seems sufficient that the polynomial $\mathsf{p}_i(x)$ is masked by a random polynomial chosen by P_1. Similarly, suppose P_1 is corrupt, for any honest party P_i, it seems sufficient that it contributes two random polynomials: one to mask its own random polynomial $\mathsf{p}_i(x)$ and one to mask $\mathsf{p}_1(x)$. In particular, we re-define $\mathsf{V}(x)$ as

$$\begin{aligned}\mathsf{V}(x) :=& \mathsf{p}_1(x) \cdot \mathsf{R}_1(x) + \cdots + \mathsf{p}_n(x) \cdot \mathsf{R}_n(x) \\ :=& \mathsf{p}_1(x) \cdot (\mathsf{R}_{1,1}(x) + \cdots + \mathsf{R}_{n,1}(x)) + \mathsf{p}_2(x) \cdot \left(\mathsf{R}_{1,2}(x) + \widetilde{\mathsf{R}}_2(x)\right) + \cdots + \mathsf{p}_n(x) \cdot \left(\mathsf{R}_{1,n}(x) + \widetilde{\mathsf{R}}_n(x)\right),\end{aligned}$$

where $(\mathsf{R}_{1,1}, \ldots, \mathsf{R}_{1,n})$ are random polynomials of degree T generated by P_1, and every other party P_i only generates two random polynomials $\mathsf{R}_{i,1}$ and $\widetilde{\mathsf{R}}_i$. Now, in the first step of our protocol, each party P_i only needs to send 2 encrypted evaluations $\mathsf{R}_{i,1}(\alpha)$ and $\widetilde{\mathsf{R}}_i(\alpha)$ to P_1, and the total communication complexity can be reduced to $O(nT)$.

Every polynomial needs a mask. However, it turns out this protocol is in fact insecure because evaluations of $\mathsf{V}(\cdot)$ may leak more information than expected! In particular, consider the case where all parties except P_n are corrupted. Then, they can jointly compute $\mathsf{V}'(\alpha) = \left(\mathsf{p}_1(\alpha) \cdot \mathsf{R}_1(\alpha) + \mathsf{p}_n(\alpha) \cdot \mathsf{R}_n(\alpha)\right)$ from $\mathsf{V}(\alpha)$. Given $3T+1$ evaluations of $\mathsf{V}'(\cdot)$, they can interpolate the rational polynomial

$$\frac{\mathsf{V}'(\alpha)}{\mathsf{p}_1(\alpha)} = \frac{\mathsf{p}_{1\backslash n}(\alpha) \cdot \mathsf{R}_1(\alpha) + \mathsf{p}_{n\backslash 1}(\alpha) \cdot \mathsf{R}_n(\alpha)}{\mathsf{p}_{1\backslash n}(\alpha)}.$$

The denominator leaks the set difference $S_1 \backslash S_n$, which shouldn't be leaked in the ideal functionality. The lesson we learned from above is that every polynomial $\mathsf{p}_i(x)$ should be masked by at least one random polynomial contributed by an honest party. However, this brings us back to our first solution where the communication complexity was $O(n^2T)$.

Idea 2: Re-use masks for other parties. Our second idea is to re-use the random polynomials of honest parties in such a way that each honest party P_i picks one random polynomial $\widetilde{\mathsf{R}}_i$ to mask its own polynomial p_i and one random polynomial R_i to mask those of all other parties. That is:

$$\mathsf{V}(x) := \mathsf{p}_1(x) \cdot \left(\widetilde{\mathsf{R}}_1(x) + \mathsf{R}_2(x) + \cdots + \mathsf{R}_n(x)\right) + \mathsf{p}_2(x) \cdot \left(\mathsf{R}_1(x) + \widetilde{\mathsf{R}}_2(x) + \mathsf{R}_3(x) + \cdots + \mathsf{R}_n(x)\right)$$

$$+\cdots+\mathsf{p}_n(x)\cdot\Big(\mathsf{R}_1(x)+\cdots+\mathsf{R}_{n-1}(x)+\widetilde{\mathsf{R}}_n(x)\Big),$$

where R_i and $\widetilde{\mathsf{R}}_i$ are random polynomials of degree T generated by party P_i. In the first step of our protocol, each party P_i still only sends 2 encrypted evaluations $\mathsf{R}_i(\alpha)$ and $\widetilde{\mathsf{R}}_i(\alpha)$ to P_1, and the total communication complexity is $O(nT)$. Consider the same attack as above. If all parties except P_n are corrupted, they can jointly compute

$$\mathsf{V}'(\alpha)=(\mathsf{p}_1(\alpha)+\cdots+\mathsf{p}_{n-1}(\alpha))\cdot\mathsf{R}_n(\alpha)+\mathsf{p}_n(x)\cdot\Big(\mathsf{R}_1(x)+\cdots+\mathsf{R}_{n-1}(x)+\widetilde{\mathsf{R}}_n(x)\Big)$$

from $\mathsf{V}(\alpha)$. Given $3T+1$ evaluations of $\mathsf{V}'(\cdot)$, they can interpolate the rational polynomial

$$\frac{\mathsf{V}'(x)}{\mathsf{p}_1(x)}=\frac{\big(\mathsf{p}_{1\setminus I}(x)+\cdots+\mathsf{p}_{n-1\setminus I}(x)\big)\cdot\mathsf{R}_n(x)+\mathsf{p}_{n\setminus I}(x)\cdot\Big(\mathsf{R}_1(x)+\cdots+\mathsf{R}_{n-1}(x)+\widetilde{\mathsf{R}}_n(x)\Big)}{\mathsf{p}_{1\setminus I}(x)}.$$

The numerator seems to be sufficiently random as both terms include random polynomials (R_n and $\widetilde{\mathsf{R}}_n$) contributed by the honest party P_n.

Return of the unexpected degree reduction. However, there is a subtle issue that we have seen when constructing a TFHE-based protocol for $\mathcal{F}_{\mathsf{CTest\text{-}int}}$. In particular, $(\mathsf{p}_1(x)+\cdots+\mathsf{p}_{n-1}(x))$ might accidentally create some roots that are not in the intersection but can be canceled out with p_1 and p_n. A concrete example is four sets $S_1=\{a\},S_2=\{b\},S_3=\{c\},S_4=\{a\}$ where $b+c=2\cdot a$.

To resolve this problem, we follow the same approach as in the TFHE-based protocol. In particular, we set each polynomial $\mathsf{p}'_i(x):=\mathsf{p}_i(x)\cdot(x-r_i)$ for a random r_i chosen by P_i and define

$$\begin{aligned}\mathsf{V}(x)=&\mathsf{p}'_1(x)\cdot\Big(\widetilde{\mathsf{R}}_1(x)+\mathsf{R}_2(x)+\cdots+\mathsf{R}_n(x)\Big)+\mathsf{p}'_2(x)\cdot\Big(\mathsf{R}_1(x)+\widetilde{\mathsf{R}}_2(x)+\mathsf{R}_3(x)+\cdots+\mathsf{R}_n(x)\Big)\\&+\cdots+\mathsf{p}'_n(x)\cdot\Big(\mathsf{R}_1(x)+\cdots+\mathsf{R}_{n-1}(x)+\widetilde{\mathsf{R}}_n(x)\Big).\end{aligned}$$

This makes our protocol secure while maintaining the communication complexity to be $O(nT)$.

2.5 Lower Bounds

We briefly discuss the communication lower bound for multi-party threshold PSI. To prove lower bound in the point-to-point network, we perform a reduction from two-party threshold PSI (for which [GS19a] showed a lower bound of $\Omega(T)$) to multi-party threshold PSI. We first prove that the total "communication complexity of any party" is $\Omega(T)$ which denotes the sum of all the bits exchanged by that party (both sent and received). As a corollary, the total communication complexity of any multi-party threshold PSI protocol is $\Omega(nT)$. We refer to Section 4 for more details about the reduction.

To prove a lower bound in the broadcast model, we rely on the communication lower bound of the multi-party set disjointness problem shown by Braverman and Oshman [BO15]. We reduce the problem of multi-party set disjointness to multi-party threshold PSI $\mathcal{F}_{\mathsf{TPSI\text{-}int}}$ and prove a lower bound $\Omega(T\log n+n)$ for any multi-party threshold PSI protocol in the broadcast network. We refer to Section 9 for more details about the reduction.

2.6 Roadmap

We describe some preliminaries and definitions in Section 3, the lower bound in Section 4, and the TFHE based private intersection cardinality testing protocols in Section 5. We describe the TAHE based private intersection cardinality testing protocol in Section 6. Finally, we present the second phase protocol to compute the actual intersection in Section 7.

3 Preliminaries

3.1 Notations

We use λ to denote the security parameters. By $\mathsf{poly}(\lambda)$ we denote a polynomial function in λ. By $\mathsf{negl}(\lambda)$ we denote a negligible function, that is, a function f such that $f(\lambda) < 1/p(\lambda)$ holds for any polynomial $p(\cdot)$ and sufficiently large λ. We use $[\![x]\!]$ to denote an encryption of x. We use $\widetilde{O}(x)$ to ignore any $\mathsf{polylog}$ factor, namely $\widetilde{O}(x) = O(x \cdot \mathsf{polylog}(x))$.

3.2 Secure Multi-Party Computation

We follow the security definitions for secure multi-party computation (MPC) in the universal composability (UC) framework [Can01]. We provide a brief overview here and refer the reader to [Can01] for more details of the security model.

We consider a protocol Π amongst n parties that implements an ideal functionality $\mathcal{F}$. We define security in the real/ideal world paradigm. In the real execution, the parties execute the protocol Π, which is allowed to call any ideal functionality $\mathcal{G}$. An environment $\mathcal{Z}$ chooses the inputs of all parties, models everything that is external to the protocol, and represents the adversary. $\mathcal{Z}$ may corrupt any subset (not all) of the parties and get access to those parties' internal tapes. In the ideal execution, n dummy parties send their inputs to the ideal functionality $\mathcal{F}$ and get back the output of the computation. A simulator Sim, also known as the ideal world adversary, emulates $\mathcal{Z}$'s view of a real protocol execution. Sim has full control of the corrupted dummy parties and emulates $\mathcal{Z}$'s view of those parties. Let $\mathsf{Real}[\mathcal{Z}, \Pi, \mathcal{G}]$ and $\mathsf{Ideal}[\mathcal{Z}, \mathsf{Sim}, \mathcal{F}]$ be the output of $\mathcal{Z}$ in the real and ideal executions, respectively. We say the protocol Π securely implements $\mathcal{F}$ if for any PPT $\mathcal{Z}$, there exists a PPT Sim such that

$$\mathsf{Real}[\mathcal{Z}, \Pi, \mathcal{G}] \overset{c}{\approx} \mathsf{Ideal}[\mathcal{Z}, \mathsf{Sim}, \mathcal{F}].$$

3.3 Multi-Party Threshold Private Set Intersection

Setting. Consider n parties $P_1, \ldots, P_n$ with input sets $S_1, \ldots, S_n$ respectively. Throughout the paper, we consider all the sets to be of equal size m. We assume that the set elements come from a field $\mathbb{F}_p$, where p is a $\Theta(\lambda)$-bit prime. Also, throughout the paper, we focus only on the point-to-point network channels. For the lower bounds, we consider a setting where every pair of parties has a point-to-point channel between them. For the upper bounds, we consider a more restrictive model – the star network, where only one central party has a point-to-point channel with every other party and the other parties cannot communicate with each other. Note that when an adversary corrupts a set of parties, they are free to communicate and share information amongst each other.

The goal of the parties is to run an MPC protocol Π at the end of which each party learns the intersection I of all the sets if certain conditions hold. In the definition of two-party threshold PSI, both parties P_1 and P_2 learn the intersection I if the size of their set difference is small, namely $|(S_1 \setminus S_2) \cup (S_2 \setminus S_1)| < 2T$. In the multi-party case, we consider two different functionalities, each of which might be better suited to different applications.

Functionalities. In the first definition, we consider functionality $\mathcal{F}_{\mathsf{TPSI\text{-}int}}$, in which each party P_i learns the intersection I if the size of its own set minus the intersection is small, namely $|S_i \setminus I| \leq T$ for some threshold T. Recall that we consider all the sets to be of equal size, hence either all the parties learn the output or all of them don't. In the second definition, we consider a functionality $\mathcal{F}_{\mathsf{TPSI\text{-}diff}}$, where each party learns the intersection I if the size of the union of all the sets minus the intersection is small, namely $|(\bigcup_{i=1}^{n} S_i) \setminus I| \leq T$. The formal definitions of the two ideal functionalities are shown in Figure 1 and Figure Figure 2.

Parameters: Parties $P_1, \ldots, P_n$. Each party has a set of m elements. Threshold $T \in \mathbb{N}$.

Inputs: Party P_i has an input set $S_i = \{a_1^i, \ldots, a_m^i\}$ where $a_j^i \in \mathbb{F}_p$ for all $j \in [m]$.

Output: Each party P_i receives the set intersection $I = \bigcap_{i=1}^{n} S_i$ if and only if $|S_i \setminus I| \leq T$.

Figure 1: Ideal functionality $\mathcal{F}_{\mathsf{TPSI\text{-}int}}$ for multi-party threshold PSI.

Parameters: Parties $P_1, \ldots, P_n$. Each party has a set of m elements. Threshold $T \in \mathbb{N}$.

Inputs: Party P_i has an input set $S_i = \{a_1^i, \ldots, a_m^i\}$ where $a_j^i \in \mathbb{F}_p$ for all $j \in [m]$.

Output: Each party P_i receives the set intersection $I = \bigcap_{i=1}^{n} S_i$ if and only if $|(\bigcup_{i=1}^{n} S_i) \setminus I| \leq T$.

Figure 2: Ideal functionality $\mathcal{F}_{\mathsf{TPSI\text{-}diff}}$ for multi-party threshold PSI.

3.4 Multi-Party Private Intersection Cardinality Testing

An important building block in our multi-party threshold PSI protocols is a multi-party protocol for private intersection cardinality testing which we define below. Consider n parties $P_1, \ldots, P_n$ with input sets $S_1, \ldots, S_n$ respectively of equal size m. Their goal is to run an MPC protocol Π at the end of which each party learns whether the size of the intersection I of all the sets is sufficiently large. As before, we consider two functionalities. In the first functionality $\mathcal{F}_{\mathsf{CTest\text{-}int}}$, each party P_i learns whether $|S_i \setminus I| \leq T$. In the second functionality $\mathcal{F}_{\mathsf{CTest\text{-}diff}}$, each party learns whether $|(\bigcup_{i=1}^{n} S_i) \setminus I| \leq T$. The formal definitions of the two ideal functionalities are presented in Figure 3 and Figure 4.

Parameters: Parties $P_1, \ldots, P_n$. Each party has a set of m elements. Threshold $T \in \mathbb{N}$.

Inputs: Party P_i has an input set $S_i = \{a_1^i, \ldots, a_m^i\}$ where $a_j^i \in \mathbb{F}_p$ for all $j \in [m]$.

Output: Each party P_i receives $\mathsf{similar}$ if $|S_i \setminus I| \leq T$ and $\mathsf{different}$ otherwise where $I = \bigcap_{i=1}^{n} S_i$.

Figure 3: Ideal functionality $\mathcal{F}_{\mathsf{CTest\text{-}int}}$ for multi-party private intersection cardinality testing.

Parameters: Parties $P_1, \ldots, P_n$. Each party has a set of m elements. Threshold $T \in \mathbb{N}$.

Inputs: Party P_i has an input set $S_i = \{a_1^i, \ldots, a_m^i\}$ where $a_j^i \in \mathbb{F}_p$ for all $j \in [m]$.

Output: Each party P_i receives $\mathsf{similar}$ if $|(\bigcup_{i=1}^n S_i) \setminus I| \le T$ and $\mathsf{different}$ otherwise where $I = \bigcap_{i=1}^n S_i$.

Figure 4: Ideal functionality $\mathcal{F}_{\mathsf{CTest\text{-}diff}}$ for multi-party private intersection cardinality testing.

3.5 Threshold Fully Homomorphic Encryption

We define the notion of a threshold fully homomorphic encryption (TFHE) scheme with distributed setup introduced in the work of Boneh et al. [BGG+18]. Also, throughout the paper, we are only interested in the n-out-of-n threshold setting.

Definition 3.1. ***(Threshold Fully Homomorphic Encryption (TFHE) with Distributed Setup)*** *Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a set of parties. A TFHE scheme consists of a tuple of PPT algorithms* $\mathsf{TFHE} = (\mathsf{TFHE.DistSetup}, \mathsf{TFHE.Enc}, \mathsf{TFHE.Eval}, \mathsf{TFHE.PartialDec}, \mathsf{TFHE.Combine})$ *with the following syntax:*

- $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{TFHE.DistSetup}(1^\lambda, i)$*: On input the security parameter λ and a party index i, the distributed setup algorithm outputs the parameters associated with the i-th party: a component of the public key pk_i, and a share of the secret key sk_i. We denote the public key of the scheme* pk *to be* $(\mathsf{pk}_1 || \ldots || \mathsf{pk}_n)$.

- $[\![m]\!] \leftarrow \mathsf{TFHE.Enc}(\mathsf{pk}, m)$*: On input a public key* pk *and a plaintext m in the message space $\mathcal{M}$, the encryption algorithm outputs a ciphertext $[\![m]\!]$.*

- $[\![y]\!] \leftarrow \mathsf{TFHE.Eval}(\mathsf{pk}, \mathsf{C}, [\![m_1]\!], \ldots, [\![m_k]\!])$*: On input a public key* pk*, a circuit* C *of polynomial size that takes k inputs each from the message space and outputs one value in the message space, and a set of ciphertexts $[\![m_1]\!], \ldots, [\![m_k]\!]$, the evaluation algorithm outputs a ciphertext $[\![y]\!]$.*

- $[\![m : \mathsf{sk}_i]\!] \leftarrow \mathsf{TFHE.PartialDec}(\mathsf{sk}_i, [\![m]\!])$*: On input a a secret key share* sk_i *and a ciphertext $[\![m]\!]$, the partial decryption algorithm outputs a partial decryption $[\![m : \mathsf{sk}_i]\!]$.*

- $m/\bot \leftarrow \mathsf{TFHE.Combine}(\mathsf{pk}, \{[\![m : \mathsf{sk}_i]\!]\}_{i \in [n]})$*: On input a public key* pk *and a set of partial decryptions $\{[\![m : \mathsf{sk}_i]\!]\}_{i \in [n]}$, the combination algorithm either outputs a plaintext m or the symbol $\bot$.*

As in a standard homomorphic encryption scheme, we require that a TFHE scheme satisfies compactness, correctness and security. We discuss these properties below.

Compactness. A TFHE scheme is said to be compact if there exists polynomials $\mathsf{poly}_1(\cdot)$ and $\mathsf{poly}_2(\cdot)$ such that for all λ, all message spaces $\mathcal{M}$ with size of each message being $\mathsf{poly}_3(\lambda)$, circuit C of size at most $\mathsf{poly}_4(\lambda)$ and $m_i \in \mathcal{M}$ for $i \in [k]$, the following condition holds. For $\{(\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{TFHE.DistSetup}(1^\lambda, j)\}_{j \in [n]}$, $[\![m_i]\!] \leftarrow \mathsf{TFHE.Enc}(\mathsf{pk}, m_i)$ for $i \in [k]$, $[\![y]\!] \leftarrow \mathsf{TFHE.Eval}(\mathsf{pk}, \mathsf{C}, [\![m_1]\!], \ldots, [\![m_k]\!])$, $[\![y : \mathsf{sk}_j]\!] = \mathsf{TFHE.PartialDec}([\![y]\!], \mathsf{sk}_j)$ for $j \in [n]$, $|\,[\![y]\!]\,| \le \mathsf{poly}_1(\lambda)$ and $|\,[\![y : \mathsf{sk}_j]\!]\,| \le \mathsf{poly}_2(\lambda)$.[5]

[5]Note that in the definition given in [BGG+18], the size of the partial decryption also grows with n for arbitrary t-out-of-n threshold schemes. Here, we focus only on the n-out-of-n threshold, in which case there is no dependence

Evaluation Correctness. Informally, a TFHE scheme is said to be correct if recombining partial decryptions of a ciphertext output by the evaluation algorithm returns the correct evaluation of the corresponding circuit on the underlying plaintexts. Formally, We say that a TAHE scheme satisfies evaluation correctness if for all λ, all message spaces $\mathcal{M}$, circuit C and $m_i \in \mathcal{M}$ for $i \in [k]$, the following condition holds. For $\{(\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{TFHE.DistSetup}(1^\lambda, j)\}_{j\in[n]}$, $[\![m_i]\!] \leftarrow \mathsf{TFHE.Enc}(\mathsf{pk}, m_i)$ for $i \in [k]$, $[\![y]\!] \leftarrow \mathsf{TFHE.Eval}(\mathsf{pk}, \mathsf{C}, [\![m_1]\!], \ldots, [\![m_k]\!])$,

$$\Pr[\mathsf{TFHE.Combine}(\mathsf{pk}, \mathsf{TFHE.PartialDec}([\![y]\!], \mathsf{sk}_i)_{i\in[n]}) = \mathsf{C}(m_1, \ldots, m_k)] = 1 - \mathsf{negl}(\lambda).$$

Semantic Security. Informally, a TFHE scheme is said to provide semantic security if any PPT adversary cannot distinguish between encryptions of arbitrarily chosen plaintext messages m_0 and m_1, even given the secret key shares corresponding to a subset $\mathcal{S}$ of the parties for any set $\mathcal{S}$ of size at most $(n-1)$. Formally, a TFHE scheme satisfies semantic security if for all λ, message space $\mathcal{M}$, for any PPT adversary $\mathcal{A}$, $\Pr[\mathsf{Expt}_{\mathsf{TFHE,sem}}(1^\lambda) = 1] \leq 1/2 + \mathsf{negl}(\lambda)$ where the experiment $\mathsf{Expt}_{\mathsf{TFHE,sem}}(1^\lambda)$ is defined as:

$\underline{\mathsf{Expt}_{\mathsf{TFHE,sem}}(1^\lambda)}$:

1. On input the security parameter 1^λ, the message space $\mathcal{M}$ and the number of parties n, the adversary $\mathcal{A}$ does the following: Pick a set $\mathcal{S}$ of size at most $(n-1)$. For each $i \in \mathcal{S}$, compute $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{TAHE.DistSetup}(1^\lambda, i)$. Pick two messages (m_0, m_1). Send $(\mathcal{S}, \{\mathsf{pk}_i, \mathsf{sk}_i\}_{i\in\mathcal{S}}, m_0, m_1)$ to the challenger.

