Black-Box Constant-Round Secure 2PC with Succinct Communication

Michele Ciampi[1], Ankit Kumar Misra[2], Rafail Ostrovsky[2], and Akash Shah[2]

[1] University of Edinburgh, Scotland
[2] University of California Los Angeles, USA

Abstract. The most fundamental performance metrics of secure multi-party computation (MPC) protocols are related to the number of messages the parties exchange (i.e., round complexity), the size of these messages (i.e., communication complexity), and the overall computational resources required to execute the protocol (i.e., computational complexity). Another quality metric of MPC protocols is related to the *black-box* or *non-black-box* use of the underlying cryptographic primitives. Indeed, the design of black-box MPC protocols, other than being of theoretical interest, usually can lead to protocols that have better computational complexity.
In this work, we aim to optimize the round and communication complexity of *black-box* secure multi-party computation in the plain model, by designing a constant-round two-party computation protocol in the malicious setting, whose communication complexity is only poly-logarithmic in the size of the function being evaluated. We successfully design such a protocol, having only *black-box* access to fully homomorphic encryption, trapdoor permutations, and hash functions. To the best of our knowledge, our protocol is the first to make black-box use of standard cryptographic primitives while achieving almost asymptotically optimal communication and round complexity.

1 Introduction

Secure multiparty computation (MPC) [Yao86, GMW87] enables multiple parties to jointly compute a function over their inputs while keeping those inputs private. In particular, a multi-party computation protocol ensures that none of the parties learn anything beyond what can already be inferred from their own inputs and the output of the computation. Since its conception, a prolific area of research has studied the notion of MPC, in particular with the aim of minimizing the computational and communication overhead that it incurs in *securely* evaluating a function. Indeed, in a model without security requirements, the parties can simply exchange their inputs and locally evaluate the function, incurring in what is arguably the minimal communication and computational cost. In particular, the size of the exchanged messages is independent of the complexity of the function/circuit being evaluated.

This area of research has as such seen significant progress, proposing protocols with better round, communication, and computational complexity [GMW87, Kil88, IPS08, BMR90, KOS03, KO04, Pas04, PW10, Wee10, Goy11, COSV17, CCG$^+$21, GMPP16]. In particular, for the case of a malicious adversary that can corrupt a majority of parties[3] we now have quite a clear picture of the (asymptotic) costs of multi-party computation. Notably, either assuming a setup or in the plain model, we have protocols that provably require a minimal number of interactions from standard polynomial-time assumptions [GS18, BL18, CCG$^+$20]. Moreover, we have a round-optimal protocol that sends a minimal amount of information overall [COWZ22, QWW18, MPP20, ABJ$^+$19]. In particular, the communication complexity of these protocols is almost independent (poly-logarithmic) from the size of the circuit being evaluated, or dependent only on its depth. The downside of all these

[3] A malicious adversary attacks the protocol following an arbitrary probabilistic polynomial-time strategy. Unless stated differently, when we talk about the security of an MPC protocol against malicious adversaries we assume that up to $n-1$ parties can be corrupted, where n is the number of parties.

protocols, is that they made non-black-box use of the underlying cryptographic primitives. In this paper, we will focus on protocols that make only *black-box* and that are *succinct*. Following [MPP20], by succinct we mean that the communication complexity depends only on the length of the inputs and outputs[4], and poly-logarithmically on the size of the circuit representing the function f that needs to be computed.

Malicious security and succinctness. A common method of obtaining a succinct maliciously secure protocol is to first design a succinct protocol that is secure only against a weaker form of adversarial behaviors, like semi-honest or semi-malicious[5], and then convert it into a maliciously secure protocol relying on *zero-knowledge* [GMR85]. In this new protocol, honest behavior is enforced by requiring each party to send, along with the MPC protocol message, a zero-knowledge proof attesting that the MPC message has been generated honestly. While this approach works, it inherently makes *non-black-box* use of the succinct semi-honest (or semi-malicious) protocol and, consequently, non-black-box of the primitives underlying it. This occurs even when the initial protocol does make black-box use of cryptographic primitives.

Black-box secure computation. Protocols that utilize the underlying primitives in a non-black-box manner generally incur a considerable efficiency overhead. Therefore, it is intriguing from both theoretical and practical angles to explore whether black-box protocols can be constructed. Such protocols would rely solely on oracle access to the input-output behavior of the underlying cryptographic primitives, without requiring any detailed or explicit representation of the primitives themselves. In this paper, we focus on protocols proven secure in the plain model and with respect to black-box simulation.[6]

Interestingly, recent results have shown how to obtain protocols with very low round complexity (even optimal in some cases), while making black-box use of standard polynomial-time assumptions, both in the plain model [IPS08, Wee10, Goy11, IKSS21, IKSS23] and assuming setup [IKSS22a, GIS18, IKSS22b, PS21, ABG$^+$20]. Notably, for the two-party case [ORS15] proposes a round-optimal (5-round) black-box protocol, and in [IKSS23] the authors extend this result to the multi-party setting. To the best of our knowledge, none of these prior works that make black-box use of cryptographic primitives (either with optimal or constant round complexity) are also succinct. As such, in this work, we ask the following natural question.

> *Is there a round-optimal* succinct *MPC protocol, maliciously secure against a dishonest majority in the* plain model*, that makes black-box use of standard cryptographic assumptions?*

1.1 Our Contributions

We give a partial answer to the above question, by designing the first constant-round *succinct* two-party protocol that makes only black-box use of fully homomorphic encryption. In particular, we prove the following claim:

> **Theorem** *(informal). Assuming black-box access to fully homomorphic encryption,* non-succinct *black-box constant-round 2PC, and collision-resistant hash functions, there exists a* succinct constant-round *2PC protocol that securely computes any function f, in the presence of malicious adversaries.*

[4] We recall that the dependence on the output size instead is inherent for any malicious-secure protocol [HW15].

[5] A semi-malicious adversary acts like a semi-honest adversary, but it can pick arbitrary randomness to execute the protocol.

[6] From now on, to avoid confusion, we will not specify anymore that our results are with respect to black-box simulation, and only use the terms *black-box* and *non-black-box* when referring to the use of cryptographic primitives.

Using the results of [ORS15], we can obtain a 5-round (which is optimal due to to [KO04]) 2PC protocol that makes black-box use of trapdoor permutations, hence we obtain the following corollary.

Corollary *(informal). Assuming black-box access to fully homomorphic encryption, trapdoor-permutations, and collision-resistant hash functions, there exists a* succinct 14-round *2PC protocol that securely computes any function f in the presence of malicious adversaries.*

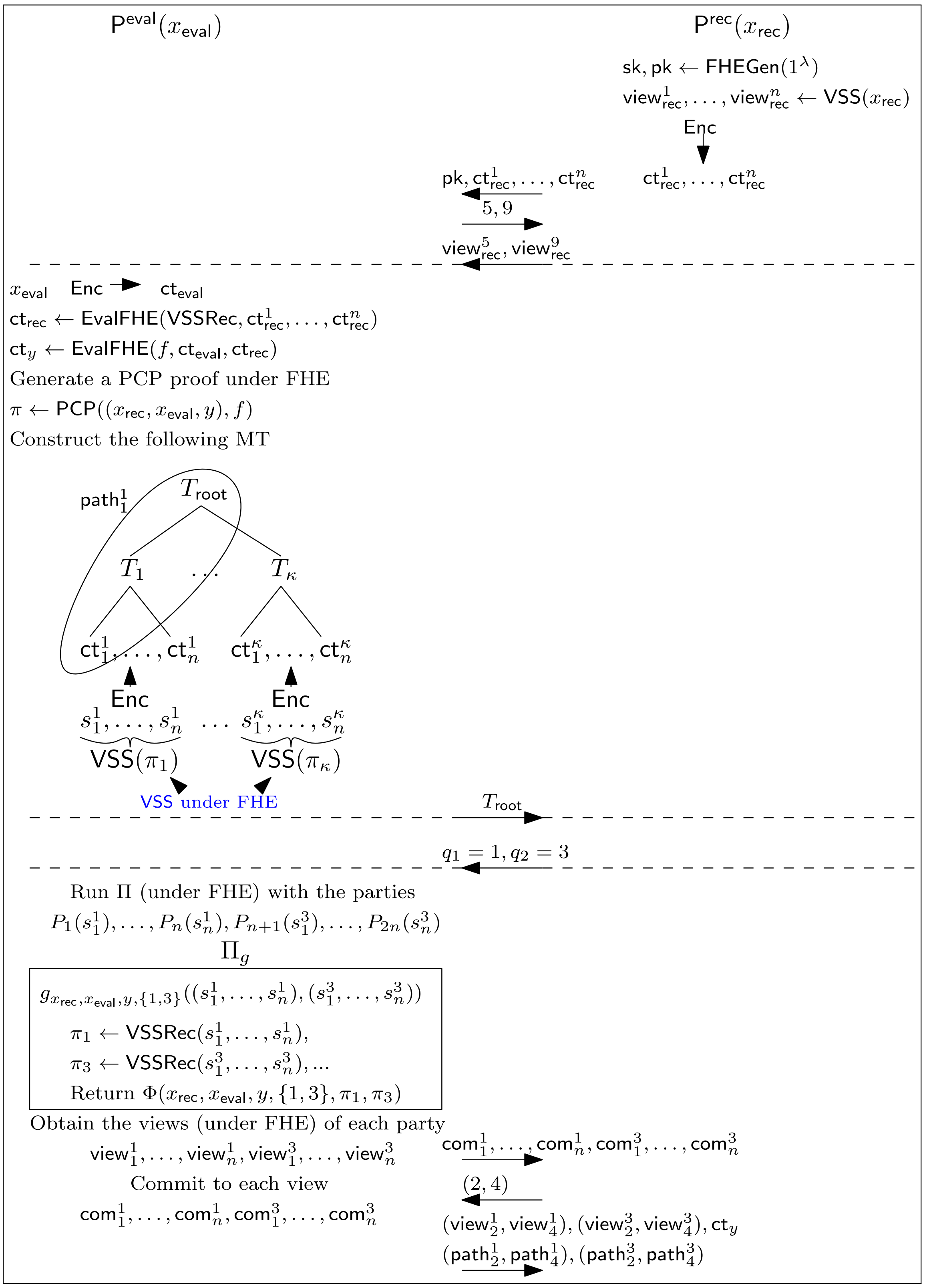
It remains an interesting open question to understand whether it is possible to match the round complexity of existing non-succinct black-box protocols and to extend our result to the multi-party setting.

1.2 Technical Overview

We start by recalling the folklore approach to designing a succinct two-party computation protocol using fully homomorphic encryption (FHE) [LTV12] for the *semi-honest* case. For simplicity, we consider the setting where only one party gets the output (a *receiver*) and denote this party with $\mathsf{P}^{\mathsf{rec}}$. The other party $\mathsf{P}^{\mathsf{eval}}$ (the *evaluator*) just provides its input and helps evaluate the function f. $\mathsf{P}^{\mathsf{rec}}$, on input x_{rec}, generates the public-secret key pair $(\mathtt{pk}, \mathtt{sk})$ of an FHE scheme, encrypts x_{rec} thus obtaining $\mathsf{ct}_{\mathsf{rec}}$, and sends $(\mathtt{pk}, \mathsf{ct}_{\mathsf{rec}})$ to $\mathsf{P}^{\mathsf{eval}}$. The evaluator encrypts its input x_{eval} using $\mathtt{pk}$, and homomorphically evaluates f under FHE using ciphertexts that contain the inputs of the two parties, thus obtaining a new ciphertext ct_y which contains $y \leftarrow f(x_{\mathsf{rec}}, x_{\mathsf{eval}})$. $\mathsf{P}^{\mathsf{eval}}$ then sends ct_y to $\mathsf{P}^{\mathsf{rec}}$, who can use the secret key $\mathtt{sk}$ to finally obtain y. Turning this protocol into one that is secure against malicious adversaries and makes only black-box use of the FHE scheme faces multiple challenges. Let us start by considering the case where $\mathsf{P}^{\mathsf{rec}}$ is malicious (denoted by $\mathsf{P}^{\mathsf{rec}\star}$).

Security against malicious $\mathsf{P}^{\mathsf{rec}}$. The adversarial $\mathsf{P}^{\mathsf{rec}}$ could generate the public key and the ciphertext of the FHE scheme in such a way that, for example, no matter what FHE evaluation $\mathsf{P}^{\mathsf{eval}}$ performs, ct_y would always contain the input of the evaluator x_{Eval}. Luckily, this problem can be easily solved using an FHE scheme that prevents exactly this type of attack. Such a scheme is called *circuit private*, and is provided in [DD22], where it is constructed by making black-box use of standard FHE. This, however, does not solve all our problems, as to properly simulate the behavior of a corrupted receiver, we need to be able to extract its input. To enable that, we modify the protocol as follows. $\mathsf{P}^{\mathsf{rec}}$ runs a verifiable secret sharing (VSS) procedure in its head using x_{rec} as the secret, thus obtaining a set of views (or shares). $\mathsf{P}^{\mathsf{rec}}$ then encrypts each share individually and sends these ciphertexts to $\mathsf{P}^{\mathsf{eval}}$. Now $\mathsf{P}^{\mathsf{rec}}$ engages in a cut-and-choose with $\mathsf{P}^{\mathsf{eval}}$, who asks $\mathsf{P}^{\mathsf{rec}}$ to *open* (i.e., to see the plaintext and the randomness) a subset of these ciphertexts. $\mathsf{P}^{\mathsf{eval}}$ checks that the openings are correct and that the views of the VSS are consistent with each other. If this is the case, then $\mathsf{P}^{\mathsf{eval}}$ continues the execution of the protocol as described above. The simulator can now extract the input of a corrupted $\mathsf{P}^{\mathsf{rec}\star}$ by simply rewinding and asking to open different subsets of views at each rewind. The cut-and-choose guarantees that if $\mathsf{P}^{\mathsf{eval}}$ verification is successful, then sufficiently many views must be consistent with each other (even among the un-opened ciphertexts), hence, we can rely on the security of VSS to claim that there is a unique value committed. This allows us to claim that there is no ambiguity about what the simulator will extract. There are some subtleties that we have overlooked, like the fact that an FHE ciphertext may be non-committing. We solve this using quite standard techniques, and we refer to the technical part of the paper for more details.

Security against malicious $\mathsf{P}^{\mathsf{eval}}$. The *only* damage that a corrupted $\mathsf{P}^{\mathsf{eval}\star}$ can do is to evaluate a function different from f, hence forcing an incorrect output to $\mathsf{P}^{\mathsf{rec}}$. The standard way to solve this problem is by requiring $\mathsf{P}^{\mathsf{rec}}$ to issue a zero-knowledge proof that ct_y contains $y \leftarrow f(x_{\mathsf{rec}}, x_{\mathsf{eval}})$. Unfortunately, this approach leads to a non-black-box use of the underlying FHE scheme. Our high-level idea is to instead run a zero-knowledge proof inside FHE, thus proving that y was computed



$\mathsf{P}^{\mathsf{eval}}(x_{\mathsf{eval}})$
$\mathsf{P}^{\mathsf{rec}}(x_{\mathsf{rec}})$
$\mathsf{sk}, \mathsf{pk} \leftarrow \mathsf{FHEGen}(1^\lambda)$
$\mathsf{view}^1_{\mathsf{rec}}, \ldots, \mathsf{view}^n_{\mathsf{rec}} \leftarrow \mathsf{VSS}(x_{\mathsf{rec}})$
Enc
$\mathsf{ct}^1_{\mathsf{rec}}, \ldots, \mathsf{ct}^n_{\mathsf{rec}}$
$\mathsf{pk}, \mathsf{ct}^1_{\mathsf{rec}}, \ldots, \mathsf{ct}^n_{\mathsf{rec}}$
$5, 9$
$\mathsf{view}^5_{\mathsf{rec}}, \mathsf{view}^9_{\mathsf{rec}}$
x_{eval} Enc $\mathsf{ct}_{\mathsf{eval}}$
$\mathsf{ct}_{\mathsf{rec}} \leftarrow \mathsf{EvalFHE}(\mathsf{VSSRec}, \mathsf{ct}^1_{\mathsf{rec}}, \ldots, \mathsf{ct}^n_{\mathsf{rec}})$
$\mathsf{ct}_y \leftarrow \mathsf{EvalFHE}(f, \mathsf{ct}_{\mathsf{eval}}, \mathsf{ct}_{\mathsf{rec}})$
Generate a PCP proof under FHE
$\pi \leftarrow \mathsf{PCP}((x_{\mathsf{rec}}, x_{\mathsf{eval}}, y), f)$
Construct the following MT
path^1_1
T_{root}
T_1 $\ldots$ T_κ
$\mathsf{ct}^1_1, \ldots, \mathsf{ct}^1_n$
$\mathsf{ct}^\kappa_1, \ldots, \mathsf{ct}^\kappa_n$
Enc
Enc
$\underbrace{s^1_1, \ldots, s^1_n}_{\mathsf{VSS}(\pi_1)}$ $\ldots$ $\underbrace{s^\kappa_1, \ldots, s^\kappa_n}_{\mathsf{VSS}(\pi_\kappa)}$
VSS under FHE
T_{root}
$q_1 = 1, q_2 = 3$
Run Π (under FHE) with the parties
$P_1(s^1_1), \ldots, P_n(s^1_n), P_{n+1}(s^3_1), \ldots, P_{2n}(s^3_n)$
Π_g
$g_{x_{\mathsf{rec}}, x_{\mathsf{eval}}, y, \{1,3\}}((s^1_1, \ldots, s^1_n), (s^3_1, \ldots, s^3_n))$
$\pi_1 \leftarrow \mathsf{VSSRec}(s^1_1, \ldots, s^1_n)$,
$\pi_3 \leftarrow \mathsf{VSSRec}(s^3_1, \ldots, s^3_n)$, ...
Return $\Phi(x_{\mathsf{rec}}, x_{\mathsf{eval}}, y, \{1,3\}, \pi_1, \pi_3)$
Obtain the views (under FHE) of each party
$\mathsf{view}^1_1, \ldots, \mathsf{view}^1_n, \mathsf{view}^3_1, \ldots, \mathsf{view}^3_n$
Commit to each view
$\mathsf{com}^1_1, \ldots, \mathsf{com}^1_n, \mathsf{com}^3_1, \ldots, \mathsf{com}^3_n$
$\mathsf{com}^1_1, \ldots, \mathsf{com}^1_n, \mathsf{com}^3_1, \ldots, \mathsf{com}^3_n$
$(2, 4)$
$(\mathsf{view}^1_2, \mathsf{view}^1_4), (\mathsf{view}^3_2, \mathsf{view}^3_4), \mathsf{ct}_y$
$(\mathsf{path}^1_2, \mathsf{path}^1_4), (\mathsf{path}^3_2, \mathsf{path}^3_4)$

Fig. 1: FHEGen and $\mathsf{EvalFHE}$ denote respectively the FHE key-generation and evaluation algorithms. $\mathsf{EvalFHE}$ takes as input the function and the set of ciphertexts on which the function should be applied homomorphically. When we write Enc with an arrow, we mean that the FHE encryption procedure was applied on plaintexts at the tail of the arrow thus generating a set of ciphertexts that we place at the tip of the arrow. ($\mathsf{VSS}, \mathsf{VSSRec}$) denote respectively the VSS sharing and reconstruction algorithms. PCP denotes the PCP prover algorithm, and Φ represents the decision algorithm of the PCP verifier. Every action that $\mathsf{P}^{\mathsf{eval}}$ performs is under FHE ciphertexts encrypted with the key pk, with the exception of the creation of the Merkle trees (only the leaves of the MT correspond to FHE ciphertexts). Also, the views of the execution of Π_g are encrypted. With abuse of notation, we denote with $\mathsf{view}^1_1, \ldots, \mathsf{view}^1_n, \mathsf{view}^3_1, \ldots, \mathsf{view}^3_n$, the ciphertexts of the view of Π_g. In this simple example, the PCP is queried on only two locations $1, 3$, and the challenges for the MPC-in-the-head part are two indices $(2, 4)$. Similar to the cut-and-choose performed to check the validity of the input of $\mathsf{P}^{\mathsf{rec}}$, this check involves a challenge of just two indices. To compute the output $\mathsf{P}^{\mathsf{rec}}$ decrypts ct_y and accepts the plaintext as the output of the computation only if the MT paths (path^1_2, path^1_4 and path^3_2, path^3_4 in this example) are valid, and the MPC-in-the-head verification succeeds. Here, path^j_i denotes the path from root to leaf with value ct^j_i.

correctly. $\mathsf{P}^{\mathsf{rec}}$ can verify the zero-knowledge proof in the clear, as it knows the FHE decryption key. If we use an information-theoretic zero-knowledge proof, we can safely evaluate it under FHE and still claim black-box use of cryptographic primitives. Unfortunately, such zero-knowledge proofs do not exist unless a strong form of setup is assumed [Ps05, AH87]. Moreover, we would need the zero-knowledge proof to be compact to keep the overall communication complexity of our protocol sublinear in $|f|$.

Quick detour on PCPs. We still want to follow this idea, but using simpler tools. The main building block we consider is a special type of information-theoretic proof system called *Probabilistic Checkable Proof* (PCP) [AS98]. In a PCP we have a prover and a verifier. The prover, on input a statement x and a witness w, generates a proof π proving that $(x, w) \in \mathsf{Rel}$, where Rel is some NP relation. This proof can be seen as a sequence of κ locations $\pi_1, \ldots, \pi_\kappa$, and the size of π depends on the size of x and w.

To verify the validity of the proof, the verifier can ask to inspect some of the proof locations by issuing a set of indices (queries) $q_1, \ldots, q_\tau$, thus receiving $\pi_{q_1}, \ldots, \pi_{q_\tau}$. The verifier then decides whether to accept or reject the proof by running a decision procedure Φ on input the theorem x, the queries $q_1, \ldots, q_\tau$ and the answers $\pi_{q_1}, \ldots, \pi_{q_\tau}$. A PCP proof is sound, in the sense that a verifier will not accept a proof for a false statement, and moreover the verification complexity, which corresponds to the number of queries and the complexity of Φ, is only polylogarithmic in the size of π.

As described, PCPs work by assuming that the verifier has access to an oracle that knows π, and honestly replies to the verifier's queries. To use PCPs in a real protocol (where such an oracle does not exist), we would need to replace the oracle with a protocol party, like the prover who generated the proof itself. However, a malicious prover may be able to cheat by crafting answers of the PCP adaptively on the queries performed by the verifier. The usual way to solve this problem is to follow the approach of [Kil92, BG01, Mic00]. In this, the prover first generates the PCP proof π and then commits to its components via a Merkle tree (MT) [Mer89]. The verifier, upon receiving the commitment, generates the PCP queries and asks the prover to open the corresponding paths in the MT.

How to use PCPs in our protocol. After this brief recap on PCPs, we are now ready to show how to modify our protocol to make sure that $\mathsf{P}^{\mathsf{eval}}$ can prove that it performed the evaluation f, without harming the succinctness of the protocol and still making black-box use of the underlying primitives.