2. The challenger runs $\{(\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{TFHE.DistSetup}(1^\lambda, j)\}_{j\in[n]\setminus\mathcal{S}}$, and provides $\{\mathsf{pk}_i\}_{i\in[n]\setminus\mathcal{S}})$ along with $\mathsf{TFHE.Enc}(\mathsf{pk}, m_b)$ to $\mathcal{A}$ where b is picked uniformly at random and $\mathsf{pk} = (\mathsf{pk}_1 || \ldots || \mathsf{pk}_n)$.

3. $\mathcal{A}$ outputs a guess b'. The experiment outputs 1 if $b = b'$.

Simulation Security. Informally, a TFHE scheme is said to provide simulation security if there exists an efficient algorithm $\mathsf{TFHE.Sim}$ that takes as input a circuit C, a ciphertext $[\![y]\!]$ that is computed by running the $\mathsf{TFHE.Eval}$ algorithm on a set of ciphertexts $[\![m_1]\!], \ldots, [\![m_k]\!]$, the output of C on the corresponding plaintexts, and outputs a set of partial decryptions corresponding to some subset of parties, such that its output is computationally indistinguishable from the output of a real algorithm that homomorphically evaluates the circuit C on the ciphertexts $[\![m_1]\!], \ldots, [\![m_k]\!]$ and outputs partial decryptions using the corresponding secret key shares for the same subset of parties. In particular, the computational indistinguishability holds even when a PPT adversary is given the secret key shares corresponding to a subset $\mathcal{S}$ of the parties, so long as $\mathsf{TFHE.Sim}$ also gets the secret key shares corresponding to the set of parties in $\mathcal{S}$.

Formally, a TFHE scheme satisfies simulation security if for all λ, message space $\mathcal{M}$, for any PPT adversary $\mathcal{A}$, there exists a simulator $\mathsf{TFHE.Sim}$ such that the following two experiments $\mathsf{Expt}_{\mathsf{TFHE,Real}}(1^\lambda)$ and $\mathsf{Expt}_{\mathsf{TFHE,Ideal}}(1^\lambda)$ are computationally indistinguishable.

$\underline{\mathsf{Expt}_{\mathsf{TFHE,Real}}(1^\lambda)}$:

on n for the sizes. Also, the definition given in [BGG$^+$18] specifies an upper bound d on the circuit depth and the size of partial decryption grows with the circuit depth. We do not include d in our definition because we will rely on circular-secure LWE to achieve the "strong" compactness.

1. On input of the security parameter 1^λ, the message space $\mathcal{M}$ and the number of parties n, the adversary $\mathcal{A}$ does the following: Pick a set $\mathcal{S}$ of size at most $(n-1)$. For each $i \in \mathcal{S}$, compute $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{TAHE.DistSetup}(1^\lambda, i)$. Send $(\mathcal{S}, \{\mathsf{pk}_i, \mathsf{sk}_i\}_{i \in \mathcal{S}})$ to the challenger.

2. The challenger runs $\{(\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{TFHE.DistSetup}(1^\lambda, 1^d, j)\}_{j \in [n] \setminus \mathcal{S}}$, and provides $(\{\mathsf{pk}_i\}_{i \in [n] \setminus \mathcal{S}})$ to $\mathcal{A}$. Set $\mathsf{pk} := (\mathsf{pk}_1 || \ldots || \mathsf{pk}_n)$.

3. The adversary picks a set of messages $m_1, \ldots, m_k$ for $k = \mathsf{poly}(\lambda)$ and a set $\mathcal{S}_1 \subseteq [k]$. It computes $[\![m_i]\!] := \mathsf{TFHE.Enc}(\mathsf{pk}, m_i; r_i)$ for each $i \in \mathcal{S}_1$ and sends $(m_1, \ldots, m_k, \{r_i\}_{i \in [\mathcal{S}_1]})$ to the challenger.

4. The challenger computes and sends $[\![m_i]\!] := \mathsf{TFHE.Enc}(\mathsf{pk}, m_i; r_i)$ to $\mathcal{A}$ for each $i \in [k] \setminus \mathcal{S}_1$.

5. $\mathcal{A}$ issues a query with a circuit C. The challenger first computes $[\![y]\!] := \mathsf{TFHE.Eval}(\mathsf{pk}, \mathsf{C}, [\![m_1]\!], \ldots, [\![m_k]\!])$ where $[\![m_i]\!] = \mathsf{TFHE.Enc}(\mathsf{pk}, m_i; r_i)$. Then, it outputs $\{\mathsf{TFHE.PartialDec}(\mathsf{sk}_i, [\![y]\!])\}_{i \notin \mathcal{S}}$ to $\mathcal{A}$.

6. The adversary may repeat step 5 $\mathsf{poly}(\lambda)$ many times.

7. At the end of the experiment, $\mathcal{A}$ outputs a distinguishing bit b.

$\mathsf{Expt}_{\mathsf{TFHE,Ideal}}(1^\lambda)$:

1. Perform steps 1-4 of the real world experiment $\mathsf{Expt}_{\mathsf{TFHE,Real}}(1^\lambda)$.

2. $\mathcal{A}$ issues a query with a circuit C. The challenger first computes $[\![y]\!] := \mathsf{TFHE.Eval}(\mathsf{pk}, \mathsf{C}, [\![m_1]\!], \ldots, [\![m_k]\!])$. Then, the challenger outputs $\mathsf{TFHE.Sim}(\mathsf{C}, \mathsf{C}(m_1, \ldots, m_k), [\![y]\!], \{\mathsf{sk}_i\}_{i \in \mathcal{S}})$ to $\mathcal{A}$.

3. The adversary may repeat step 2 $\mathsf{poly}(\lambda)$ many times.

4. At the end of the experiment, $\mathcal{A}$ outputs a distinguishing bit b.

Imported Theorem 1 ([BGG+18, BJMS18])**.** *Assuming circular-secure LWE, there exists a TFHE scheme for the n-out-of-n threshold access structure.*[6]

3.6 Threshold Additive Homomorphic Encryption

We define a threshold additive homomorphic encryption scheme (TAHE) with distributed setup by following the definition of threshold fully homomorphic encryption above but restricting it to only additive homomorphism. A TAHE scheme with distributed setup consists of the following PPT algorithms $\mathsf{TAHE} = (\mathsf{TAHE.DistSetup}, \mathsf{TAHE.Enc}, \mathsf{TAHE.Eval}, \mathsf{TAHE.PartialDec}, \mathsf{TAHE.Combine})$ with the only difference from TFHE being that in the algorithm $\mathsf{TAHE.Eval}$, the circuit C is only allowed to be linear. That is, by a linear circuit, we mean that $\mathsf{C}(x_1, \ldots, x_k) = (\Sigma_{i=1}^k a_i \cdot x_i + b)$ for some values $(a_1, \ldots, a_k, b)$ hardwired into the circuit.

[6]Note that we require circular-secure LWE and not standard LWE only because we require "strong" compactness where the size of the ciphertext and partial decryption doesn't grow with the circuit depth.

Instantiations. We note that several popular additively homomorphic encryption schemes in literature such as ElGamal encryption [Gam84] (based on the Decisional Diffie Hellman assumption) and Paillier encryption [Pai99] (based on the Decisional Composite Residuosity assumption). can in fact be easily converted into a TAHE scheme with the security properties we require. We refer the reader to the work of Hazay and Venkitasubramaniam [HV17] for more details.

Re-randomization. We implicitly assume that each homomorphic evaluation on a set of ciphertexts is concluded with a refresh operation, where the party adds the resulting ciphertext with an independently generated ciphertext that encrypts zero. This is required in order to ensure that the randomness of the final ciphertext is independent of the randomness of the original set of ciphertexts. For the schemes we mentioned above, a homomorphically evaluated ciphertext is statistically identical to a fresh ciphertext. That is, for any $\{(\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{TAHE.DistSetup}(1^\lambda, j)\}_{j\in[n]}$, any linear circuit C with input $m_1, \ldots, m_k$, any $[\![m_i]\!] \leftarrow \mathsf{TAHE.Enc}(\mathsf{pk}, m_i)$, it holds that

$$\mathsf{TAHE.Eval}(\mathsf{pk}, \mathsf{C}, [\![m_1]\!], \ldots, [\![m_k]\!]) \equiv \mathsf{TAHE.Enc}(\mathsf{pk}, \mathsf{C}(m_1, \ldots, m_k)).$$

3.7 Linear Algebra

In the security proofs of our protocols, we will make use of a few lemmas about polynomials stated below. The proofs are postponed to Appendix A.

Imported Lemma 1 (Lemma 2 from [GS19a])**.** *Let $\mathbb{F}$ be a finite field of order $q = \Omega(2^\lambda)$. Let polynomial $p(x) \in \mathbb{F}[x]$ be an arbitrary but fixed non-zero polynomial of degree at most d_p and let $R(x) \in \mathbb{F}[x]$ be a uniformly random polynomial of degree d_R. Then*

$$\Pr[\gcd(p(x), R(x)) \neq 1] \leq \mathsf{negl}(\lambda).$$

Lemma 3.2. *Let $\mathbb{F}$ be a finite field of order $q = \Omega(2^\lambda)$. Fix any $n = O(\mathsf{poly}(\lambda))$. For all polynomials $p_1(x), ..., p_n(x) \in \mathbb{F}[x]$ such that $\gcd(p_1, ..., p_n) = 1$, for all $1 \leq i < n$,*

$$\Pr_{r_j}[\gcd(p'_1 + ... + p'_i, p'_{i+1}, ..., p'_n) \neq 1] \leq \mathsf{negl}(\lambda)$$

where for all $j \in [n]$, $p'_j(x) := p(x) \cdot (x - r_j)$ and $r_j \xleftarrow{\$} \mathbb{F}$.

Lemma 3.3. *Let $\mathbb{F}$ be a finite field of prime order q. Fix any $n = O(\mathsf{poly}(\lambda))$. For all polynomials $p_1(x), \ldots, p_n(x) \in \mathbb{F}[x]$ each of degree α with $\gcd(p_1, \ldots, p_n) = 1$, let $R_1(x), \ldots, R_n(x) \in \mathbb{F}[x]$ be random polynomials with degree $\beta \geq \alpha$. Specifically, $R_1(x) = \sum_{j=0}^{\beta} r_{1,j} x^j, \ldots, R_n(x) = \sum_{j=0}^{\beta} r_{n,j} x^j$ where $r_{i,j} \xleftarrow{\$} \mathbb{F}$ are sampled independently and uniformly at random. Let $U(x) = \sum_{i=1}^{n}(p_i(x) \cdot R_i(x)) = \sum_{j=0}^{\alpha+\beta} u_j x^j$, then u_j's are distributed uniformly and independently over $\mathbb{F}$.*

4 Communication Lower Bound

In this section, we prove communication lower bounds for multi-party threshold PSI protocols in the point-to-point network model. Recall that we consider all parties to have sets of the same size m. We show that any secure protocol must have communication complexity at least $\Omega(n \cdot T)$, where n denotes the number of parties and T is the threshold parameter for both functionalities $\mathcal{F}_{\mathsf{TPSI\text{-}int}}$ and $\mathcal{F}_{\mathsf{TPSI\text{-}diff}}$.

4.1 Lower Bound for $\mathcal{F}_{\textsf{TPSI-int}}$

Before proving the lower bound, we first prove another related theorem below.

Theorem 4.1. *For any multi-party threshold PSI protocol for functionality $\mathcal{F}_{\textsf{TPSI-int}}$ that is secure against a semi-honest adversary that can corrupt up to $(n-1)$ parties, for every party P_i, the communication complexity of P_i is $\Omega(T)$.*[7]

Proof. Suppose this is not true. That is, suppose there exists a secure multi-party threshold PSI protocol Π for functionality $\mathcal{F}_{\textsf{TPSI-int}}$ in which for some party P_{i^*}, $\textsf{CC}(P_{i^*}) = o(T)$ where $\textsf{CC}(\cdot)$ denotes the communication complexity. We will now use this protocol Π as a subroutine to design a secure two-party threshold PSI protocol which has communication complexity $o(T)$.

Consider two parties Q_1 and Q_2 with input sets X_1 and X_2 (of same size m) who wish to run a secure two-party threshold PSI protocol for the following functionality: both parties learn the output if $|(X_1 \setminus X_2) \cup (X_2 \setminus X_1)| \leq 2 \cdot T$. We invoke the multi-party threshold PSI protocol Π with threshold T as follows: Q_1 emulates the role of party P_{i^*} with input set $S_{i^*} = X_1$ and Q_2 emulates the role of all the other $(n-1)$ parties with each of their input sets as X_2. From the definition of the functionality $\mathcal{F}_{\textsf{TPSI-int}}$, Q_1 learns the output at the end of the protocol if and only if $|X_1 \setminus I| \leq T$. Similarly, Q_2 learns the output at the end of the protocol if and only if $|X_2 \setminus I| \leq T$. Notice that since $|X_1| = |X_2|$ and $I = X_1 \cap X_2$, $|X_1 \setminus I| = |X_2 \setminus I|$. Thus, the parties learn the output if and only if $(|X_1 \setminus I|) + (|X_2 \setminus I|) \leq 2 \cdot T$, namely $|(X_1 \setminus X_2) \cup (X_2 \setminus X_1)| \leq 2 \cdot T$, which is the functionality of the two-party threshold PSI. Therefore, correctness is easy to observe. For security, notice that if Q_1 is corrupt, we can simulate it by considering only a corrupt P_{i^*} in the underlying protocol Π and if Q_2 is corrupt, we can simulate it by considering all parties except P_{i^*} to be corrupt in the underlying protocol Π.

Finally, notice that the communication complexity of the two-party protocol is exactly the same as $\textsf{CC}(P_{i^*})$ in the multi-party protocol Π, which is $o(T)$. However, recall from the work of Ghosh and Simkin [GS19a] that any two-party threshold PSI for this functionality has communication complexity lower bound $\Omega(T)$ leading to a contradiction. Thus, the assumption that there exists a secure multi-party PSI protocol Π in which for some party P_{i^*}, $\textsf{CC}(P_{i^*}) = o(T)$ is wrong and this completes the proof of the theorem. $\square$

It is easy to observe that as a corollary of the above theorem, in a setting with only point-to-point channels (which also includes the star network), the overall communication complexity of the protocol must be at least n times the minimum communication complexity that each party is involved in, giving the lower bound of $\Omega(n \cdot T)$. Formally,

Corollary 4.2. *For any multi-party threshold PSI protocol for functionality $\mathcal{F}_{\textsf{TPSI-int}}$ that is secure against a semi-honest adversary that can corrupt up to $(n-1)$ parties, the communication complexity is $\Omega(n \cdot T)$.*

4.2 Lower Bound for $\mathcal{F}_{\textsf{TPSI-diff}}$

The lower bound proof for functionality $\mathcal{F}_{\textsf{TPSI-diff}}$ is very similar to the one for $\mathcal{F}_{\textsf{TPSI-int}}$. The only difference is that in the reduction, we invoke the two-party threshold PSI protocol where both

[7]We define the communication complexity of a party P_i in any protocol execution as the complexity of all the communication that P_i is involved in. That is, the complexity of the messages both incoming to and outgoing from P_i.

parties learn the output if $|(X_1 \setminus X_2) \cup (X_2 \setminus X_1)| \leq T$ instead of $2 \cdot T$ as in the previous case. We elaborate on the proof for the sake of completeness.

Once again, before we prove the lower bound, we first prove another related theorem below.

Theorem 4.3. *For any multi-party threshold private intersection protocol for functionality $\mathcal{F}_{\mathsf{TPSI\text{-}diff}}$ that is secure against a semi-honest adversary that can corrupt up to $(n-1)$ parties, for every party P_i, the communication complexity of P_i is $\Omega(T)$.*

Proof. Suppose this is not true. That is, suppose there exists a secure multi-party threshold PSI protocol Π for functionality $\mathcal{F}_{\mathsf{TPSI\text{-}diff}}$ in which for some party P_{i^*}, $\mathsf{CC}(P_{i^*}) = o(T)$. We will now use this protocol Π as a subroutine to design a secure two-party threshold PSI protocol which has communication complexity $o(T)$.

Consider two parties Q_1 and Q_2 with input sets X_1 and X_2 (of same size) who wish to run a secure two-party threshold PSI protocol for the following functionality: both parties learn the output if $|(X_1 \setminus X_2) \cup (X_2 \setminus X_1)| \leq T$. We invoke the multi-party threshold PSI protocol Π with threshold T as follows: Q_1 emulates the role of party P_{i^*} with input set $S_{i^*} = X_1$ and Q_2 emulates the role of all the other $(n-1)$ parties with each of their input sets as X_2. From the definition of the functionality $\mathcal{F}_{\mathsf{TPSI\text{-}diff}}$, the parties learn the output at the end of the protocol if and only if $|(\bigcup_{i=1}^{n} S_i) \setminus I| \leq T$, namely $|(X_1 \setminus X_2) \cup (X_2 \setminus X_1)| \leq T$, which is the functionality of the two-party threshold PSI primitive. Thus, correctness is easy to observe. For security, notice that if Q_1 is corrupt, we can simulate it by considering only a corrupt P_{i^*} in the underlying protocol Π and if Q_2 is corrupt, we can simulate it by considering all parties except P_{i^*} to be corrupt in the underlying protocol Π.

Finally, notice that the communication complexity of the two-party protocol is exactly the same as $\mathsf{CC}(P_{i^*})$ in the multi-party protocol Π, which is $o(T)$. However, recall from the work of Ghosh and Simkin [GS19a] that any two-party threshold PSI for this functionality has communication complexity lower bound $\Omega(T)$ leading to a contradiction. Thus, the assumption that there exists a secure multi-party threshold PSI protocol Π in which for some party P_{i^*}, $\mathsf{CC}(P_{i^*}) = o(T)$ is wrong and this completes the proof of the theorem. □

Similarly as in the proof for $\mathcal{F}_{\mathsf{TPSI\text{-}int}}$, we get the following corollary from the above theorem:

Corollary 4.4. *For any multi-party threshold PSI protocol for functionality $\mathcal{F}_{\mathsf{TPSI\text{-}diff}}$ that is secure against a semi-honest adversary that can corrupt up to $(n-1)$ parties, the communication complexity is $\Omega(n \cdot T)$ where n is the number of parties and T is the threshold parameter.*

5 TFHE-Based Private Intersection Cardinality Testing

In this section, we present two protocols for private intersection cardinality testing, one for functionalities $\mathcal{F}_{\mathsf{CTest\text{-}int}}$ (described in Figure 3) and the other for $\mathcal{F}_{\mathsf{CTest\text{-}diff}}$ (described in Figure 4). Both protocols are based on n-out-of-n threshold fully homomorphic encryption with distributed setup. The former functionality states that the intersection must be of size at least $(m - T)$ where m is the size of each set. The latter functionality requires the difference between the union of all the sets and the intersection be of size at most T. Due to the possibility of elements appearing in a strict subset of the sets, these two functionalities are not equivalent.

5.1 Protocol for Functionality $\mathcal{F}_{\mathsf{CTest\text{-}int}}$

In this protocol, we compute the cardinality predicate b where $b = 1$ if and only if $\forall i, |S_i \setminus I| \leq T$. The communication complexity of this protocol involves sending $O(nT)$ TFHE ciphertexts and performing a single decryption of the result. We briefly describe the approach below.

Each party P_i first encodes their set S_i as a polynomial $\mathsf{p}_i(x) := \prod_{a \in S_i}(x - a) \in \mathbb{F}[x]$. Each of these polynomials are then randomized as $\mathsf{p}'_i(x) := \mathsf{p}_i(x) \cdot (x - r_i)$ where P_i uniformly samples $r_i \xleftarrow{\$} \mathbb{F}$. The central party also picks a random $z \xleftarrow{\$} \mathbb{F}$ which is sent to every other party. Each party P_i then computes $e_{i,j} := \mathsf{p}'_i(j)$ for $j \in [2T + 3]$ and $e'_i := \mathsf{p}'_i(z)$. P_i sends the ciphertexts $[\![e_{i,j}]\!] := \mathsf{TFHE.Enc}(\mathsf{pk}, e_{i,j})$ and $[\![e'_i]\!] := \mathsf{TFHE.Enc}(\mathsf{pk}, e'_i)$ to P_1. Party P_1 considers the rational polynomial

$$\mathsf{p}'(x) = \frac{\mathsf{p}'_2(x) + \cdots + \mathsf{p}'_n(x)}{\mathsf{p}'_1(x)}$$

and homomorphically computes $2T + 3$ encrypted evaluations

$$\left(j, \left[\!\!\left[\frac{e_{2,j} + \cdots + e_{n,j}}{e_{1,j}}\right]\!\!\right]\right)$$

for $j = [2T + 3]$. Using these encrypted evaluations, P_1 homomorphically computes an encrypted rational polynomial $[\![\mathsf{p}^*(x)]\!]$ using rational polynomial interpolation. Note that $\mathsf{p}^*(x) = \mathsf{p}'(x)$ if $\mathsf{p}'(x)$ has degree at most $2T + 2$. Furthermore, P_1 can homomorphically compute an encryption of the predicate $b := \left(\mathsf{p}^*(z) \stackrel{?}{=} \frac{e'_2 + \cdots + e'_n}{e'_1}\right)$. Finally the parties jointly perform a threshold decryption of $[\![b]\!]$ and party P_1 learns the output which is sent to every other party.

There are a few subtleties in parsing this protocol. First, the protocol attempts to interpolate the rational polynomial $\mathsf{p}'(x) = \frac{\mathsf{p}'_2(x) + \cdots + \mathsf{p}'_n(x)}{\mathsf{p}'_1(x)}$ using $2T+3$ points.[8] This is because $\mathsf{p}'(x)$ has degree at most $2T + 2$ if the intersection is of size at least $m - T$. In particular, if $|I| \geq m - T$, then the roots that encode the intersection will be canceled out in the numerator and denominator of $\mathsf{p}'(x)$ leaving a rational polynomial of degree at most $2T + 2$. If this is the case, then $\mathsf{p}^*(x) = \mathsf{p}'(x)$ and hence $\mathsf{p}^*(z) = \mathsf{p}'(z) = \frac{e'_2 + \cdots + e'_n}{e'_1}$. Otherwise the equality only holds with negligible probability. Another subtlety in the protocol is that we add a random term $(x - r_i)$ in each polynomial $\mathsf{p}'_i(x)$. Note that there is a subtle issue if we only use $(\mathsf{p}_2(x) + \cdots + \mathsf{p}_n(x))$ in the numerator. In particular, it is possible for the sum of these polynomials to accidentally generate roots that are not in the intersection and cancel out with $\mathsf{p}_1(x)$. To prevent this issue we randomize each polynomial $\mathsf{p}_i(x)$ as $\mathsf{p}'_i(x) = \mathsf{p}_i(x) \cdot (x - r_i)$ for some random value r_i. The full protocol is detailed in Figure 5.

Theorem 5.1. *Assuming threshold FHE with distributed setup, protocol $\Pi_{\mathsf{TFHE\text{-}CTest\text{-}int}}$ (Figure 5) securely realizes $\mathcal{F}_{\mathsf{CTest\text{-}int}}$ (Figure 3).*

Proof. **Correctness.** We first prove the protocol is correct. By the correctness of the TFHE scheme, we only need to show that the computed predicate $b = 1$ if and only if $\forall i, |S_i \setminus I| \leq T$. First consider the case where the protocol should output similar. Since

$$\mathsf{p}'(x) = \frac{\mathsf{p}'_2(x) + \cdots + \mathsf{p}'_n(x)}{\mathsf{p}'_1(x)} = \frac{\mathsf{p}_{2\setminus I}(x) \cdot (x - r_2) + \cdots + \mathsf{p}_{n\setminus I}(x) \cdot (x - r_n)}{\mathsf{p}_{1\setminus I}(x) \cdot (x - r_1)},$$

[8] A rational polynomial $p(x) = f(x)/g(x)$ where f, g are of degree d can be uniquely interpolated with $(2d + 1)$ evaluations.

Parameters: Parties $P_1, \dots, P_n$. Each party has a set of m elements. Threshold $T \in \mathbb{N}$. $\mathbb{F}$ is a finite field where $|\mathbb{F}| = \Omega(2^\lambda)$.

Inputs: Party P_i has an input set $S_i = \{a_1^i, \dots, a_m^i\}$ where $a_j^i \in \mathbb{F}$ for all $j \in [m]$.

Output: Each party P_i receives $\mathsf{similar}$ if $|S_i \setminus I| \le T$ and $\mathsf{different}$ otherwise where $I = \bigcap_{i=1}^n S_i$.

Protocol:

1. Each party P_i generates $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{TFHE.DistSetup}(1^\lambda, i)$ and sends pk_i to P_1. Then P_1 sends $\mathsf{pk} = (\mathsf{pk}_1 \| \dots \| \mathsf{pk}_n)$ to all the other parties.
2. P_1 picks a random value $z \in \mathbb{F}$ and sends it to all the other parties.
3. Each party P_i does the following:
   (a) Define the polynomial $\mathsf{p}_i(x) := \prod_{a \in S_i}(x - a)$ and randomize it by $\mathsf{p}_i'(x) := \mathsf{p}_i(x) \cdot (x - r_i)$ where $r_i \overset{\$}{\leftarrow} \mathbb{F}$.
   (b) Compute $e_{i,j} := \mathsf{p}_i'(j)$ for $j \in [2T+3]$ and $e_i' := \mathsf{p}_i'(z)$.
   (c) Send encrypted evaluations $[\![e_{i,j}]\!] := \mathsf{TFHE.Enc}(\mathsf{pk}, e_{i,j})$ for all $j \in [2T+3]$ and $[\![e_i']\!] := \mathsf{TFHE.Enc}(\mathsf{pk}, e_i')$ to P_1.
4. P_1 does the following:
   (a) Use the algorithm $\mathsf{TFHE.Eval}$ to homomorphically compute an encryption $[\![\mathsf{p}^*(x)]\!]$ by rational polynomial interpolation from encrypted evaluations $\left(j, \left[\!\!\left[\frac{e_{2,j}+\cdots+e_{n,j}}{e_{1,j}}\right]\!\!\right]\right)$ for $j \in [2T+3]$.
   (b) Homomorphically compute the encrypted predicate $[\![b]\!]$ where $b = 1$ if $\mathsf{p}^*(z) = \frac{e_2'+\cdots+e_n'}{e_1'}$ and 0 otherwise.
5. P_1 sends $[\![b]\!]$ to all parties who respond with $[\![b : \mathsf{sk}_i]\!] := \mathsf{TFHE.PartialDec}(\mathsf{sk}_i, [\![b]\!])$. P_1 broadcasts $b := \mathsf{TFHE.Combine}(\mathsf{pk}, \{[\![b : \mathsf{sk}_i]\!]\}_{i \in [n]})$ and all parties output $\mathsf{similar}$ if $b = 1$ and $\mathsf{different}$ otherwise.

Figure 5: Multi-party private intersection cardinality testing protocol $\Pi_{\mathsf{TFHE\text{-}CTest\text{-}int}}$ for $\mathcal{F}_{\mathsf{CTest\text{-}int}}$.

the degree of each term $\mathsf{p}_{i\setminus I}(x) \cdot (x - r_i)$ is at most $T+1$ and therefore the rational polynomial interpolation requires a total of $(2T+3)$ evaluation points. Therefore $\mathsf{p}^*(x) = \mathsf{p}'(x)$ and $\mathsf{p}^*(z) = \mathsf{p}'(z) = \frac{e_2'+\cdots+e_n'}{e_1'}$. Thus $b = 1$ as required.

Now consider the case where the protocol should output $\mathsf{different}$, namely when $|I| < m - T$. Observe that $\gcd(\mathsf{p}_{1\setminus I}, \cdots, \mathsf{p}_{n\setminus I}) = 1$ by construction and therefore Lemma 3.2 states that

$$\gcd\left(\mathsf{p}_{2\setminus I}'(x) + \cdots + \mathsf{p}_{n\setminus I}'(x), \mathsf{p}_{1\setminus I}'(x)\right) = 1$$

except with negligible probability, where $\mathsf{p}_{i\setminus I}'(x) := \mathsf{p}_{i\setminus I}(x) \cdot (x - r_i)$. Assuming $\gcd\left(\mathsf{p}_{2\setminus I}'(x) + \cdots + \mathsf{p}_{n\setminus I}'(x), \mathsf{p}_{1\setminus I}'(x)\right)$ 1, it then follows that the degree of the rational polynomial $\mathsf{p}'(x)$ is the degree of $\mathsf{p}_{2\setminus I}'(x) + \cdots + \mathsf{p}_{n\setminus I}'(x)$ plus the degree of $\mathsf{p}_{1\setminus I}'(x)$. The former must have a leading term with degree $(m - |I| + 1) > (T+1)$. Similarly, the latter also has degree $(m - |I| + 1) > T + 1$. Hence the degree of $\mathsf{p}'(x)$ is at least $2T+4$. The probability of $b=1$ is $\Pr_z[\mathsf{p}'(z)=\mathsf{p}^*(z)]$ where $\mathsf{p}^*(x)$ is the polynomial interpolated by P_1 using $(2T+3)$ evaluations. However, since the degree of $\mathsf{p}'(x)$ is at least $2T+4$, $\Pr_z[\mathsf{p}'(z)=\mathsf{p}^*(z)] \leq \mathsf{negl}(\lambda)$.

Communication Cost. Each party sends $(2T+4)$ TFHE encryptions and one partial decryption to P_1 where each plaintext is a field element. P_1 sends one ciphertext to every other party. The size of each encryption and each partial decryption is $\mathsf{poly}(\lambda)$. Thus, the overall communication complexity is $O(n \cdot T \cdot \mathsf{poly}(\lambda))$ in a star network and the protocols runs in $O(1)$ rounds.

Security. Consider an environment $\mathcal{Z}$ who corrupts a set $\mathcal{S}^*$ of n^* parties where $n^* < n$. The simulator Sim has output $w \in \{\mathsf{similar}, \mathsf{different}\}$ from the ideal functionality. Sim sets a bit $b^*=1$ if $w = \mathsf{similar}$ and $b^*=0$ otherwise. Also, for each corrupt party P_i, Sim has as input the tuple (S_i, r_i) indicating the party's input and randomness for the protocol. The strategy of the simulator Sim for our protocol is described below.

1. Sim runs the distributed key generation algorithm $\mathsf{TFHE.DistSetup}(1^\lambda, i)$ of the TFHE scheme honestly on behalf of each honest party P_i as in the real world. Note that Sim also knows $(\{\mathsf{sk}_i\}_{i \in S^*})$ as it knows the randomness for the corrupt parties.

2. In Steps 2-4 of the protocol, Sim plays the role of the honest parties exactly as in the real world except that on behalf of every honest party P_i, whenever P_i has to send any ciphertext, compute $[\![0]\!] = \mathsf{TFHE.Enc}(0)$ using fresh randomness.

3. In Step 5, on behalf of each honest party P_i, instead of sending the value $[\![b : \mathsf{sk}_i]\!]$ by running the honest $\mathsf{TFHE.PartialDec}$ algorithm as in the real world, Sim computes the partial decryptions by running the simulator $\mathsf{TFHE.Sim}$ as follows: $\{[\![b : \mathsf{Sim}_i]\!]\}_{i \in [n] \setminus \mathcal{S}^*} \leftarrow \mathsf{TFHE.Sim}(\mathsf{C}, b^*, [\![b]\!], \{\mathsf{sk}_i\}_{i \in \mathcal{S}^*})$ where the circuit C denotes the whole computation done by P_1 in the real world to evaluate bit b. On behalf of the honest party P_i the simulator sends $[\![b : \mathsf{Sim}_i]\!]$. This corresponds to the ideal world.

We now show that the above simulation strategy is successful against all environments $\mathcal{Z}$ that corrupt parties in a semi-honest manner. We will show this via a series of computationally indistinguishable hybrids where the first hybrid Hybrid_0 corresponds to the real world and the last hybrid Hybrid_2 corresponds to the ideal world.

- Hybrid_0 **- Real World:** In this hybrid, consider a simulator SimHyb that plays the role of the honest parties as in the real world.
- Hybrid_1 **- Simulate Partial Decryptions:** - In this hybrid, in Step 5, SimHyb simulates the partial decryptions generated by the honest parties as done in the ideal world. That is, the simulator calls $\{[\![b : \mathsf{Sim}_i]\!]\}_{i \in [n] \setminus \mathcal{S}} \leftarrow \mathsf{TFHE.Sim}(\mathsf{C}, b^*, [\![b]\!], \{\mathsf{sk}_i\}_{i \in \mathcal{S}})$. On behalf of the honest party P_i the simulator sends $[\![b : \mathsf{Sim}_i]\!]$ instead of $[\![b : \mathsf{sk}_i]\!]$.
- Hybrid_2 **- Switch Encryptions:** In this hybrid, SimHyb now computes every ciphertext generated on behalf of any honest party as encryptions of 0 as done by Sim in the ideal world. This hybrid corresponds to the ideal world.

We now show that every pair of consecutive hybrids is computationally indistinguishable.

Lemma 5.2. *Assuming the simulation security of the threshold fully homomorphic encryption scheme,* Hybrid_0 *is computationally indistinguishable from* Hybrid_1*.*

Proof. The only difference between the two hybrids is that in Hybrid_0, the simulator SimHyb generates the partial decryptions of the TFHE scheme on behalf of the honest parties as in the real world while in Hybrid_1, they are simulated by running the simulator $\mathsf{TFHE.Sim}$. We now show that if there exists an environment $\mathcal{Z}$ that can distinguish between these two hybrids with some non-negligible probability ϵ, we will come up with a reduction $\mathcal{A}$ that can break the simulation security of the TFHE scheme.

$\mathcal{A}$ interacts with a challenger $\mathcal{C}$ in the simulation security game for TFHE and with the environment $\mathcal{Z}$ in the game between Hybrid_0 and Hybrid_1. $\mathcal{A}$ corrupts the same set of parties as done by $\mathcal{Z}$ in its game with $\mathcal{C}$. Further, $\mathcal{A}$ forwards the public key-secret key pairs $(\mathsf{pk}_i, \mathsf{sk}_i)$ for the corrupt parties it receives from $\mathcal{Z}$ to the challenger and the public keys pk_i for the honest parties from $\mathcal{C}$ to $\mathcal{Z}$. $\mathcal{A}$ also forwards to $\mathcal{C}$ the set of messages to be encrypted along with the randomness for the ones encrypted by the adversary, received from $\mathcal{Z}$. Similarly, it forwards the ciphertexts received from $\mathcal{C}$ to $\mathcal{Z}$. Finally, $\mathcal{A}$ sends the circuit C that denotes the whole computation done by P_1 in the real world to evaluate bit b and receives a set of partial decryptions on behalf of each honest party which it forwards to $\mathcal{A}$. It continues interacting with $\mathcal{Z}$ as in Hybrid_0 in the rest of its interaction. It is easy to see that if $\mathcal{C}$ sent honestly generated partial decryptions, the interaction between $\mathcal{A}$ and $\mathcal{Z}$ exactly corresponds to Hybrid_0 and if the partial decryptions were simulated, the interaction between $\mathcal{A}$ and $\mathcal{Z}$ exactly corresponds to Hybrid_1. Thus, if $\mathcal{Z}$ can distinguish between the two hybrids with non-negligible probability ϵ, $\mathcal{A}$ can break the simulation security of the TFHE scheme with the same probability ϵ which is a contradiction. □

Lemma 5.3. *Assuming the semantic security of the threshold fully homomorphic encryption scheme,* Hybrid_1 *is computationally indistinguishable from* Hybrid_2*.*

Proof. The only difference between the two hybrids is that in Hybrid_1, the simulator SimHyb generates the encryptions of the TFHE scheme on behalf of the honest parties as in the real world while in Hybrid_2, they are generated as encryptions of 0. We now show that if there exists an adversarial environment $\mathcal{Z}$ that can distinguish between these two hybrids with some non-negligible probability ϵ, we will come up with a reduction $\mathcal{A}$ that can break the semantic security of the TFHE scheme.

$\mathcal{A}$ interacts with a challenger $\mathcal{C}$ in the semantic security game for TFHE and with the environment $\mathcal{Z}$ in the game between Hybrid_1 and Hybrid_2. $\mathcal{A}$ corrupts the same set of parties as done by $\mathcal{Z}$ in its game with $\mathcal{C}$. Further, $\mathcal{A}$ forwards the public key-secret key pairs $(\mathsf{pk}_i, \mathsf{sk}_i)$ for the corrupt parties it receives from $\mathcal{Z}$ to the challenger and the public keys pk_i for the honest parties from $\mathcal{C}$ to $\mathcal{Z}$. $\mathcal{A}$ also forwards the pair of 0 and the set of honestly generated plaintexts to be encrypted, to the challenger and receives back a ciphertext for each of them which it uses in its interaction with $\mathcal{Z}$. It continues interacting with $\mathcal{Z}$ as in Hybrid_1 in the rest of its interaction. It is easy to see that if $\mathcal{C}$ sent honestly generated ciphertexts, the interaction between $\mathcal{A}$ and $\mathcal{Z}$ exactly corresponds to Hybrid_1 and if the ciphertexts were generated as encryptions of 0, the interaction between $\mathcal{A}$ and $\mathcal{Z}$ exactly corresponds to Hybrid_2. Thus, if $\mathcal{Z}$ can distinguish between the two hybrids with non-negligible probability ϵ, $\mathcal{A}$ can break the semantic security of the TFHE scheme with the same probability ϵ which is a contradiction. □

□

5.2 Protocol for Functionality $\mathcal{F}_{\mathsf{CTest\text{-}diff}}$

This protocol will compute the cardinality predicate b where $b = 1$ if and only if $|(\bigcup_{i=1}^{n} S_i) \setminus I| \leq T$. The core idea behind the protocol is that P_1 (the star of the network) and P_i first run a protocol to compute an encryption (via TFHE) of their set differences $D_{1,i} = S_1 \setminus S_i$ and $D_{i,1} = S_i \setminus S_1$ with $O(T)$ communication complexity if $|S_1 \setminus S_i| \leq T$. Before we describe how this is achieved, notice that at this point, the protocol enables P_1 to reconstruct an encryption of $(\bigcup_{i=1}^{n} S_i) \setminus I = \bigcup_{i \in [n] \setminus \{1\}} (D^*_{1,i} \cup D^*_{i,1})$ and a predicate b where $b = 1$ if and only if $|(\bigcup_{i=1}^{n} S_i) \setminus I| \leq T$. P_1 can then send this encryption to all parties to run threshold decryption.

We now describe in more detail how the encryption of $D_{1,i}$ and $D_{i,1}$ are computed. The idea follows from the two-party protocol of Ghosh and Simkin [GS19a]. Each party P_i encodes their set S_i as $\mathsf{p}_i(x) := \Pi_{a \in S_i}(x - a) \in \mathbb{F}[x]$. P_i then computes $e_{i,j} := \mathsf{p}_i(j)$ for $j \in [2T+1]$ and $e'_i := \mathsf{p}_i(z)$ on a special random point $z \in \mathbb{F}$ (picked uniformly at random by P_1). Party P_i encrypts these values as $[\![e_{i,j}]\!], [\![e'_i]\!]$ and sends them to P_1. Party P_1 considers the rational polynomial

$$\widetilde{\mathsf{p}}_i(x) = \frac{\mathsf{p}_i(x)}{\mathsf{p}_1(x)} = \frac{\mathsf{p}_{i \setminus 1}(x)}{\mathsf{p}_{1 \setminus i}(x)}$$

and homomorphically computes $2T + 1$ encrypted evaluations $\left(j, \left[\!\!\left[\frac{e_{i,j}}{e_{1,j}}\right]\!\!\right]\right)$ for $j = [2T + 1]$. Using these encrypted evaluations, P_1 homomorphically computes an encrypted rational polynomial $[\![\widetilde{\mathsf{p}}^*_i(x)]\!]$ using rational polynomial interpolation. P_1 then homomorphically reconstructs the roots of $\mathsf{p}_{i \setminus 1}(x)$ and $\mathsf{p}_{1 \setminus i}(x)$ from $\widetilde{\mathsf{p}}^*_i$ to obtain $\left[\!\!\left[D^*_{i,1}\right]\!\!\right], \left[\!\!\left[D^*_{1,i}\right]\!\!\right]$. Note that $\widetilde{\mathsf{p}}^*_i(x) = \widetilde{\mathsf{p}}_i(x)$ if $\widetilde{\mathsf{p}}_i(x)$ has degree at most $2T$, in which case $D^*_{i,1} = D_{i,1}$ and $D^*_{1,i} = D_{1,i}$.