The statement that $\mathsf{P}^{\mathsf{eval}}$ needs to prove via the PCP consists of $(y, x_{\mathsf{rec}}, x_{\mathsf{eval}})$, and the witness simply consists of f. We have that $((y, x_{\mathsf{rec}}, x_{\mathsf{eval}}), f) \in \mathsf{Rel}$ iff $f(x_{\mathsf{rec}}, x_{\mathsf{eval}}) = y$. We are now ready to show a first attempt to modify our protocol.

$\mathsf{P}^{\mathsf{eval}}$, using the statement and the witness just defined, generates the PCP proof under FHE, thus obtaining a list of ciphertexts $\mathsf{ct}_i^{(\pi)} \leftarrow \mathsf{Enc}(\mathsf{pk}, \pi_i)$ for each $i \in [\kappa]$. $\mathsf{P}^{\mathsf{eval}}$ then constructs the Merkle tree using these ciphertexts, and sends the root to $\mathsf{P}^{\mathsf{rec}}$. $\mathsf{P}^{\mathsf{rec}}$ sends the challenges of the PCP, and $\mathsf{P}^{\mathsf{eval}}$ opens the MT paths according to these challenges. $\mathsf{P}^{\mathsf{rec}}$ checks that the MT paths have been opened correctly, decrypts the leaves of the MT that it got, and runs the PCP verification procedure Φ in the clear. This approach makes sure that the output y obtained by decrypting ct_y does correspond to $y = f(x_{\mathsf{rec}}, x_{\mathsf{eval}})$.

Unfortunately, this approach still has some issues. We recall that the PCP verification Φ requires the knowledge of the statement which in our case corresponds to $(y, x_{\mathsf{rec}}, x_{\mathsf{eval}})$. Note that the statement includes the input of the party x_{eval} that we must protect from $\mathsf{P}^{\mathsf{rec}\star}$ (which we recall is the PCP verifier). We also note that just giving in the clear some of the components of π may leak information about the statement (hence the input of $\mathsf{P}^{\mathsf{eval}}$).

We first argue how to solve the latter issue: hiding the components of π, while still allowing $\mathsf{P}^{\mathsf{rec}}$ to verify the PCP proof. In Fig. 1 we provide a more high-level description of our protocol to help visualize the components that we will now introduce. The high level idea is to let $\mathsf{P}^{\mathsf{eval}}$ issue a *zero-knowledge* (ZK) proof (non-compact) for the following statement *"There exists a reply $\pi_{q_1}, \ldots, \pi_{q_\tau}$ to the queries $q_1, \ldots, q_\tau$ that makes the PCP verifier's decision algorithm Φ accept the statement*

$(y, x_{\mathsf{rec}}, x_{\mathsf{eval}})$". To make this simple idea work, we need to connect the PCP answers $(\pi_{q_1}, \ldots, \pi_{q_\tau})$ opened via the MT, with the statement proven by the ZK proof. The idea is to follow an MPC-in-the-head approach [IKOS07] and combine it with PCPs. In particular, instead of generating the MT of π, we generate a MT of a secret sharing of π. In more detail, for each element of the proof π_i we run a VSS, thus obtaining n shares, and we construct a MT T_i using these shares. This process leads to κ different Merkle trees. These κ MTs are then used to construct a final MT, which represents our commitment T_{root} to the PCP, that is sent to $\mathsf{P}^{\mathsf{rec}}$.

For every j, upon receiving a query q_j from $\mathsf{P}^{\mathsf{rec}}$, $\mathsf{P}^{\mathsf{eval}}$ opens a path to the q_j-th leaf of T_{root} (see Fig. 1 for an example), which corresponds to the MT root T_{q_j}. We recall that T_{q_j} contains in its leaves a verifiable secret sharing of π_{q_j}. Then $\mathsf{P}^{\mathsf{eval}}$ uses all these shares of $\pi_{q_1}, \ldots, \pi_{q_\tau}$ as input of an information theoretic-MPC protocol Π_g that evaluates the following function $g_{x_{\mathsf{rec}}, x_{\mathsf{eval}}, y, \{q_j\}_{i \in [\tau]}}(\cdot)$. The function g is parametrized by the input of the parties, the output of the computation $y \leftarrow f(x_{\mathsf{rec}}, x_{\mathsf{eval}})$, and by the PCP queries, and takes as input τ sets of VSS shares $S_1, \ldots, S_\tau$. For each $j \in [\tau]$ the function g runs the VSS reconstruction algorithm on the set of shares S_j, thus obtaining π_{q_j}, and then runs the PCP verification algorithm Φ on input the reconstructed values $\pi_{q_1}, \ldots, \pi_{q_\tau}$, the queries $\{q_j\}_{j \in [\tau]}$ and the PCP theorem $(y, x_{\mathsf{rec}}, x_{\mathsf{eval}})$.

The party $\mathsf{P}^{\mathsf{eval}}$ runs Π_g in its head and commits to the obtained views using a standard commitment scheme. Then, $\mathsf{P}^{\mathsf{rec}}$ asks to open a subset of the views of Π_g with the corresponding MT paths, and checks the consistency of the views and the correctness of the MT openings. For example, if $\mathsf{P}^{\mathsf{rec}}$ asks to see the view of the first player of Π_g, then $\mathsf{P}^{\mathsf{eval}}$ will send for each $j \in [\tau]$: 1) The path that goes from the MT root T_{root} to T_{q_j}; 2) the Merkle tree path that from the root T_{q_j} leads to the first VSS share of π_{q_j} which we denote with $s_1^{q_j}$, and 3) the opening of the first commitment that contains the view of the first player in Π_g (with the inputs $s_1^{q_j}$). $\mathsf{P}^{\mathsf{rec}}$ will accept y as a valid output if the following checks succeed (for every query): 1) the path from T_{q_j} to $s_1^{q_j}$ is valid, 2) the view of the first party that executed Π_g uses s_j^1 and the view of this party is consistent with all other opened parties' views for Π_g; and 3) The path from T_{root} to T_{q_j} is valid (we refer to Fig. 1 for an example).

Now that we have prevented any leakage of x_{eval} from the proof π, we need to fix the other problem we mentioned. Indeed, recall that in the above approach, g is still parametrized by the input of $\mathsf{P}^{\mathsf{eval}}$. Hence, we are still leaking x_{eval}. To solve this issue, the idea is to turn x_{eval} into an additional witness of the MPC-in-the-head proof. To extract the witness in the simulation, we can rely on the existence of an efficient extraction procedure to retrieve the witness and an efficient sampler to generate queries whose distribution is indistinguishable from that of the PCP verifier. However, to simplify the simulation, we adopt an alternative approach described below. A similar method was used in a concurrent work [IOS25] to construct doubly efficient zero-knowledge proofs for RAM programs.

To do that, we commit to a secret sharing of x_{eval}, and include these shares as inputs of the MPC-in-the-head proof. Again, we need to somehow connect the fact that the same x_{eval} is used in both the generation of the PCP proof and as the prover input of the MPC-in-the head protocol.

We can solve this problem easily by relying on a special type of PCP where the statement has two components: an *explicit* and an *implicit* input. The explicit part is read entirely by the verifier, and the verifier is only given oracle access to the implicit part (the queries to this part are counted as part of query complexity). In our case, (y, x_{rec}) represents the explicit input, whereas x_{eval} represents an implicit input. Equipped with this special PCP, we can now just treat x_{eval} exactly as π to create the connection between the PCP and the MPC-in-the-head. The core technique is very similar to the one we have just described, and we refer the reader to the technical part of the paper for all the details. PCPs that enjoy this special property are called PCPs of proximity (PCPPs) [BGH+05, BGH+06, Din07]. However, PCPPs need to be handled with some care as they enjoy a weaker form of soundness. In the technical part of the paper, we will show how to deal with this and other minor subtleties that we need to face.

The reader may have noticed that in our informal theorem, we claimed to need, on top of FHE and collision-resistant hash functions (needed to construct the Merkle tree), a generic non-compact

2PC protocol, that we have not mentioned in this high-level overview. We will use this additional 2PC to generate the challenges of the cut-and-choose and the challenges of the MPC-in-the-head part. This will be needed during the simulation to program the challenge or to rewind and change the challenges of these sub-protocol executions. More details are provided in the technical part of the paper.

Finally, our protocol enables black-box cryptography without impacting concrete efficiency. While the non-black-box solution [COWZ22] achieves function-size-independent communication, our protocol's communication still depends only polylogarithmically on the function size due to PCP verifier complexity. Recent advances in efficient PCP-based proofs (e.g., StarkWare [Sta]) further support its practical feasibility. In terms of computational cost, our protocol heavily relies on FHE, with the most expensive part being PCP execution under FHE. Though current FHE constructions are concretely inefficient, there are efforts to develop hardware-based solutions (e.g., DPRIVE by DARPA [DAR], Duality [BCPR]). Notably, in the malicious secure setting, to the best of our knowledge, even succinct constant-round non-black-box protocols rely on FHE or other concretely inefficient tools (e.g., functional encryption [SW05, BSW11], homomorphic secret sharing [BGI16]).

Other related works. Other than the works mentioned previously, a line of research where communication complexity plays an important role is that of *input-hiding secure computation* [MRK03, IP07, ADT11]. In this, the goal is to hide the size of the parties' inputs. The techniques used to design such protocols are similar to those mentioned earlier in the context of succinct communication. Indeed, prior approaches on input-hiding computation are based on somehow similar combinations of fully-homomorphic encryption and proof systems [CV12, LNO13, COV15].

2 Preliminaries

In this section, we define the tools and primitives required for our construction. In addition, see Appendix A for detailed definitions of commitment schemes and encoding schemes.

Notation. We use λ to denote the computational security parameter. For integers a, b s.t. $a \leq b$, $[a, b]$ denotes set $\{a, a+1, \ldots, b\}$. For integer m, we use $[m]$ to denote set $[1, m]$. We use $=$ to denote equality, $\leftarrow$ to denote assignment, and $\overset{\$}{\leftarrow}$ to denote sampling a uniform distribution. We use $\|$ to denote concatenation.

Encoding schemes. An encoding scheme comprises of two algorithms, Encode and Decode. Encode maps a message $x \in \mathbb{F}_q^k$ to codeword $X \in \mathbb{F}_q^K$. Decode takes as input (possibly corrupted) codeword $X' \in \mathbb{F}_q^K$ and outputs x or $\perp$. For any $x \neq x'$, the Hamming distance between $\mathsf{Encode}(x)$ and $\mathsf{Encode}(x')$ is $\geq \delta K$. If $\exists x \in \mathbb{F}_q^k$ such that Hamming distance between X' and $\mathsf{Encode}(x)$ is $< \delta K/2$, then $\mathsf{Decode}(X')$ outputs x, else $\mathsf{Decode}(X')$ outputs $\perp$. We use Reed-Muller [Ree54, Mul54] codes in our paper. See Appendix A.2 for details.

2.1 Secure Multiparty Computation

Here, we provide a formal definition of secure multiparty computation, with emphasis on the two-party scenario. Much of the following is taken verbatim from [IKSS21], which in turn uses definitions from [Gol01].

A multiparty protocol is cast by specifying a random process that maps tuples of inputs to tuples of outputs (one for each party). We refer to such a process as a functionality. The security of a protocol is defined with respect to a functionality f. In particular, let n denote the number of parties. A (non-reactive) n-party functionality f is a (possibly randomized) mapping of n inputs to n outputs. A multiparty protocol with security parameter λ for computing the functionality f is a protocol running in time $\mathsf{poly}(\lambda)$ and satisfying the following correctness requirement: if parties $P_1, \ldots, P_n$ with inputs $x_1, \ldots, x_n$ respectively, all run an honest execution of the protocol, then the joint distribution of the outputs $y_1, \ldots, y_n$ of the parties is statistically close to $f(x_1, \ldots, x_n)$.

Defining security. We assume that the reader is familiar with the standard simulation-based definitions of secure multiparty computation in the standalone setting. We provide a self-contained definition for completeness and refer to [Gol01] for a more complete description. The security of a protocol (w.r.t. a functionality f) is defined by comparing the real-world execution of the protocol with an ideal-world evaluation of f by a trusted party. More concretely, it is required that for every adversary $\mathcal{A}$, which attacks the real-world execution of the protocol, there exists an adversary $\mathcal{S}$, also referred to as the simulator, which can *achieve the same effect* in the ideal-world execution. We denote $\vec{x} = (x_1, \ldots, x_n)$.

The real execution. In the real execution, the n-party protocol Π for computing f is executed in the presence of an adversary $\mathcal{A}$. The honest parties follow the instructions of Π. The adversary $\mathcal{A}$ takes as input the security parameter λ, the set $I \subset [n]$ of corrupted parties, the inputs of the corrupted parties, and an auxiliary input z. $\mathcal{A}$ sends all messages in place of corrupted parties and may follow an arbitrary polynomial-time strategy.

The above interaction of $\mathcal{A}$ with a protocol Π defines a random variable $\mathsf{REAL}_{\Pi,\mathcal{A}(z),I}(\lambda, \vec{x})$ whose value is determined by the coin tosses of the adversary and the honest players. This random variable contains the output of the adversary (which may be an arbitrary function of its view) as well as the outputs of the uncorrupted parties.

The ideal execution - security with abort. An ideal execution for a function f proceeds as follows:

- **Send inputs to the trusted party:** As before, the parties send their inputs to the trusted party, and we let x'_i denote the value sent by P_i.
- **Trusted party sends output to the adversary:** The trusted party computes $f(x'_1, \ldots, x'_n) = (y_1, \ldots, y_n)$ and sends $\{y_i\}_{i \in I}$ to the adversary.
- **Adversary instructs trusted party to abort or continue:** This is formalized by having the adversary send either a continue or abort message to the trusted party. (A semi-honest adversary never aborts.) In the latter case, the trusted party sends to each uncorrupted party P_i its output value y_i. In the former case, the trusted party sends the special symbol $\perp$ to each uncorrupted party.
- **Outputs:** $\mathcal{S}$ outputs an arbitrary function of its view, and the honest parties output the values obtained from the trusted party.

In the case of secure two-party computation, where an adversary corrupts only one of the two parties, the above behavior of an ideal-world trusted party is captured by functionality $\mathcal{F}_{\mathsf{2PC}}$, described in Figure 2 below.

Functionality $\mathcal{F}_{\mathsf{2PC}}$

Parameters. Description of function f. Let $\mathcal{H} \in \{1, 2\}$ and $\mathcal{M} \in \{1, 2\} \setminus \{\mathcal{H}\}$ denote the index of honest party and corrupt party, respectively.

1. $\mathcal{F}_{\mathsf{2PC}}$ receives input $x_{\mathcal{H}}$ from honest party $P_{\mathcal{H}}$ and input $x_{\mathcal{M}}$ of the corrupt party from simulator $\mathcal{S}$.
2. $\mathcal{F}_{\mathsf{2PC}}$ computes $y \leftarrow f(x_1, x_2)$ and sends y to $\mathcal{S}$.
3. If $\mathcal{F}_{\mathsf{2PC}}$ receives abort from $\mathcal{S}$, it sends abort to honest party $P_{\mathcal{H}}$. Otherwise, $\mathcal{F}_{\mathsf{2PC}}$ sends y to honest party $P_{\mathcal{H}}$.

Fig. 2: Functionality $\mathcal{F}_{\mathsf{2PC}}$ for maliciously secure 2PC.

As before, the interaction of $\mathcal{S}$ with the trusted party defines a random variable $\mathsf{IDEAL}_{f,\mathcal{S}(z),I}(\lambda, \vec{x})$. Having defined the real and the ideal world executions, we can now proceed to define the security notion.

Definition 1. *Let λ be the security parameter, f an n-party randomized functionality, and Π an n-party protocol for $n \in \mathbb{N}$. We say that Π t-securely computes f in the presence of malicious adversaries if for every PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{S}$ such that, for any $I \subset [n]$ with $|I| \leq t$, the following quantity is negligible in λ:*

$$|\Pr[\mathsf{REAL}_{\Pi,\mathcal{A}(z),I}(\lambda, \vec{x}) = 1] - \Pr[\mathsf{IDEAL}_{f,\mathcal{S}(z),I}(\lambda, \vec{x}) = 1]|,$$

where $\vec{x} = \{x_i\}_{I \in [n]} \in \{0,1\}^*$ and $z \in \{0,1\}^*$.

Note that for two-party computation, $n = 2$ and $t = 1$ in Definition 1.

Privacy with knowledge of outputs (PKO). Ishai et al. [IKP10] considered a weakening of the security definition, where the trusted party first delivers the output to the ideal world adversary, which then provides an output to be delivered to all the honest parties. They called this security notion *privacy with knowledge of outputs* (PKO), and showed a transformation from this notion to security with selective abort using unconditional MACs. Note that the notions of security with abort and security with *selective* abort are identical in the case of two-party computation, and thus the result of [IKP10] equips us with a statistical reduction from PKO to security with abort (Definition 1).

The behavior of an ideal-world trusted party which guarantees PKO is captured by functionality $\mathcal{F}_{\mathsf{2PC\text{-}PKO}}$, described in Figure 3 below.

Functionality $\mathcal{F}_{\mathsf{2PC\text{-}PKO}}$

Parameters. Description of function f. Let $\mathcal{H} \in \{1,2\}$ and $\mathcal{M} \in \{1,2\} \setminus \{\mathcal{H}\}$ denote the index of honest party and corrupt party, respectively.

1. $\mathcal{F}_{\mathsf{2PC\text{-}PKO}}$ receives input $x_{\mathcal{H}}$ from honest party $P_{\mathcal{H}}$ and input $x_{\mathcal{M}}$ of the corrupt party from simulator $\mathcal{S}$.
2. $\mathcal{F}_{\mathsf{2PC\text{-}PKO}}$ computes $y \leftarrow f(x_1, x_2)$, and sends y to $\mathcal{S}$.
3. $\mathcal{F}_{\mathsf{2PC\text{-}PKO}}$ receives abort or y' from $\mathcal{S}$, and sends the same to honest party $P_{\mathcal{H}}$.

Fig. 3: Functionality $\mathcal{F}_{\mathsf{2PC\text{-}PKO}}$ for 2PC satisfying PKO.

In Section 3.2, we show that our construction securely realizes $\mathcal{F}_{\mathsf{2PC\text{-}PKO}}$ as defined above, and we defer to the result of [IKP10] for a black-box extension to standard malicious security with abort.

Zero-knowledge from "MPC-in-the-head". Ishai et al. [IKOS07] proposed the following approach to constructing zero-knowledge proofs, using secure multiparty computation. For NP relation $\mathcal{R}(x, w)$ and language $L = \{x : \exists w\ R(x, w) = \mathtt{accept}\}$, consider a prover $\mathcal{P}$ having NP statement x and witness w, and a verifier $\mathcal{V}$ having only x. In order to prove $x \in L$ without leaking w, $\mathcal{P}$ locally emulates $n(= O(\lambda))$ players $p_1, \ldots, p_n$, and gives each p_i a share w_i of the witness w. Then, $\mathcal{P}$ runs "in the head" an execution of an n-party MPC protocol Π_g, which takes public input x and private input w_i from each player p_i, computes a reconstruction of witness w, and outputs $R(x, w)$ to each player. Let $g(x, w_1, \ldots, w_n)$ denote this functionality realized by Π_g.

In this process, each p_i obtains a view View_i of the protocol execution, which includes x, w_i, and all incoming messages from other players. A pair of views $(\mathsf{View}_i, \mathsf{View}_j)$ is said to be consistent (w.r.t. Π_g and x) if all outgoing messages implicit in View_i, x are identical to the incoming messages reported in View_j, and vice versa. Then, $\mathcal{P}$ reveals some random t $(= n/c)$ views to $\mathcal{V}$, and $\mathcal{V}$ accepts if every pair of these views is consistent, and each has output $\mathtt{accept}$. Clearly, if the sharing scheme guarantees t-privacy, then zero knowledge is ensured.

Our result makes use of the above approach, and as shown in Theorem 4 (and in [IKOS07]), soundness holds in the malicious setting if MPC protocol Π_g satisfies t-privacy in the semi-honest setting (obtained by relaxing Definition 1 to semi-honest adversaries), as well as a notion of t-robustness in the malicious setting, defined as follows.

Definition 2 (t-Robustness). *We say that n-party protocol Π_g realizes g with perfect t-robustness if it is perfectly correct in the presence of a semi-honest adversary, and furthermore for any computationally unbounded malicious adversary corrupting at most t players, and for any inputs $(x, w_1, \dots, w_n)$, the following robustness property holds. If there is no $(w'_1, \dots, w'_n)$ such that $g(x, w'_1, \dots, w'_n) =$ `accept`, then the probability that some uncorrupted player outputs `accept` in an execution of Π_g is 0.*

Note that t-robustness is simply a weaker notion of t-security as defined in Definition 1. In our construction, we execute MPC-in-the-head using the information theoretic MPC protocol of [DN07]. We state their result formally below.

Theorem 1. *[DN07, Section 5] Let $t < n/3$. There exists a perfectly correct, t-private and t-robust MPC protocol that has a communication cost of $\tilde{O}(n|C|)$, where $|C|$ denotes the size of circuit C.*

2.2 Verifiable Secret Sharing

A verifiable secret sharing scheme [CGMA85] is a pair of $(n + 1)$-player MPC protocols, which implement the following functionality. In the first protocol, referred to as the *sharing* protocol, one player called the dealer holds a secret, and it sends shares of this secret to the remaining n players, called shareholders. In the second protocol, referred to as the *reconstruction* protocol, the shareholders reveal their shares to each other, and they reconstruct the secret initially shared by the dealer.

For any adversary corrupting up to t shareholders, the VSS scheme guarantees the following security properties. It guarantees *privacy*, which ensures that if the dealer is honest, then the adversary learns no information about the underlying secret. It also guarantees *strong commitment*, which ensures that, even if the dealer is malicious, then either it gets disqualified by all the honest parties during the sharing protocol, or the honest parties are able to reconstruct a unique secret via the reconstruction protocol. We now give a formal definition of verifiable secret sharing schemes.

Definition 3 (Verifiable secret sharing scheme). *A t-out-of-n perfectly secure verifiable secret sharing (VSS) scheme is a pair of $(n + 1)$-party protocols $\mathsf{VSS} = (\mathsf{Share}, \mathsf{Reconstruct})$, defined as follows:*

- Share. *In the sharing protocol, a dealer $\mathcal{D}$ runs on input a secret s and randomness r, while all other players P_i run on randomness r_i, for $i \in [n]$. Players exchange messages in multiple rounds, and each player P_i ultimately receives a share View_i.*
- $\mathsf{Reconstruct}$. *In the reconstruction protocol, each player P_i sends its view View_i to each other player, and on input the views of all players (including bad views), each player computes the reconstruction function recons on the received views to output reconstructed secret s, i.e., $s \leftarrow \mathsf{recons}(\mathsf{View}_1, \dots, \mathsf{View}_n)$.*

We require a VSS scheme to satisfy the following three conditions:

- **Correctness.** *If the dealer $\mathcal{D}$ runs Share honestly with input secret s, then each honest player outputs s at the end of $\mathsf{Reconstruct}$.*
- **Privacy.** *If the dealer $\mathcal{D}$ runs Share honestly with input secret s, then the view of any adversary corrupting upto t players, before running $\mathsf{Reconstruct}$, is distributed independently of s.*

- **Strong Commitment.** *If the dealer $\mathcal{D}$ is dishonest, then either (1) all the honest parties disqualify $\mathcal{D}$ during* Share*, output $\perp$, and refuse to run* Reconstruct*, or (2) the honest parties do not disqualify $\mathcal{D}$ during* Share*, in which case they all output a unique secret $s^* \neq \perp$ at the end of* Reconstruct*.*

We recall the following result of [GIKR01]. We observe that their VSS construction is information-theoretic and has a deterministic reconstruction protocol, making it suitable for our purposes.