In the final protocol, P_1 homomorphically computes encrypted predicates b_i where $b_i = 1$ iff $\widetilde{\mathsf{p}}^*_i(z) = \frac{e'_i}{e'_1}$ for each $i \in [n] \setminus \{1\}$ and encrypted predicate b' where $b' = 1$ iff $\left|\bigcup_{i \in [n] \setminus \{1\}} (D^*_{1,i} \cup D^*_{i,1})\right| \leq T$. The output predicate b is homomorpically computed as $[\![b]\!] = \left[\!\!\left[b' \cdot \prod_{i \in [n] \setminus \{1\}} b_i\right]\!\!\right]$ and jointly decrypted by all the parties. The protocol is formally described in Figure 6.

Theorem 5.4. *Assuming threshold FHE with distributed setup, protocol* $\Pi_{\mathsf{TFHE\text{-}CTest\text{-}diff}}$ *(Figure 6) securely realizes* $\mathcal{F}_{\mathsf{CTest\text{-}diff}}$ *(Figure 4).*

Proof. **Correctness.** We first prove the protocol is correct. By the correctness of the TFHE scheme, we only need to show that the computed predicate $b = 1$ if and only if $|(\bigcup_{i=1}^{n} S_i) \setminus I| \leq T$. First consider the case where the protocol should output similar. Since

$$\widetilde{\mathsf{p}}_i(x) = \frac{\mathsf{p}_i(x)}{\mathsf{p}_1(x)} = \frac{\mathsf{p}_{i \setminus 1}(x)}{\mathsf{p}_{1 \setminus i}(x)},$$

both the numerator and denominator have degree at most T and therefore the rational polynomial interpolation requires at most $(2T+1)$ evaluation points. Hence $\widetilde{\mathsf{p}}^*_i(x) = \widetilde{\mathsf{p}}_i(x)$ and $\widetilde{\mathsf{p}}^*_i(z) = \widetilde{\mathsf{p}}_i(z) = \frac{e'_i}{e'_1}$, thus $b_i = 1$. Since the roots of $\mathsf{p}_{i \setminus 1}$ is simply the set difference $D_{i,1} = S_i \setminus S_1$, we have $D^*_{i,1} = D_{i,1} = S_i \setminus S_1$. Similarly $D^*_{1,i} = S_1 \setminus S_i$. Since $\left|\bigcup_{i \in [n] \setminus \{1\}} (D^*_{1,i} \cup D^*_{i,1})\right| = |(\bigcup_{i=1}^{n} S_i) \setminus I| \leq T$, we have $b' = 1$. Hence the protocol will output $b = 1$.

Now consider the case where the protocol should output different, namely $|(\bigcup_{i=1}^{n} S_i) \setminus I| > T$. There are two possible cases. In the first case, $|S_i \setminus S_1| > T$ for some i. Then $\widetilde{\mathsf{p}}_i$ has degree at least

Parameters: Parties $P_1, \ldots, P_n$. Each party has a set of m elements. Threshold $T \in \mathbb{N}$. $\mathbb{F}$ is a finite field where $|\mathbb{F}| = \Omega(2^\lambda)$.

Inputs: Party P_i has an input set $S_i = \{a_1^i, \ldots, a_m^i\}$ where $a_j^i \in \mathbb{F}$ for all $j \in [m]$.

Output: Each party P_i receives $\mathsf{similar}$ if $|(\bigcup_{i=1}^n S_i) \setminus I| \leq T$ and $\mathsf{different}$ otherwise where $I = \bigcap_{i=1}^n S_i$.

Protocol:

1. Each party P_i generates $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{TFHE.DistSetup}(1^\lambda, i)$ and sends pk_i to P_1. Then P_1 sends $\mathsf{pk} = (\mathsf{pk}_1 \| \ldots \| \mathsf{pk}_n)$ to all the other parties.
2. P_1 picks a random value $z \in \mathbb{F}$ and sends it to all parties.
3. Each party P_i does the following:
   (a) Define the polynomial $\mathsf{p}_i(x) := \prod_{a \in S_i}(x - a)$.
   (b) Compute $e_{i,j} := \mathsf{p}_i(j)$ for $j \in [2T+1]$ and $e'_i := \mathsf{p}_i(z)$.
   (c) Send encrypted evaluations $[\![e_{i,j}]\!] := \mathsf{TFHE.Enc}(\mathsf{pk}, e_{i,j})$ for $j \in [2T+1]$ and $[\![e'_i]\!] := \mathsf{TFHE.Enc}(\mathsf{pk}, e'_i)$ to P_1.
4. P_1 does the following:
   (a) For each $i \in [n] \setminus \{1\}$, use the algorithm $\mathsf{TFHE.Eval}$ to homomorphically compute an encryption $[\![\widetilde{\mathsf{p}}_i^*(x)]\!]$ by rational polynomial interpolation from $2T+1$ encrypted evaluations $\left(j, \left[\!\left[\frac{e_{i,j}}{e_{1,j}}\right]\!\right]\right)$ for $j \in [2T+1]$.
   (b) For each $i \in [n] \setminus \{1\}$, homomorphically compute the encrypted predicate $[\![b_i]\!]$ where $b_i = 1$ if $\widetilde{\mathsf{p}}_i^*(z) = \frac{e'_i}{e'_1}$ and 0 otherwise.
   (c) For each $i \in [n] \setminus \{1\}$, homomorphically compute the encrypted roots $\left[\!\left[D_{i,1}^*\right]\!\right]$, $\left[\!\left[D_{1,i}^*\right]\!\right]$ of the numerator and denominator of $\widetilde{\mathsf{p}}_i^*(x)$, respectively.
   (d) Homomorphically compute the encrypted predicate $[\![b']\!]$ where $b' = 1$ if $\left|\bigcup_{i \in [n] \setminus \{1\}}(D_{1,i}^* \cup D_{i,1}^*)\right| \leq T$ and 0 otherwise.
5. P_1 sends $[\![b]\!] = \left[\!\left[b' \cdot \prod_{i \in [n] \setminus \{1\}} b_i\right]\!\right]$ to all parties who respond with $[\![b : \mathsf{sk}_i]\!] := \mathsf{TFHE.PartialDec}(\mathsf{sk}_i, [\![b]\!])$. P_1 broadcasts $b := \mathsf{TFHE.Combine}(\mathsf{pk}, \{[\![b : \mathsf{sk}_i]\!]\}_{i \in [n]})$ and all parties output $\mathsf{similar}$ if $b = 1$ and $\mathsf{different}$ otherwise.

Figure 6: Multi-party private intersection cardinality testing protocol $\Pi_{\mathsf{TFHE\text{-}CTest\text{-}diff}}$ for $\mathcal{F}_{\mathsf{CTest\text{-}diff}}$

$2T+2$ but $\widetilde{\mathsf{p}}_i^*$ is interpolated from $2T+1$ evaluation points, hence $b'_i = 0$ with all but negligible probability. In the second case, $|S_i \setminus S_1| \leq T$ for all $i \in [n] \setminus \{1\}$. Then $D_{i,1}^* = D_{i,1} = S_i \setminus S_1$, $D_{1,i}^* = S_1 \setminus S_i$, and $b_i = 1$ for all i. Since $|(\bigcup_{i=1}^n S_i) \setminus I| > T$, $b' = 0$. In both cases, we have $b = b' \cdot \prod_{i \in [n] \setminus \{1\}} b_i = 0$ with all but negligible probability.

Communication Cost. Each party sends $(2T+2)$ TFHE encryptions and one partial decryption to P_1 where each plaintext is a field element. P_1 sends one ciphertext to every other party. The size of each encryption and each partial decryption is $\mathsf{poly}(\lambda)$. Thus, the overall communication complexity is $O(n \cdot T \cdot \mathsf{poly}(\lambda))$ in a star network and the protocols runs in $O(1)$ rounds.

Security. The proof of security is identical to the proof of Theorem 5.1. We describe it below for the sake of completeness.
Consider an environment $\mathcal{Z}$ who corrupts a set $\mathcal{S}^*$ of n^* parties where $n^* < n$. The simulator Sim has input $w \in \{\mathsf{similar}, \mathsf{different}\}$ from the ideal functionality. Sim sets a bit $b^* = 1$ if $w = \mathsf{similar}$ and $b^* = 0$ otherwise. Also, for each corrupt party P_i, Sim has as input the tuple (S_i, r_i) indicating the party's input and randomness for the protocol. The strategy of the simulator Sim for our protocol is described below.

1. Sim runs the distributed key generation algorithm $\mathsf{TFHE.DistSetup}(1^\lambda, i)$ of the TFHE scheme honestly on behalf of each honest party P_i as in the real world. Note that Sim also knows $(\{\mathsf{sk}_i\}_{i \in S^*})$ as it knows the randomness for the corrupt parties.

2. In Steps 2-4 of the protocol, Sim plays the role of the honest parties exactly as in the real world except that on behalf of every honest party P_i, whenever P_i has to send any ciphertext, compute $[\![0]\!] = \mathsf{TFHE.Enc}(0)$ using fresh randomness.

3. In Step 5, on behalf of each honest party P_i, instead of sending the value $[\![b : \mathsf{sk}_i]\!]$ by running the honest $\mathsf{TFHE.PartialDec}$ algorithm as in the real world, Sim computes the partial decryptions by running the simulator $\mathsf{TFHE.Sim}$ as follows: $\{[\![b : \mathsf{Sim}_i]\!]\}_{i \in [n] \setminus \mathcal{S}^*} \leftarrow \mathsf{TFHE.Sim}(\mathsf{C}, b^*, [\![b]\!], \{\mathsf{sk}_i\}_{i \in \mathcal{S}^*})$ where the circuit C denotes the whole computation done by P_1 in the real world to evaluate bit b. On behalf of the honest party P_i the simulator sends $[\![b : \mathsf{Sim}_i]\!]$. This corresponds to the ideal world.

We now show that the above simulation strategy is successful against all environments $\mathcal{Z}$ that corrupt parties in a semi-honest manner. We will show this via a series of computationally indistinguishable hybrids where the first hybrid Hybrid_0 corresponds to the real world and the last hybrid Hybrid_2 corresponds to the ideal world.

- Hybrid_0 **- Real World:** In this hybrid, consider a simulator SimHyb that plays the role of the honest parties as in the real world.
- Hybrid_1 **- Simulate Partial Decryptions:** - In this hybrid, in Step 5, SimHyb simulates the partial decryptions generated by the honest parties as done in the ideal world. That is, the simulator calls $\{[\![b : \mathsf{Sim}_i]\!]\}_{i \in [n] \setminus \mathcal{S}} \leftarrow \mathsf{TFHE.Sim}(\mathsf{C}, b^*, [\![b]\!], \{\mathsf{sk}_i\}_{i \in \mathcal{S}})$. On behalf of the honest party P_i the simulator sends $[\![b : \mathsf{Sim}_i]\!]$ instead of $[\![b : \mathsf{sk}_i]\!]$.
- Hybrid_2 **- Switch Encryptions:** In this hybrid, SimHyb now computes every ciphertext generated on behalf of any honest party as encryptions of 0 as done by Sim in the ideal world. This hybrid corresponds to the ideal world.

We now show that every pair of consecutive hybrids is computationally indistinguishable.

Lemma 5.5. *Assuming the simulation security of the threshold fully homomorphic encryption scheme,* Hybrid_0 *is computationally indistinguishable from* Hybrid_1*.*

Proof. This is identical to the proof of Lemma 5.2. □

Lemma 5.6. *Assuming the semantic security of the threshold fully homomorphic encryption scheme,* Hybrid_1 *is computationally indistinguishable from* Hybrid_2.

Proof. This is identical to the proof of Lemma 5.3. □

□

6 TAHE-Based Protocol for $\mathcal{F}_{\mathsf{CTest\text{-}diff}}$

In this section, we present a multi-party protocol for private intersection cardinality testing for functionality $\mathcal{F}_{\mathsf{CTest\text{-}diff}}$ based on threshold additive homomorphic encryption with distributed setup. That is, the parties learn whether their sets satisfy $|(\bigcup_{i=1}^n S_i) \setminus I| \leq T$. Our protocol works in the star network communication model where P_1 is the central party.

In our construction we will make use of a sub-protocol that realizes multi-party homomorphic matrix multiplication, which enables the parties to homomorphically multiply two encrypted matrices. The details will be presented in Section 6.1. In addition, we need a secure multi-party computation (MPC) protocol that computes the minimal polynomial of a linearly recurrent sequence, which we discuss in Section 6.2. Finally, we present our protocol in Section 6.3.

6.1 Multi-Party Homomorphic Matrix Multiplication

In the sub-protocol for multi-party homomorphic matrix multiplication, two encrypted matrices $[\![A]\!]$ and $[\![B]\!]$ are publicly known to all the parties and the secret key of the TAHE scheme is shared among all the parties. At the end of the sub-protocol, all the parties learn as output a fresh encryption of the multiplied matrix $[\![A \cdot B]\!]$, and nothing else is leaked.

Our protocol is a generalization of the two-party homomorphic matrix multiplication protocol [KMWF07]. We briefly describe our protocol inspired by [Bea91]. First each party P_i generates two uniformly random matrices R_i^A and R_i^B sends encryption $[\![R_i^A]\!], [\![R_i^B]\!]$ to party P_1. Then P_1 homomorphically computes three encrypted matrices $[\![R^A]\!] = [\![\sum_{i=1}^n R_i^A]\!]$, $[\![M^A]\!] = [\![A + R^A]\!]$, and $[\![M^B]\!] = [\![B + \sum_{i=1}^n R_i^B]\!]$ and sends back to all the parties. The parties jointly decrypt $[\![M^A]\!]$ and $[\![M^B]\!]$ to P_1. Now P_1 has two matrices (M^A, M^B) in the clear, and each party P_i holds $[\![R^A]\!], [\![M^A]\!], [\![M^B]\!]$. Note that

$$\begin{aligned} AB = &\left(M^A - \sum_{i=1}^n R_i^A\right) \cdot \left(M^B - \sum_{i=1}^n R_i^B\right) \\ = &M^A M^B - \sum_{i=1}^n M^A R_i^B - \sum_{i=1}^n R_i^A M^B + \sum_{i=1}^n R^A R_i^B. \end{aligned}$$

To compute an encryption of AB, each party P_i homomorphically computes $[\![R^A R_i^B - M^A R_i^B - R_i^A M^B]\!]$ and sends to P_1. Finally P_1 can homomorphically compute $[\![AB]\!]$ from a linear combination of existing ciphertexts. Our multi-party protocol Π_{MMult} is presented in Figure 7.

Correctness. Correctness of the protocol mainly follows from the correctness of TAHE. We show that the final ciphertext computed by P_1 in Step 4c is an encryption of $A \cdot B$. Note that

Input: Each party P_i inputs $(\mathsf{pk}, [\![A]\!], [\![B]\!], \mathsf{sk}_i)$, where pk is the public key of the TAHE scheme, $[\![A]\!]$ and $[\![B]\!]$ are item-wise encryption of matrix $A \in \mathbb{F}^{k\times s}$ and matrix $B \in \mathbb{F}^{s\times \ell}$, sk_i is the secret key share of party P_i for TAHE.

Output: Each party receives a fresh encryption of $A \cdot B$, namely $[\![A \cdot B]\!]$.

Protocol:

1. Each party P_i samples two random matrices $R_i^A \xleftarrow{\$} \mathbb{F}^{k\times s}, R_i^B \xleftarrow{\$} \mathbb{F}^{s\times \ell}$, and sends encryption of them $([\![R_i^A]\!], [\![R_i^B]\!])$ to P_1.
2. Let $R^A = \sum_{i=1}^n R_i^A$, $M^A = A + R^A$, $R^B = \sum_{i=1}^n R_i^B$, $M^B = B + R^B$. P_1 homomorphically compute $[\![R^A]\!], [\![M^A]\!], [\![M^B]\!]$, and sends them to all the other parties.
3. Each party P_i does the following:
   (a) Homomorphically compute $\mathsf{ct}_i = [\![R^A R_i^B - M^A R_i^B - R_i^A M^B]\!]$ and send to P_1.
   (b) Compute $[\![M^A : \mathsf{sk}_i]\!] \leftarrow \mathsf{TAHE.PartialDec}(\mathsf{sk}_i, [\![M^A]\!])$ and $[\![M^B : \mathsf{sk}_i]\!] \leftarrow \mathsf{TAHE.PartialDec}(\mathsf{sk}_i, [\![M^B]\!])$, and send $([\![M^A : \mathsf{sk}_i]\!], [\![M^B : \mathsf{sk}_i]\!])$ to P_1.
4. P_1 does the following:
   (a) Compute $M^A \leftarrow \mathsf{TAHE.Combine}(\mathsf{pk}, \{[\![M^A : \mathsf{sk}_i]\!]\}_{i\in[n]})$ and $M^B \leftarrow \mathsf{TAHE.Combine}(\mathsf{pk}, \{[\![M^B : \mathsf{sk}_i]\!]\}_{i\in[n]})$.
   (b) Compute an encryption of the product of the two matrices $[\![M^A \cdot M^B]\!]$.
   (c) Homomorphically compute

$$[\![A \cdot B]\!] = \left[\!\!\left[M^A \cdot M^B + \sum_{i=1}^n \left(R^A \cdot R_i^B - M^A \cdot R_i^B - R_i^A \cdot M^B\right) \right]\!\!\right]$$

   and send to all the other parties.

Figure 7: Multi-party homomorphic matrix multiplication protocol Π_{MMult}.

$M^A = A + R^A = A + \sum_{i=1}^n R_i^A$ and $M^B = B + R^B = B + \sum_{i=1}^n R_i^B$. We have

$$\begin{aligned} AB &= \left(M^A - \sum_{i=1}^n R_i^A\right) \cdot \left(M^B - \sum_{i=1}^n R_i^B\right) \\ &= M^A M^B - \sum_{i=1}^n M^A R_i^B - \sum_{i=1}^n R_i^A M^B + \sum_{i=1}^n R^A R_i^B \\ &= M^A M^B + \sum_{i=1}^n \left(R^A R_i^B - M^A R_i^B - R_i^A M^B\right). \end{aligned}$$

Communication Cost. The protocol requires $O(1)$ rounds of communication in a star network and the total communication complexity is $O((ks + s\ell) \cdot n \cdot \mathsf{poly}(\lambda))$.

Security. Intuitively speaking, the protocol does not leak any information apart from the output and the security is relying on the security of the TAHE scheme and the fact that any subset of parties cannot recover the secret key of the TAHE scheme. However, there is a subtle issue in formally proving security of the protocol. In particular, if we formalize the ideal functionality for the multi-party homomorphic matrix multiplication, every party holds two encrypted matrices and the secret key is shared among all the parties. Since the distinguisher can choose the inputs for all the parties, it knows all the secret key shares of the TAHE scheme, hence we can not rely on the security of TAHE. The fundamental problem is that when formalizing the ideal functionality, we have to involve cryptographic primitives in the inputs, which makes it hard to argue security based on these primitives. We get around this problem by including all the sub-protocols in our final multi-party private intersection cardinality testing protocol in Section 6.3 and provide a single security proof. This approach complicates the security proof, but it avoids including cryptographic primitives in the inputs.

6.2 Computing Minimal Polynomial

In Section 6.3, we will see that intersection cardinality testing can be reduced to determining whether the determinant of a matrix is 0 or not. The latter problem can be reduced to computing the minimal polynomial of that matrix, which can be further reduced to computing the minimal polynomial of a linearly recurrent sequence $\mathbf{a} = \{\mathbf{u}^\top A^i \mathbf{v}\}_{i\in\mathbb{N}}$ for the matrix A and random vectors $\mathbf{u}$ and $\mathbf{v}$. We take the following theorems from [KMWF07] (Corollary 2 and Lemma 3) and refer the reader to [VZGG13,KMWF07] for formal definitions of minimal polynomials and linearly recurrent sequences.

Imported Theorem 2. *Let $A \in \mathbb{F}^{k\times k}$ and let $m_A(\cdot)$ be the minimal polynomial of matrix A. Then $\det(A) = 0$ if and only if $m_A(0) = 0$.*

Imported Theorem 3. *Let $A \in \mathbb{F}^{k\times k}$ and let $m_A(\cdot)$ be the minimal polynomial of matrix A. For $\mathbf{u}, \mathbf{v} \in \mathbb{F}^k$ chosen uniformly at random, let $m_{\mathbf{a}}(\cdot)$ be the minimal polynomial of the linearly recurrent sequence $\mathbf{a} = \{\mathbf{u}^\top A^i \mathbf{v}\}_{i\in\mathbb{N}}$. Then $m_A = m_{\mathbf{a}}$ with probability at least $1 - 2k/|\mathbb{F}|$.*

To compute the minimal polynomial of the sequence $\mathbf{a}$, the first $2k$ entries of the sequence will suffice. Specifically, computing the minimal polynomial can be reduced to computing the greatest common division (GCD) of two polynomial of degree $2k$ [KMWF07, Appendix A.2]. Using the fast Extended Euclidean algorithm [VZGG13, Chapter 11], the computation can be carried out using an arithmetic circuit of size $O(k \log k)$.

In our multi-party intersection cardinality testing protocol, the first $2k$ entries of $\mathbf{a}$ will be additively secret shared among all the parties, and the parties will run a secure multi-party computation (MPC) protocol to jointly compute the minimal polynomial of $\mathbf{a}$ and $m_{\mathbf{a}}(0)$, which equals 0 if and only if the matrix A is singular. The ideal functionality $\mathcal{F}_{\mathsf{MinPoly}}$ for the multi-party minimal polynomial computation is defined in Figure 8. We will need an MPC protocol that realizes $\mathcal{F}_{\mathsf{MinPoly}}$ with communication complexity at most $\widetilde{O}(k^2 \cdot n \cdot \mathsf{poly}(\lambda))$. Any such protocol suffices, and we denote by Π_{MinPoly} the MPC protocol realizing $\mathcal{F}_{\mathsf{MinPoly}}$.