Theorem 2 ($\lfloor \frac{n-1}{4} \rfloor$-out-of-$n$ VSS, [GIKR01]). *There exists an efficient 2-round (n, t)-VSS protocol when $n > 4t$.*

2.3 Extractable Commitments

Informally, a commitment scheme is said to be extractable (with over-extraction) if there exists a PPT extractor that extracts the committed value conditioned on the commitment being well-formed. Formally, we report a weaker version of the definition of [PW09]. While the definition of [PW09] requires an extractor capable of extracting the underlying messages from *accepting* transcripts, we only require an extractor to extract underlying messages from *well-formed* transcripts, as described below.

Definition 4 (Extractable commitment scheme). *Consider a statistically binding, computationally hiding commitment scheme $\Pi_{\mathsf{comExt}} = (\mathcal{C}, \mathcal{R})$. Then Π_{comExt} is said to be* extractable *if there exists an expected* PPT *algorithm* Ext*, called the extractor, that given oracle access to any malicious* PPT *committer $\mathcal{C}^*$, outputs a transcript τ and a message m such that the following hold:*

- *τ is identically distributed to the view of $\mathcal{C}^*$ when interacting with an honest receiver $\mathcal{R}$ in the commitment phase.*
- *The probability that τ is a well-formed transcript and $m = \perp$ is negligible.*
- *If $m \neq \perp$ then $\Pr[(\exists \tilde{m} \neq m, \tilde{r}_c) : \mathsf{Dec}(\tau, \tilde{m}, \tilde{r}_c) = 1] \leq \mathsf{negl}(\lambda)$.*

We use the construction of an extractable commitment scheme given in [PW09], which is based on the works of [Ros04, PRS02, DDN00].

2.4 Vector Commitments

Vector commitments [Mer89, LY10, CF13] enable the commitment of a vector of messages in such a way that the commitment can later be opened with respect to a specific index. We provide its formal definition below.

Definition 5 (Vector commitment scheme). *A vector commitment scheme* VC=(Setup, Com, Open, Verify) *is a tuple of protocols between two PPT interactive algorithms, a committer $\mathcal{C}$ and a receiver $\mathcal{R}$, defined as follows:*

- $\mathsf{ck} \overset{\$}{\leftarrow} \mathsf{Setup}(1^\lambda, n)$*: Given λ and the size of the vector $n = \mathsf{poly}(\lambda)$,* Setup *outputs commitment key* ck*. The* Setup *algorithm is executed by $\mathcal{R}$, who sends* ck *to $\mathcal{C}$.*
- $(c, \mathsf{st}) \overset{\$}{\leftarrow} \mathsf{Com}(\mathsf{ck}, \mathbf{m} = (m_1, \ldots, m_n))$*:* Com *takes as input* ck *and a message vector $\mathbf{m}$ of length n, and outputs commitment c and committer state* st*.* Com *is executed by $\mathcal{C}$, who sends the commitment c to $\mathcal{R}$.*
- $\sigma_i \leftarrow \mathsf{Open}(\mathsf{ck}, m, i, \mathsf{st})$*: On input* ck*, message m, index $i \in [n]$ and* st*,* Open *outputs proof σ_i for the statement that the message at the i^{th} index of the committed message vector is m.* Open *is executed by $\mathcal{C}$, who sends σ_i to $\mathcal{R}$.*

– $b \leftarrow \mathsf{Verify}(\mathsf{ck}, c, i, m, \sigma)$*: Given* ck*,* c*,* $i \in [n]$*, message* m *and proof* σ*,* Verify *validates proof* σ *for the statement that* c *is a commitment on message vector* $\mathbf{m}$ *such that* $m_i = m$ *and outputs* `accept`/`reject`*.* Verify *algorithm is executed by* $\mathcal{R}$*.*

We require a vector commitment scheme VC to be complete and concise, and satisfy computational hiding and index binding. These are defined as follows.

Definition 6 (Completeness). *A vector commitment scheme* VC *is complete if for all* $\lambda \in \mathbb{N}$ *and* $n \in \mathsf{poly}(\lambda)$*, if* $\mathsf{ck} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, n)$*,* c *is a commitment on vector* $\mathbf{m} = (m_1, \ldots, m_n)$ *obtained on executing* Com*, and* σ_i *is a proof for index* i *computed using* Open*, then it holds that* $\mathsf{Verify}(\mathsf{ck}, c, i, m_i, \sigma_i) = \mathtt{accept}$*.*

Definition 7 (Conciseness). *A vector commitment scheme* VC *is concise if for all* $\lambda \in \mathbb{N}$ *and* $n \in \mathsf{poly}(\lambda)$*, if* $\mathsf{ck} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, n)$*,* c *is a commitment on vector* $\mathbf{m} = (m_1, \ldots, m_n)$ *obtained on executing* Com*, and* σ_i *is a proof for index* i *computed using* Open*, then the sizes of* c *and* σ_i *are both independent of* n*.*

Definition 8 (Statistical hiding). *A vector commitment scheme* VC *is statistically hiding if for every* $\lambda \in \mathbb{N}$ *and* $n \in \mathsf{poly}(\lambda)$*, any unbounded adversary* $\mathcal{A}$ *has a success probability* $\leq \frac{1}{2} + \mathsf{negl}(\lambda)$ *in the following security game:*

– $\mathcal{A}$ *sends* ck*,* $\mathbf{m}_0 = (m_{0,1}, \ldots, m_{0,n})$ *and* $\mathbf{m}_1 = (m_{1,1}, \ldots, m_{1,n})$*, such that* $\mathbf{m}_0 \neq \mathbf{m}_1$*, but they agree on some* $k < n$ *positions.*
– *The challenger samples* $b \xleftarrow{\$} \{0, 1\}$ *and computes* $(c, \mathsf{st}) \xleftarrow{\$} (\mathsf{ck}, \mathbf{m}_b)$*. Then, for each* i *such that* $m_{0,i} = m_{1,i}$*, it computes* $\sigma_i \leftarrow \mathsf{Open}(\mathsf{ck}, m_{b,i}, i, \mathsf{st})$*. It sends* c *and all such* σ_i*.*
– $\mathcal{A}$ *outputs* b'*, and succeeds if* $b = b'$*.*

Definition 9 (Index binding). *A vector commitment scheme* VC *is index binding if for every* $\lambda \in \mathbb{N}$*,* $n \in \mathsf{poly}(\lambda)$*, and* PPT *adversary* $\mathcal{A}$*, we have:*

$$\Pr\left[\begin{array}{c|c} \mathsf{Verify}(\mathsf{ck}, c, i, m, \sigma) = \mathtt{accept} \ \wedge & \mathsf{ck} \xleftarrow{\$} \mathsf{Setup}(1^\lambda); (c, i, m, m', \sigma, \sigma') \xleftarrow{\$} \mathcal{A}(\mathsf{ck}), \\ \mathsf{Verify}(\mathsf{ck}, c, i, m', \sigma') = \mathtt{accept} & \text{where } i \in [n] \text{ and } m \neq m' \end{array}\right] \leq \mathsf{negl}(\lambda).$$

We use ubiquitous Merkle trees [Mer89] to realize vector commitments in our construction, with ck comprising of a key of length λ. The execution time of Com is $\tilde{O}(n)$ and size of commitment is λ. The computation time of Open and Verify and the size of proof is $O(\lambda \log(n))$. While Merkle trees are index binding, we can make our vector commitment scheme statistically hiding by committing each m_i using the statistically hiding scheme of [HM96] (based on collision-resistant hash functions), and committing the resulting commitments using a Merkle tree.

2.5 Fully Homomorphic Encryption

We recall the notion of fully homomorphic encryption (FHE), defined as follows.

Definition 10 (Fully homomorphic encryption scheme). *A (multi-hop) fully homomorphic encryption (FHE) scheme is a tuple of PPT algorithms* $(\mathsf{Gen}^{\mathsf{FHE}}, \mathsf{Enc}^{\mathsf{FHE}}, \mathsf{Dec}^{\mathsf{FHE}}, \mathsf{Eval}^{\mathsf{FHE}})$*, defined as follows:*

– $\mathsf{Gen}^{\mathsf{FHE}}(1^\lambda)$*: The key generation algorithm takes security parameter* λ *and outputs a key pair* $(\mathtt{sk}, \mathtt{pk})$*.*
– $\mathsf{Enc}^{\mathsf{FHE}}(\mathtt{pk}, m)$*: The encryption algorithm takes a public key* pk *and a message* m *as input, and it outputs the corresponding ciphertext* c*.*

- $\mathsf{Dec}^{\mathsf{FHE}}(\mathtt{sk}, \mathsf{ct})$: *The decryption algorithm takes a secret key* sk *and a ciphertext c as input, and it outputs the decrypted message m.*
- $\mathsf{Eval}^{\mathsf{FHE}}(\mathtt{pk}, (\mathsf{ct}_1, \ldots, \mathsf{ct}_n), f)$: *The evaluation algorithm takes as input a public key* pk, *a string description f of any polynomial-size function, and input ciphertexts $(\mathsf{ct}_1, \ldots, \mathsf{ct}_n)$, where n is the number of inputs of f, and it outputs a new ciphertext* ct.

Let $\mathcal{M}$ denote the space of plaintext messages. An FHE scheme is required to satisfy the standard notions of correctness and CPA-security fulfilled by any public key encryption scheme, and defined as follows:

Definition 11 (Correctness). *An FHE scheme is* correct *if $\forall \lambda \in \mathbb{N}, m \in \mathcal{M}$, if $(\mathtt{sk}, \mathtt{pk}) \xleftarrow{\$} \mathsf{Gen}^{\mathsf{FHE}}(1^\lambda)$ and $c \xleftarrow{\$} \mathsf{Enc}^{\mathsf{FHE}}(\mathtt{pk}, m)$, then $\mathsf{Dec}^{\mathsf{FHE}}(\mathtt{sk}, c) = m$.*

Definition 12 (CPA-security). *An FHE scheme is* CPA-secure *if for all $\lambda \in \mathbb{N}$ and* PPT *adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, we have:*

$$\Pr\left[b = b' \middle| \begin{array}{l} (\mathtt{sk}, \mathtt{pk}) \xleftarrow{\$} \mathsf{Gen}^{\mathsf{FHE}}(1^\lambda); (m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_0(\mathtt{pk}) \\ b \xleftarrow{\$} \{0,1\}; c \xleftarrow{\$} \mathsf{Enc}^{\mathsf{FHE}}(\mathtt{pk}, m_b); b' \leftarrow \mathcal{A}_1(c, \mathsf{st}) \end{array} \right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

Further, we characterize the correctness of multiple evaluations by an FHE scheme using the following definition of multi-hop homomorphic correctness.

Definition 13 (Multi-hop homomorphic correctness). *An FHE scheme has multi-hop homomorphic correctness if for all λ and functions f, if $(\mathtt{sk}, \mathtt{pk}) \xleftarrow{\$} \mathsf{Gen}^{\mathsf{FHE}}(1^\lambda)$ and the set of correctly generated ciphertexts under* pk *is defined as:*

$$\begin{aligned} C_{\mathtt{pk}} = \{c \mid \big(m \in \mathcal{M} \ \wedge\ c \xleftarrow{\$} \mathsf{Enc}^{\mathsf{FHE}}(\mathtt{pk}, m)\big) \ \vee\ \exists\ f\ \big(\mathsf{ct}_1, \ldots, \mathsf{ct}_n \in C_{\mathtt{pk}} \ \wedge \\ c \xleftarrow{\$} \mathsf{Eval}^{\mathsf{FHE}}(\mathtt{pk}, (\mathsf{ct}_1, \ldots, \mathsf{ct}_n), f)\big)\}, \end{aligned}$$

then for all correctly generated ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_n \in C_{\mathtt{pk}}$, we have:

$$\Pr[f(\mathsf{Dec}^{\mathsf{FHE}}(\mathtt{sk}, \mathsf{ct}_1), \ldots, \mathsf{Dec}^{\mathsf{FHE}}(\mathtt{sk}, \mathsf{ct}_n)) = \mathsf{Dec}^{\mathsf{FHE}}(\mathtt{sk}, \mathsf{Eval}^{\mathsf{FHE}}(\mathtt{pk}, (\mathsf{ct}_1, \ldots, \mathsf{ct}_n), f))] = 1.$$

Our result will also make use of an additional property of FHE called malicious circuit-privacy, which ensures the correctness and privacy of evaluation even with maliciously generated keys and ciphertexts. To the best of our knowledge, the only known constructions of malicious circuit-private FHE make non-black-box use of an underlying FHE scheme. Thus, we resort to the following weaker notion introduced by [DD22].

Definition 14 (Φ-circuit privacy). *Let $\Phi : \mathcal{F} \to \{0,1\}^*$ be a (leakage) function. We say an FHE scheme is Φ-(maliciously) circuit private if for all λ, there exists an unbounded simulator $\mathcal{S}_{\mathsf{FHE}}$ with one-time oracle access to f such that for all public keys* pk, *ciphertexts $c = (c_1, \ldots, c_n)$, and functions $f \in \mathcal{F}$, the distributions $\mathcal{S}^f_{\mathsf{FHE}}(\mathtt{pk}, c, \Phi(f))$ and $\mathsf{Eval}^{\mathsf{FHE}}(\mathtt{pk}, c, f)$ are computationally indistinguishable.*

The construction given by [DD22] makes only black-box use of an underlying *high-rate* FHE scheme [GH19, BDGM19], and it satisfies $\Phi_{\mathsf{depth},\mathsf{width}}$-circuit privacy, i.e., it leaks only the depth and width of the circuit evaluated. As we show in Theorem 4, this is sufficient for our protocol.

2.6 Probabilistically Checkable Proofs of Proximity (PCPPs)

Informally, for an NP relation $\mathcal{R}$, probabilistically checkable proofs (PCPs) [AS98] are proofs that can be verified by reading only a few bits of the proof. PCPs of Proximity (PCPPs) [BGH$^+$05, BGH$^+$06, Din07] considers a pair language L in which input is of the form (u, v). The first part of the input $u \in \{0,1\}^\ell$ is the explicit part that is read entirely by the verifier, and the second part $v \in \{0,1\}^n$ is the implicit part to which verifier has oracle access (queries to v are counted as part of query complexity). The explicit input is of the form $u = (u', n)$, where n is the length of v. Let $\mathcal{R}$ denote the NP relation corresponding to NP Language L.

Definition 15 (PCPPs). *A probabilistic proof system $(\mathcal{P}, \mathcal{V})$ for an NP relation $\mathcal{R}((u,v),w) \in \mathsf{DTIME}(T(n))$ with parameters: query complexity of $\mathcal{V}$ (Q), runtime of $\mathcal{P}$ (t_p), runtime of $\mathcal{V}$ (t_v), amount of randomness used by $\mathcal{V}$ (r_v), proof length (ℓ_p), soundness error (ϵ) and proximity parameter (δ) is a Probabilistically Checkable Proofs of Proximity (PCPPs) system if the following holds for every pair (u,v):*

- *Input: $\mathcal{P}$ receives $u \in \{0,1\}^\ell, v \in \{0,1\}^n$, and $w \in \{0,1\}^\mu$ as input, and $\mathcal{V}$ receives u as input.*
- *Prover's proof: $\mathcal{P}$ generates a proof π with $|\pi| \leq \ell_p(n)$ in time $\leq t_p(n)$.*
- *Verifier's queries: $\mathcal{V}$ is decomposed into a pair of algorithms: query algorithm* Query *and decision algorithm Φ. $\mathcal{V}$ tosses atmost $r_v = r_v(n)$ many coins; let r denote the randomness obtained as a result of these coin tosses. $\mathcal{V}$ computes* $\mathsf{Query}(u,r)$ *to obtain $Q = Q(n)$ many oracle queries $q_1, \ldots, q_Q$ to v and π. Let $b_1, \ldots, b_Q$ denote the answers to these queries. $\mathcal{V}$ computes $\Phi(u, r, b_1, \ldots, b_Q)$ to obtain output* $\mathsf{ver} = \mathtt{accept}/\mathtt{reject}$. *The runtime of $\mathcal{V}$ is within $t_v(n)$.*
- *Completeness: If $\mathcal{R}((u,v),w) =$* $\mathtt{accept}$*, then $\mathcal{V}$ accepts with probability 1.*
- *Soundness: Suppose (u, v^*) is such that for every v, for which there exists w s.t. $\mathcal{R}((u,v),w) =$* $\mathtt{accept}$*, we have that v^* is δ-far away from v. Then, $\mathcal{V}$ accepts (u, v^*) with probability at most $\epsilon = \epsilon(n)$.*

Succinctly, we refer to such a system as a $(Q, t_p, t_v, r_v, \ell_p, \epsilon, \delta)$-PCPP system.

We recall the following result on the existence of "quasi-optimal" PCPP proof system for any NP relation $\mathcal{R}$, implicit in [BCGT13].[7]

Theorem 3 (Quasilinear PCPPs, [BCGT13]). *Let $\mathcal{R}(\mathbf{x}, w) \in \mathsf{DTIME}(T(n))$ be an NP relation, where $n = |\mathbf{x}|$ and $T : \mathbb{Z}^+ \to \mathbb{Z}^+$. Then there exists a PCPP system $(\mathcal{P}, \mathcal{V})$ for $\mathcal{R}$, with the following parameters:*

- *Query Complexity: $Q = \mathsf{polylog}(T(n))$.*
- *Prover Complexity: $t_p = \tilde{O}((T(n))$.*
- *Verifier Complexity: $t_v = \mathsf{polylog}(T(n))$.*
- *Verifier Randomness Size: $r_v = \mathsf{polylog}(T(n))$.*
- *Proof Size: $\ell_p = \tilde{O}((T(n)))$.*
- *Soundness error: $\epsilon = \mathsf{negl}(n)$.*
- *Proximity parameter: $\delta = 1/\mathsf{polylog}(T(n))$.*

3 Succinct black-box 2PC protocol

In this section, we describe our succinct, constant-round, maliciously secure two-party computation protocol Π_{2PC}, making only black-box use of cryptographic primitives. This is followed by a formal proof that our protocol satisfies privacy with knowledge of outputs (PKO). We recall that the protocol can then be extended to satisfy standard malicious security using the techniques of [IKP10]. Finally, we discuss the round and communication complexity of our protocol.

[7] Theorem 1 in [BCGT13] states this for a PCP, but this can be extended to the PCPP case [BC23]. A similar PCPP with a slightly higher prover complexity appears in [BGH+06].

3.1 Our Construction

Given parties P_1 and P_2 with inputs x_1 and x_2 respectively, our aim is to evaluate a function $y = f(x_1, x_2)$, with communication complexity sublinear (in particular, logarithmic) in the size of the description of f. As explained in Section 1.2, one party (here, P_2) will play the role of $\mathsf{P}^{\mathsf{eval}}$, i.e., perform the actual computation of f and prove, using a PCPP system, that this computation was performed correctly. The other party (here, P_1) will play the role of $\mathsf{P}^{\mathsf{rec}}$, i.e., receive the output y, verify that it was computed correctly, and finally send y in the clear to P_2. A high level description of Π_{2PC} now follows.

Our protocol requires setup for an FHE scheme and for P_2 to make vector commitments. The required keys are sampled by P_1 in an initialization phase.

The next phase handles processing of P_1's input x_1. Here, we essentially require P_1 to communicate an FHE ciphertext of x_1 to P_2, but in such a manner that x_1 is extractable by an efficient simulator. To this end, for $n = O(\lambda)$ and $t = \lfloor \frac{n-1}{4} \rfloor$, P_1 locally generates t-out-of-n VSS shares (i.e., views) of x_1, and sends P_2 an FHE ciphertext of each view, along with an extractable commitment containing that view and the randomness used to encrypt it. P_1 then opens a random set of t commitments.[8] P_2 verifies that the opened views are mutually consistent, and that their corresponding ciphertexts are generated honestly using the opened randomness. If these checks are successful, P_2 uses all n ciphertexts to reconstruct (under FHE) P_1's input x_1.

Having obtained an encryption of x_1, P_2 now encrypts its own input x_2 under FHE and homomorphically evaluates $y = f(x_1, x_2)$, thus obtaining an encryption of y, which it sends to P_1.

In the next phase, P_2 generates a PCPP proof that y was evaluated honestly, as follows. It first encodes x_2 into a codeword X_2. It then computes, under FHE, a witness w for the correct decoding of X_2 to x_2 *and* the correct evaluation of $y = f(x_1, x_2)$. This is followed by an execution of the PCPP prover $\mathcal{P}$ (again, under FHE) to obtain proof π that w is a valid witness for the above computation of y from x_1 and some X_2' (which is δ-close to X_2, and thus also decodes to x_2). To enable verification by P_1 (later in the protocol), and extraction of x_2 by an efficient simulator, P_2 now generates t-out-of-n VSS views (under FHE) of each location of X_2 and π. It finally sends P_1 an extractable commitment of the ith set of encrypted views of X_2, and a vector commitment of the ith set of encrypted views of π, for each $i \in [n]$.

Next, using randomness received from P_1, P_2 computes (under FHE) the queries for verification of π. Each query thus generated asks for either a location of π or a location of X_2. In the former case, P_2 selects (under FHE) the views of the corresponding location of π and their proofs of vector opening. In the latter case, P_2 selects (under FHE) the views of the corresponding location of X_2.

Now, P_2 runs the verifier decision algorithm locally, under FHE, in the MPC-in-the-head paradigm [IKOS10]. Specifically, P_2 locally emulates n players with the above queried views as private inputs and x_1, y, r_M as public inputs. Treating the decision algorithm Φ of the PCPP verifier as a relation, with its "witness" (queried locations of π and X_2) shared among the n players by VSS, P_2 runs a robust and secure MPC protocol "in the head" to output as Φ would, on the reconstruction of π and X_2 at the queried locations. P_2 commits the views thus generated.

In the final verification phase, for a random set of t indices, P_2 opens the corresponding views of both X_2 and the MPC-in-the-head execution, and it sends P_1 the proofs of vector opening for the corresponding (encrypted) views of the queried locations of π. P_1 then verifies that all opened views are mutually consistent and all vector openings are valid. If successful, P_1 concludes that the computation was done correctly. P_1 obtains y by decryption and sends it to P_2.

We now provide a detailed description of our protocol. We list the parameters used in our construction in Figure 4. We present our protocol in Figure 5.