Here we describe two such protocols with communication complexity $\widetilde{O}(k \cdot n \cdot \mathsf{poly}(\lambda))$ based on TAHE. In the first protocol, after the TAHE setup, each party P_i sends $\left[\!\left[r_i^0 \right]\!\right], \ldots, \left[\!\left[r_i^{2k-1} \right]\!\right]$ to P_1 and P_1 homomorphically computes $\left[\!\left[a^j \right]\!\right]$ for all j. Afterwards P_1 can homomorphically evaluate a circuit C that computes a predicate $b \overset{?}{=} (m_{\mathbf{a}}(0) = 0)$, following the ideas from [FH96, CDN01]. Finally the parties jointly decrypt the encrypted output. Since the size and depth of C are both

Parameters: Parties $P_1, \dots, P_n$.

Inputs: Each party P_i inputs $2k$ field elements $r_i^0, r_i^1, \dots, r_i^{2k-1} \in \mathbb{F}$.

Output: Let $a^j = \sum_{i=1}^{n} r_i^j$ for $j = 0, 1, \dots, 2k-1$. Compute the minimal polynomial $m_{\mathbf{a}}$ of the sequence $\{a^j\}_{j=0}^{2k-1}$. Each party receives 0 if $m_{\mathbf{a}}(0) = 0$ and 1 otherwise.

Figure 8: Ideal functionality $\mathcal{F}_{\mathsf{MinPoly}}$ for multi-party minimal polynomial computation.

$O(k \log k)$, the total communication complexity of this protocol is $O(k \log k \cdot n \cdot \mathsf{poly}(\lambda))$ and the round complexity is $O(k \log k)$.

As a second protocol, the parties jointly compute another C' that takes $\{a^j\}_{j=0}^{2k-1}$ and a random PRF key r as input and outputs a Yao's garbled circuit [Yao86] that computes C. This approach is inspired by the work of Damgård et al. [DIK+08]. Since both $\{a^j\}$ and r are additively shared among all the parties, this MPC can be done similarly as in the previous protocol, namely P_1 first obtains $[\![a^j]\!]$ and $[\![r]\!]$ and then homomorphically evaluates C'. Since the size C' is $\widetilde{O}(k \cdot \mathsf{poly}(\lambda))$ and the depth of C' is constant assuming PRG is a circuit in NC^1 [**?**], the total communication complexity of this protocol is $\widetilde{O}(k \cdot n \cdot \mathsf{poly}(\lambda))$ and the round complexity is $O(1)$.

6.3 Our Protocol

In this section we present our multi-party private intersection cardinality testing protocol. That is, the parties learn whether their sets satisfy $|(\bigcup_{i=1}^{n} S_i) \setminus I| \le T$.

At a high level, our protocol first encodes each party P_i's set as a polynomial $\mathsf{p}_i(x) = \sum_{j=1}^{m} x^{a_j^i}$, and let $\mathsf{p}(x) := (n-1)\mathsf{p}_1(x) - \sum_{i=2}^{n} \mathsf{p}_i(x)$. Notice that a term x^a is cancelled out in the polynomial p if and only if the element a is in the set intersection I. Therefore, the number of monomials in p is exactly $|(\bigcup_{i=1}^{n} S_i) \setminus I|$.

To determine if the number of monomials in p is $\le T$, we can apply the polynomial sparsity test of Grigorescu et al. [GJR10] similarly as in [GS19a]. In particular, pick a field $\mathbb{F}_q$, sample $u \xleftarrow{\$} \mathbb{F}_q$ uniformly at random, and compute the Hankel matrix

$$H = \begin{bmatrix} \mathsf{p}(u^0) & \mathsf{p}(u^1) & \dots & \mathsf{p}(u^T) \\ \mathsf{p}(u^1) & \mathsf{p}(u^2) & \dots & \mathsf{p}(u^{T+1}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathsf{p}(u^T) & \mathsf{p}(u^{T+1}) & \dots & \mathsf{p}(u^{2T}) \end{bmatrix}.$$

Determining if the number of monomials in p is $\le T$ can be reduced to computing the determinant of H. In particular, we take the following theorem from [GJR10, Theorem 3] and [GS19a, Theorem 1].

Imported Theorem 4. *Let $q > T(T+1)(p-1)2^{\kappa}$ be a prime. If the number of monomials in p is $\le T$, then $\Pr[\det(H) = 0] = 1$, and if the number of monomials in p is $> T$, then $\Pr[\det(H) = 0] \le 2^{-\kappa}$,*

In our multi-party private intersection cardinality testing protocol, the parties will first jointly compute an encryption of H under TAHE and then jointly determine the singularity of H. Recall that by Imported Theorem 2 and 3, determining the singularity of H can be reduced to computing the minimal polynomial of a linearly recurrent sequence $\mathbf{a} = \{\mathbf{u}^\top H^i \mathbf{v}\}_{i \in \mathbb{N}}$ for random vectors $\mathbf{u}$

and $\mathbf{v}$. If P_1 picks $\mathbf{u}, \mathbf{v}$ randomly and sends to all the other parties, then all the parties can jointly compute encryption of $\mathbf{a}$ by multi-party homomorphic matrix multiplication. Afterwards $\mathbf{a}$ can be additively shared among the parties. Finally the parties run a multi-party minimal polynomial computation protocol to jointly compute the minimal polynomial of $\mathbf{a}$. The protocol is presented in Figure 9.

Theorem 6.1. *Let $q > T(T+1)(p-1)2^\kappa$ be a prime. Assuming threshold additive homomorphic encryption scheme with distributed setup, the protocol $\Pi_{\mathsf{CTest\text{-}diff}}$ (Figure 9) securely realizes $\mathcal{F}_{\mathsf{CTest\text{-}diff}}$ in the $\mathcal{F}_{\mathsf{MinPoly}}$-hybrid model.*

Proof. **Correctness.** By the correctness of TAHE, at the end of Step 2, P_1 obtains an encryption of the Hankel matrix H. By the correctness of TAHE and Π_{MMult}, at the end of Step 3e, P_1 gets encryptions of $\{\mathbf{u}^\top H^j \mathbf{v}\}_{j=0}^{2k-1}$ for random $\mathbf{u}$ and $\mathbf{v}$, namely encryptions of the first $2k$ entries of the linearly recurrent sequence $\mathbf{a} = \{\mathbf{u}^\top H^i \mathbf{v}\}_{i\in\mathbb{N}}$. By the correctness of TAHE, at the end of Step 3h, the first $2k$ entries of $\mathbf{a}$ are secret shared among the n parties. Specifically, each party P_i holds $r_i^0, \ldots, r_i^{2k-1}$ such that $a^j = \sum_{i=1}^n r_i^j$ for $j = 0, 1, \ldots, 2k-1$. By the correctness of $\mathcal{F}_{\mathsf{MinPoly}}$, in Step 3i all the parties learn a bit b and $b = 0$ if and only if $m_{\mathbf{a}}(0) = 0$, where $m_{\mathbf{a}}$ is the minimal polynomial of the sequence $\{a^j\}_{j=0}^{2k-1}$. By Imported Theorem 2 and 3, $b = 0$ if and only if $\det(H) = 0$ with all but negligible probability. Finally, by Theorem 4, $b = 0$ if and only if $|(\bigcup_{i=1}^n S_i) \setminus I| \le T$ with all but negligible probability. Therefore the protocol is correct with all but negligible probability.

Communication Cost. The bottleneck of protocol is Step 3d. The total round complexity is $O(\log T)$ in a star network and the total communication complexity is $\widetilde{O}(T^2 \cdot n \cdot \mathsf{poly}(\lambda))$.

Security. We construct a PPT Sim which simulates the view of the corrupted parties. We consider two cases of corrupted parties. In the first case P_1 is corrupted and in the second case P_1 is honest.

Case 1:* P_1 *is corrupted. Assume WLOG that the corrupted parties are $P_1, P_2, \ldots, P_t$ where $1 \le t \le n-1$. The simulator Sim has output $w \in \{\mathsf{similar}, \mathsf{different}\}$ from the ideal functionality. Sim sets a bit $b^* = 1$ if $w = \mathsf{similar}$ and $b^* = 0$ otherwise. Also, for each corrupt party P_i, Sim has as input the tuple (S_i, r_i) indicating the party's input and randomness for the protocol. The strategy of the simulator Sim for our protocol is described below.

1. Invoke the corrupted parties with their corresponding inputs and randomness.
2. Play the role of the honest parties as follows: Run the protocol honestly except that in Step 2b, each honest party P_i sets $Z_i := 0^{k\times k}$ and sends $[\![Z_i]\!]$ to P_1.
3. In Step 3i, play the role of $\mathcal{F}_{\mathsf{MinPoly}}$ and respond b^*.

Next we argue that the view of the corrupted parties generated by Sim is computationally indistinguishable to their view in the real world from $\mathcal{Z}$'s point of view, via the following hybrids.

- Hybrid_0: The view of corrupted parties and the output of honest parties in the real world.
- Hybrid_1: Same as Hybrid_0 but the output form $\mathcal{F}_{\mathsf{MinPoly}}$ is replaced by 0 if $|(\bigcup_{i=1}^n S_i) \setminus I| \le T$ and 1 otherwise. This hybrid is computationally indistinguishable from Hybrid_0 because of the correctness of the protocol.

1. **TAHE Key Generation.** Each party P_i generates $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{TAHE.DistSetup}(1^\lambda, i)$ and sends pk_i to P_1. Then P_1 sends $\mathsf{pk} = (\mathsf{pk}_1 \| \dots \| \mathsf{pk}_n)$ to all the other parties.
2. **Computing Encrypted Hankel Matrix H.**
   (a) P_1 picks a uniform random $u \xleftarrow{\$} \mathbb{F}_q$ and sends to all other parties.
   (b) Each party P_i $(i = 2, 3, \dots, n)$ sets a polynomial $\mathsf{p}_i(x) = \sum_{j=1}^{m} x^{a_j^i}$ in $\mathbb{F}_q[X]$, computes the Hankel matrix
   $$H_i = \begin{bmatrix} \mathsf{p}_i(u^0) & \mathsf{p}_i(u^1) & \dots & \mathsf{p}_i(u^T) \\ \mathsf{p}_i(u^1) & \mathsf{p}_i(u^2) & \dots & \mathsf{p}_i(u^{T+1}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathsf{p}_i(u^T) & \mathsf{p}_i(u^{T+1}) & \dots & \mathsf{p}_i(u^{2T}) \end{bmatrix},$$
   and sends an item-wise encryption of the matrix $[\![H_i]\!]$ to P_1.
   (c) P_1 does the following:
      i. Set a polynomial $\mathsf{p}_1(x) = \sum_{j=1}^{m} (n-1) \cdot x^{a_j^1}$ in $\mathbb{F}_q[X]$, compute the Hankel matrix H_1 on p_1 and u, and compute an item-wise encryption of the matrix $[\![H_1]\!]$.
      ii. Homomorphically compute $[\![H]\!] = [\![H_1 - \sum_{i=2}^{n} H_i]\!]$ and send to all the other parties.
3. **Matrix Singularity Testing of H.**
   (a) Let $k = T + 1$, then $H \in \mathbb{F}_q^{k \times k}$.
   (b) P_1 samples a uniform random vector $\mathbf{v} \xleftarrow{\$} \mathbb{F}_q^k$ and sends an item-wise encryption $[\![\mathbf{v}]\!]$ to all the other parties.
   (c) Parties run $\log k$ instances of Π_{MMult} (sequentially) to compute $\left[\!\!\left[H^{2^j}\right]\!\!\right]$ for $1 \le j \le \log k$, where $H^{2^j} = H^{2^{j-1}} \cdot H^{2^{j-1}}$.
   (d) Parties run $\log k$ instances of Π_{MMult} to compute the following homomorphic matrix multiplications, where $X|Y$ denotes the concatenation of two matrices X and Y.
   $$\begin{aligned} &[\![H\mathbf{v}]\!] = [\![H]\!] \cdot [\![\mathbf{v}]\!] \\ &[\![H^3\mathbf{v}|H^2\mathbf{v}]\!] = [\![H^2]\!] \cdot [\![H\mathbf{v}|\mathbf{v}]\!] \\ &[\![H^7\mathbf{v}|H^6\mathbf{v}|H^5\mathbf{v}|H^4\mathbf{v}]\!] = [\![H^4]\!] \cdot [\![H^3\mathbf{v}|H^2\mathbf{v}|H\mathbf{v}|\mathbf{v}]\!] \\ &\vdots \\ &\left[\!\!\left[H^{2k-1}\mathbf{v}|\dots|H^{k+1}\mathbf{v}|H^k\mathbf{v}\right]\!\!\right] = \left[\!\!\left[H^k\right]\!\!\right] \cdot \left[\!\!\left[H^{k-1}\mathbf{v}|\dots|H\mathbf{v}|\mathbf{v}\right]\!\!\right] \end{aligned}$$
   (e) P_1 samples another uniform random vector $\mathbf{u} \xleftarrow{\$} \mathbb{F}_q^k$ and homomorphically computes $[\![h^j]\!] = [\![\mathbf{u}^\top H^j \mathbf{v}]\!]$ for all $j = 0, 1, \dots, 2k-1$.
   (f) Each party P_i (except P_1) samples $r_i^j \xleftarrow{\$} \mathbb{F}_q^k$ for each $j = 0, 1, \dots, 2k-1$ and sends an encryption $[\![r_i^j]\!]$ to P_1.
   (g) P_1 homomorphically computes $[\![r_1^j]\!] = [\![h^j - \sum_{i=2}^{n} r_i^j]\!]$ for each $j = 0, 1, \dots, 2k-1$ and sends to all the other parties.
   (h) Each party P_i computes $[\![r_1^j : \mathsf{sk}_i]\!] \leftarrow \mathsf{TAHE.PartialDec}(\mathsf{sk}_i, [\![r_1^j]\!])$ for each $j = 0, 1, \dots, 2k-1$ and sends to P_1. P_1 computes $r_1^j \leftarrow \mathsf{TAHE.Combine}(\mathsf{pk}, \{[\![r_1^j : \mathsf{sk}_i]\!]\}_{i \in [n]})$ for each $j = 0, 1, \dots, 2k-1$.
   (i) Parties invoke an instance of $\mathcal{F}_{\mathsf{MinPoly}}$ where each party P_i inputs $r_i^0, \dots, r_i^{2k-1}$ and obtains a bit b.
4. **Output.** Each party P_i outputs $\mathsf{similar}$ if $b = 0$ and $\mathsf{different}$ otherwise.

Figure 9: Multi-party private intersection cardinality testing protocol $\Pi_{\mathsf{CTest\text{-}diff}}$.

- Hybrid_2: Same as Hybrid_1 except that in Step 3b of each instance of Π_{MMult}, the messages sent from honest parties are replaced by $\mathsf{TAHE.Sim}(M^A, [\![M^A]\!], \{\mathsf{sk}_i\}_{i=1}^t)$ and $\mathsf{TAHE.Sim}(M^B, [\![M^B]\!], \{\mathsf{sk}_i\}_{i=1}^t)$; in Step 3h of $\Pi_{\mathsf{CTest\text{-}diff}}$, the messages sent from honest parties are replaced by $\mathsf{TAHE.Sim}(r_1^j, [\![r_1^j]\!], \{\mathsf{sk}_i\}_{i=1}^t)$ for each $j = 0, 1, \ldots, 2k-1$. Here we omit the circuits C in $\mathsf{TAHE.Sim}$ but they are all public linear functions. This hybrid is computationally indistingushable from Hybrid_3 based on the simulation security of TAHE. In particular, if there exists a PPT distinguisher $\mathcal{D}$ that can distinguish Hybrid_1 and Hybrid_2, then we can construct a PPT adversary $\mathcal{A}$ that breaks the simulation security of TAHE.

  $\mathcal{A}$ plays the simulation security game as follows. $\mathcal{A}$ generates the view in the same way as Hybrid_1 except the following: (a) $\mathcal{A}$ generates $(\mathsf{pk}_i, \mathsf{sk}_i)$ for each corrupted party P_i and sends to the challenger; then it receives the pk_i for each honest party P_i. (b) For each encryption, $\mathcal{A}$ computes it and sends the message along with the randomness to the challenger. (c) In Step 3b of each instance of Π_{MMult} and Step 3h of $\Pi_{\mathsf{CTest\text{-}diff}}$, $\mathcal{A}$ queries the challenger for partial decryption of the honest parties. Finally, $\mathcal{A}$ outputs the view of the corrupted parties along with the output of honest parties to $\mathcal{D}$. If $\mathcal{A}$ is playing the real game, then $\mathcal{D}$ receives Hybrid_1; otherwise $\mathcal{D}$ receives Hybrid_2. Hence $\mathcal{A}$ can break the simulation security of TAHE if $\mathcal{D}$ can distinguish Hybrid_1 from Hybrid_2.

- Hybrid_3: Same as Hybrid_2 but in Step 3f, sample $r_1^j, \ldots, r_{n-1}^j$ uniformly at random and let $r_n^j := h^j - \sum_{i=1}^{n-1} r_i^j$ for all $j = 0, 1, \ldots, 2k-1$. This hybrid is statistically identical to Hybrid_2.

- Hybrid_4: Same as Hybrid_3 but in Step 3f, party P_n homomorphically computes $\left[\!\!\left[h^j - \sum_{i=1}^{n-1} r_i^j\right]\!\!\right]$ instead of computing $\left[\!\!\left[r_n^j\right]\!\!\right]$ directly, where $[\![h^j]\!]$ is computed in Step 3e and $\left[\!\!\left[r_i^j\right]\!\!\right]$'s are computed in Step 3f. This hybrid is statistically identical to Hybrid_3 because a homomorphically evaluated ciphertext is statistically identical to a fresh ciphertext.

  Notice that now in Hybrid_4, the honest values of h^j's are not needed to generate the view of corrupted parties.

- Hybrid_5: Same as Hybrid_4 except that in each instance of Π_{MMult}, we first sample two matrices M^A, M^B uniformly at random; then in Step 1 of Π_{MMult}, let $R_n^A := M^A - A - \sum_{i=1}^{n-1} R_i^A$, $R_n^B := M^B - B - \sum_{i=1}^{n-1} R_i^B$ for party P_n. This hybrid is statistically identical to Hybrid_4.

- Hybrid_6: Same as Hybrid_5 except that in each instance of Π_{MMult}, we first compute $[\![M^A]\!]$, $[\![M^B]\!]$, $[\![AB]\!]$, and $[\![M^A M^B]\!]$; then in Step 1 of Π_{MMult}, party P_n homomorphically computes $[\![R_n^A]\!] = \left[\!\!\left[M^A - A - \sum_{i=1}^{n-1} R_i^A\right]\!\!\right]$ and $[\![R_n^B]\!] = \left[\!\!\left[M^B - B - \sum_{i=1}^{n-1} R_i^B\right]\!\!\right]$; in Step 3a of Π_{MMult}, P_n homomorphically computes $\mathsf{ct}_n = \left[\!\!\left[AB - M^A M^B - \sum_{i=1}^{n-1}(R^A R_i^B - M^A R_i^B - R_i^A M^B)\right]\!\!\right]$. This hybrid is statistically identical to Hybrid_5 because a homomorphically evaluated ciphertext is statistically identical to a fresh ciphertext.

- Hybrid_7: Same as Hybrid_6 except that in each instance of Π_{MMult}, $[\![AB]\!]$ is replaced by an encryption of an all-zero matrix. Hybrid_7 is computationally indistinguishable from Hybrid_6 based on the semantic security of TAHE. We show the indistinguishability via a sequence of $(2\log k + 1)$ hybrids.

  In $\mathsf{Hybrid}_{7,h}$, in the last h instances of Π_{MMult}, $[\![A \cdot B]\!]$ is replaced by an encryption of an all-zero matrix. Then $\mathsf{Hybrid}_{7,0} = \mathsf{Hybrid}_6$ and $\mathsf{Hybrid}_{7,2\log k} = \mathsf{Hybrid}_7$. Next we prove that

$\mathsf{Hybrid}_{7,h-1}$ is computationally indistinguishable from $\mathsf{Hybrid}_{7,h}$ for all $h \in [2\log k]$.