[8] This random set of indices can be generated by a black-box 2PC protocol [ORS15].

Parameters

- $n = O(\lambda)$ and $t = \lfloor \frac{n-1}{4} \rfloor$.
- Description of function $f : \{0,1\}^{\ell_1} \times \{0,1\}^{\ell_2} \to \{0,1\}^{\ell_o}$, where ℓ_1, ℓ_2 and ℓ_o denotes the length of input from party P_1, the length of input from party P_2, and the output length ℓ_o respectively.
- Let $\mathcal{E}$ be a (k, K, δ')-Reed-Muller encoding scheme, where $k = \ell_2$, $K = O(k^{1+\alpha})$, $0 < \delta < \frac{1}{2}$ is a constant, for any constant $\alpha > 0$.
- Let $(\mathsf{Share}, \mathsf{Reconstruct})$ be an information theoretic t-out-of-n verifiable secret sharing (VSS) scheme with reconstruction function recons.
- Let $\Pi_{\mathsf{com}} = (\mathcal{C}, \mathcal{R})$ be a standard commitment scheme.
- Let $\Pi_{\mathsf{comExt}} = (\mathcal{C}, \mathcal{R})$ be an extractable commitment scheme.
- Let $\mathsf{VC} = (\mathsf{Gen}, \mathsf{Com}, \mathsf{Open}, \mathsf{Verify})$ be a vector commitment scheme.
- Let $(\mathsf{Gen}^{\mathsf{FHE}}, \mathsf{Enc}^{\mathsf{FHE}}, \mathsf{Dec}^{\mathsf{FHE}}, \mathsf{Eval}^{\mathsf{FHE}})$ be a $\Phi_{\mathsf{depth},\mathsf{width}}$-circuit private fully homomorphic encryption (FHE) scheme.
- Let $(\mathcal{P}, \mathcal{V} = (\mathsf{Query}, \Phi))$ be an information-theoretic quasilinear PCPP system for the NP relation $\mathcal{R}_f(\cdot)$, and let this be a $(Q, t_p, t_v, r_v, \ell_p, \epsilon, \delta)$-PCPP system. The relation $\mathcal{R}_f(\cdot)$ takes as input a pair (u, v), where $u = (x_1, y, \ell_2)$, $v = X_2$, and as witness w, the trace of the computation M described below:
  1. On input (x_1, y, ℓ_2, X_2), compute $x_2 \leftarrow \mathcal{E}.\mathsf{Decode}(X_2)$.
  2. Verify that x_2 is of length ℓ_2.
  3. Compute $y' \leftarrow f(x_1, x_2)$.
  4. Output $y' = y$.

  $\mathcal{R}_f((u, v), w)$ outputs `accept` if the trace of the computation is consistent with the input pair (u, v), and the computation outputs `accept`; otherwise, $\mathcal{R}_f((u, v), w)$ outputs `reject`.
- Let Π_g be an n-player MPC protocol, which takes public input (u, r) and private input $(z_{i,k})_{k \in Q}$ from each player p_i $(i \in [n])$, and outputs to each player $g(u, r, (z_{1,k})_{k \in [Q]}, \ldots, (z_{n,k})_{k \in [Q]}) = \Phi(u, r, (b_k \leftarrow \mathsf{recons}(z_{1,k}, \ldots, z_{n,k}))_{k \in [Q]})$, where Φ is the decision algorithm of PCPP verifier $\mathcal{V}$. Let Π_g have perfect t-robustness (in the malicious model) and perfect/statistical t-privacy (in the semi-honest model).

Fig. 4: Parameters of succinct black-box protocol Π_{2PC} for 2-party computation.

Protocol Π_{2PC}

Inputs. P_1 and P_2 hold respective inputs $x_1 \in \{0,1\}^{\ell_1}$ and $x_2 \in \{0,1\}^{\ell_2}$.

Initialization Phase.

1. P_1 samples $\mathsf{ck} \xleftarrow{\$} \mathsf{VC.Setup}(1^\lambda)$ and $(\mathtt{sk}, \mathtt{pk}) \xleftarrow{\$} \mathsf{Gen}^{\mathsf{FHE}}(1^\lambda)$, and sends $(\mathsf{ck}, \mathtt{pk})$ to P_2.

Processing input of P_1.

2. P_1 does the following:
   (a) Locally emulate a dealer with input x_1, along with n players $p_1, \ldots, p_n$, and run the VSS sharing protocol Share "in the head" to obtain n views $\mathsf{View}_1^{(x_1)}, \ldots, \mathsf{View}_n^{(x_1)}$ belonging to $p_1, \ldots, p_n$ respectively.
   (b) For each $i \in [n]$, sample random r_i and compute $\mathsf{ct}_i^{(x_1)} \leftarrow \mathsf{Enc}^{\mathsf{FHE}}(\mathtt{pk}, \mathsf{View}_i^{(x_1)}; r_i)$.
   (c) Send $\left(\mathsf{ct}_i^{(x_1)}\right)_{i \in [n]}$ to P_2, and send an extractable commitment $\mathtt{com}_i^{(x_1)}$ of $(\mathsf{View}_i^{(x_1)} \| r_i)$ to P_2 for each $i \in [n]$, using Π_{comExt}.
3. P_1 and P_2 invoke the $\mathcal{F}_{\mathsf{2PC}}$ functionality to compute a function that randomly samples indices $i_1^{(2)}, \ldots, i_t^{(2)} \xleftarrow{\$} [n]$. Let $\mathcal{I}^{(2)}$ denote the set $\{i_1^{(2)}, \ldots, i_t^{(2)}\}$.
4. For each $i \in \mathcal{I}^{(2)}$, P_1 decommits $\mathtt{com}_i^{(x_1)}$ to $(\mathsf{View}_i^{(x_1)} \| r_i)$, using Π_{comExt}.
5. P_2 checks that the views $(\mathsf{View}_i^{(x_1)})_{i \in \mathcal{I}^{(2)}}$ of VSS are valid and mutually consistent, and that $\mathsf{ct}_i = \mathsf{Enc}^{\mathsf{FHE}}(\mathtt{pk}, \mathsf{View}_i^{(x_1)}; r_i)$ for each $i \in \mathcal{I}^{(2)}$. If successful, P_2 continues to the next step; else it aborts.
6. P_2 locally emulates n players $p_1, \ldots, p_n$ with inputs $\mathsf{ct}_1^{(x_1)}, \ldots, \mathsf{ct}_n^{(x_1)}$ respectively, and runs (under FHE, using $\mathsf{Eval}^{\mathsf{FHE}}(\mathtt{pk}, \cdot, \cdot)$ for computations) the VSS reconstruction protocol $\mathsf{Reconstruct}$ "in the head" to obtain an FHE ciphertext $\mathsf{ct}^{(x_1)}$ corresponding to P_1's reconstructed input x_1.

Computation of function f.

7. P_2 encrypts its input x_2 under FHE, i.e., $\mathsf{ct}^{(x_2)} \xleftarrow{\$} \mathsf{Enc}(\mathtt{pk}, x_2)$, and performs the homomorphic evaluation of the function f, i.e., $\mathsf{ct}^{(y)} \xleftarrow{\$} \mathsf{Eval}^{\mathsf{FHE}}(\mathtt{pk}, (\mathsf{ct}^{(x_1)}, \mathsf{ct}^{(x_2)}), f)$. Let y denote the plaintext encrypted under $\mathsf{ct}^{(y)}$.

Computation of PCPP proof, and its processing.

8. P_2 does the following:
   (a) Compute $X_2 = (X_{2,1}, \ldots, X_{2,K}) \leftarrow \mathcal{E}.\mathsf{Encode}(x_2)$. For each $i \in [K]$, compute $\mathsf{ct}_i^{(X_2)} \xleftarrow{\$} \mathsf{Enc}(\mathtt{pk}, X_{2,i})$. Let $\mathsf{CT}^{(X_2)} = (\mathsf{ct}_i^{(X_2)})_{i\in[K]}$ denote a vector of ciphertexts.
   (b) Encrypt ℓ_2 under FHE, to get $\mathsf{ct}^{(\ell_2)} \xleftarrow{\$} \mathsf{Enc}(\mathtt{pk}, \ell_2)$.
   (c) Using ciphertexts $\mathsf{ct}^{(x_1)}$, $\mathsf{ct}^{(\ell_2)}$, $\mathsf{ct}^{(y)}$, and $\mathsf{CT}^{(X_2)}$, compute witness w, i.e., the trace of the computation M (described in Figure 4), under FHE. Let the resultant ciphertext be denoted by $\mathsf{ct}^{(w)}$.
   (d) Using ciphertexts $\mathsf{ct}^{(x_1)}$, $\mathsf{ct}^{(\ell_2)}$, $\mathsf{ct}^{(y)}$, $\mathsf{CT}^{(X_2)}$, and $\mathsf{ct}^{(w)}$, execute the prover $\mathcal{P}$ of PCPP system under FHE with explicit input $u \leftarrow (x_1, y, \ell_2)$, implicit input $v = X_2$, and witness w, to obtain PCPP proof π, having length ℓ_p. In this evaluation, bits of the PCPP proof are encrypted individually. As a result, P_2 obtains a vector of ciphertexts $\mathsf{CT}^{(\pi)} = \left(\mathsf{ct}_1^{(\pi)}, \ldots, \mathsf{ct}_{\ell_p}^{(\pi)}\right)$.
   (e) For each $j \in [K]$, locally emulate a dealer with input $\mathsf{ct}_j^{(X_2)}$, along with n players $p_1, \ldots, p_n$, and run (under FHE) the VSS sharing protocol Share "in the head" to obtain ciphertexts $(\mathsf{ct}_{1,j}^{(X_2)}, \ldots, \mathsf{ct}_{n,j}^{(X_2)})$, where $\mathsf{ct}_{i,j}^{(X_2)}$ encrypts the view $\mathsf{View}_{i,j}^{(X_2)}$, for $i \in [n]$. Let $\mathsf{CT}_i^{(X_2)}$ denote the vector of ciphertexts $\left(\mathsf{ct}_{i,1}^{(X_2)}, \ldots, \mathsf{ct}_{i,K}^{(X_2)}\right)$, for $i \in [n]$.
   (f) For each $j \in [\ell_p]$, locally emulate a dealer with input $\mathsf{ct}_j^{(\pi)}$, along with n players $p_1, \ldots, p_n$, and run (under FHE) the VSS sharing protocol Share "in the head" to obtain ciphertexts $(\mathsf{ct}_{1,j}^{(\pi)}, \ldots, \mathsf{ct}_{n,j}^{(\pi)})$, where $\mathsf{ct}_{i,j}^{(\pi)}$ encrypts the view $\mathsf{View}_{i,j}^{(\pi)}$, for $i \in [n]$. Let $\mathsf{CT}_i^{(\pi)}$ denote the vector of ciphertexts $\left(\mathsf{ct}_{i,1}^{(\pi)}, \ldots, \mathsf{ct}_{i,\ell_p}^{(\pi)}\right)$, for $i \in [n]$.
   (g) For each $i \in [n]$, compute $(\mathtt{com}_i^{(\pi)}, \mathsf{st}_i^{(\pi)}) \xleftarrow{\$} \mathsf{VC.Com}(\mathsf{ck}, \mathsf{CT}_i^{(\pi)})$.
   (h) Send $(\mathsf{ct}^{(y)}, (\mathtt{com}_i^{(\pi)})_{i\in[n]})$ to P_1, and send an extractable commitment $\mathtt{com}_i^{(X_2)}$ of $\mathsf{CT}_i^{(X_2)}$ for each $i \in [n]$, using Π_{comExt}.

Generating queries of PCPP Verifier.

9. P_1 samples $r_M \xleftarrow{\$} \{0,1\}^{r_v}$, and sends r_M to P_2.
10. Using $\mathsf{ct}^{(x_1)}$, $\mathsf{ct}^{(y)}$, $\mathsf{ct}^{(\ell_2)}$ and $\mathsf{ct}^{(r_M)} \xleftarrow{\$} \mathsf{Enc}^{\mathsf{FHE}}(\mathtt{pk}, r_M)$, P_2 runs the query algorithm of PCPP verifier $\mathcal{V}$, i.e., $\mathsf{Query}(\cdot)$, under FHE with inputs $(u = (x_1, y, \ell_2), r_M)$ to obtain queries $q_1, \ldots, q_Q \in [\ell_p + K]$. For a query q, if $q \le \ell_p$, the query refers to the q^{th} bit of π. Otherwise, it refers to the $(q - \ell_p)^{\text{th}}$ bit of X_2. In the FHE evaluation described above, each query is encrypted individually. As a result, P_2 obtains a vector of ciphertexts $\mathsf{CT}^{(q)} = (\mathsf{ct}_1^{(q)}, \ldots, \mathsf{ct}_Q^{(q)})$.

Building verifiable responses to queries of PCPP Verifier

11. For each $i \in [n]$, P_2 does the following:
    (a) For each index $j \in [\ell_p]$, compute proof of vector opening $\sigma_{i,j} \leftarrow \mathsf{VC.Open}(\mathsf{ck}, \mathsf{ct}_{i,j}^{(\pi)}, j, \mathsf{st}_i^{(\pi)})$ and compute $\widehat{\nu}_{i,j} \xleftarrow{\$} \mathsf{Enc}^{\mathsf{FHE}}(\mathtt{pk}, (\mathsf{ct}_{i,j}^{(\pi)}, \sigma_{i,j}))$. Prepare a list of tuples $O_i \leftarrow (\widehat{\nu}_{i,j})_{j\in[\ell_p]}$.
    (b) Consider a selector function which takes as input i and a list $Z = (Z_1, \ldots, Z_\ell)$ of length ℓ, and outputs Z_i if $i \in [\ell]$, otherwise it outputs $\perp$. For each $k \in [Q]$:
       i. Using ciphertexts $\mathsf{ct}_k^{(q)}$, $\mathsf{CT}_i^{(\pi)}$ and $\mathsf{CT}_i^{(X_2)}$, compute selector function under FHE with input index q_k and list $\left(\mathsf{View}_{i,1}^{(\pi)}, \ldots, \mathsf{View}_{i,\ell_p}^{(\pi)}, \mathsf{View}_{i,1}^{(X_2)}, \ldots, \mathsf{View}_{i,K}^{(X_2)}\right)$ to obtain $\psi_{i,k}$. If $q_k \le \ell_p$, $\psi_{i,k} = \mathsf{View}_{i,q_k}^{(\pi)}$; else $\psi_{i,k} = \mathsf{View}_{i,q_k-\ell_p}^{(X_2)}$. Let the resultant ciphertext be denoted by $\widehat{\psi}_{i,k}$.
       ii. Similarly, using ciphertexts $\mathsf{ct}_k^{(q)}$ and O_i, compute selector function under FHE with input index q_k and list $((\mathsf{ct}_{i,j}^{(\pi)}, \sigma_{i,j}))_{j\in[\ell_p]}$ to obtain $\rho_{i,k}$. If $q_k \le \ell_p$, $\rho_{i,k} = (\mathsf{ct}_{i,q_k}^{(\pi)}, \sigma_{i,q_k})$; else $\rho_{i,k} = \perp$. Let the resultant ciphertext be denoted by $\widehat{\rho}_{i,k}$.
    (c) Let A_i and R_i denote the lists $(\widehat{\psi}_{i,k})_{k\in[Q]}$ and $(\widehat{\rho}_{i,k})_{k\in[Q]}$ respectively.

MPC-in-the-head emulation of PCPP verifier.

12. Using ciphertexts $\mathsf{ct}^{(x_1)}, \mathsf{ct}^{(y)}, \mathsf{ct}^{(\ell_2)}, \mathsf{ct}^{(r_M)}$ and A_i, P_2 locally emulates n players $p_1, \ldots, p_n$, and runs (under FHE) protocol Π_g "in the head" with public inputs $u = (x_1, y, \ell_2)$ and r_M, and

private input of party p_i as $(\psi_{i,k})_{k\in[Q]}$, for $i \in [n]$.[a] P_2 obtains ciphertexts $(\mathsf{ct}_1^{(V)}, \ldots, \mathsf{ct}_n^{(V)})$, where $\mathsf{ct}_i^{(V)}$ encrypts the view of party p_i, i.e., $\mathsf{View}_i^{(V)}$, for $i \in [n]$.

13. P_2 sends a commitment $\mathtt{com}_i^{(V)}$ of $\mathsf{ct}_i^{(V)}$ to P_1 for each $i \in [n]$, using $\Pi_{\mathtt{com}}$.

<u>Verification of MPC-in-the-head emulation of PCPP verifier.</u>

14. P_1 and P_2 invoke the $\mathcal{F}_{\mathsf{2PC}}$ functionality to compute a function that randomly samples indices $i_1^{(1)}, \ldots, i_t^{(1)} \overset{\$}{\leftarrow} [n]$. Let $\mathcal{I}^{(1)}$ denote the set $\{i_1^{(1)}, \ldots, i_t^{(1)}\}$.
15. For each $i \in \mathcal{I}^{(1)}$,
    (a) P_2 decommits $\mathtt{com}_i^{(V)}$ to $\mathsf{ct}_i^{(V)}$ using $\Pi_{\mathtt{com}}$. Also, P_2 decommits $\mathtt{com}_i^{(X_2)}$ to $\mathsf{CT}_i^{(X_2)}$ using $\Pi_{\mathtt{comExt}}$.
    (b) P_2 sends R_i to P_1.
16. P_1 decrypts $\mathsf{ct}^{(y)}$ to obtain the output $y \leftarrow \mathsf{Dec}^{\mathsf{FHE}}(\mathtt{sk}, \mathsf{ct}^{(y)})$, and it does the following (if any check fails, P_1 aborts):
    (a) For each $i \in \mathcal{I}^{(1)}$:
        i. Execute $\mathsf{Query}(u = (x_1, y, \ell_2), r_M)$ to obtain queries $q_1, \ldots, q_Q$.
        ii. Parse $\mathsf{CT}_i^{(X_2)}$ as $(\mathsf{ct}_{i,1}^{(X_2)}, \ldots, \mathsf{ct}_{i,K}^{(X_2)})$. For each $j \in [K]$, compute $\mathsf{View}_{i,j}^{(X_2)} \leftarrow \mathsf{Dec}^{\mathsf{FHE}}(\mathtt{sk}, \mathsf{ct}_{i,j}^{(X_2)})$.
        iii. Parse R_i as $(\widehat{\rho}_{i,k})_{k\in[Q]}$. For each $k \in [Q]$, if $q_k \leq \ell_p$, compute $(\mathsf{ct}_{i,q_k}^{(\pi)}, \sigma_{i,q_k}) \leftarrow \mathsf{Dec}^{\mathsf{FHE}}(\mathtt{sk}, \widehat{\rho}_{i,k})$ and check if $\mathsf{VC.Verify}(\mathsf{ck}, \mathtt{com}_i^{(\pi)}, q_k, \mathsf{ct}_{i,q_k}^{(\pi)}, \sigma_{i,q_k}) = \mathtt{accept}$. Compute $\mathsf{View}_{i,q_k}^{(\pi)} \leftarrow \mathsf{Dec}^{\mathsf{FHE}}(\mathtt{sk}, \mathsf{ct}_{i,q_k}^{(\pi)})$.
        iv. Compute $\mathsf{View}_i^{(V)} \leftarrow \mathsf{Dec}^{\mathsf{FHE}}(\mathtt{sk}, \mathsf{ct}_i^{(V)})$.
        v. Check the consistency of individual view $\mathsf{View}_i^{(V)}$. Also, check consistency of $\mathsf{View}_i^{(V)}$ with respect to list $(\mathsf{View}_{i,j}^{(X_2)})_{j\in[K]}$ and $(\mathsf{View}_{i,k}^{(\pi)})_{k\in[\ell_p]}$ for each k, where $q_k \leq \ell_p$.
    (b) Check the global consistency of the views $(\mathsf{View}_i^{(V)})_{i\in\mathcal{I}^{(1)}}$.
    (c) Check that the output of Π_g is $\mathtt{accept}$.
    (d) Send y to P_2, and output y.
17. P_2 outputs y.

[a] Recall the definition of Π_g as given in Figure 4, and observe that, if computed honestly on inputs $u = (x_1, y, \ell_2)$, $r = r_M$, and $z_{i,k} = \psi_{i,k}$ for each $i \in [n], k \in [Q]$, we get, for all $k \in [Q]$, $b_k = \pi_{q_k}$ if $q_k \leq \ell_p$; else $b_k = X_{2,q_k-\ell_p}$ as required.

Fig. 5: Succinct black-box protocol Π_{2PC} for 2-party computation.

A note on handling maliciously formed ciphertexts from P_1. In step 2(c) of the protocol, P_1 sends ciphertexts $\mathsf{ct}_1^{(x_1)}, \ldots, \mathsf{ct}_n^{(x_1)}$ to P_2. It is possible (with noticeable probability) that a sublinear (in n) number of these ciphertexts may be maliciously formed, i.e., they may not correspond to any underlying plaintext. As a result, even though we show in the proof of Theorem 4 that at least $n - t$ ciphertexts encrypt VSS views that are mutually consistent, we cannot guarantee that the unique secret x_1^* will be reconstructed in step 6 of the protocol, since the presence of maliciously formed ciphertexts could cause an execution of $\mathsf{Eval}^{\mathsf{FHE}}$ to output something else. This creates a difficulty in simulating the view of a corrupted P_1^*. To address this, we augment our protocol.

We consider a circuit $\mathsf{C}_{\mathsf{Dummy}}$ that takes $v_1, \ldots, v_n$ as input and outputs $\sum_{i=1}^{n}(v_i - v_i)$. On receiving ciphertexts $\mathsf{ct}_1^{(x_1)}, \ldots, \mathsf{ct}_n^{(x_1)}$, P_2 evaluates $\mathsf{ct}_{\mathsf{Dummy}} \overset{\$}{\leftarrow} \mathsf{Eval}^{\mathsf{FHE}}(\mathtt{pk}, (\mathsf{ct}_i^{(x_1)})_{i\in[n]}, \mathsf{C}_{\mathsf{Dummy}})$. If all ciphertexts are well-formed, $\mathsf{ct}_{\mathsf{Dummy}}$ will be an encryption of 0^{ℓ_1}. For any ciphertext ct that is computed by P_2 and may be sent to P_1, where ct encrypts a message of length ℓ, P_2 obtains an encryption of 0^{ℓ} from $\mathsf{ct}_{\mathsf{Dummy}}$, adds it to ct, and updates the ciphertext ct. If the ciphertexts $\mathsf{ct}_1^{(x_1)}, \ldots, \mathsf{ct}_n^{(x_1)}$ are well-formed, then the message encrypted in ct remains the same and the computation remains unchanged. As we will see in the proof, if any of these ciphertexts are maliciously formed, then the view of the corrupt P_1^* is now simulatable due to the Φ-circuit privacy of the FHE scheme [DD22] (see Definition 14).

3.2 Security Analysis

We now provide a proof that our protocol Π_{2PC} satisfies privacy with knowledge of outputs (PKO).

Theorem 4. *Assuming the primitives listed in Figure 4, protocol Π_{2PC} securely realizes functionality $\mathcal{F}_{\mathsf{2PC\text{-}PKO}}$.*

Proof (sketch). Here we provide descriptions of our ideal world simulators for a malicious adversary corrupting one of the two parties, and we give a high-level sketch of the security proof. The complete proof can be found in Appendix B.

Case 1: P_1 is corrupt. Simulator $\mathcal{S}_1$ for malicious party P_1^* is described in Figure 6. $\mathcal{S}_1$ computes $\mathsf{ct}_{\mathsf{Dummy}}$ (see Step 8 of $\mathcal{S}_1$) and sends updated ciphertexts as described in the note at the end of Section 3.1. Moreover, $\mathcal{S}_1$ also updates the ciphertexts with this procedure in all the intermediate hybrids.

Simulator $\mathcal{S}_1$

1. $\mathcal{S}_1$ receives a message of the form $(\mathsf{ck}, \mathsf{pk})$ from P_1^*.
2. $\mathcal{S}_1$ receives a message of the form $(\mathsf{ct}_i^{(x_1)})_{i\in[n]}$ from P_1^*.
3. For all $i \in [n]$, $\mathcal{S}_1$ runs the extractor Ext of extractable commitment scheme Π_{comExt}, with oracle access to P_1^*, to obtain $(\mathsf{com}_i^{(x_1)}, \mathsf{View}_i^{(x_1)*} \parallel r_i^*)$.
4. $\mathcal{S}_1$ simulates $\mathcal{F}_{\mathsf{2PC}}$ and sends the output $i_1^{(2)}, \ldots, i_t^{(2)}$ to P_1^*.
5. $\mathcal{S}_1$ interacts with P_1^* as honest party P_2 in step 4 of the protocol. If any of the checks in step 5 fail, $\mathcal{S}_1$ sends abort to $\mathcal{F}_{\mathsf{2PC\text{-}PKO}}$.
6. $\mathcal{S}_1$ aborts the simulation if Ext failed on more than $t/3$ indices $i \in [n]$, in step 3 of the simulation. Else, let the set of indices on which Ext succeeds in extracting be denoted by $\mathcal{J}$. $\mathcal{S}_1$ aborts the simulation if more than $t/3$ ciphertexts $\mathsf{ct}_i^{(x_1)} \neq \mathsf{Enc}^{\mathsf{FHE}}(\mathsf{pk}, \mathsf{View}_i^{(x_1)*}; r_i^*)$, for $i \in \mathcal{J}$. Else, let $\mathcal{J}' \subseteq \mathcal{J}$ denote the set of indices for which ciphertext $\mathsf{ct}_i^{(x_1)} = \mathsf{Enc}^{\mathsf{FHE}}(\mathsf{pk}, \mathsf{View}_i^{(x_1)*}; r_i^*)$. $\mathcal{S}_1$ locally emulates n players $p_1, \ldots, p_n$ with inputs $\mathsf{View}_1^{(x_1)*}, \ldots, \mathsf{View}_n^{(x_1)*}$, respectively, and it runs the VSS reconstruction protocol $\mathsf{Reconstruct}$ "in the head" to obtain x_1^*. $\mathcal{S}_1$ aborts the simulation if reconstruction fails (which occurs only if there is no set of $n-t$ mutually consistent views $\mathsf{View}_i^{(x_1)*}$).
7. $\mathcal{S}_1$ sends x_1^* to $\mathcal{F}_{\mathsf{2PC\text{-}PKO}}$. $\mathcal{S}_1$ receives y from $\mathcal{F}_{\mathsf{2PC\text{-}PKO}}$.
8. $\mathcal{S}_1$ computes $\mathsf{ct}_{\mathsf{Dummy}} \leftarrow \mathsf{Eval}(\mathsf{pk}, (\mathsf{ct}_i^{(x_1)})_{i\in[n]}, \mathsf{C}_{\mathsf{Dummy}})$.
9. $\mathcal{S}_1$ does the following:
   (a) Compute $\mathsf{ct}^{(y)*} \xleftarrow{\$} \mathsf{Enc}^{\mathsf{FHE}}(\mathsf{pk}, y)$. Recall that in the FHE construction [DD22] we use, ciphertexts leak the width and depth of the evaluated circuit. Therefore, we apply an identity function with the same width and depth as the circuit which reconstructs x_1^* from its views *and* evaluates the function f, on the ciphertext $\mathsf{ct}^{(y)*}$ to obtain an updated ciphertext $\mathsf{ct}^{(y)}$. We prepare all the ciphertexts that are evaluated under FHE in the real world but computed in the plaintext and then encrypted in the simulation, in this manner.
   (b) $\mathcal{S}_1$ randomly samples indices $i_1^{(1)}, \ldots, i_t^{(1)} \xleftarrow{\$} [n]$. Let $\mathcal{I}^{(1)}$ denote the set $\{i_1^{(1)}, \ldots, i_t^{(1)}\}$.
   (c) Set $X_2 = (X_{2,1}, \ldots, X_{2,K}) \leftarrow 0^K$. For $j \in [K]$,
      i. Locally emulate a dealer with input $X_{2,j}$, along with n players $p_1, \ldots, p_n$, and run the VSS sharing protocol Share "in the head" to obtain n views $\mathsf{View}_{1,j}^{(X_2)}, \ldots, \mathsf{View}_{n,j}^{(X_2)}$ of $p_1, \ldots, p_n$ respectively.
      ii. For each $i \in [n]$, compute $\mathsf{ct}_{i,j}^{(X_2)} \xleftarrow{\$} \mathsf{Enc}^{\mathsf{FHE}}(\mathsf{pk}, \mathsf{View}_{i,j}^{(X_2)})$.
   (d) Let $\mathsf{CT}_i^{(X_2)}$ denote the ciphertext vector $\left(\mathsf{ct}_{i,1}^{(X_2)}, \ldots, \mathsf{ct}_{i,K}^{(X_2)}\right)$, for $i \in [n]$.
   (e) Set $\pi = (\pi_1, \ldots, \pi_{\ell_p}) \leftarrow 0^{\ell_p}$. For $j \in [\ell_p]$,
      i. Locally emulate a dealer with input π_j, along with n players $p_1, \ldots, p_n$, and run the VSS sharing protocol Share "in the head" to obtain n views $\mathsf{View}_{1,j}^{(\pi)}, \ldots, \mathsf{View}_{n,j}^{(\pi)}$ of $p_1, \ldots, p_n$ respectively.
      ii. For each $i \in [n]$, compute $\mathsf{ct}_{i,j}^{(\pi)} \xleftarrow{\$} \mathsf{Enc}^{\mathsf{FHE}}(\mathsf{pk}, \mathsf{View}_{i,j}^{(\pi)})$.
   (f) Let $\mathsf{CT}_i^{(\pi)}$ denote the ciphertext vector $\left(\mathsf{ct}_{i,1}^{(\pi)}, \ldots, \mathsf{ct}_{i,\ell_p}^{(\pi)}\right)$, for $i \in [n]$.

(g) For each $i \in [n]$, compute $(\mathtt{com}_i^{(\pi)}, \mathsf{st}_i^{(\pi)}) \xleftarrow{\$} \mathsf{VC.Com}(\mathsf{ck}, \mathsf{CT}_i^{(\pi)})$.

(h) Send $(\mathsf{ct}^{(y)}, (\mathtt{com}_i^{(\pi)})_{i \in [n]})$ to P_1^*, and send an extractable commitment $\mathtt{com}_i^{(X_2)}$ of $\mathsf{CT}_i^{(X_2)}$ for each $i \in [n]$, using Π_{comExt}.

10. $\mathcal{S}_1$ receives r_M from P_1^* and computes verifier queries $\mathcal{Q} = (q_1, \ldots, q_Q) \xleftarrow{\$} \mathsf{Query}(u = (x_1^*, y, \ell_2), r_M)$.
11. For each $i \in [n]$, $\mathcal{S}_1$ does the following:
    (a) For each $k \in [Q]$:
        i. If $q_k \leq \ell_p$, set $\psi_{i,k} \leftarrow \mathsf{View}_{i,q_k}^{(\pi)}$; otherwise $\psi_{i,k} \leftarrow \mathsf{View}_{i,q_k - \ell_p}^{(X_2)}$.
        ii. If $q_k \leq \ell_p$, set $\rho_{i,k} \leftarrow (\mathsf{ct}_{i,q_k}^{(\pi)}, \mathsf{VC.Open}(\mathsf{ck}, \mathsf{ct}_{i,q_k}^{(\pi)}, j, \mathsf{st}_i^{(\pi)}))$; otherwise $\rho_{i,k} \leftarrow \bot$. Compute $\widehat{\rho}_{i,k} \xleftarrow{\$} \mathsf{Enc}^{\mathsf{FHE}}(\mathsf{pk}, \rho_{i,k})$.

    Let R_i denote the list $(\widehat{\rho}_{i,k})_{k \in [Q]}$.
12. Invoke (in the clear) simulator $\mathcal{S}_{\Pi_g}$ corresponding to MPC-in-the-head protocol Π_g, with set of indices of parties $\mathcal{I}^{(1)}$, public inputs $u = (x_1^*, y, \ell_2)$ and r_M, and private input of party p_i as $(\psi_{i,k})_{k \in [Q]}$, for $i \in \mathcal{I}^{(1)}$, and output of the protocol set to $\mathtt{accept}$. $\mathcal{S}_1$ obtains $\{\mathsf{View}_i^{(V)}\}_{i \in \mathcal{I}^{(1)}} \leftarrow \mathcal{S}_{\Pi_g}(\mathcal{I}^{(1)}, (x_1^*, y, \ell_2, r_M), (\psi_{i,j})_{i \in \mathcal{I}^{(1)}, j \in [Q]}, \mathtt{accept})$.
13. For $i \in [n] \setminus \mathcal{I}^{(1)}$, generate arbitrary MPC views $\mathsf{View}_i^{(V)}$.
14. Compute $(\mathsf{ct}_i^{(V)} \leftarrow \mathsf{Enc}^{\mathsf{FHE}}(\mathsf{pk}, \mathsf{View}_i^{(V)}))_{i \in [n]}$.
15. $\mathcal{S}_1$ sends a commitment $\mathtt{com}_i^{(V)}$ of $\mathsf{ct}_i^{(V)}$ to P_1^* for each $i \in [n]$, using $\Pi_{\mathtt{com}}$.
16. Simulate functionality $\mathcal{F}_{\mathsf{2PC}}$ with output $\mathcal{I}^{(1)}$ and send $\mathcal{I}^{(1)}$ to P_1^*.
17. Send $(\mathtt{com}_i^{(V)})_{i \in [n]}$ to P_1^*.
18. For each $i \in \mathcal{I}^{(1)}$,
    (a) $\mathcal{S}_1$ decommits $\mathtt{com}_i^{(V)}$ to $\mathsf{ct}_i^{(V)}$ using $\Pi_{\mathtt{com}}$. Also, $\mathcal{S}_1$ decommits $\mathtt{com}_i^{(X_2)}$ to $\mathsf{CT}_i^{(X_2)}$ using Π_{comExt}.
    (b) $\mathcal{S}_1$ sends R_i to P_1^*.
19. $\mathcal{S}_1$ receives y' from P_1^*, and sends the same to $\mathcal{F}_{\mathsf{2PC\text{-}PKO}}$.

Fig. 6: Simulator $\mathcal{S}_1$ for malicious party P_1^*.

Next, we give a high-level description of our hybrid argument. In $\mathcal{H}_1$, $\mathcal{S}_1$ runs the extractor Ext of scheme Π_{comExt} to obtain $\mathsf{View}_i^{(x_1)*}$ from commitment $\mathtt{com}_i^{(x_1)}$ for at least $n - t$ indices in $[n]$. Through cut-and-choose arguments, the extractability property of scheme Π_{comExt}, and the strong commitment property of the VSS scheme, we show that if all the ciphertexts are well-formed, then $\mathcal{S}_1$ succeeds in extracting input x_1^* of the corrupt P_1^*. In $\mathcal{H}_2$, we use the extracted input x_1^* in the simulation and prove indistinguishability from the perspective of P_1^* by relying on the circuit privacy of the FHE scheme (Definition 14). In $\mathcal{H}_3$, we perform all computations that are supposed to be computed under FHE in the clear, and then encrypt them. Indistinguishability follows from the circuit privacy of the FHE scheme. In $\mathcal{H}_4$, $\mathcal{S}_1$ simulates $\mathcal{F}_{\mathsf{2PC}}$ with an output of its choice from the same distribution. In $\mathcal{H}_5$, $\mathcal{S}_1$ replaces the views $(\mathsf{View}_i^{(V)})_{i \in [n] \setminus \mathcal{I}^{(1)}}$ that are never opened with arbitrary MPC-in-the-head views. Indistinguishability follows from the computational hiding of commitment scheme Π_{com}. In $\mathcal{H}_6$, $\mathcal{S}_1$ invokes the simulator $\mathcal{S}_{\Pi_g}$ of the MPC protocol Π_g to simulate the opened views. Indistinguishability follows from the security of Π_g. In $\mathcal{H}_7$, $\mathcal{S}_1$ replaces the VSS shares of the PCPP proof that will never be opened by P_1^* with VSS shares of 0^{ℓ_p}. Indistinguishability follows from the computational hiding property of vector commitments. In $\mathcal{H}_8$, $\mathcal{S}_1$ replaces the VSS shares of the PCPP proof of each index $i \in \mathcal{I}^{(1)}$ with VSS shares of 0^{ℓ_p}. Indistinguishability follows from the t-privacy of VSS. In $\mathcal{H}_9$ and $\mathcal{H}_{10}$, $\mathcal{S}_1$ proceeds to replace the VSS shares of the codeword X_2 with VSS shares of 0^K in a similar manner as for π. In $\mathcal{H}_{11}$, $\mathcal{S}_2$ uses the output obtained from the $\mathcal{F}_{\mathsf{2PC\text{-}PKO}}$ functionality to simulate the view of P_1^*, replacing a local computation of $y = f(x_1^*, x_2)$ with an invocation of $\mathcal{F}_{\mathsf{2PC\text{-}PKO}}$. Observe that, at this point, $\mathcal{S}_1$'s behavior coincides exactly with that in the ideal-world execution, as given in Figure 6. We conclude that the joint distribution of P_1^*'s view and P_2's output is identical in real and ideal worlds.

Case 2: P_2 is corrupt. We construct a simulator $\mathcal{S}_2$ for malicious party P_2^*, as shown in Figure 7.

Simulator $\mathcal{S}_2$

1. $\mathcal{S}_2$ samples $\mathsf{ck} \xleftarrow{\$} \mathsf{VC.Setup}(1^\lambda)$ and $(\mathtt{sk}, \mathtt{pk}) \xleftarrow{\$} \mathsf{Gen}^{\mathsf{FHE}}(1^\lambda)$, and sends $(\mathsf{ck}, \mathtt{pk})$ to P_2^*.
2. $\mathcal{S}_2$ randomly samples indices $i_1^{(2)*}, \ldots, i_t^{(2)*} \xleftarrow{\$} [n]$. Let $\mathcal{I}^{(2)*}$ denote the set $\{i_1^{(2)*}, \ldots, i_t^{(2)*}\}$.
3. $\mathcal{S}_2$ does the following:
   (a) Locally emulate a dealer with input a dummy all-zeros string 0^{ℓ_1}, along with n players $p_1, \ldots, p_n$, and run the VSS sharing protocol Share to obtain n views $\mathsf{View}_1^{(x_1)*}, \ldots, \mathsf{View}_n^{(x_1)*}$.
   (b) Sample $\mathtt{sk}$, $\mathtt{pk}$, $(r_i)_{i\in[n]}$ as in Π_{2PC}, and for each $i \in [n]$, compute $\mathsf{ct}_i^{(x_1)*} \xleftarrow{\$} \mathsf{Enc}^{\mathsf{FHE}}(\mathtt{pk}, \mathsf{View}_i^{(x_1)*}; r_i)$.
   (c) Send $\left(\mathsf{ct}_i^{(x_1)*}\right)_{i\in[n]}$ to P_2^*, and send extractable commitments $\left(\mathtt{com}_i^{(x_1)*}\right)_{i\in[n]}$ of $(\mathsf{View}_i^{(x_1)*} \parallel r_i)_{i\in[n]}$ using Π_{comExt}.
4. $\mathcal{S}_2$ emulates the $\mathcal{F}_{\mathsf{2PC}}$ functionality to output $\mathcal{I}^{(2)*}$ to P_2^*.
5. $\mathcal{S}_2$ decommits $(\mathtt{com}_i^{(x_1)*})_{i\in\mathcal{I}^{(2)*}}$ to $(\mathsf{View}_i^{(x_1)*} \parallel r_i)_{i\in\mathcal{I}^{(2)*}}$.
6. $\mathcal{S}_2$ receives a message of the form $(\mathsf{ct}^{(y)*}, \left(\mathtt{com}_i^{(\pi)*}\right)_{i\in[n]})$ from P_2^*, and for each $i \in [n]$, it runs the extractor Ext of extractable commitment scheme Π_{comExt}, with oracle access to P_2^*, to obtain $(\mathtt{com}_i^{(X_2)*}, \mathsf{CT}_i^{(X_2)*})$.
7. $\mathcal{S}_2$ interacts with P_2^* exactly as party P_1 interacts with P_2 in Steps 9 to 16(c) of protocol Π_{2PC}.
8. $\mathcal{S}_2$ aborts the simulation if Ext failed on more than $t/2$ indices $i \in [n]$, in Step 5 of $\mathcal{S}_2$ above.
9. For each $i \in [n]$, $\mathcal{S}_2$ parses $\mathsf{CT}_i^{(X_2)*}$ as $\left(\mathsf{ct}_{i,1}^{(X_2)*}, \ldots, \mathsf{ct}_{i,K}^{(X_2)*}\right)$, and for each $j \in [K]$, computes $\mathsf{View}_{i,j}^{(X_2)*} \leftarrow \mathsf{Dec}^{\mathsf{FHE}}(\mathtt{sk}, \mathsf{ct}_{i,j}^{(X_2)*})$.
10. For each $j \in [K]$, $\mathcal{S}_2$ locally emulates n players $p_1, \ldots, p_n$ with inputs $\mathsf{View}_{1,j}^{(X_2)*}, \ldots, \mathsf{View}_{n,j}^{(X_2)*}$ respectively, and it runs the VSS reconstruction protocol $\mathsf{Reconstruct}$ "in the head" to obtain $X_{2,j}^*$. $\mathcal{S}_2$ aborts the simulation if reconstruction fails (which occurs only if there is no set of $n-t$ mutually consistent views $\mathsf{View}_{i,j}^{(X_2)*}$).
11. $\mathcal{S}_2$ decodes $X_2^* \leftarrow (X_{2,1}^*, \ldots, X_{2,K}^*)$ to get $x_2^* \leftarrow \mathcal{E}.\mathsf{Decode}(X_2^*)$.
12. $\mathcal{S}_2$ sends x_2^* to $\mathcal{F}_{\mathsf{2PC\text{-}PKO}}$ and receives $y^* = f(x_1, x_2^*)$.
13. $\mathcal{S}_2$ sends y^* to P_2^* and $\mathcal{F}_{\mathsf{2PC\text{-}PKO}}$.

Fig. 7: Simulator $\mathcal{S}_2$ for malicious party P_2^*.

We now give a high level overview of our hybrid argument. We start with a real-world execution $\mathcal{H}_0$ where $\mathcal{S}_2$ runs Π_{2PC} using input x_1. In $\mathcal{H}_1$, $\mathcal{S}_2$ runs extraction, decryption, reconstruction, and decoding on the committed (encrypted) views of X_2, to obtain P_2^*'s input x_2^*. This succeeds with overwhelming probability, since most of the commitments must be well-formed (and thus extractable), and most of the underlying views must be mutually consistent, or else P_2^* would be caught by cut-and-choose in both $\mathcal{H}_0$ and $\mathcal{H}_1$. Moreover, P_2^*'s view is indistinguishable by Definition 4. In $\mathcal{H}_2$, instead of outputting y to P_2^* in the end as per Π_{2PC}, $\mathcal{S}_2$ invokes $\mathcal{F}_{\mathsf{2PC\text{-}PKO}}$ and outputs $y^* = f(x_1, x_2^*)$. By the soundness of our PCPP + MPC-in-the-head proof, it can be shown that y and y^* are indistinguishable. In $\mathcal{H}_3$, $\mathcal{S}_2$ simulates $\mathcal{F}_{\mathsf{2PC}}$ with a uniformly random output $\mathcal{I}^{(2)}$. In $\mathcal{H}_4$, using knowledge of $\mathcal{I}^{(2)*}$, $\mathcal{S}_2$ replaces the $n-t$ commitments of views of x_1 which will not be opened with those of 0^{ℓ_1} instead. Due to the hiding property of the commitment scheme, P_2^* cannot distinguish. In $\mathcal{H}_5$, $\mathcal{S}_2$ replaces the $n-t$ ciphertexts of views of x_1, corresponding to the same unopened views as before, with those of 0^{ℓ_1} instead. Due to the CPA-security of the FHE scheme, P_2^* cannot distinguish, and moreover an honest P_2 would now start reconstructing $\mathcal{S}_2$'s input (under FHE) to be 0^{ℓ_1} from its $n-t$ consistent views. Finally, in $\mathcal{H}_6$, $\mathcal{S}_2$ replaces even the commitments and ciphertexts that do get opened with those of 0^{ℓ_1}. By the t-privacy of the VSS scheme, the opened views are indistinguishable from those in $\mathcal{H}_5$ for P_2^*. Observe that, at this point, $\mathcal{S}_2$'s behavior coincides exactly with that in the ideal-world execution, as given in Figure 7. We conclude that the joint distribution of P_2^*'s view and P_1's output (y and y^*, respectively) is identical in the real and ideal worlds.

3.3 Complexity Analysis

We instantiate the primitives listed in Figure 4 using the corresponding instantiations discussed in Section 2, and thus provide the round and communication complexities of our protocol Π_{2PC}. Step-wise calculations of communication rounds and costs can be found in Appendix C.

Round complexity. The number of rounds of communication required in Π_{2PC} as described in Figure 5, without any additional parallelization, is 21. However, we are able to compress this to a final round complexity of *14 rounds*, by performing some rounds of $\mathcal{F}_{\mathsf{2PC}}$ invocations in parallel with previous rounds of Π_{2PC} (see Appendix C for more details).

Communication complexity. The total asymptotic communication complexity of Π_{2PC} is $O\big((\ell_1 + \ell_2^{1+\alpha} + \ell_o + \mathsf{polylog}(|f|)) \cdot \mathsf{poly}(\lambda)\big)$ (see Appendix C). Clearly, Π_{2PC} is succinct, as it achieves a communication complexity only polylogarithmic in the size of the circuit computing function f. Moreover, our communication complexity is *almost linear* in the sizes of the inputs and output of f.

3.4 Conclusion

Note that applying the technique of [IKP10], for extension from PKO to malicious security with abort, affects neither the round complexity nor the asymptotic communication complexity of Π_{2PC}, as it only requires computing MACs on the output y, using private MAC keys given as input to the 2PC, and sending these MACs to the parties along with the output. The following corollary is immediate from Theorem 4 and the discussion above.