Assume there exists a PPT distinguisher $\mathcal{D}$ that can distinguish between $\mathsf{Hybrid}_{7,h-1}$ and $\mathsf{Hybrid}_{7,h}$, then we construct a PPT $\mathcal{A}$ to break the semantic security of TAHE. $\mathcal{A}$ plays the semantic security game as follows. $\mathcal{A}$ generates the view in the same way as $\mathsf{Hybrid}_{7,h-1}$ except the following: (a) $\mathcal{A}$ generates $(\mathsf{pk}_i, \mathsf{sk}_i)$ for each corrupted party P_i and sends to the challenger; then it receives the pk_i for each honest party P_i. (b) In the h-th from last instance of Π_{MMult}, $\mathcal{A}$ sends AB and an all-zero matrix to the challenger and gets back an encryption; then $\mathcal{A}$ uses that as $[\![AB]\!]$. Strictly speaking, $\mathcal{A}$ can only send one pair of messages to the challenger, so we need another sequence of hybrids where $\mathcal{A}$ changes one message at a time. We omit the details here. Finally, $\mathcal{A}$ outputs the view of the corrupted parties along with the output of honest parties to $\mathcal{D}$. If the challenger returns an encryption of $A \cdot B$, then $\mathcal{D}$ receives $\mathsf{Hybrid}_{7,h-1}$; otherwise $\mathcal{D}$ receives $\mathsf{Hybrid}_{7,h}$. Hence $\mathcal{A}$ can break the semantic security of TAHE if $\mathcal{D}$ can distinguish between $\mathsf{Hybrid}_{7,h-1}$ and $\mathsf{Hybrid}_{7,h}$.

Notice that now in Hybrid_7, the honest values of A and B are not needed in Π_{MMult} when generating the view of corrupted parties.

- Hybrid_8: Same as Hybrid_7 except that in Step 2b the messages from honest parties are replaced by encryption of all-zero matrices. This hybrid is computationally indistingushable from Hybrid_7 based on the semantic security of TAHE.

  Assume there exists a PPT distinguisher $\mathcal{D}$ that can distinguish between Hybrid_7 and Hybrid_8, then we construct a PPT $\mathcal{A}$ to break the semantic security of TAHE. $\mathcal{A}$ plays the semantic security game as follows. $\mathcal{A}$ generates the view in the same way as Hybrid_7 except the following: (a) $\mathcal{A}$ generates $(\mathsf{pk}_i, \mathsf{sk}_i)$ for each corrupted party P_i and sends to the challenger; then it receives the pk_i for each honest party P_i. (b) In Step 2b, $\mathcal{A}$ sends honest matrices and an all-zero matrices to the challenger and gets back encryption of one of them; then $\mathcal{A}$ uses that as the messages from honest parties. Strictly speaking, $\mathcal{A}$ can only send one pair of messages to the challenger, so we need a sequence of hybrids where $\mathcal{A}$ changes one message at a time. We omit the details here. Finally, $\mathcal{A}$ outputs the view of the corrupted parties along with the output of honest parties to $\mathcal{D}$. If the challenger returns encryption of honest matrices, then $\mathcal{D}$ receives Hybrid_7; otherwise $\mathcal{D}$ receives Hybrid_8. Hence $\mathcal{A}$ can break the semantic security of TAHE if $\mathcal{D}$ can distinguish between Hybrid_7 and Hybrid_8.

- Hybrid_9: Same as Hybrid_8 but in each instance of Π_{MMult}, $[\![AB]\!]$ is computed from honestly computed A and B. Hybrid_9 is computationally indistinguishable from Hybrid_8 based on the semantic security of TAHE. We can argue the indistinguishability via a sequence of $(2\log k+1)$ hybrids similarly as the argument between Hybrid_6 and Hybrid_7.

- Hybrid_{10}: Same as Hybrid_9 except that in Steps 1 and 3a of each instance of Π_{MMult}, party P_n computes its encryptions from R_n^A and R_n^B as described in the protocol. This hybrid is statistically identical to Hybrid_9 because a fresh ciphertext is statistically identical to a homomorphically evaluated ciphertext.

- Hybrid_{11}: Same as Hybrid_{10} except that in each instance of Π_{MMult}, P_n samples R_n^A and R_n^B honestly. This hybrid is statistically identical to Hybrid_{10}.

- Hybrid_{12}: Same as Hybrid_{11} but in Step 3f, party P_n computes $\left[\!\left[r_n^j\right]\!\right]$ directly from r_n^j for all $j = 0, 1, \ldots, 2k-1$. This hybrid is statistically identical to Hybrid_{11} because a fresh ciphertext is statistically identical to a homomorphically evaluated ciphertext.

- Hybrid_{13}: Same as Hybrid_{12} but in Step 3f, party P_n samples r_n^j honestly for all $j = 0, 1, \ldots, 2k-1$. This hybrid is statistically identical to Hybrid_{12}.
- Hybrid_{14}: Same as Hybrid_{13} except that in Steps 3b and 3h of each instance of Π_{MMult}, the messages sent from honest parties are computed honestly by $\mathsf{TAHE.PartialDec}$. This hybrid is computationally indistingushable from Hybrid_{13} based on the simulation security of TAHE. We can prove the indistinguishability similarly as in the argument between Hybrid_1 and Hybrid_2, This hybrid outputs the view generated by Sim and the output of honest parties in the ideal world.

Case 2: P_1 is honest. Assume WLOG that the corrupted parties are $P_t, P_{t+1}, \ldots, P_n$ where $2 \le t \le n$. The simulator Sim has output $w \in \{\mathsf{similar}, \mathsf{different}\}$ from the ideal functionality. Sim sets a bit $b^* = 1$ if $w = \mathsf{similar}$ and $b^* = 0$ otherwise. Also, for each corrupt party P_i, Sim has as input the tuple (S_i, r_i) indicating the party's input and randomness for the protocol. The strategy of the simulator Sim for our protocol is described below.

1. Invoke the corrupted parties with their corresponding inputs and randomness.
2. Play the role of the honest parties as follows: Run the protocol honestly except that in Step 2(c)ii, P_1 sets $Z := 0^{k \times k}$ and sends $[\![Z]\!]$ to all the other parties.
3. In Step 3i, play the role of $\mathcal{F}_{\mathsf{MinPoly}}$ and respond b^*.

Next we argue that the view of the corrupted parties generated by Sim is computationally indistinguishable to their view in the real world from $\mathcal{Z}$'s point of view, via the following hybrids.

- Hybrid_0: The view of corrupted parties and the output of honest parties in the real world.
- Hybrid_1: Same as Hybrid_0 but the output form $\mathcal{F}_{\mathsf{MinPoly}}$ is replaced by 0 if $|(\bigcup_{i=1}^n S_i) \setminus I| \le T$ and 1 otherwise. This hybrid is computationally indistinguishable from Hybrid_0 because of the correctness of the protocol.
- Hybrid_2: Same as Hybrid_1 except that all the encryptions sent from P_1 are replaced by fresh encryptions. In particular, in Steps 2(c)ii, 3g of $\Pi_{\mathsf{CTest\text{-}diff}}$ and Steps 2, 4c of Π_{MMult}, P_1 sends fresh encryptions instead of homomorphically computed ciphertexts to the corrupted parties. This hybrid is statistically identical to Hybrid_1 because a fresh ciphertext is statistically identical to a homomorphically evaluated ciphertext.
- Hybrid_3: Same as Hybrid_2 but all the encryptions sent from P_1 are replaced by encryptions of zero. In particular, in Steps 2(c)ii, 3g of $\Pi_{\mathsf{CTest\text{-}diff}}$ and Steps 2, 4c of Π_{MMult}, P_1 sends encryption of zeros to the corrupted parties. Hybrid_3 is computationally indistinguishable from Hybrid_2 based on the semantic security of TAHE.

  Assume there exists a PPT distinguisher $\mathcal{D}$ that can distinguish between Hybrid_2 and Hybrid_3, then we construct a PPT $\mathcal{A}$ to break the semantic security of TAHE. $\mathcal{A}$ plays the semantic security game as follows. $\mathcal{A}$ generates the view in the same way as Hybrid_2 except the following: (a) $\mathcal{A}$ generates $(\mathsf{pk}_i, \mathsf{sk}_i)$ for each corrupted party P_i and sends to the challenger; then it receives the pk_i for each honest party P_i. (b) In Steps 2(c)ii, 3g of $\Pi_{\mathsf{CTest\text{-}diff}}$ and Steps 2, 4c of Π_{MMult}, $\mathcal{A}$ sends honest values and zeros to the challenger and gets back encryption of one of them; then $\mathcal{A}$ uses that as the messages sent to the corrupted parties. Strictly speaking,

$\mathcal{A}$ can only send one pair of messages to the challenger, so we need a sequence of hybrids where $\mathcal{A}$ changes one message at a time (from the last to the first). We omit the details here. Finally, $\mathcal{A}$ outputs the view of the corrupted parties along with the output of honest parties to $\mathcal{D}$. If the challenger returns encryption of honest values, then $\mathcal{D}$ receives Hybrid_2; otherwise $\mathcal{D}$ receives Hybrid_3. Hence $\mathcal{A}$ can break the semantic security of TAHE if $\mathcal{D}$ can distinguish between Hybrid_2 and Hybrid_3.

- Hybrid_4: Same as Hybrid_3 except that all the encryptions sent from P_1 in Step 3g of $\Pi_{\mathsf{CTest\text{-}diff}}$ and Steps 2, 4c of Π_{MMult} are replaced by fresh encryptions of honestly computed values. Hybrid_4 is computationally indistinguishable from Hybrid_3 based on the semantic security of TAHE.
- Hybrid_5: Same as Hybrid_3 except that all the encryptions sent from P_1 in Step 3g of $\Pi_{\mathsf{CTest\text{-}diff}}$ and Steps 2, 4c of Π_{MMult} are replaced by homomorphically computed ciphertexts. Hybrid_4 is computationally indistinguishable from Hybrid_5 because a fresh ciphertext is statistically identical to a homomorphically evaluated ciphertext. This hybrid outputs the view generated by Sim and the output of honest parties in the ideal world.

□

Corollary 6.2. *Assuming TAHE with distributed setup, protocol $\Pi_{\mathsf{CTest\text{-}diff}}$ (Figure 9) securely realizes $\mathcal{F}_{\mathsf{CTest\text{-}diff}}$ in the star network communication model with communication complexity $\widetilde{O}(n \cdot T^2 \cdot \mathsf{poly}(\lambda))$ and round complexity $O(\log T)$.*

7 Threshold PSI for $\mathcal{F}_{\mathsf{TPSI\text{-}diff}}$

Recall that in a multi-party threshold PSI protocol for functionality $\mathcal{F}_{\mathsf{TPSI\text{-}diff}}$ defined in Figure 2, each party wishes to learn the intersection of all their sets if $|(\bigcup_{i=1}^{n} S_i) \setminus I| \leq T$, that is, if the size of the union of all their sets minus the intersection is less than the threshold T. In this section, we describe our multi-party threshold PSI protocol based on any protocol for multi-party private intersection cardinality testing. We rely on threshold additive homomorphic encryption (TAHE) with distributed setup.

Theorem 7.1. *Assuming threshold additive homomorphic encryption with distributed setup, protocol $\Pi_{\mathsf{TPSI\text{-}diff}}$ (Figure 10) securely realizes $\mathcal{F}_{\mathsf{TPSI\text{-}diff}}$ in the $\mathcal{F}_{\mathsf{CTest\text{-}diff}}$-hybrid model in the star network communication model. Our protocol is secure against a semi-honest adversary that can corrupt up to $(n-1)$ parties.*

The protocol runs in a constant number of rounds and the communication complexity is $O(n \cdot T \cdot \mathsf{poly}(\lambda))$ in the $\mathcal{F}_{\mathsf{CTest\text{-}diff}}$-hybrid model. We then instantiate the $\mathcal{F}_{\mathsf{CTest\text{-}diff}}$-hybrid with the two protocols from the previous sections: one based on TFHE from Section 5.2 that has constant round complexity and $O(n \cdot T \cdot \mathsf{poly}(\lambda))$ communication complexity and the other based on TAHE from Section 6 that has round complexity $O(\log T)$ and communication complexity $\widetilde{O}(n \cdot T^2 \cdot \mathsf{poly}(\lambda))$. Formally, we get the following corollaries:

Corollary 7.2. *Assuming TFHE with distributed setup, protocol $\Pi_{\mathsf{TPSI\text{-}diff}}$ (Figure 10) securely realizes $\mathcal{F}_{\mathsf{TPSI\text{-}diff}}$ in the star network communication model with communication complexity $O(n \cdot T \cdot \mathsf{poly}(\lambda))$.*

Corollary 7.3. *Assuming TAHE with distributed setup, protocol $\Pi_{\mathsf{TPSI\text{-}diff}}$ (Figure 10) securely realizes $\mathcal{F}_{\mathsf{TPSI\text{-}diff}}$ in the star network communication model with communication complexity $\widetilde{O}(n \cdot T^2 \cdot \mathsf{poly}(\lambda))$ and round complexity $O(\log T)$.*

Our threshold PSI protocol for functionality $\mathcal{F}_{\mathsf{TPSI\text{-}int}}$ is almost identical and we describe the details in Appendix 8.

7.1 Protocol

Consider n parties $P_1, \ldots, P_n$ with input sets $S_1, \ldots, S_n$ of size m and a star network where the central party is P_1. The parties first run the private intersection cardinality testing protocols for functionality $\mathcal{F}_{\mathsf{CTest\text{-}diff}}$ from the previous sections and proceed if $|(\bigcup_{i=1}^{n} S_i) \setminus I| \leq T$. Then, each party P_i encodes its set as a polynomial $\mathsf{p}'_i(x) = (x - r_i) \cdot \prod_{j=1}^{m}(x - a_j^i)$ where r_i is picked uniformly at random. The parties then compute $(3T+4)$ evaluations of the following polynomial $\mathsf{V}(\cdot)$ on points $1, \ldots, (3T+4)$ using threshold additive homomorphic encryption: $\mathsf{V}(x) = \sum_{i=1}^{n} (\mathsf{p}'_i(x) \cdot \mathsf{R}_i(x))$ where each $\mathsf{R}_i(\cdot)$ is a uniformly random polynomial of degree T that is computed as an addition of n random polynomials - one generated by each party. Then, each party P_i interpolates the degree $(3T+3)$ rational polynomial $\frac{\mathsf{V}(\cdot)}{p'_i(\cdot)}$ using the $(3T+4)$ evaluations (we elaborate more on how we arrive at the degree of the rational polynomial while describing the correctness in Appendix **??**). Finally, each party outputs the intersection as $S_i \setminus D_i$ where D_i denotes the roots of the above interpolated polynomial. Our protocol is formally described in Figure 10.

7.2 Security Proof

In this section, we formally prove Theorem 7.1.

Correctness. If $|(\bigcup_{i=1}^{n} S_i) \setminus I| > T$, then the protocol terminates after the first step – private intersection cardinality testing. If, on the other hand, $|(\bigcup_{i=1}^{n} S_i) \setminus I| \leq T$, observe that polynomial $\mathsf{V}(x)$ can be rewritten as $\sum_{i=1}^{n} \mathsf{p}'_i(x) \cdot U_i(x)$ where each U_i is a uniformly random polynomial of degree at most $T+1$. Now, from the correctness of the TAHE scheme, each party P_i learns $3T+4$ evaluations of the rational polynomial:

$$\mathsf{q}_i(x) = \frac{\mathsf{V}(x)}{\mathsf{p}'_i(x)} = \frac{\sum_{i=1}^{n} \mathsf{p}'_i(x) \cdot U_i(x)}{\mathsf{p}'_i(x)} = \frac{\sum_{i=1}^{n} \mathsf{p}_{i \setminus I}(x) \cdot (x - r_i) \cdot U_i(x)}{\mathsf{p}_{i \setminus I}(x) \cdot (x - r_i)}.$$

Since $|S_i - I| \leq T$ for each $i \in [n]$, the numerator is a polynomial of degree at most $2T+2$ and the denominator is a polynomial of degree at most $T+1$. Further, since each U_i is uniformly random, by Lemma 3.3, the numerator is a random degree $2T+2$ polynomial. From Imported Lemma 1, the gcd of the polynomials in the numerator and denominator is 1 and hence no other terms will get canceled out. Therefore, each party P_i can interpolate this rational polynomial using $3T+4$ evaluation points and thereby learn the numerator and denominator. Finally, observe that for each party P_i, the roots of the denominator contains the set $S_i \setminus I$ and a random r_i, from which P_i can easily compute the intersection I.

Communication Cost. The first phase of the protocol, namely private intersection cardinality testing, has a communication complexity of $O(n \cdot T \cdot \mathsf{poly}(\lambda))$ when instantiated with the scheme based on threshold fully homomorphic encryption as shown in Section 5.2 and a communication

Parameters: Parties $P_1, \ldots, P_n$. Each party has a set of m elements. Threshold $T \in \mathbb{N}$. $\mathbb{F}$ is a finite field where $|\mathbb{F}| = \Omega(2^\lambda)$.

Inputs: Party P_i has an input set $S_i = \{a_1^i, \ldots, a_m^i\}$ where $a_j^i \in \mathbb{F}$ for all $j \in [m]$.

Output: Each party P_i receives the set intersection $I = \bigcap_{i=1}^n S_i$ if and only if $|(\bigcup_{i=1}^n S_i) \setminus I| \leq T$.

Protocol:

1. **Private Intersection Cardinality Testing.** The parties invoke $\mathcal{F}_{\mathsf{CTest\text{-}diff}}$ on inputs $S_1, \ldots, S_n$ and receive back $w \in \{\mathsf{similar}, \mathsf{different}\}$. If $w = \mathsf{different}$ then all parties output $\perp$.
2. **TAHE Key Generation.** Each party P_i generates $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{TAHE.DistSetup}(1^\lambda, i)$ and sends pk_i to P_1. Then P_1 sends $\mathsf{pk} = (\mathsf{pk}_1 \| \ldots \| \mathsf{pk}_n)$ to all the other parties.
3. **Evaluations of Random Polynomial.** In this phase the parties will evaluate a polynomial
$$\mathsf{V}(x) = \sum_{i=1}^{n} \Big(\mathsf{p}'_i(x) \cdot \Big(\mathsf{R}_1(x) + \ldots + \mathsf{R}_{i-1}(x) + \widetilde{\mathsf{R}}_i(x) + \mathsf{R}_{i+1}(x) \ldots + \mathsf{R}_n(x) \Big) \Big)$$
for $x \in [3T+4]$ where the terms are defined as follows.
   (a) Each party P_i defines $\mathsf{p}_i(x) = \prod_{j=1}^m (x - a_j^i)$ and $\mathsf{p}'_i(x) = \mathsf{p}_i(x) \cdot (x - r_i)$ where $r_i \xleftarrow{\$} \mathbb{F}$.
   (b) Each party P_i uniformly samples $\mathsf{R}_i, \widetilde{\mathsf{R}}_i \xleftarrow{\$} \mathbb{F}[x]$ of degree $T+1$, computes $\mathsf{R}_i(x)$ for $x \in [3T+4]$ and sends encrypted $[\![\mathsf{R}_i(x)]\!]$ to P_1.
   (c) For each $i \in [n], x \in [3T+4]$, party P_1 sends $[\![e_{i,x}]\!] = \Big[\!\!\Big[\sum_{j \in [n] \setminus \{i\}} \mathsf{R}_j(x)\Big]\!\!\Big]$ to P_i.
   (d) For each $x \in [3T+4]$, each party P_i sends $[\![v_{i,x}]\!] = \Big[\!\!\Big[\mathsf{p}'_i(x) \cdot \Big(e_{i,x} + \widetilde{\mathsf{R}}_i(x)\Big)\Big]\!\!\Big]$ to P_1.
   (e) For each $x \in [3T+4]$, P_1 sends $[\![v_x]\!] = [\![\sum_{i=1}^n v_{i,x}]\!]$ to all P_i.
   (f) For each $x \in [3T+4]$, each party P_i sends $[\![v_x : \mathsf{sk}_i]\!] \leftarrow \mathsf{TAHE.PartialDec}(\mathsf{sk}_i, [\![v_x]\!])$ to P_1.
   (g) For each $x \in [3T+4]$, P_1 sends $\mathsf{V}(x) = \mathsf{TAHE.Combine}(\mathsf{pk}, \{[\![v_x : \mathsf{sk}_i]\!]\}_{i \in [n]})$ to all P_i.
4. **Computing Set Intersection.** Each party P_i does the following:
   (a) Interpolate $\mathsf{q}_i(x)$ to be the degree $3T+3$ rational polynomial such that $\mathsf{q}_i(x) = \frac{\mathsf{V}(x)}{\mathsf{p}'_i(x)}$ for $x \in [3T+4]$ and the gcd of the numerator and denominator is 1. Let D_i be the roots of the denominator of $\mathsf{q}_i(x)$.
   (b) Output the set intersection $I = S_i \setminus D_i$.

Figure 10: Multi-party threshold PSI protocol $\Pi_{\mathsf{TPSI\text{-}diff}}$ for functionality $\mathcal{F}_{\mathsf{TPSI\text{-}diff}}$.

complexity of $\widetilde{O}(n \cdot T^2 \cdot \mathsf{poly}(\lambda))$ when instantiated with the scheme based on threshold additive homomorphic encryption as shown in Section 6.

We now analyze the communication cost for the second phase where the parties compute the concrete intersection. The TAHE key generation is independent of the set sizes and the threshold T and has a communication complexity of only $O(n \cdot \mathsf{poly}(\lambda))$. The bottleneck of the protocol is in Step 3, that is, evaluating the random polynomial. In Steps 3b, 3d, and 3f, every party sends $3T+4$ encryptions or partial decryptions to P_1 hence the cost for these steps is $O(n \cdot T \cdot \mathsf{poly}(\lambda))$. In Steps 3c, 3e, and 3g, P_1 sends $3T+4$ ciphertexts or plaintexts to every other party so the cost of these steps is $O(n \cdot T \cdot \mathsf{poly}(\lambda))$. Finally, the last stage, namely computing the set intersection, does not involve any communication. Thus, the overall communication cost for computing the intersection is $O(n \cdot T \cdot \mathsf{poly}(\lambda))$.