Corollary 1. *Assuming high-rate fully homomorphic encryption, enhanced trapdoor permutations, and collision-resistant hash functions, there exists a black-box* 14-round *maliciously secure 2PC protocol that computes any function f (having input sizes ℓ_1, ℓ_2 and output size ℓ_o) with communication complexity $O\big((\ell_1 + \ell_2^{1+\alpha} + \ell_o + \mathsf{polylog}(|f|)) \cdot \mathsf{poly}(\lambda)\big)$, where λ is the security parameter and α is any positive constant.*

Acknowledgements. We would like to thank anonymous Eurocrypt reviewers for their helpful comments. M. Ciampi was partially supported by the Input Output Research Hub (IORH) of the University of Edinburgh. R. Ostrovsky was supported in part by NSF grants CNS-2246355, CCF-2220450, US-Israel BSF grant 2022370, and by Sunday Group.

References

ABG$^+$20. Benny Applebaum, Zvika Brakerski, Sanjam Garg, Yuval Ishai, and Akshayaram Srinivasan. Separating two-round secure computation from oblivious transfer. In Thomas Vidick, editor, *ITCS 2020*, volume 151, pages 71:1–71:18. LIPIcs, January 2020.

ABJ$^+$19. Prabhanjan Ananth, Saikrishna Badrinarayanan, Aayush Jain, Nathan Manohar, and Amit Sahai. From FE combiners to secure MPC and back. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part I*, volume 11891 of *LNCS*, pages 199–228, December 2019.

ADT11. Giuseppe Ateniese, Emiliano De Cristofaro, and Gene Tsudik. (If) size matters: Size-hiding private set intersection. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 156–173, March 2011.

AH87. William Aiello and Johan Håstad. Perfect zero-knowledge languages can be recognized in two rounds. In *28th FOCS*, pages 439–448. IEEE Computer Society Press, October 1987.

AS98. Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of np. *Journal of the ACM (JACM)*, 45(1):70–122, 1998.

BC23. Eli Ben-Sasson and Alessandro Chiesa. Personal communication, 2023.

BCGT13. Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, and Eran Tromer. On the concrete efficiency of probabilistically-checkable proofs. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 585–594. ACM Press, June 2013.

BCPR. Ahmad Al Badawi, David Bruce Cousins, Yuriy Polyakov, and Kurt Rohloff. Hardware Acceleration of Fully Homomorphic Encryption: Making Privacy-Preserving Machine Learning Practical. Accessed: 2025-02-22.

BDGM19. Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Leveraging linear decryption: Rate-1 fully-homomorphic encryption and time-lock puzzles. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 407–437, December 2019.

BDOZ11. Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 169–188, May 2011.

BG01. Boaz Barak and Oded Goldreich. Universal arguments and their applications. Cryptology ePrint Archive, Report 2001/105, 2001.

BGH$^+$05. Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Short pcps verifiable in polylogarithmic time. In *20th Annual IEEE Conference on Computational Complexity (CCC 2005), 11-15 June 2005, San Jose, CA, USA*, pages 120–134. IEEE Computer Society, 2005.

BGH$^+$06. Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. Robust PCPs of proximity, shorter PCPs, and applications to coding. *SIAM Journal on Computing*, 36(4):889–974, 2006.

BGI16. Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under DDH. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 509–539, August 2016.

BL18. Fabrice Benhamouda and Huijia Lin. k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 500–532, April / May 2018.

BMR90. Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *22nd ACM STOC*, pages 503–513. ACM Press, May 1990.

BSW11. Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273, March 2011.

CCG$^+$20. Arka Rai Choudhuri, Michele Ciampi, Vipul Goyal, Abhishek Jain, and Rafail Ostrovsky. Round optimal secure multiparty computation from minimal assumptions. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 291–319, November 2020.

CCG$^+$21. Arka Rai Choudhuri, Michele Ciampi, Vipul Goyal, Abhishek Jain, and Rafail Ostrovsky. Oblivious transfer from trapdoor permutations in minimal rounds. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part II*, volume 13043 of *LNCS*, pages 518–549, November 2021.

CF13. Dario Catalano and Dario Fiore. Vector commitments and their applications. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 55–72, February / March 2013.

CGMA85. Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science*, SFCS '85, page 383–395, USA, 1985. IEEE Computer Society.

COSV17. Michele Ciampi, Rafail Ostrovsky, Luisa Siniscalchi, and Ivan Visconti. Round-optimal secure two-party computation from trapdoor permutations. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 678–710, November 2017.

COV15. Melissa Chase, Rafail Ostrovsky, and Ivan Visconti. Executable proofs, input-size hiding secure computation and a new ideal world. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 532–560, April 2015.

COWZ22. Michele Ciampi, Rafail Ostrovsky, Hendrik Waldner, and Vassilis Zikas. Round-optimal and communication-efficient multiparty computation. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part I*, volume 13275 of *LNCS*, pages 65–95, May / June 2022.

CV12. Melissa Chase and Ivan Visconti. Secure database commitments and universal arguments of quasi knowledge. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 236–254, August 2012.

DAR. DARPA. DPRIVE: Data Protection in Virtual Environments. Accessed: 2025-02-22.

DD22. Nico Döttling and Jesko Dujmovic. Maliciously Circuit-Private FHE from Information-Theoretic Principles. In Dana Dachman-Soled, editor, *3rd Conference on Information-Theoretic Cryptography (ITC 2022)*, volume 230 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 4:1–4:21, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

DDN00. Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000.

Din07. Irit Dinur. The PCP theorem by gap amplification. *Journal of the ACM (JACM)*, 54(3):12–es, 2007.

DN07. Ivan Damgård and Jesper Buus Nielsen. Scalable and unconditionally secure multiparty computation. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007*, pages 572–590, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

GH19. Craig Gentry and Shai Halevi. Compressible FHE with applications to PIR. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 438–464, December 2019.

GIKR01. Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. The round complexity of verifiable secret sharing and secure multicast. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, STOC '01, page 580–589, New York, NY, USA, 2001. Association for Computing Machinery.

GIS18. Sanjam Garg, Yuval Ishai, and Akshayaram Srinivasan. Two-round MPC: Information-theoretic and black-box. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part I*, volume 11239 of *LNCS*, pages 123–151, November 2018.

GMPP16. Sanjam Garg, Pratyay Mukherjee, Omkant Pandey, and Antigoni Polychroniadou. The exact round complexity of secure computation. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 448–476, May 2016.

GMR85. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th ACM STOC*, pages 291–304. ACM Press, May 1985.

GMW87. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.

Gol01. Oded Goldreich. *Foundations of Cryptography.* Cambridge University Press, 2001.

Goy11. Vipul Goyal. Constant round non-malleable protocols using one way functions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 695–704. ACM Press, June 2011.

GS18. Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 468–499, April / May 2018.

HM96. Shai Halevi and Silvio Micali. Practical and provably-secure commitment schemes from collision-free hashing. In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 201–215, August 1996.

HOWW19. Ariel Hamlin, Rafail Ostrovsky, Mor Weiss, and Daniel Wichs. Private anonymous data access. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 244–273, May 2019.

HW15. Pavel Hubacek and Daniel Wichs. On the communication complexity of secure function evaluation with long output. In Tim Roughgarden, editor, *ITCS 2015*, pages 163–172. ACM, January 2015.

IKOS07. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *39th ACM STOC*, pages 21–30. ACM Press, June 2007.

IKOS10. Yuval Ishai, Abishek Kumarasubramanian, Claudio Orlandi, and Amit Sahai. On invertible sampling and adaptive security. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 466–482, December 2010.

IKP10. Yuval Ishai, Eyal Kushilevitz, and Anat Paskin. Secure multiparty computation with minimal interaction. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 577–594, August 2010.

IKSS21. Yuval Ishai, Dakshita Khurana, Amit Sahai, and Akshayaram Srinivasan. On the round complexity of black-box secure MPC. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 214–243, Virtual Event, August 2021.

IKSS22a. Yuval Ishai, Dakshita Khurana, Amit Sahai, and Akshayaram Srinivasan. Round-optimal black-box protocol compilers. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part I*, volume 13275 of *LNCS*, pages 210–240, May / June 2022.

IKSS22b. Yuval Ishai, Dakshita Khurana, Amit Sahai, and Akshayaram Srinivasan. Round-optimal black-box secure computation from two-round malicious OT. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022, Part II*, volume 13748 of *LNCS*, pages 441–469, November 2022.

IKSS23. Yuval Ishai, Dakshita Khurana, Amit Sahai, and Akshayaram Srinivasan. Round-optimal black-box MPC in the plain model. In *CRYPTO 2023, Part I*, LNCS, pages 393–426, August 2023.

IOS25. Yuval Ishai, Rafail Ostrovsky, and Akash Shah. Zero-knowledge ram: Doubly efficient and black-box. In *Eurocrypt*, Lecture Notes in Computer Science. Springer, 2025.

IP07. Yuval Ishai and Anat Paskin. Evaluating branching programs on encrypted data. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 575–594, February 2007.

IPS08. Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 572–591, August 2008.

Kil88. Joe Kilian. Founding cryptography on oblivious transfer. In *20th ACM STOC*, pages 20–31. ACM Press, May 1988.

Kil92. Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *24th ACM STOC*, pages 723–732. ACM Press, May 1992.

KO04. Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 335–354, August 2004.

KOS03. Jonathan Katz, Rafail Ostrovsky, and Adam Smith. Round efficiency of multi-party computation with a dishonest majority. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 578–595, May 2003.

LNO13. Yehuda Lindell, Kobbi Nissim, and Claudio Orlandi. Hiding the input-size in secure two-party computation. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 421–440, December 2013.

LTV12. Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Howard J. Karloff and Toniann Pitassi, editors, *44th ACM STOC*, pages 1219–1234. ACM Press, May 2012.

LY10. Benoît Libert and Moti Yung. Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 499–517, February 2010.

Mer89. Ralph C. Merkle. A certified digital signature. In *CRYPTO*. Springer, 1989.

Mic00. Silvio Micali. Computationally sound proofs. *SIAM J. Comput.*, 30(4):1253–1298, 2000.

MPP20. Andrew Morgan, Rafael Pass, and Antigoni Polychroniadou. Succinct non-interactive secure computation. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 216–245, May 2020.

MRK03. Silvio Micali, Michael O. Rabin, and Joe Kilian. Zero-knowledge sets. In *44th FOCS*, pages 80–91. IEEE Computer Society Press, October 2003.

Mul54. David E. Muller. Application of boolean algebra to switching circuit design and to error detection. *Trans. I R E Prof. Group Electron. Comput.*, 3(3):6–12, 1954.

ORS15. Rafail Ostrovsky, Silas Richelson, and Alessandra Scafuro. Round-optimal black-box two-party computation. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 339–358, August 2015.

Pas04. Rafael Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In László Babai, editor, *36th ACM STOC*, pages 232–241. ACM Press, June 2004.

PRS02. Manoj Prabhakaran, Alon Rosen, and Amit Sahai. Concurrent zero knowledge with logarithmic round-complexity. In *Proceedings of the 43rd Symposium on Foundations of Computer Science*, FOCS '02, page 366–375, USA, 2002. IEEE Computer Society.

Ps05. Rafael Pass and Abhi shelat. Unconditional characterizations of non-interactive zero-knowledge. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 118–134, August 2005.

PS21. Arpita Patra and Akshayaram Srinivasan. Three-round secure multiparty computation from black-box two-round oblivious transfer. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 185–213, Virtual Event, August 2021.

PW09. Rafael Pass and Hoeteck Wee. Black-box constructions of two-party protocols from one-way functions. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 403–418, March 2009.

PW10. Rafael Pass and Hoeteck Wee. Constant-round non-malleable commitments from sub-exponential one-way functions. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 638–655, May / June 2010.

QWW18. Willy Quach, Hoeteck Wee, and Daniel Wichs. Laconic function evaluation and applications. In Mikkel Thorup, editor, *59th FOCS*, pages 859–870. IEEE Computer Society Press, October 2018.

Ree54. Irving S. Reed. A class of multiple-error-correcting codes and the decoding scheme. *Trans. IRE Prof. Group Inf. Theory*, 4:38–49, 1954.

Ros04. Alon Rosen. A note on constant-round zero-knowledge proofs for np. In Moni Naor, editor, *Theory of Cryptography*, pages 191–202, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

Sta. StarkWare. A Framework for Efficient STARKs. Accessed: 2025-02-22.

SW05. Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473, May 2005.

Wee10. Hoeteck Wee. Black-box, round-efficient secure computation via non-malleability amplification. In *51st FOCS*, pages 531–540. IEEE Computer Society Press, October 2010.

Yao86. Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.

A Further Preliminaries

Here we define commitment schemes and encoding schemes, required for our construction.

A.1 Commitment Schemes

We define a commitment scheme as follows.

Definition 16 (Commitment scheme). *A commitment scheme $\Pi_{\mathtt{com}} = (\mathcal{C}, \mathcal{R})$ is a two-phase protocol between two* PPT *interactive algorithms, a committer $\mathcal{C}$ and a receiver $\mathcal{R}$, with the following two phases:*

- **Commitment phase.** *In this first phase, $\mathcal{C}$ on input a message m and a randomness r_c interacts with $\mathcal{R}$ on input r_r. Let $\tau = \langle \mathcal{C}(m, r_c), \mathcal{R}(r_r) \rangle$ denote the corresponding commitment transcript.*
- **Decommitment phase.** *In this second phase, the committer $\mathcal{C}$ reveals m', and $\mathcal{R}$ accepts the value committed in τ to be m' if and only if $\mathcal{C}$ proves that τ can be produced on input m'.*

We only consider commitment schemes where the decommitment phase consists of a single message from the committer to the receiver.

Let $\mathsf{Dec}(\tau, m, r_c)$ denote the polynomial-time deterministic algorithm that on input a commitment transcript τ, committer message m, and randomness r_c, outputs `accept` or `reject` to denote whether the decommitment was accepted or rejected, respectively. We report the classic definitions of completeness, binding, and hiding. We refer the reader to [Gol01] for further details.

Definition 17 (Completeness). *A commitment scheme $(\mathcal{C}, \mathcal{R})$ is said to be* complete *if for any message m, committer randomness r_c, and receiver randomness r_r, $\mathsf{Dec}(\tau, m, r_c)$ outputs* `accept`*, where $\tau = \langle \mathcal{C}(m, r_c), \mathcal{R}(r_r) \rangle$.*

Definition 18 (Binding). *A commitment scheme $(\mathcal{C}, \mathcal{R})$ is said to be* statistically (resp., computationally) binding *if for every unbounded (resp., PPT) malicious committer $\mathcal{C}^*$, there exists a negligible function ν, such that $\mathcal{C}^*$ succeeds in the following game with probability at most $\nu(\lambda)$:*

- *On input the security parameter λ, $\mathcal{C}^*$ interacts with honest receiver $\mathcal{R}$ in the commitment phase, and $\mathcal{R}$ obtains the commitment τ.*
- *$\mathcal{C}^*$ outputs pairs (m_0, r_0) and (m_1, r_1).*
- *$\mathcal{C}^*$ succeeds if $\mathsf{Dec}(\tau, m_0, r_0) = \mathsf{Dec}(\tau, m_1, r_1) = \mathtt{accept}$ and $m_0 \neq m_1$.*

If $\nu(\lambda) = 0$, we refer to the above as a *perfectly binding* commitment scheme.

Definition 19 (Hiding). *A commitment scheme $(\mathcal{C}, \mathcal{R})$ is said to be* computationally (resp., statistically) hiding *if for every malicious* PPT *(resp., unbounded) receiver $\mathcal{R}^*$, and every pair of messages (m_0, m_1), the view of $\mathcal{R}^*$ after a commitment phase where $\mathcal{C}$ commits to m_0 is computationally (resp., statistically) indistinguishable from the view of $\mathcal{R}^*$ after a commitment phase where $\mathcal{C}$ commits to m_1.*

A.2 Encoding Schemes

We define an encoding scheme as follows.

Definition 20 (Encoding Scheme). *An encoding scheme over the field $\mathbb{F}_q$ with message length k, codeword length K, relative distance δ, and relative decoding radius ρ is defined by a pair of algorithms* $(\mathsf{Encode}, \mathsf{Decode})$ *as follows:*

1. $\mathsf{Encode} : \mathbb{F}_q^k \to \mathbb{F}_q^K$. *The algorithm* Encode *maps a message* $\mathbf{x} \in \mathbb{F}_q^k$ *to a codeword* $\mathbf{X} \in \mathbb{F}_q^K$ *such that for any* $\mathbf{x} \neq \mathbf{x}'$*, the Hamming distance between* $\mathsf{Encode}(\mathbf{x})$ *and* $\mathsf{Encode}(\mathbf{x}')$ *is at least* δK.
2. $\mathsf{Decode} : \mathbb{F}_q^K \to \mathbb{F}_q^k$. *The algorithm* Decode *takes as input a possibly corrupted codeword* $\mathbf{X}' \in \mathbb{F}_q^K$. *For* $\mathbf{x} \in \mathbb{F}_q^k$*, if the Hamming distance between* $\mathbf{X}'$ *and* $\mathsf{Encode}(\mathbf{x})$ *is at most* ρK*, where* $\rho < \delta/2$*, then* Decode *outputs* $\mathbf{x}$.

A family of encoding schemes $(\mathsf{Encode}_k, \mathsf{Decode}_k)$ *with* $q = \mathsf{polylog}(k)$*,* $K = K(k)$ *for some polynomial* $K(\cdot)$*, and a fixed* $\delta > 0$*, is said to be efficiently encodable (resp. decodable) if* Encode *(resp.* Decode*) runs in probabilistic polynomial time (PPT) in* k*. We denote* $\mathcal{E} = (\mathsf{Encode}, \mathsf{Decode})$ *as a* (k, K, δ)*-encoding scheme with relative decoding radius* ρ*. Let* $\mathcal{C} = \{\mathcal{E}.\mathsf{Encode}(\mathbf{x}) \mid \forall \mathbf{x} \in \mathbb{F}_q^k\} \subseteq \mathbb{F}_q^K$ *represent the codeword space of* $\mathcal{E}$.

For any $\alpha > 0$ and $0 < \delta < \frac{1}{2}$, there exists a Reed-Muller code [Ree54, Mul54] with codeword size $K = O(k^{1+\alpha})$ [HOWW19].

B Proof of Theorem 4

Here we provide a complete formal proof of Theorem 4, i.e., Π_{2PC} securely realizes $\mathcal{F}_{\mathsf{2PC\text{-}PKO}}$. We use hybrid arguments to argue security against a malicious adversary corrupting any one of the two parties.

Case 1: P_1 is corrupt. Simulator $\mathcal{S}_1$ for malicious party P_1^* is described in Figure 6. $\mathcal{S}_1$ computes $\mathsf{ct}_{\mathsf{Dummy}}$ (see Step 8 of $\mathcal{S}_1$) and sends updated ciphertexts as described in the note at the end of Section 3.1. Moreover, $\mathcal{S}_1$ also updates the ciphertexts with this procedure in all the intermediate hybrids.

Let $\mathcal{D}_{\mathcal{R}}$ and $\mathcal{D}_{\mathcal{I}}$ denote the joint distribution of view of malicious party P_1 and output of P_2 in the real and ideal world executions, respectively. Similarly, let $\mathcal{D}_i$ denote the joint distribution of view of malicious party P_1 and output of P_2 when $\mathcal{S}_i$ runs experiment $\mathcal{H}_i$. We then have the following sequence of hybrid experiments.

Experiment $\mathcal{H}_0$. In this experiment, $\mathcal{S}_1$ interacts with P_1^* by following the computation prescribed in protocol Π_{2pc} with input $x_2 \in \{0,1\}^{\ell_2}$. As honest party's output $\mathcal{S}_1$ outputs whatever P_2 outputs in the protocol. Note that this experiment corresponds to a real world execution, i.e., $\mathcal{D}_{\mathcal{R}} \equiv \mathcal{D}_0$.

Experiment $\mathcal{H}_1$. In this experiment, $\mathcal{S}_1$ interacts with P_1^* identically to $\mathcal{H}_0$, except it also tries to extract input x_1^*. In more detail, $\mathcal{S}_1$ does the following:

– For each $i \in [n]$, instead of interacting as receiver $\mathcal{R}$ in the commit phase of Π_{comExt} with P_1^* to receive commitment $\mathtt{com}_i^{(x_1)}$, $\mathcal{S}_1$ runs the extractor Ext of Π_{comExt}, with oracle access to P_1^*, to obtain $(\mathtt{com}_i^{(x_1)*}, (\mathsf{View}_i^{(x_1)*} \parallel r_i^*))$. $\mathcal{S}_1$ aborts the simulation if Ext fails on more than $t/3$ indices $i \in [n]$. Define $\mathcal{J} = \{i \in [n] : \mathsf{Ext}$ succeeds on $i\}$.
– After executing step 5 of the protocol, if all the checks passed, $\mathcal{S}_1$ aborts the simulation if $|\mathcal{J}| < n - \frac{t}{3}$.

- $\mathcal{S}_1$ aborts the simulation if more than $\frac{t}{3}$ indices are there in $\mathcal{J}$ such that $\mathsf{ct}_i^{(x_1)} \neq \mathsf{Enc}^{\mathsf{FHE}}(\mathtt{pk}, \mathsf{View}_i^{(x_1)*}; r_i^*)$. Let $\mathcal{J}' = \{i \in \mathcal{J} : \mathsf{ct}_i = \mathsf{Enc}^{\mathsf{FHE}}(\mathtt{pk}, \mathsf{View}_i^{(x_1)*}; r_i^*)\}$.
- $\mathcal{S}_1$ locally emulates n players $p_1, \dots, p_n$, with input $\mathsf{View}_i^{(x_1)*}$ of p_i if $i \in \mathcal{J}'$, and input $\perp$ of p_i otherwise. It then runs the VSS reconstruction protocol $\mathsf{Reconstruct}$ "in the head" to obtain x_1^*. $\mathcal{S}_1$ aborts the simulation if reconstruction failed (which occurs only if there is no set of $n-t$ mutually consistent views among $\{\mathsf{View}_i^{(x_1)*}\}_{i \in \mathcal{J}'}$).

There are two differences in view of P_1^* in hybrid $\mathcal{H}_0$ and $\mathcal{H}_1$. Namely, view of P_1^* in the commit phase, and simulation abort.

By Definition 4, the extracted commitment $\mathtt{com}_i^{(x_i)*}$ is identically distributed to the commitment that would have otherwise been received from P_1^*, implying the view of P_1^*'s in commitment phase is identically distributed in both cases.

Next, let us analyze the probability of simulation abort. $\mathcal{S}_1$ aborts in either of the following cases (we use the fact that t is at least a constant multiple of n; in particular, $t = \lfloor \frac{n-1}{4} \rfloor > \frac{n}{5}$):

- Extractor Ext fails on more than $t/3$ commitments $\mathtt{com}_i^{(x_1)*}$. From Definition 4, we know that if $\mathtt{com}_i^{(x_2)*}$ is a well-formed commitment, then Ext yields a valid opening $(\mathsf{View}_i^{(x_1)*}, r_i^*) \neq \perp$ with overwhelming probability. By contrapositive, if Ext fails on some $\mathtt{com}_i^{(x_1)*}$, then with overwhelming probability, it is an ill-formed commitment. Moreover, if P_1^* sends more than $t/3$ ill-formed commitments, then the probability that all opened commitments $(\mathtt{com}_i^{(x_1)*})_{i \in \mathcal{I}^{(2)}}$ in step 4 of the protocol are well-formed is $< \binom{n-\frac{t}{3}}{t} / \binom{n}{t} = \mathsf{negl}(n)$. We conclude that, if $\mathcal{S}_1$ has not already sent abort to P_2^* after opening $(\mathtt{com}_i^{(x_1)*})_{i \in \mathcal{I}^{(2)}}$, then there is a negligible probability of more than $t/3$ commitments being ill-formed, and thus of Ext failing on more than $t/3$ commitments.
- If P_1^* sends more than $t/3$ ciphertexts such that $\mathsf{ct}_i^{(x_1)} \neq \mathsf{Enc}^{\mathsf{FHE}}(\mathtt{pk}, \mathsf{View}_i^{(x_1)*}; r_i^*)$, then the probability that in step 5 of the protocol, $\mathsf{ct}_i^{(x_1)} = \mathsf{Enc}^{\mathsf{FHE}}(\mathtt{pk}, \mathsf{View}_i^{(x_1)*}; r_i^*)$, for $i \in \mathcal{I}^{(2)}$ (conditioned on successful decommitments) is $< \binom{\mathcal{J}-\frac{t}{3}}{t} / \binom{\mathcal{J}}{t} = \mathsf{negl}(n)$, since $\mathcal{J} \geq n - \frac{t}{3}$.
- $\mathsf{Reconstruct}$ fails if there is no set of $n-t$ views among $\{\mathsf{View}_{i,j}^{(x_1)*}\}_{i \in \mathcal{J}'}$ which are all mutually consistent. In this case, the probability that views $(\mathsf{View}_i^{(x_1)*})_{i \in \mathcal{I}^{(2)}}$ opened in step 5 of the protocol are all mutually consistent (conditioned on successful decommitments and well-formedness of ciphertexts) is $< \binom{n-t}{t} / \binom{|\mathcal{J}'|}{t} = \mathsf{negl}(n)$, since $|\mathcal{J}'| \geq n - \frac{2t}{3}$. We conclude that, if $\mathcal{S}_1$ has not already sent abort to P_1^* in step 5 of the protocol, then the probability that $\mathcal{S}_1$ fails to extract x_i^* is negligible.

Therefore, $\mathcal{S}_1$ aborts the simulation with negligible probability, and $\mathcal{D}_0 \approx \mathcal{D}_1$.

Experiment $\mathcal{H}_2$. In this experiment, $\mathcal{S}$ proceeds identically to $\mathcal{H}_1$, except that it does the following: $\mathcal{S}_1$ computes $\mathsf{ct}^{(x_1)*} \xleftarrow{\$} \mathsf{Enc}^{\mathsf{FHE}}(\mathtt{pk}, x_1^*)$ and applies an identity circuit of width and depth equal to the circuit in the computation in step 6 of the protocol and uses it in the computation instead of the ciphertext reconstructed in step 6 of the protocol, i.e., $\mathsf{ct}^{(x_1)}$. In hybrid $\mathcal{H}_1$, let $\hat{J} = \{j_1, \dots, j_p\}$ denote a set of indices of size $p \geq n-t$, such that ciphertexts $\mathsf{ct}_{j_k}^{(x_1)*} = \mathsf{Enc}^{\mathsf{FHE}}(\mathtt{pk}, \mathsf{View}_{j_k}^{(x_1)*}, r_{j_k}^*)$, for each $k \in [p]$ and $\{\mathsf{View}_{j_k}^{(x_1)*}\}_{k \in [p]}$ are mutually consistent. Now, there are two cases:

1. *Case 1*: If all the ciphertexts $\mathsf{ct}_1^{(x_1)}, \dots, \mathsf{ct}_n^{(x_1)}$ are well-formed then $\mathsf{ct}^{(x_1)*}$ and $\mathsf{ct}^{(x_1)}$ encrypt the same value, i.e., x_1^*. The view of P_1^* is computationally indistinguishable from its view in $\mathcal{H}_1$ due to circuit-privacy of FHE scheme (Definition 14).
2. *Case 2*: If there exists a ciphertext in $\mathsf{ct}_1^{(x_1)}, \dots, \mathsf{ct}_n^{(x_1)}$ that is malformed, then due to circuit-privacy of FHE scheme, the view of P_1^* in this case too is computationally indistinguishable from its view in $\mathcal{H}_1$.

Therefore, $\mathcal{D}_1 \approx \mathcal{D}_2$.

Experiment $\mathcal{H}_3$. In this experiment $\mathcal{S}_1$, after extracting x_1^*, $\mathcal{S}_1$ proceeds identically to $\mathcal{H}_2$, except that $\mathcal{S}_1$ computes everything in the clear and encrypts it before committing or sending it to P_1^*. In more detail, $\mathcal{S}_1$ does the following:

1. Performs all the computation steps in line 7 - 8.(f) in the clear.
2. In step 8.(g) of the protocol, for each $j \in [\ell_p]$,
   (a) Let $(\mathsf{View}_{i,j}^{\pi})_{i\in[n]}$ denote the VSS share of bit j of the PCPP proof π obtained in clear execution in step 8.(f) of the protocol.
   (b) For each $i \in [n]$, encrypt $\mathsf{ct}_{i,j}^{(\pi)} \xleftarrow{\$} \mathsf{Enc}^{\mathsf{FHE}}(\mathtt{pk}, \mathsf{View}_{i,j}^{(\pi)})$ and apply identity circuit of width and depth equivalent to the circuit corresponding to computation to obtain $\mathsf{View}_{i,j}^{(\pi)}$. Let $\mathsf{CT}_i^{(\pi)}$ denote the vector of ciphertexts $\left(\mathsf{ct}_{i,1}^{(\pi)}, \dots, \mathsf{ct}_{i,\ell_p}^{(\pi)}\right)$, for $i \in [n]$.
3. Now, commitments as per step 8.(g) of the protocol.
4. Similarly, for each $j \in [K]$,
   (a) Let $(\mathsf{View}_{i,j}^{X_2})_{i\in[n]}$ denote the VSS share of bit j encoding X_2 obtained in clear execution in step 8.(e) of the protocol.
   (b) For each $i \in [n]$, encrypt $\mathsf{ct}_{i,j}^{(X_2)} \xleftarrow{\$} \mathsf{Enc}^{\mathsf{FHE}}(\mathtt{pk}, \mathsf{View}_{i,j}^{(X_2)})$ and apply identity circuit of width and depth equivalent to the circuit corresponding to computation to obtain $\mathsf{View}_{i,j}^{(X_2)}$. Let $\mathsf{CT}_i^{(X_2)}$ denote the vector of ciphertexts $\left(\mathsf{ct}_{i,1}^{(X_2)}, \dots, \mathsf{ct}_{i,K}^{(X_2)}\right)$, for $i \in [n]$.
5. Compute $\mathsf{ct}^{(y)} \xleftarrow{\$} \mathsf{Enc}^{\mathsf{FHE}}(\mathtt{pk}, y)$ and apply identity circuit of appropriate width and depth.
6. Now, make commitments as per step 8.(h) in the protocol.
7. Execute steps 10, 11.(a)-(b), 12 in the clear.
8. Now, given views $(\mathsf{View}_i^{(V)})_{i\in[n]}$, compute ciphertexts $(\mathsf{ct}_i^{(V)})_{i\in[n]}$ by following similar process as above.
9. Proceed following steps 13,14,15.(a) of the protocol.
10. In step 15.(b), for each $j \in [n]$, given $(\rho_{i,k})_{k\in[Q]}$ proceed as above to obtain ciphertext list R_i and send it to P_1^*.

The view of P_1^* is computationally indistinguishable in Hybrids $\mathcal{H}_2$ and $\mathcal{H}_3$ is due to the circuit-privacy of FHE scheme. Thus, $\mathcal{D}_2 \approx \mathcal{D}_3$.

Experiment $\mathcal{H}_4$. In hybrid $\mathcal{H}_4$, $\mathcal{S}_1$ proceeds identically to $\mathcal{H}_3$ except that it samples $i_1^{(1)}, \dots, i_t^{(1)} \xleftarrow{\$} [n]$ before step 8 of the actual protocol. Let the set be denoted by $\mathcal{I}^{(1)}$. It then simulates $\mathcal{F}_{\mathsf{2pc}}$ with output set $\mathcal{I}^{(1)}$ in step 14 of the actual protocol. Since, the distribution of set $\mathcal{I}^{(1)}$ in $\mathcal{H}_4$ is identical to that in $\mathcal{H}_3$, we have $\mathcal{D}_3 \equiv \mathcal{D}_4$.

Experiment $\mathcal{H}_5$. In this hybrid, after executing the MPC-in-the-head emulation of protocol Π_g, for all $i \in [n] \setminus \mathcal{I}^{(1)}$, $\mathcal{S}_1$ replaces $\mathsf{View}_i^{(V)}$ with an arbitrary MPC view. The only difference in the view of P_1^* are the commitments $\mathtt{com}_i^{(V)}$, for $i \in [n] \setminus \mathcal{I}^{(1)}$ as these commitments are never opened to P_1^*. Due to the computational hiding property of commitment scheme Π_{com} (see Definition 19), we have $\mathcal{D}_4 \approx \mathcal{D}_5$.

Experiment $\mathcal{H}_6$. In this hybrid, $\mathcal{S}_1$ doesn't emulate Π_g with parties $p_1, \dots, p_n$ in the head. Instead, $\mathcal{S}_1$ invokes simulator $\mathcal{S}_{\Pi_g}$ corresponding to protocol Π_g, with set of indices of parties $\mathcal{I}^{(1)}$, public inputs $u = (x_1^*, y, \ell_2)$ and r_M, and private input of party p_i as $(\psi_{i,k})_{k\in[Q]}$, for $i \in \mathcal{I}^{(1)}$, and output of the protocol set to $\mathtt{accept}$. $\mathcal{S}_1$ obtains $\{\mathsf{View}_i^{(V)}\}_{i\in\mathcal{I}^{(1)}} \leftarrow \mathcal{S}_{\Pi_g}(\mathcal{I}^{(1)}, (x_1^*, y, \ell_2, r_M), (\psi_{i,j})_{i\in\mathcal{I}^{(1)}, j\in[Q]}, \mathtt{accept})$. The only change in view of P_1^* in $\mathcal{H}_6$ from $\mathcal{H}_5$ is the views $\{\mathsf{View}_i^{(V)}\}_{i\in\mathcal{I}^{(1)}}$ opened to P_1^*. Since, Π_g is a perfectly t-private, the distribution of views $\{\mathsf{View}_i^{(V)}\}_{i\in\mathcal{I}^{(1)}}$ output in emulating Π_g "in-the-head" in $\mathcal{H}_5$ is identical to the distribution of views output by $\mathcal{S}_{\Pi_g}$ in $\mathcal{H}_6$. Therefore $\mathcal{D}_5 \equiv \mathcal{D}_6$.

Experiment $\mathcal{H}_7$. In this hybrid, $\mathcal{S}_1$ behaves identical to $\mathcal{H}_6$, except

1. For $j \in [\ell_p]$, $\mathcal{S}_1$ locally emulates a dealer with input 0 along with n players $p_1, \ldots, p_n$ and it runs the VSS sharing protocol Share "in the head" to obtain n views $\mathsf{View}^{(\pi)*}_{1,j}, \ldots, \mathsf{View}^{(\pi)*}_{n,j}$.
2. With views $(\mathsf{View}^{(\pi)*}_{i,j})_{i\in[n],j\in[\ell_p]}$, $\mathcal{S}_1$ computes as in step 2.(b) in $\mathcal{H}_3$ to obtain vector of ciphertexts $\mathsf{CT}^{(\pi)*}_i$, for each $i \in [n]$. For $i \in [n] \setminus \mathcal{I}^{(1)}$, instead of committing to $\mathsf{CT}^{(\pi)}_i$ using vector commitment to obtain commitments $\mathtt{com}^{(\pi)}_i$, $\mathcal{S}_1$ sends commitment to $\mathsf{CT}^{(\pi)*}_i$ to P^*_1.

In brief, in this hybrid, for the commitments $\mathtt{com}^{(\pi)}_i$ for $i \in [n] \setminus \mathcal{I}^{(1)}$, that are never opened by P^*_1, $\mathcal{S}_1$ replaces the vector commitments to encryptions of VSS shares of actual bits of the PCPP proof π with encryptions of VSS shares of 0.

The only difference in the view of P^*_1 are the commitments $\mathtt{com}^{(\pi)}_i$ for $i \in [n] \setminus \mathcal{I}^{(1)}$ as no index of these vector commitments are ever opened to P^*_1. Due to the computational hiding property of vector commitment scheme (see Definition 5 and Definition 8), the view of P_1 in $\mathcal{H}_6$ and $\mathcal{H}_7$ is computationally indistinguishable.

Experiment $\mathcal{H}_8$. In this hybrid, $\mathcal{S}_1$ proceeds identically to $\mathcal{H}_7$, except that

For each $i \in [\mathcal{I}^{(1)}]$,

1. $\mathcal{S}_1$ commits to $\mathsf{CT}^{(\pi)*}_i$.
2. In the simulation, for each $k \in [Q]$, where $q_k \leq \ell_p$, $\mathcal{S}_1$ sets $\psi_{i,k} \leftarrow \mathsf{View}^{(\pi)*}_{i,q_k}$ and $\rho_{i,k} \leftarrow (\mathsf{ct}^{\pi*}_{i,q_k}, \mathsf{VC.Open}(\mathsf{ck}, \mathsf{ct}^{(\pi*)}_{i,q_k}, q_k, \mathsf{st}^{(\pi*)}_i))$, for all $i \in [n]$.

In effect, $\mathcal{S}_1$ replaces VSS shares of actual bits of PCPP proof π with VSS shares of 0. This causes the following changes in view of P^*_1: for $i \in \mathcal{I}^{(1)}$,

1. Commitments $\mathtt{com}^{(\pi)}_i$,
2. Values $\psi_{i,k}$ used in the MPC-in-the-head simulation, for $k \in [Q]$ such that $q_k \leq \ell_p$.
3. Entry at index k in vectors of ciphertexts R_i, where $q_k \leq \ell_p$.

From the privacy property of VSS shares (Definition 3), the distrbution of view of P^*_1 in hybrids $\mathcal{H}_8$ and $\mathcal{H}_7$ are identical. Thus, $\mathcal{D}_8 \equiv \mathcal{D}_7$.

Experiment $\mathcal{H}_9$. $\mathcal{S}_1$ proceeds almost identically as in $\mathcal{H}_8$, except for the following change.

1. For $j \in [K]$, $\mathcal{S}_1$ locally emulates a dealer with input 0 along with n players $p_1, \ldots, p_n$ and it runs the VSS sharing protocol Share "in the head" to obtain n views $\mathsf{View}^{(X_2)*}_{1,j}, \ldots, \mathsf{View}^{(X_2)*}_{n,j}$.
2. With views $(\mathsf{View}^{(X_2)*}_{i,j})_{i\in[n],j\in[K]}$, $\mathcal{S}_1$ computes as in step 4.(b) in $\mathcal{H}_3$ to obtain vector of ciphertexts $\mathsf{CT}^{(X_2)*}_i$, for each $i \in [n]$. For $i \in [n] \setminus \mathcal{I}^{(1)}$, instead of committing to $\mathsf{CT}^{(X_2)}_i$ in commit phase of Π_{comExt} to obtain commitments $\mathtt{com}^{(X_2)}_i$, $\mathcal{S}_1$ commits to $\mathsf{CT}^{(X_2)*}_i$.

In brief, in this hybrid, for the commitments $\mathtt{com}^{(X_2)}_i$ for $i \in [n] \setminus \mathcal{I}^{(1)}$, that are never opened by P^*_1, $\mathcal{S}_1$ replaces the commitments to encryptions of codeword X_2 (encoding of input x_2) with encryptions of VSS shares of 0.

The only difference in the view of P^*_1 are the commitments $\mathtt{com}^{(X_2)}_i$ for $i \in [n] \setminus \mathcal{I}^{(1)}$ as these commitments are never opened to P^*_1. Due to the computational hiding property of Π_{comExt} (see Definition 19), the view of P_1 in $\mathcal{H}_8$ and $\mathcal{H}_9$ is computationally indistinguishable.

Experiment $\mathcal{H}_{10}$. In this hybrid, $\mathcal{S}_1$ proceeds identically to $\mathcal{H}_9$, except that for each $i \in [\mathcal{I}^{(1)}]$,

1. $\mathcal{S}_1$ commits to $\mathsf{CT}_i^{(X_2)*}$.
2. In the simulation, for each $k \in [Q]$, where $q_k > \ell_p$, $\mathcal{S}_1$ sets $\psi_{i,k} \leftarrow \mathsf{View}_{i,q_k-\ell_p}^{(X_2)*}$, for all $i \in [n]$.

In effect, $\mathcal{S}_1$ replaces VSS shares of actual bits of codeword X_2 with VSS shares of 0. This causes the following changes in view of P_1^*: for $i \in \mathcal{I}^{(1)}$,

1. Commitments $\mathtt{com}_i^{(X_2)}$,
2. Values $\psi_{i,k}$ used in the MPC-in-the-head simulation, for $k \in [Q]$ such that $q_k > \ell_p$.

From the privacy property of VSS shares (Definition 3), the distrbution of view of P_1^* in hybrids $\mathcal{H}_9$ and $\mathcal{H}_{10}$ are identical. Thus, $\mathcal{D}_9 \equiv \mathcal{D}_{10}$.

Experiment $\mathcal{H}_{11}$. Observe that, we have made the simulation independent of input of x_2, except for the step in which we compute y. In this hybrid, $\mathcal{S}_1$ sends extracted x_1^* to $\mathcal{F}_{\mathsf{2PC\text{-}PKO}}$. $\mathcal{F}_{\mathsf{2PC\text{-}PKO}}$ sends $y = f(x_1^*, x_2)$ to $\mathcal{S}_1$. If $\mathcal{S}_1$ has not aborted until the penultimate step of the protocol and receives y from $\mathcal{S}_1$, $\mathcal{S}_1$ forwards y to $\mathcal{F}_{\mathsf{2PC\text{-}PKO}}$. In this hybrid $\mathcal{S}_1$ doesn't need input of honest party P_2 in any step of the simulation. Thus, we do not provide x_2 to $\mathcal{S}_1$. This hybrid corresponds to the ideal world as the simulator $\mathcal{S}_1$ interacts with ideal functionality $\mathcal{F}_{\mathsf{2PC\text{-}PKO}}$ with the description provided in Figure 6. The view of P_1^* and output of honest party P_2 is identically distributed in both the hybrids. Thus, $\mathcal{D}_{10} \equiv \mathcal{D}_{\mathcal{I}}$.

Combining the hybrids. By a combination of the above hybrid arguments, we get $\mathcal{D}_{\mathcal{R}} \approx \mathcal{D}_{\mathcal{I}}$, i.e., the joint distribution of view of malicious party P_1^* and output of honest party P_2 is computationally indistinguishable in the real and ideal worlds. Therefore, we conclude that Π_{2PC} securely realizes $\mathcal{F}_{\mathsf{2PC\text{-}PKO}}$ in the presence of a malicious adversary corrupting P_1.

Case 2: P_2 is corrupt. We construct a simulator $\mathcal{S}_2$ for malicious party P_2^*, as shown in Figure 7.

Let $\mathcal{D}_{\mathcal{R}}$ and $\mathcal{D}_{\mathcal{I}}$ denote the joint distribution of view of malicious party P_2^* and output of honest party P_1, in the real and ideal world executions, respectively. Similarly, let $\mathcal{D}_i$ denote the joint distribution of view of malicious party P_1 and output of honest party P_2 when $\mathcal{S}_2$ runs experiment $\mathcal{H}_i$. We then have the following sequence of hybrid experiments.

Experiment $\mathcal{H}_0$. In this experiment, input x_1 is given to $\mathcal{S}_2$ and it runs an honest execution of Π_{2PC} with P_2^*, acting as party P_1 with input x_1. As the honest party's output, $\mathcal{S}_2$ outputs y. Note that this experiment corresponds to the real world execution, i.e., $\mathcal{D}_{\mathcal{R}} \equiv \mathcal{D}_0$.

Experiment $\mathcal{H}_1$. In this experiment, $\mathcal{S}_2$ executes the protocol identically to $\mathcal{H}_0$, except it also tries to extract P_2^*'s input x_2^*. In more detail, $\mathcal{S}_2$ does the following:

– For each $i \in [n]$, instead of simply receiving commitment $\mathtt{com}_i^{(X_2)}$, $\mathcal{S}_2$ runs the extractor Ext of extractable commitment scheme $\Pi_{\mathtt{comExt}}$, with oracle access to P_2^*, to obtain $(\mathtt{com}_i^{(X_2)*}, \mathsf{CT}_i^{(X_2)*})$. $\mathcal{S}_2$ aborts the simulation if Ext fails on more than $t/2$ indices $i \in [n]$. Define $\mathcal{J} = \{i \in [n] : \mathsf{Ext}$ succeeds on $i\}$.
– After completing the rest of Π_{2PC} honestly, $\mathcal{S}_2$ aborts the simulation if $|\mathcal{J}| < n - \frac{t}{2}$ in the previous step.
– For each $i \in \mathcal{J}$, $\mathcal{S}_2$ parses $\mathsf{CT}_i^{(X_2)*}$ as $\left(\mathsf{ct}_{i,1}^{(X_2)*}, \ldots, \mathsf{ct}_{i,K}^{(X_2)*}\right)$, and for each $j \in [K]$, computes $\mathsf{View}_{i,j}^{(X_2)*} \leftarrow \mathsf{Dec}^{\mathsf{FHE}}(\mathtt{sk}, \mathsf{ct}_{i,j}^{(X_2)*})$.

- For each $j \in [K]$, $\mathcal{S}_2$ locally emulates n players $p_1, \dots, p_n$, with input $\mathsf{View}_{i,j}^{(X_2)*}$ of p_i if $i \in \mathcal{J}$, and input $\perp$ of p_i otherwise. It then runs the VSS reconstruction protocol $\mathsf{Reconstruct}$ "in the head" to obtain $X_{2,j}^*$. $\mathcal{S}_2$ aborts the simulation if reconstruction fails for some $j \in [K]$ (which occurs only if there is no set of $n-t$ mutually consistent views among $\{\mathsf{View}_{i,j}^{(X_2)*}\}_{i \in \mathcal{J}}$).
- $\mathcal{S}_2$ decodes $X_2^* \leftarrow (X_{2,1}^*, \dots, X_{2,K}^*)$ to get $x_2^* \leftarrow \mathcal{E}.\mathsf{Decode}(X_2^*)$.