Therefore, when the private intersection cardinality testing protocol is instantiated with the TFHE-based protocol, the overall communication complexity is $O(n \cdot T \cdot \mathsf{poly}(\lambda))$ and when instantiated with the TAHE-based scheme, the overall communication complexity is $\widetilde{O}(n \cdot T^2 \cdot \mathsf{poly}(\lambda))$ for some apriori fixed polynomial $\mathsf{poly}(\cdot)$ and is independent of the size of each input set m.

Security. Consider an environment $\mathcal{Z}$ who corrupts a set $\mathcal{S}^*$ of n^* parties where $n^* < n$. The simulator Sim has the output of the functionality $\mathcal{F}_{\mathsf{TPSI\text{-}diff}}$, namely the intersection set I or $\perp$. Sim sets $w = \mathsf{similar}$ if the output is I and $w = \mathsf{different}$ if the output is $\perp$. In addition, Sim has the tuple (S_i, r_i) for each corrupt party P_i indicating the party's input and randomness for the protocol. The strategy of the simulator Sim for our multi-party threshold PSI protocol is described below.

(a) Private Intersection Cardinality Testing: Sim plays the role of the ideal functionality $\mathcal{F}_{\mathsf{CTest\text{-}diff}}$ and responds with w.

(b) TAHE Key Generation: Sim runs the distributed key generation algorithm $\mathsf{TAHE.DistSetup}(1^\lambda, i)$ of the TAHE scheme honestly on behalf of each honest party P_i as in the real world. Note that Sim also knows $(\{\mathsf{sk}_i\}_{i \in \mathcal{S}^*})$ as it knows the randomness for the corrupt parties.

(c) Evaluations of Random Polynomial: Sim does the following:

1. Encode the intersection set $I = \{b_1, \ldots, b_{|I|}\}$ as a polynomial as follows: $\mathsf{p}_I(x) = \Pi_{i=1}^{|I|}(x - b_i)$.
2. Pick a random polynomial $U(\cdot)$ of degree $2T + 2$ and set the polynomial $\mathsf{V}(x)$ as follows: $\mathsf{V}(x) = \mathsf{p}_I(x) \cdot U(x)$.
3. In Steps 3b-3e, on behalf of every honest party P_i, whenever P_i has to send any ciphertext, send $[\![0]\!]$ using fresh randomness.
4. For each $x \in [3T + 4]$, let $[\![v_x]\!]$ denote the ciphertext that is sent to all the parties at the end of Step 3f.
5. In Step 3f, for each $j \in [3T + 4]$, on behalf of each honest party P_i, instead of computing $\{[\![v_x : \mathsf{sk}_i]\!]\}$ by running the honest $\mathsf{TAHE.PartialDec}$ algorithm as in the real world, Sim computes the partial decryptions by running the simulator $\mathsf{TAHE.Sim}$ as follows: $\{[\![v_x : \mathsf{sk}_i]\!]\} \leftarrow \mathsf{TAHE.Sim}(\mathsf{C}, \mathsf{V}(x), [\![v_x]\!], \{\mathsf{sk}_i\}_{i \in \mathcal{S}^*})$, where C is the public linear circuit to compute $\mathsf{V}(x)$ by P_1.
6. Finally, in Step 3g, if P_1 is honest, send the evaluations of polynomial $\mathsf{V}(x)$ as in the real world description.

Hybrids. We now show that the above simulation strategy is successful against all environments $\mathcal{Z}$ that corrupt parties in a semi-honest manner. That is, the view of the corrupt parties along with the output of the honest parties is computationally indistinguishable in the real and ideal worlds. We will show this via a series of computationally indistinguishable hybrids where the first hybrid Hybrid_0 corresponds to the real world and the last hybrid Hybrid_4 corresponds to the ideal world.

- Hybrid_0 **- Real World:** In this hybrid, consider a simulator SimHyb that plays the role of the honest parties as in the real world.
- Hybrid_1 **- Private Intersection Cardinality Testing:** In this hybrid, SimHyb plays the role of the ideal functionality $\mathcal{F}_{\mathsf{CTest\text{-}diff}}$ and responds with $\mathsf{similar}$ if $|(\bigcup_{i=1}^{n} S_i) \setminus I| \leq T$ and $\mathsf{different}$ otherwise.
- Hybrid_2 **- Simulate Partial Decryptions:** In this hybrid, in the evaluations of random polynomial, SimHyb simulates the partial decryptions generated by the honest parties in Step 3f as done in the ideal world. That is, for each $j \in [3T+4]$, SimHyb computes the partial decryptions as $\{[\![v_x : \mathsf{sk}_i]\!]\} \leftarrow \mathsf{TAHE.Sim}(\mathsf{C}, \mathsf{V}(x), [\![v_x]\!], \{\mathsf{sk}_i\}_{i \in \mathcal{S}^*})$. Observe that the polynomial $\mathsf{V}(\cdot)$ is still computed as in the real world (and in Hybrid_2).
- Hybrid_3 **- Switch Polynomial Computation:** In this hybrid, the polynomial $\mathsf{V}(\cdot)$ is no longer computed as in the real world. Instead, SimHyb now picks a random polynomial $U(\cdot)$ of degree $2T+2$ and sets the polynomial $\mathsf{V}(\cdot)$ as follows: $\mathsf{V}(x) = \mathsf{p}_I(x) \cdot U(x)$.
- Hybrid_4 **- Switch Encryptions:** In this hybrid, in the evaluations of random polynomial, SimHyb now computes every ciphertext generated on behalf of any honest party as encryptions of 0 as done by Sim in the ideal world. This hybrid corresponds to the ideal world.

We now show that every pair of consecutive hybrids is computationally indistinguishable.

Lemma 7.4. Hybrid_0 *is computationally indistinguishable from* Hybrid_1 *based on the correctness of our protocol.*

Proof. The only difference between the two hybrids is that in Hybrid_0, the simulator SimHyb calls $\mathcal{F}_{\mathsf{CTest\text{-}diff}}$ honestly while in Hybrid_1, SimHyb plays the role of the ideal functionality $\mathcal{F}_{\mathsf{CTest\text{-}diff}}$ responds with $\mathsf{similar}$ if $|(\bigcup_{i=1}^{n} S_i) \setminus I| \leq T$ and $\mathsf{different}$ otherwise. The output of $\mathcal{F}_{\mathsf{CTest\text{-}diff}}$ in Hybrid_1 is always correct. Based on the correctness of our protocol $\Pi_{\mathsf{TPSI\text{-}diff}}$, the output of $\mathcal{F}_{\mathsf{CTest\text{-}diff}}$ in Hybrid_0 is correct with overwhelming probability. Hence Hybrid_0 and Hybrid_1 are computationally indistinguishable. □

Lemma 7.5. *Assuming the simulation security of the threshold additive homomorphic encryption scheme,* Hybrid_1 *is computationally indistinguishable from* Hybrid_2*.*

Proof. The only difference between the two hybrids is that in Hybrid_1, the simulator SimHyb generates the partial decryptions of the TAHE scheme on behalf of the honest parties as in the real world while in Hybrid_2, they are simulated by running the simulator $\mathsf{TAHE.Sim}$. We now show that if there exists an adversarial environment $\mathcal{Z}$ that can distinguish between these two hybrids with some non-negligible probability ϵ, we will come up with a reduction $\mathcal{A}$ that can break the simulation security of the TAHE scheme.

$\mathcal{A}$ interacts with a challenger $\mathcal{C}$ in the simulation security game for TAHE and with the environment $\mathcal{Z}$ in the game between Hybrid_1 and Hybrid_2. $\mathcal{A}$ corrupts the same set of parties as done by $\mathcal{Z}$ in its game with $\mathcal{C}$. Further, $\mathcal{A}$ forwards the public key-secret key pairs $(\mathsf{pk}_i, \mathsf{sk}_i)$ for the corrupt parties it receives from $\mathcal{Z}$ to the challenger and the public keys pk_i for the honest parties from $\mathcal{C}$ to $\mathcal{Z}$. $\mathcal{A}$ also forwards to $\mathcal{C}$ the set of messages to be encrypted along with the randomness for the ones encrypted by the adversary, received from $\mathcal{Z}$. Similarly, it forwards the ciphertexts received from $\mathcal{C}$ to $\mathcal{Z}$. Finally, $\mathcal{A}$ sends the circuit corresponding to the evaluation of polynomial $\mathsf{V}(\cdot)$ to $\mathcal{C}$ and receives a set of partial decryptions on behalf of each honest party which it forwards to $\mathcal{A}$. It continues interacting with $\mathcal{Z}$ as in Hybrid_1 in the rest of its interaction. It is easy to see that if $\mathcal{C}$ sent honestly generated partial decryptions, the interaction between $\mathcal{A}$ and $\mathcal{Z}$ exactly corresponds to Hybrid_1 and if the partial decryptions were simulated, the interaction between $\mathcal{A}$ and $\mathcal{Z}$ exactly corresponds to Hybrid_2. Thus, if $\mathcal{Z}$ can distinguish between the two hybrids with non-negligible probability ϵ, $\mathcal{A}$ can break the simulation security of the TAHE scheme with the same probability ϵ which is a contradiction. □

Lemma 7.6. Hybrid_2 *is statistically close to* Hybrid_3*.*

Proof. The only difference between the two hybrids is the way the polynomial $\mathsf{V}(\cdot)$ is computed. In Hybrid_3, $\mathsf{V}(x) = \mathsf{p}_I(x) \cdot U(x)$ where $U(\cdot)$ is a uniformly random polynomial of degree $2T+2$. In Hybrid_2,[9]

$$\begin{aligned}
\mathsf{V}(x) =& \sum_{i=1}^{n} \Big(\mathsf{p}'_i(x) \cdot \Big(\mathsf{R}_1(x) + \ldots + \mathsf{R}_{i-1}(x) + \widetilde{\mathsf{R}}_i(x) + \mathsf{R}_{i+1}(x) \ldots + \mathsf{R}_n(x)\Big)\Big)\\
=& \mathsf{p}_I(x) \cdot \sum_{i=1}^{n} \Big(\mathsf{p}'_{i\setminus I}(x) \cdot \Big(\mathsf{R}_1(x) + \ldots + \mathsf{R}_{i-1}(x) + \widetilde{\mathsf{R}}_i(x) + \mathsf{R}_{i+1}(x) \ldots + \mathsf{R}_n(x)\Big)\Big)\\
=& \mathsf{p}_I(x) \cdot \left[\sum_{i\in\mathcal{S}^*} \mathsf{p}'_{i\setminus I}(x) \cdot \left(\widetilde{\mathsf{R}}_i(x) + \sum_{j\in\mathcal{S}^*} \mathsf{R}_j(x)\right) + \sum_{i\in\mathcal{S}^*} \mathsf{p}'_{i\setminus I}(x) \cdot \left(\sum_{j\in[n]\setminus\mathcal{S}^*} \mathsf{R}_j(x)\right)\right.\\
&\left. + \sum_{i\in[n]\setminus\mathcal{S}^*} \mathsf{p}'_{i\setminus I}(x) \cdot \left(\widetilde{\mathsf{R}}_i(x) + \sum_{j\in[n]} \mathsf{R}_i(x)\right)\right]\\
=& \mathsf{p}_I(x) \cdot [A(x) + B(x)]
\end{aligned}$$

where

$$\begin{aligned}
A(x) &= \sum_{i\in\mathcal{S}^*} \mathsf{p}'_{i\setminus I}(x) \cdot \left(\widetilde{\mathsf{R}}_i(x) + \sum_{j\in\mathcal{S}^*} \mathsf{R}_j(x)\right)\\
B(x) &= \left(\sum_{i\in\mathcal{S}^*} \mathsf{p}'_{i\setminus I}(x) \cdot \left(\sum_{j\in[n]\setminus\mathcal{S}^*} \mathsf{R}_i(x)\right)\right) + \sum_{i\in[n]\setminus\mathcal{S}^*} \mathsf{p}'_{i\setminus I}(x) \cdot \left(\widetilde{\mathsf{R}}_i(x) + \sum_{j\in[n]} \mathsf{R}_j(x)\right).
\end{aligned}$$

Note that for all $i \in [n]$, $\mathsf{Deg}(\mathsf{p}_{i\setminus I}(x)) \leq T$, $\mathsf{Deg}(\mathsf{p}'_{i\setminus I}(x)) \leq T+1$, $\mathsf{Deg}(\mathsf{R}_i(x)) = \mathsf{Deg}(\widetilde{\mathsf{R}}_i(x)) = T+1$. Thus, $\mathsf{Deg}(A(x)) = \deg(B(x)) = 2T+2$. Now, suppose we prove that $B(x)$ is statistically close to a uniformly random polynomial of degree $2T+2$, then in Hybrid_2,

$$\begin{aligned}
\mathsf{V}(x) &\equiv \mathsf{p}_I(x) \cdot \Big[A(x) + U_1(x)\Big]\\
&\equiv \mathsf{p}_I(x) \cdot U_2(x)
\end{aligned}$$

where $U_1(\cdot)$ and $U_2(\cdot)$ are uniformly random polynomials of degree $2T+2$. Thus, $\mathsf{V}(x)$ in Hybrid_2 is statistically close to the distribution of $\mathsf{V}(x)$ in Hybrid_3. Therefore, the only remaining step is to prove the below claim.

[9]Here, $\mathsf{p}'_{i\setminus I}(x) = \frac{\mathsf{p}'_i(x)}{\mathsf{p}_I(x)} = \mathsf{p}_{i\setminus I}(x) \cdot (x - r_i)$.

Claim 7.7. *$B(x)$ is statistically close to a uniformly random polynomial of degree $2T+2$.*

Proof.

$$B(x) = \left(\sum_{i\in\mathcal{S}^*} \mathsf{p}'_{i\setminus I}(x)\cdot\left(\sum_{j\in[n]\setminus\mathcal{S}^*}\mathsf{R}_i(x)\right)\right) + \sum_{i\in[n]\setminus\mathcal{S}^*}\mathsf{p}'_{i\setminus I}(x)\cdot\left(\widetilde{\mathsf{R}}_i(x)+\sum_{j\in[n]}\mathsf{R}_j(x)\right)$$
$$= \left(\sum_{i\in\mathcal{S}^*}\mathsf{p}'_{i\setminus I}(x)\right)\cdot\widetilde{U}(x) + \sum_{i\in[n]\setminus\mathcal{S}^*}\left(\mathsf{p}'_{i\setminus I}(x)\cdot U_i(x)\right).$$

where $\widetilde{U}(x), U_i(x)$ are uniformly random polynomials of degree $T+1$.

Note that by the definition of the set intersection I,

$$\gcd(\mathsf{p}_{1\setminus I}(x),\ldots,\mathsf{p}_{n\setminus I}(x)) = 1.$$

By a direct invocation of Lemma 3.2,

$$\gcd\left(\sum_{i\in\mathcal{S}^*}\mathsf{p}'_{i\setminus I}(x), \mathsf{p}'_{j_1\setminus I}(x), \mathsf{p}'_{j_2\setminus I}(x),\ldots,\mathsf{p}'_{j_t\setminus I}(x)\right) = 1$$

except with negligible probability where $j_1,\ldots,j_t$ denotes the indices of the honest parties. Then, by Lemma 3.3, we can conclude that $B(x)$ is statistically close to a uniformly random polynomial of degree $2T+2$. □

□

Lemma 7.8. *Assuming the semantic security of the threshold additive homomorphic encryption scheme,* Hybrid_3 *is computationally indistinguishable from* Hybrid_4*.*

Proof. The only difference between the two hybrids is that in Hybrid_3, the simulator SimHyb generates the encryptions of the TAHE scheme on behalf of the honest parties as in the real world while in Hybrid_4, they are generated as encryptions of 0. We now show that if there exists an adversarial environment $\mathcal{Z}$ that can distinguish between these two hybrids with some non-negligible probability ϵ, we will come up with a reduction $\mathcal{A}$ that can break the semantic security of the TAHE scheme.

$\mathcal{A}$ interacts with a challenger $\mathcal{C}$ in the semantic security game for TAHE and with the environment $\mathcal{Z}$ in the game between Hybrid_3 and Hybrid_4. $\mathcal{A}$ corrupts the same set of parties as done by $\mathcal{Z}$ in its game with $\mathcal{C}$. Further, $\mathcal{A}$ forwards the public key-secret key pairs $(\mathsf{pk}_i, \mathsf{sk}_i)$ for the corrupt parties it receives from $\mathcal{Z}$ to the challenger and the public keys pk_i for the honest parties from $\mathcal{C}$ to $\mathcal{Z}$. $\mathcal{A}$ also forwards the pair of 0 and the set of honestly generated plaintexts to be encrypted, to the challenger and receives back a ciphertext for each of them which it uses in its interaction with $\mathcal{Z}$. It continues interacting with $\mathcal{Z}$ as in Hybrid_3 in the rest of its interaction. It is easy to see that if $\mathcal{C}$ sent honestly generated ciphertexts, the interaction between $\mathcal{A}$ and $\mathcal{Z}$ exactly corresponds to Hybrid_3 and if the ciphertexts were generated as encryptions of 0, the interaction between $\mathcal{A}$ and $\mathcal{Z}$ exactly corresponds to Hybrid_4. Thus, if $\mathcal{Z}$ can distinguish between the two hybrids with non-negligible probability ϵ, $\mathcal{A}$ can break the semantic security of the TAHE scheme with the same probability ϵ which is a contradiction. □

8 Threshold PSI for Functionality $\mathcal{F}_{\mathsf{TPSI\text{-}int}}$

In this section, we give a multiparty threshold PSI protocol for the functionality $\mathcal{F}_{\mathsf{TPSI\text{-}int}}$. Recall that in a multi-party threshold PSI protocol for functionality $\mathcal{F}_{\mathsf{TPSI\text{-}int}}$ defined in Figure 1, each party P_i wishes to learn the intersection of all the sets if $|S_i \setminus I| \leq T$, that is, if the size of its own set minus the intersection is less than the threshold T. Our protocol is almost identical to the protocol from Section 7 for functionality $\mathcal{F}_{\mathsf{TPSI\text{-}diff}}$ with the only difference being in the first step of the protocol, we run the multiparty private intersection cardinality testing protocol for functionality $\mathcal{F}_{\mathsf{TPSI\text{-}int}}$ instead of $\mathcal{F}_{\mathsf{TPSI\text{-}diff}}$. The rest of the protocol is the same. We elaborate on the details here for completeness.

As before, we formally prove the following theorem:

Theorem 8.1. *Assuming the existence of threshold additive homomorphic encryption, protocol* $\Pi_{\mathsf{TPSI\text{-}int}}$ *(Figure 11) securely realizes* $\mathcal{F}_{\mathsf{TPSI\text{-}int}}$ *in the* $\mathcal{F}_{\mathsf{CTest\text{-}int}}$*-hybrid model in the star network communication model. Our protocol is secure against a semi-honest adversary that can corrupt up to* $(n-1)$ *parties.*

The protocol runs in a constant number of rounds and the communication complexity is $O(n \cdot T \cdot \mathsf{poly}(\lambda))$ in the $\mathcal{F}_{\mathsf{CTest\text{-}int}}$-hybrid model. We then instantiate the $\mathcal{F}_{\mathsf{CTest\text{-}int}}$-hybrid with the protocol based on TFHE from Section 5.1 that has constant round complexity and $O(n \cdot T \cdot \mathsf{poly}(\lambda))$ communication complexity. Formally, we get the following corollary:

Corollary 8.2. *Assuming TFHE with distributed setup, protocol* $\Pi_{\mathsf{TPSI\text{-}int}}$ *(Figure 11) securely realizes* $\mathcal{F}_{\mathsf{TPSI\text{-}diff}}$ *in the star network communication model with communication complexity* $O(n \cdot T \cdot \mathsf{poly}(\lambda))$.

8.1 Protocol

Our protocol, which is almost identical to the protocol from Section 7 is described below. The change is highlighted in red.

Correctness. If $|S_i \setminus I| > T$, then the protocol terminates after the first step - the private intersection cardinality testing. Note that since $|S_i|$ is the same for all i, the protocol either terminates for all the parties or proceeds for all the parties. If, on the other hand, $|S_i \setminus I| \leq T$, the rest of the correctness analysis is identical to the one performed for the protocol in Section 7.

Communication Complexity. The communication complexity analysis is identical to the one performed for the protocol in Section 7.

Security Proof The proof is identical to the proof of Theorem 7.1 from Section 7.

9 Other Communication Models

Throughout the paper we focus on the communication lower bounds in point-to-point networks and design protocols in the star network (which can be implemented on point-to-point network). In this section, we initiate the study of multiparty threshold PSI on networks with broadcast channels. We

Parameters: Parties $P_1, \ldots, P_n$. Each party has a set of m elements. Threshold $T \in \mathbb{N}$. $\mathbb{F}$ is a finite field where $|\mathbb{F}| = \Omega(2^\lambda)$.

Inputs: Party P_i has an input set $S_i = \{a_1^i, \ldots, a_m^i\}$ where $a_j^i \in \mathbb{F}$ for all $j \in [m]$.

Output: Each party P_i receives the set intersection $I = \bigcap_{i=1}^n S_i$ if and only if $|S_i \setminus I| \leq T$.

Protocol:

1. **Private Intersection Cardinality Testing.** The parties invoke $\mathcal{F}_{\mathsf{CTest\text{-}diff}}$ on inputs $S_1, \ldots, S_n$ and receive back $w \in \{\mathsf{similar}, \mathsf{different}\}$. If $w = \mathsf{different}$ then all parties output $\perp$.
2. **TAHE Key Generation.** Each party P_i generates $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{TAHE.DistSetup}(1^\lambda, i)$ and sends pk_i to P_1. Then P_1 sends $\mathsf{pk} = (\mathsf{pk}_1 \| \ldots \| \mathsf{pk}_n)$ to all the other parties.
3. **Evaluations of Random Polynomial.** In this phase the parties will evaluate a polynomial
$$\mathsf{V}(x) = \sum_{i=1}^{n} \Big(\mathsf{p}_i'(x) \cdot \Big(\mathsf{R}_1(x) + \ldots + \mathsf{R}_{i-1}(x) + \widetilde{\mathsf{R}}_i(x) + \mathsf{R}_{i+1}(x) \ldots + \mathsf{R}_n(x) \Big) \Big)$$
for $x \in [3T+4]$ where the terms are defined as follows.
   (a) Each party P_i defines $\mathsf{p}_i(x) = \prod_{j=1}^m (x - a_j^i)$ and $\mathsf{p}_i'(x) = \mathsf{p}_i(x) \cdot (x - r_i)$ where $r_i \xleftarrow{\$} \mathbb{F}$.
   (b) Each party P_i uniformly samples $\mathsf{R}_i, \widetilde{\mathsf{R}}_i \xleftarrow{\$} \mathbb{F}[x]$ of degree $T+1$, computes $\mathsf{R}_i(x)$ for $x \in [3T+4]$ and sends encrypted $[\![\mathsf{R}_i(x)]\!]$ to P_1.
   (c) For each $i \in [n], x \in [3T+4]$, party P_1 sends $[\![e_{i,x}]\!] = \Big[\!\Big[\sum_{j \in [n] \setminus \{i\}} \mathsf{R}_j(x)\Big]\!\Big]$ to P_i.
   (d) For each $x \in [3T+4]$, each party P_i sends $[\![v_{i,x}]\!] = \Big[\!\Big[\mathsf{p}_i'(x) \cdot \Big(e_{i,x} + \widetilde{\mathsf{R}}_i(x)\Big)\Big]\!\Big]$ to P_1.
   (e) For each $x \in [3T+4]$, P_1 sends $[\![v_x]\!] = [\![\sum_{i=1}^n v_{i,x}]\!]$ to all P_i.
   (f) For each $x \in [3T+4]$, each party P_i sends $[\![v_x : \mathsf{sk}_i]\!] \leftarrow \mathsf{TAHE.PartialDec}(\mathsf{sk}_i, [\![v_x]\!])$ to P_1.
   (g) For each $x \in [3T+4]$, P_1 sends $\mathsf{V}(x) = \mathsf{TAHE.Combine}(\mathsf{pk}, \{[\![v_x : \mathsf{sk}_i]\!]\}_{i \in [n]})$ to all P_i.
4. **Computing Set Intersection.** Each party P_i does the following:
   (a) Interpolate $\mathsf{q}_i(x)$ to be the degree $3T+3$ rational polynomial such that $\mathsf{q}_i(x) = \frac{\mathsf{V}(x)}{\mathsf{p}_i'(x)}$ for $x \in [3T+4]$ and the gcd of the numerator and denominator is 1. Let D_i be the roots of the denominator of $\mathsf{q}_i(x)$.
   (b) Output the set intersection $I = S_i \setminus D_i$.

Figure 11: Multi-party threshold PSI protocol $\Pi_{\mathsf{TPSI\text{-}int}}$ for functionality $\mathcal{F}_{\mathsf{TPSI\text{-}int}}$.

give a new lower bound for the functionality $\mathcal{F}_{\mathsf{TPSI\text{-}int}}$ in the broadcast model of communication. Note that the lower bound of $\Omega(n \cdot T)$ from Section 4 for the point-to-point network does not necessarily apply to the broadcast setting.

At a high level, we reduce the problem of multiparty set disjointness to multiparty threshold PSI for the ideal functionality $\mathcal{F}_{\mathsf{TPSI\text{-}int}}$. In the multiparty set disjointness problem, there are n parties each holding a set $X_i \subseteq [N]$. The parties' task is to determine if $\bigcap_{i=1}^n X_i = \emptyset$. It is shown by Braverman and Oshman [BO15] that the communication complexity of multiparty set disjointness in the broadcast model is $\Omega(N \log n + n)$. Given this result, we prove the communication lower bound for multiparty threshold PSI:

Theorem 9.1. *For any multiparty threshold private set intersection protocol for functionality*

$\mathcal{F}_{\mathsf{TPSI\text{-}int}}$*, the communication complexity in the broadcast model is* $\Omega(T \log n + n)$ *where* n *is the number of parties and* T *is the threshold parameter.*

Proof. We prove the theorem by giving a reduction from multiparty set disjointness to multiparty threshold PSI. Assume there is a multiparty threshold PSI protocol $\Pi_{\mathsf{TPSI\text{-}int}}$ that achieves communication complexity $o(T \log n + n)$ in the broadcast model, then we can solve multiparty set disjointness with communication complexity $o(N \log n + n)$, which leads to a contradiction. We show the reduction in the following.

Given an instance of multiparty set disjointness where there are n parties each holding a set $X_i \subseteq [N]$, we construct an instance of multiparty threshold PSI as follows. There are n parties. Each party P_i has a set of size N. We set the sets as $S_i := X_i \cup U_i$, where $|S_i| = N$, and U_i consists of unique elements that can only appear in S_i. Notice that $\bigcap_{i=1}^{n} X_i \neq \emptyset$ if and only if $\bigcap_{i=1}^{n} S_i \neq \emptyset$, namely $|S_i - \bigcap_{i=1}^{n} S_i| \leq (N-1)$. The n parties in the multiparty set disjointness problem run the multiparty threshold PSI protocol $\Pi_{\mathsf{TPSI\text{-}int}}$ where each party P_i inputs set S_i and the threshold is set to be $T := N-1$. If $\Pi_{\mathsf{TPSI\text{-}int}}$ outputs the set intersection, then the parties output $\bigcap_{i=1}^{n} X_i \neq \emptyset$; otherwise the parties output $\bigcap_{i=1}^{n} X_i = \emptyset$. □

We leave further exploration in the broadcast model including a lower bound for the other functionality $\mathcal{F}_{\mathsf{TPSI\text{-}diff}}$ as well as designing more efficient protocols as interesting open problems.

References

[Bea91] Donald Beaver. Efficient multiparty protocols using circuit randomization. In *CRYPTO*, 1991.

[Ben94] Josh Benaloh. Dense probabilistic encryption. May 1994.

[BFK+19] Saikrishna Badrinarayanan, Rex Fernando, Venkata Koppula, Amit Sahai, and Brent Waters. Output compression, mpc, and io for turing machines. In *ASIACRYPT*, 2019.

[BGG+18] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In *CRYPTO*, 2018.

[BJMS18] Saikrishna Badrinarayanan, Aayush Jain, Nathan Manohar, and Amit Sahai. Threshold multi-key fhe and applications to round-optimal mpc. *IACR Cryptology ePrint Archive*, 2018:580, 2018.

[BO15] Mark Braverman and Rotem Oshman. On information complexity in the broadcast model. In *PODC*, 2015.

[BPSW07] Justin Brickell, Donald E Porter, Vitaly Shmatikov, and Emmett Witchel. Privacy-preserving remote diagnostics. In *CCS*, 2007.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, 2001.

[CDN01] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. Multiparty computation from threshold homomorphic encryption. In *EUROCRYPT*, 2001.

[DCT10] Emiliano De Cristofaro and Gene Tsudik. Practical private set intersection protocols with linear complexity. In *FC*, 2010.

[DCW13] Changyu Dong, Liqun Chen, and Zikai Wen. When private set intersection meets big data: an efficient and scalable protocol. In *CCS*, 2013.

[DIK$^+$08] Ivan Damgård, Yuval Ishai, Mikkel Krøigaard, Jesper Buus Nielsen, and Adam D. Smith. Scalable multiparty computation with nearly optimal work and resilience. In *CRYPTO*, 2008.

[FH96] Matthew K. Franklin and Stuart Haber. Joint encryption and message-efficient secure computation. *J. Cryptology*, 1996.

[FNP04] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *EUROCRYPT*, 2004.

[Gam84] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO*, 1984.

[GJR10] Elena Grigorescu, Kyomin Jung, and Ronitt Rubinfeld. A local decision test for sparse polynomials. *Information Processing Letters*, 2010.

[GN19] Satrajit Ghosh and Tobias Nilges. An algebraic approach to maliciously secure private set intersection. In *EUROCRYPT*, 2019.

[GS19a] Satrajit Ghosh and Mark Simkin. The communication complexity of threshold private set intersection. In *CRYPTO*, 2019.

[GS19b] Satrajit Ghosh and Mark Simkin. The communication complexity of threshold private set intersection, 2019. ia.cr/2019/175.

[HFH99] Bernardo A. Huberman, Matt Franklin, and Tad Hogg. Enhancing privacy and trust in electronic communities. In *In Proc. of the 1st ACM Conference on Electronic Commerce*, 1999.

[HN10] Carmit Hazay and Kobbi Nissim. Efficient set operations in the presence of malicious adversaries. In *PKC*, 2010.

[HOS17] Per A. Hallgren, Claudio Orlandi, and Andrei Sabelfeld. Privatepool: Privacy-preserving ridesharing. In *CSF*, 2017.

[HV17] Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. Scalable multi-party private set-intersection. In Serge Fehr, editor, *PKC*, 2017.

[HW15] Pavel Hubáček and Daniel Wichs. On the communication complexity of secure function evaluation with long output. In *ITCS*, 2015.

[IKN$^+$17] Mihaela Ion, Ben Kreuter, Erhan Nergiz, Sarvar Patel, Shobhit Saxena, Karn Seth, David Shanahan, and Moti Yung. Private intersection-sum protocol with applications to attributing aggregate ad conversions. 2017. ia.cr/2017/735.

[KKRT16] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. Efficient batched oblivious PRF with applications to private set intersection. In *CCS*, 2016.

[KMP$^+$17] Vladimir Kolesnikov, Naor Matania, Benny Pinkas, Mike Rosulek, and Ni Trieu. Practical multi-party private set intersection from symmetric-key techniques. In *CCS*, 2017.

[KMWF07] Eike Kiltz, Payman Mohassel, Enav Weinreb, and Matthew Franklin. Secure linear algebra using linearly recurrent sequences. In *TCC*, 2007.

[KS05] Lea Kissner and Dawn Song. Privacy-preserving set operations. In *CRYPTO*, 2005.

[Mea86] C. Meadows. A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In *IEEE S and P*, 1986.

[MTZ03] Yaron Minsky, Ari Trachtenberg, and Richard Zippel. Set reconciliation with nearly optimal communication complexity. *IEEE Transactions on Information Theory*, 2003.

[MZ17] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *IEEE S and P*, 2017.

[NMH$^+$10] Shishir Nagaraja, Prateek Mittal, Chi-Yao Hong, Matthew Caesar, and Nikita Borisov. Botgrep: Finding p2p bots with structured graph analysis. In *USENIX security symposium*, 2010.

[OOS16] Michele Orrù, Emmanuela Orsini, and Peter Scholl. Actively secure 1-out-of-n OT extension with application to private set intersection. In *CT-RSA*, 2016.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *EUROCRYPT*, 1999.

[PRTY19] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. Spot-light: Lightweight private set intersection from sparse ot extension. In *CRYPTO*, 2019.

[PRTY20] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. PSI from paxos: Fast, malicious private set intersection. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT*, 2020.

[PSSZ15] Benny Pinkas, Thomas Schneider, Gil Segev, and Michael Zohner. Phasing: Private set intersection using permutation-based hashing. In *USENIX*, 2015.

[PSWW18] Benny Pinkas, Thomas Schneider, Christian Weinert, and Udi Wieder. Efficient circuit-based PSI via cuckoo hashing. In *EUROCRYPT*, 2018.

[PSZ14] Benny Pinkas, Thomas Schneider, and Michael Zohner. Faster private set intersection based on ot extension. In *USENIX*, 2014.

[RR17a] Peter Rindal and Mike Rosulek. Improved private set intersection against malicious adversaries. In *EUROCRYPT*, 2017.

[RR17b] Peter Rindal and Mike Rosulek. Malicious-secure private set intersection via dual execution. In *CCS*, 2017.

[TPKC07] Juan Ramón Troncoso-Pastoriza, Stefan Katzenbeisser, and Mehmet Celik. Privacy preserving error resilient dna searching through oblivious automata. In *CCS*, 2007.

[VZGG13] Joachim Von Zur Gathen and Jürgen Gerhard. *Modern computer algebra*. Cambridge university press, 2013.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, 1986.

[ZC18] Yongjun Zhao and Sherman SM Chow. Can you find the one for me? privacy-preserving matchmaking via threshold psi. 2018. ia.cr/2018/184.

A Linear Algebra Proofs

A.1 Proof of Lemma 3.3

This lemma is a generalization of Lemma 2 in the work of Kissner and Song [KS05]. They proved the lemma for two polynomials and we generalize it to multiple polynomials. We first extend [KS05, Lemma 2] for two polynomials to the case where p_1 and p_2 do not necessarily have the same degree. Formally, we first prove the following lemma.

Lemma A.1. *Let $\mathbb{F}$ be a finite field of prime order q. Fix any $n = O(\mathsf{poly}(\lambda))$. For any two polynomials $p_1(x), p_2(x) \in \mathbb{F}[x]$ of degrees α_1 and α_2 respectively, if $R_1(x) = \sum_{j=0}^{\beta_1} r_{1,j}x^j, R_2(x) = \sum_{j=0}^{\beta_2} r_{2,j}x^j \in \mathbb{F}[x]$ where $\alpha_1 + \beta_1 = \alpha_2 + \beta_2 \geq \alpha_1 + \alpha_2$ and $r_{i,j} \xleftarrow{\$} \mathbb{F}$ are sampled independently and uniformly at random. Let $S(x) = p_1(x) \cdot R_1(x) + p_2(x) \cdot R_2(x)$. Then $S(x) = \gcd(p_1(x), p_2(x)) \cdot U(x)$, where $U(x) = \sum_{j=0}^{\alpha_1+\beta_1-\gamma} u_j x^j$, in which γ is the degree of $\gcd(p_1(x), p_2(x))$, u_j's are distributed uniformly and independently over $\mathbb{F}$.*

Proof. Let $g = \gcd(p_1, p_2)$, and let $p_1(x) = g(x) \cdot q_1(x), p_2(x) = g(x) \cdot q_2(x)$. We know that g has degree γ, hence q_1, q_2 has degrees $\alpha_1 - \gamma$ and $\alpha_2 - \gamma$, respectively. In addition, $\gcd(q_1, q_2) = 1$. Since

$$S(x) = p_1(x) \cdot R_1(x) + p_2(x) \cdot R_2(x) = g(x) \cdot (q_1(x) \cdot R_1(x) + q_2(x) \cdot R_2(x)),$$

we only need to show that $q_1(x) \cdot R_1(x) + q_2(x) \cdot R_2(x) = \sum_{j=0}^{\alpha_1+\beta_1-\gamma} u_j x^j$ where u_j's are distributed uniformly and independently over $\mathbb{F}$.

Given any fixed polynomial U of degree $\alpha_1 + \beta_1 - \gamma$, we calculate the number of (R_1, R_2) pairs such that $q_1 \cdot R_1 + q_2 \cdot R_2 = U$. Let us assume for this particular U there exists at least one pair of (R_1, R_2) such that $q_1 \cdot R_1 + q_2 \cdot R_2 = U$. Then for any other satisfying pair (R_1', R_2'), we have that

$$q_1 \cdot (R_1 - R_1') + q_2 \cdot (R_2 - R_2') = 0,$$
$$q_1 \cdot (R_1 - R_1') = q_2 \cdot (R_2' - R_2).$$

Since $\gcd(q_1, q_2) = 1$ and $\mathbb{F}$ is a finite field of prime order, we have $q_2 | (R_1 - R_1')$ and $q_1 | (R_2' - R_2)$. Let $R_1 - R_1' = q_2 \cdot h$, then $R_2' - R_2 = q_1 \cdot h$, where h is a polynomial with degree

$$\begin{aligned} d =& \deg(U) - \deg(q_1) - \deg(q_2) \\ =&(\alpha_1 + \beta_1 - \gamma) - (\alpha_1 - \gamma) - (\alpha_2 - \gamma) \\ =&\beta_1 - \alpha_2 + \gamma. \end{aligned}$$

Every pair of satisfying (R_1', R_2') decides a unique polynomial h and every polynomial h corresponds to a unique satisfying pair (R_1', R_2'). Hence the total number of (R_1', R_2') pairs equals the total number of degree-d polynomial h in $\mathbb{F}[x]$, which is $q^{\beta_1-\alpha_2+\gamma+1}$.

Now consider the polynomial $(q_1 \cdot R_1 + q_2 \cdot R_2)$. We know that for any fixed polynomial U as a possible result, there are $q^{\beta_1-\alpha_2+\gamma+1}$ pairs of (R_1, R_2) that lead to the result. Since there are a total number of $q^{\beta_1+\beta_2+2}$ possible pairs of (R_1, R_2), the total number of possible resulting U is

$$\frac{q^{\beta_1+\beta_2+2}}{q^{\beta_1-\alpha_2+\gamma+1}} = q^{\alpha_2+\beta_2-\gamma+1} = q^{\alpha_1+\beta_1-\gamma+1},$$

which is exactly the total number of U. Therefore, for randomly generated R_1, R_2, each possible polynomial with degree $\alpha_1 + \beta_1 - \gamma$ will be the result with equal probability. □

Given Lemma A.1,

$$\begin{aligned}
&p_1(x) \cdot R_1(x) + p_2(x) \cdot R_2(x) \cdots + p_n(x) \cdot R_n(x) \\
=&\gcd(p_1(x), p_2(x)) \cdot U_2(x) + p_3(x) \cdot R_3(x) \cdots + p_n(x) \cdot R_n(x) \\
=&\gcd(p_1(x), p_2(x), p_3(x)) \cdot U_3(x) + p_4(x) \cdot R_4(x) \cdots + p_n(x) \cdot R_n(x) \\
=&\ldots \\
=&\gcd(p_1(x), \ldots, p_n) \cdot U_n(x),
\end{aligned}$$

where $U_i(x) = \sum_{j=0}^{\alpha+\beta-\gamma_i} u_{i,j} x^j$, in which γ_i is the degree of $\gcd(p_1(x), \ldots, p_i(x))$, $u_{i,j}$'s are distributed uniformly and independently over $\mathbb{F}$. Since $\gcd(p_1, \ldots, p_n) = 1$, we have $\gamma_n = 0$. Hence $\sum_{i=1}^{n}(p_i(x) \cdot R_i(x)) = \sum_{j=0}^{\alpha+\beta} u_j x^j$, then u_j's are distributed uniformly and independently over $\mathbb{F}$.

A.2 Proof of Lemma 3.2

Fix any i such that $1 \le i < n$. Let's denote $p_i^*(x) := p'_1(x) + ... + p'_i(x)$. Observe that $\gcd(p_i^*, p'_{i+1}, \ldots, p'_n) = \gcd(p_i^*, \gcd(p'_{i+1}, \ldots, p'_n)) = \gcd(p_i^*, \gcd(p_{i+1}, \ldots, p_n))$ since the probability that for $j > i$, r_j is a root of p'_k, where $k > i, k \neq j$ is negligible.

Consider any root v of $\gcd(p_{i+1}, \ldots, p_n)$. We now analyze the event: $\Pr_{r_j}[p_i^*(v) = 0]$. First, note that since $\gcd(p_1, ..., p_n) = 1$, there exists $k \le i$ s.t. $(x - v) \nmid p_k(x)$. Further, since r_k is picked uniformly at random, $\Pr_{r_k}[r_k = v] \le \mathsf{negl}(\lambda)$. Therefore, except with negligible probability, there exists $k <= i$ s.t. $(x - v) \nmid p'_k(x)$.

As such it must hold that

$$\begin{gathered}
p'_1(v) + ... + p'_{k-1}(v) + p'_{k+1}(v) + ... + p'_i(v) = -p'_k(v), \\
\frac{p'_1(v) + ... + p'_{k-1}(v) + p'_{k+1}(v) + ... + p'_i(v)}{p_k(v)} + v = r_k.
\end{gathered}$$

Since $r_k \in \mathbb{F}$ is picked uniformly at random, the probability of this event is $1/q = \mathsf{negl}(\lambda)$. Taking a union bound over all the roots of $\gcd(p_{i+1}, \ldots, p_n)$ yields

$$\Pr_{r_j}[\gcd(p_i^*, \gcd(p_{i+1}, \ldots, p_n)) \neq 1] \le \mathsf{negl}(\lambda)$$

and this completes the proof.