By Definition 4, the extracted commitment $\mathtt{com}_i^{(X_2)*}$ is identically distributed to the commitment that would have otherwise been received from P_2^*, implying P_2^*'s view is identically distributed in both cases. The only other difference between the views of P_2^* in $\mathcal{H}_0$ and $\mathcal{H}_1$ comes from a potential simulation abort in either of the following cases (we use the fact that t is at least a constant multiple of n; in particular, $t = \lfloor \frac{n-1}{4} \rfloor > \frac{n}{5}$):

- Extractor Ext fails on more than $t/2$ commitments $\mathtt{com}_i^{(X_2)*}$. From Definition 4, we know that if $\mathtt{com}_i^{(X_2)*}$ is a well-formed commitment, then Ext yields a valid opening $\mathsf{CT}_i^{(X_2)*} \neq \perp$ with overwhelming probability. By contrapositive, if Ext fails on some $\mathtt{com}_i^{(X_2)*}$, then with overwhelming probability, it is an ill-formed commitment. Moreover, if P_2^* sends more than $t/2$ ill-formed commitments, then the probability that all earlier-opened commitments $(\mathtt{com}_i^{(X_2)*})_{i \in \mathcal{I}^{(1)}}$ are well-formed is $< \binom{n-\frac{t}{2}}{t} / \binom{n}{t} = \mathsf{negl}(n)$. We conclude that, if $\mathcal{S}_2$ has not already sent abort to P_2^* after opening $(\mathtt{com}_i^{(X_2)*})_{i \in \mathcal{I}^{(1)}}$, then there is a negligible probability of more than $t/2$ commitments being ill-formed, and thus of Ext failing on more than $t/2$ commitments.
- $\mathsf{Reconstruct}$ fails for some $j \in [K]$, which occurs only if there is no set of $n-t$ views among $\{\mathsf{View}_{i,j}^{(X_2)*}\}_{i \in \mathcal{J}}$ which are all mutually consistent. In this case, the probability that views $(\mathsf{View}_{i,j}^{(X_2)*})_{i \in \mathcal{I}^{(1)}}$ opened to $\mathcal{S}_2$ earlier are all mutually consistent (conditioned on these views being decommitted successfully) is $< \binom{n-t}{t} / \binom{|\mathcal{J}|}{t} = \mathsf{negl}(n)$, since $|\mathcal{J}| \geq n - \frac{t}{2}$. We conclude that, if $\mathcal{S}_2$ has not already sent abort to P_2^* after checking $(\mathsf{View}_{i,j}^{(X_2)*})_{i \in \mathcal{I}^{(1)}}$, then there is a negligible probability of $\mathsf{Reconstruct}$ failing.

Therefore, $\mathcal{S}_2$ aborts the simulation with negligible probability, and $\mathcal{D}_0 \approx \mathcal{D}_1$.

Experiment $\mathcal{H}_2$. In this experiment, $\mathcal{S}_2$ executes the protocol identically to $\mathcal{H}_1$, except in the end it does not send $y \leftarrow \mathsf{Dec}^{\mathsf{FHE}}(\mathtt{sk}, \mathsf{ct}^{(y)*})$ to P_2^*. Instead, it invokes $\mathcal{F}_{\mathsf{2PC\text{-}PKO}}$ with P_2^*'s extracted input x_2^*, and sends the output $y^* = f(x_1, x_2^*)$ to both P_2^* and $\mathcal{F}_{\mathsf{2PC\text{-}PKO}}$. $\mathcal{F}_{\mathsf{2PC\text{-}PKO}}$ then sends output y^* to honest P_1.

We now show that y and y^*, as defined above, are computationally indistinguishable. Our argument, as shown below, relies on the soundness of MPC-in-the-head with protocol Π_g and of the PCPP system $(\mathcal{P}, \mathcal{V})$, as well as the correctness and strong commitment properties of VSS.

- Firstly, if the views $(\mathsf{View}_i^{(V)*})_{i \in \mathcal{I}^{(1)}}$ received from P_2^* are all mutually consistent and contain output $\mathtt{accept}$, then there exists $(\psi_{i,k})_{i \in [n], k \in [Q]}$ such that $\Phi(u = (x_1, y, \ell_2), r_M, (b_k \leftarrow \mathsf{recons}(\psi_{1,k}, \dots, \psi_{n,k}))_{k \in [Q]}) = \mathtt{accept}$, with overwhelming probability. We follow the reasoning of [IKOS07] to justify this. If all inconsistencies in the complete set of views $(\mathsf{View}_i^{(V)*})_{i \in [n]}$ can be resolved by eliminating at most t views, then the corresponding execution of Π_g can be achieved by an adversary corrupting at most t players. By the perfect t-robustness of Π_g, if there is no $(\psi_{i,k})_{i \in [n], k \in [Q]}$ satisfying the above relation Φ, then all the uncorrupted ($\geq n-t$) players output $\mathtt{reject}$ in Π_g. Since t is at least a constant fraction of n, the probability that none of the opened views $(\mathsf{View}_i^{(V)*})_{i \in \mathcal{I}^{(1)}}$ contains output $\mathtt{reject}$ is $\mathsf{negl}(n)$. On the other hand, if there is no set of $n-t$ consistent views, then $(\mathsf{View}_i^{(V)*})_{i \in \mathcal{I}^{(1)}}$ reveals at least one inconsistency with overwhelming probability.
- Next, if for each $i \in \mathcal{I}^{(1)}$, the view $\mathsf{View}_i^{(V)}$ is consistent with $(\mathsf{View}_{i,j}^{(X_2)})_{j \in [K]}$, and also with $\mathsf{View}_{i,k}^{(\pi)}$ for each k (where $q_k \leq \ell_p$), then for each $i \in [n]$, with overwhelming probability,

$(\psi_{i,k})_{k\in[Q]}$ from above contains views $(\mathsf{View}^{(\pi)}_{i,q_k})_{q_k\le\ell_p}$ and $(\mathsf{View}^{(X_2)}_{i,q_k-\ell_p})_{q_k>\ell_p}$, which are all consistent across indices $i\in[n]$, except on at most t indices. (If more than t indices are inconsistent, the random choice of $\mathcal{I}^{(1)}$ reveals at least one inconsistency with overwhelming probability.) By the strong commitment property of Definition 3, there must exist $(\pi_{q_k})_{q_k\le\ell_p}$ and $(X_{2,q_k-\ell_p})_{q_k>\ell_p}$, for $k\in[Q]$, such that these are VSS-shared in the views $(\mathsf{View}^{(\pi)}_{i,q_k})_{q_k\le\ell_p}$ and $(\mathsf{View}^{(X_2)}_{i,q_k-\ell_p})_{q_k>\ell_p}$ respectively, and Φ accepts as above.
- Finally, if queries $(q_k)_{k\in[Q]}$ have been generated honestly using $\mathsf{Query}(u=(x_1,y,\ell_2),r_M)$, then by the soundness property of PCPP given by Definition 15, since Φ outputs $\mathtt{accept}$ as above, it must be the case (w.h.p.) that $\mathcal{R}((u=(x_1,y,\ell_2),X_2'),w)=\mathtt{accept}$, for some witness w and X_2' which is δ-close to $X_2=(X_{2,1},\dots,X_{2,K})$. In other words, by the definition of $\mathcal{R}(\cdot)$ in Figure 4, there exists (w.h.p., by Theorem 3) X_2' such that $x_2\leftarrow\mathcal{E}.\mathsf{Decode}(X_2')$, x_2 is of length ℓ_2, and $y=f(x_1,x_2)$.
- Since $\delta<1/2$, we have $\mathcal{E}.\mathsf{Decode}(X_2)=\mathcal{E}.\mathsf{Decode}(X_2')=x_2$. Moreover, by the correctness and strong commitment of VSS (Definition 3), we have with overwhelming probability, $x_2=x_2^*$, the input extracted by $\mathcal{S}_2$. This follows because $\mathcal{S}_2$ extracted x_2^* using the same $(\mathsf{View}^{(X_2)}_{i,j})_{j\in[K]}$ in the soundness proof above, of which $\ge n-t$ views must be consistent as argued earlier.

Overall, we find that if all required consistency checks are successful, then with overwhelming probability, there exists x_2 of length ℓ_2 such that $y=f(x_1,x_2)$, and moreover $x_2=x_2^*$. Therefore, y and $y^*=f(x_1,x_2^*)$ are computationally indistinguishable to both P_2^* and P_1, and we conclude $\mathcal{D}_1\approx\mathcal{D}_2$.

Experiment $\mathcal{H}_3$. In this experiment, $\mathcal{S}_2$ executes the protocol identically to $\mathcal{H}_2$, except instead of invoking $\mathcal{F}_{\mathsf{2PC}}$ with P_2^* to get $\mathcal{I}^{(2)}$, it randomly samples indices $i_1^{(2)*},\dots,i_t^{(2)*}\xleftarrow{\$}[n]$ on its own. Let $\mathcal{I}^{(2)*}$ denote the set $\{i_1^{(2)*},\dots,i_t^{(2)*}\}$. $\mathcal{S}_2$ then emulates the $\mathcal{F}_{\mathsf{2PC}}$ functionality itself to output $\mathcal{I}^{(2)*}$, and thus P_2^* receives $\mathcal{I}^{(2)*}$ instead of $\mathcal{I}^{(2)}$. Next, $\mathcal{S}_2$ decommits $(\mathtt{com}_i^{(x_1)})_{i\in\mathcal{I}^{(2)*}}$ instead of $(\mathtt{com}_i^{(x_1)})_{i\in\mathcal{I}^{(2)}}$. Since both are uniformly random subsets of $[n]$, P_2^*'s view distribution in $\mathcal{H}_3$ is clearly identical with that in $\mathcal{H}_2$, i.e., $\mathcal{D}_2\equiv\mathcal{D}_3$.

Experiment $\mathcal{H}_4$. In this experiment, $\mathcal{S}_2$ executes the protocol identically to $\mathcal{H}_3$, except that it samples $\mathcal{I}^{(2)*}$ at the very beginning, and on the indices of views which do not get decommitted (i.e., $i\notin\mathcal{I}^{(2)*}$), it replaces the commitment to a view of x_1 with another commitment to a view of a dummy all-zeros string 0^{ℓ_1} of the same length. Since these replaced commitments are never opened, correctness is preserved. In more detail, $\mathcal{S}_2$ does the following:

- $\mathcal{S}_2$ locally emulates a dealer with input a dummy all-zeros string 0^{ℓ_1}, along with n players $p_1,\dots,p_n$, and it runs the VSS sharing protocol Share to obtain n views $\mathsf{View}_1^{(x_1)*},\dots,\mathsf{View}_n^{(x_1)*}$.
- For each $i\in\mathcal{I}^{(2)}$, $\mathcal{S}_2$ sends (as before in $\mathcal{H}_3$) an extractable commitment $\mathtt{com}_i^{(x_1)}$ of $(\mathsf{View}_i^{(x_1)}\,\|\,r_i)$, for view $\mathsf{View}_i^{(x_1)}$ of x_1.
- For each $i\in[n]\setminus\mathcal{I}^{(2)}$, $\mathcal{S}_2$ sends an extractable commitment $\mathtt{com}_i^{(x_1)*}$ of $(\mathsf{View}_i^{(x_1)*}\,\|\,r_i)$, for view $\mathsf{View}_i^{(x_1)*}$ of 0^{ℓ_1}, instead of $\mathtt{com}_i^{(x_1)}$.
- $\mathcal{S}_2$ sends (as before in $\mathcal{H}_2$) $(\mathsf{ct}_i^{(x_1)})_{i\in[n]}$ to P_2^*.

$\mathcal{S}_2$ executes the rest of the protocol identically to $\mathcal{H}_3$. Since commitments are performed independently of each other, it suffices to show that P_2^* cannot distinguish between $\mathtt{com}_i^{(x_1)}$ and $\mathtt{com}_i^{(x_1)*}$ for any $i\in[n]\setminus\mathcal{I}^{(2)*}$. This follows directly from Definition 19; $\mathtt{com}_i^{(x_1)}$ and $\mathtt{com}_i^{(x_1)*}$ are generated by the commitment phase of the computationally hiding scheme $\Pi_{\mathtt{comExt}}$, on inputs $(\mathsf{View}_i^{(x_1)}\,\|\,r_i)$ and $(\mathsf{View}_i^{(x_1)*}\,\|\,r_i)$ respectively, and are thus computationally indistinguishable for P_2^*, a PPT receiver. We thus have $\mathcal{D}_3\approx\mathcal{D}_4$.

Experiment $\mathcal{H}_5$. In this experiment, $\mathcal{S}_2$ executes the protocol identically to $\mathcal{H}_4$, except that, on the indices of views which do not get decommitted (i.e., $i\notin\mathcal{I}^{(2)*}$), it replaces the ciphertext of the view of x_1 with another ciphertext of the view of 0^{ℓ_1}. In more detail, $\mathcal{S}_2$ does the following:

- $\mathcal{S}_2$ computes ciphertext $\mathsf{ct}_i^{(x_1)*} \xleftarrow{\$} \mathsf{Enc}^{\mathsf{FHE}}(\mathtt{pk}, \mathsf{View}_i^{(x_1)*})$ for each $i \in [n]$.
- $\mathcal{S}_2$ sends the same commitments to P_2^* as before in $\mathcal{H}_3$.
- For each $i \in \mathcal{I}^{(2)}$, $\mathcal{S}_2$ sends (as in $\mathcal{H}_4$) $\mathsf{ct}_i^{(x_1)}$, which encrypts $\mathsf{View}_i^{(x_1)}$ of x_1.
- For each $i \in [n] \setminus \mathcal{I}^{(2)}$, $\mathcal{S}_2$ sends $\mathsf{ct}_i^{(x_1)*}$, which encrypts $\mathsf{View}_i^{(x_1)*}$ of 0^{ℓ_1}.

$\mathcal{S}_2$ executes the rest of the protocol identically to $\mathcal{H}_4$. Since encryptions are performed independently of each other, it suffices to show that P_2^* cannot distinguish between $\mathsf{ct}_i^{(x_1)}$ and $\mathsf{ct}_i^{(x_1)*}$ for any $i \in [n] \setminus \mathcal{I}^{(2)}$. Towards a contradiction, suppose there exists $i \in [n] \setminus \mathcal{I}^{(2)}$ such that P_2^* can distinguish between $\mathsf{ct}_i^{(x_1)}$ and $\mathsf{ct}_i^{(x_1)*}$, which are obtained by encrypting $\mathsf{View}_i^{(x_1)}$ and $\mathsf{View}_i^{(x_1)*}$ respectively, with noticeable probability (over the randomness of $\mathsf{Gen}^{\mathsf{FHE}}$ and $\mathsf{Enc}^{\mathsf{FHE}}$). Then, we can construct an adversary $\mathcal{A}$ for the CPA-security game of the FHE scheme as follows: On receiving $\mathtt{pk}$ from the challenger, $\mathcal{A}$ sends back $(m_0 = \mathsf{View}_i^{(x_1)}, m_1 = \mathsf{View}_i^{(x_1)*})$. On receiving $\mathsf{ct}_b \xleftarrow{\$} \mathsf{Enc}^{\mathsf{FHE}}(\mathtt{pk}, m_b)$ for $b \xleftarrow{\$} \{0,1\}$ from the challenger, $\mathcal{A}$ sends ct_b to P_2^*, and forwards P_2^*'s output to the challenger. Then, $\mathcal{A}$ succeeds in this game with noticeable probability, and this contradicts the CPA-security assumption of the FHE scheme. Hence, if the FHE scheme is CPA-secure, we have $\mathcal{D}_4 \approx \mathcal{D}_5$.

Experiment $\mathcal{H}_6$. In this experiment, $\mathcal{S}_2$ executes the protocol identically to $\mathcal{H}_5$, except that even on indices of views to be decommitted (i.e., $i \in \mathcal{I}^{(2)*}$), it replaces the commitments and ciphertexts of the views of x_1 with those of 0^{ℓ_1}. Formally, for each $i \in \mathcal{I}^{(2)*}$, $\mathcal{S}_2$ sends $\mathsf{ct}_i^{(x_1)*}$ and $\mathtt{com}_i^{(x_1)*}$ to P_2^*, instead of $\mathsf{ct}_i^{(x_1)}$ and $\mathtt{com}_i^{(x_1)}$ respectively. After emulating $\mathcal{F}_{\mathsf{2PC}}$ which outputs $\mathcal{I}^{(2)*}$, $\mathcal{S}_2$ decommits $(\mathtt{com}_i^{(x_1)*})_{i \in \mathcal{I}^{(2)*}}$ (instead of $(\mathtt{com}_i^{(x_1)})_{i \in \mathcal{I}^{(2)*}}$) to $(\mathsf{View}_i^{(x_1)*})_{i \in \mathcal{I}^{(2)*}}$.

We first argue that the distributions of $(\mathsf{View}_i^{(x_1)})_{i \in \mathcal{I}^{(2)*}}$, which are VSS shares of x_1, and $(\mathsf{View}_i^{(x_1)*})_{i \in \mathcal{I}^{(2)*}}$, which are VSS shares of 0^{ℓ_1}, are identical. This follows directly from the privacy condition of Definition 3; since $|\mathcal{I}^{(2)}| = t$, the joint view of these t indices is distributed independently of the underlying secret (x_1 or 0^{ℓ_1}), which is the only input to computing these views. Moreover, since $(\mathsf{ct}_i^{(x_1)*})_{i \in \mathcal{I}^{(2)*}}$ and $(\mathsf{ct}_i^{(x_1)})_{i \in \mathcal{I}^{(2)*}}$ each take as input only $(\mathsf{View}_i^{(x_1)*})_{i \in \mathcal{I}^{(2)*}}$ and $(\mathsf{View}_i^{(x_1)})_{i \in \mathcal{I}^{(2)*}}$ respectively, the former two must be identically distributed as well. By the same argument, $(\mathtt{com}_i^{(x_1)*})_{i \in \mathcal{I}^{(2)*}}$ and $(\mathtt{com}_i^{(x_1)})_{i \in \mathcal{I}^{(2)*}}$ are also identically distributed. Thus, we have $\mathcal{D}_5 \equiv \mathcal{D}_6$. Further, note that experiment $\mathcal{H}_6$ exactly corresponds to the ideal world execution in Figure 7, i.e., $\mathcal{D}_6 \equiv \mathcal{D}_{\mathcal{I}}$.

Combining the hybrids. By a combination of the above hybrid arguments, we get $\mathcal{D}_{\mathcal{R}} \approx \mathcal{D}_{\mathcal{I}}$, i.e., the joint distribution of view of malicious party P_2^* and output of honest party P_1 is computationally indistinguishable in the real and ideal worlds. Therefore, we conclude that Π_{2PC} securely realizes $\mathcal{F}_{\mathsf{2PC\text{-}PKO}}$ in the presence of a malicious adversary corrupting P_2. □

C Complexity Analysis of Π_{2PC}

Round complexity. We first enumerate the number of rounds of communication required in Π_{2PC} as described in Figure 5, without any additional parallelization. The steps of Π_{2PC} which involve communication and the corresponding number of rounds are as follows:

- Steps 1 and 2(c): 3 rounds for extractable commitment [PW09].
- Step 3: 5 rounds for $\mathcal{F}_{\mathsf{2PC}}$ invocation [ORS15].
- Step 4: 1 round for decommitment.
- Step 8(h): 3 rounds for extractable commitment.
- Step 9: 1 round.
- Step 13: 1 round for standard non-interactive commitment.
- Step 14: 5 rounds for $\mathcal{F}_{\mathsf{2PC}}$ invocation.

- Steps 15(a)–(b): 1 round for decommitment.
- Step 16(d): 1 round.

This implies a total of 21 rounds. However, note that we can further compress the number of rounds, by performing some rounds of the $\mathcal{F}_{\mathsf{2PC}}$ invocation in parallel with previous rounds of $\mathit{\Pi}_{\mathsf{2PC}}$.

A key observation here is that, given any 2PC protocol, we can apply the following (information theoretic) transformation to ensure that neither party learns any information about the output until the *last* round of the protocol.

- In addition to inputs x_1 and x_2 to the 2PC, each party P_i ($i \in \{1,2\}$) also supplies (as private inputs) uniformly random one-time pads $k_{i,0}$ and $k_{i,i}$, and information theoretic MAC [BDOZ11] key $\mathit{\Delta}_i$.
- The 2PC protocol does the following:
  - Compute $y = f(x_1, x_2)$.
  - Compute MACs M_1 and M_2 of y, using keys $\mathit{\Delta}_1$ and $\mathit{\Delta}_2$ respectively.
  - For $(c_0, c_1, c_2) \leftarrow (y \oplus k_{1,0} \oplus k_{2,0}, M_1 \oplus k_{2,2}, M_2 \oplus k_{1,1})$, output (c_0, c_1) to P_1 and (c_0, c_2) to P_2.
- In parallel with the last round of the 2PC protocol, P_1 and P_2 send their keys $k_{1,0}, k_{1,1}$ and $k_{2,0}, k_{2,2}$, respectively, to each other.
- Locally, each party P_i ($i \in \{1,2\}$) decrypts the 2PC output to obtain y and M_i (as $y \leftarrow c_0 \oplus k_{1,0} \oplus k_{2,0}$ and $M_i \leftarrow c_i \oplus k_{3-i,3-i}$), and verifies that M_i is a valid MAC on y. If successful, it outputs y; otherwise $\perp$.

It is clear that the output of the underlying 2PC reveals nothing to either party P_i, as both c_0 and c_i contain the other party's one-time pads, which are only revealed to P_i in parallel with the last round of 2PC. And thus, we may assume without loss of generality that a 2PC protocol (in particular, the one we use in $\mathit{\Pi}_{\mathsf{2PC}}$ to generate random indices) leaks no information to the parties about its output until the last round of the protocol.

Then, $\mathit{\Pi}_{\mathsf{2PC}}$ remains secure if all rounds, except for the last one, of each $\mathcal{F}_{\mathsf{2PC}}$ invocation (Steps 3 and 14) are performed in parallel with previous rounds of $\mathit{\Pi}_{\mathsf{2PC}}$. This brings the round cost down by 3 in Step 3 (since $\mathit{\Pi}_{\mathsf{2PC}}$ only has 3 rounds before this step), and 4 in Step 14. Therefore, instead of 21, we are able to compress the round complexity of $\mathit{\Pi}_{\mathsf{2PC}}$ to 14.

Communication complexity. We now calculate the amount of communication required between the two parties in $\mathit{\Pi}_{\mathsf{2PC}}$. The steps of $\mathit{\Pi}_{\mathsf{2PC}}$ which involve communication and the corresponding amount of communication are as follows (recall that $n = O(\lambda)$):

- Step 1: The keys communicated have size $O(\mathsf{poly}(\lambda))$.
- Step 2(c): The views of x_1 have size $O(\ell_1 \cdot \mathsf{poly}(\lambda))$ [GIKR01], and thus the ciphertexts and extractable commitments [PW09] communicated have size $O(\ell_1 \cdot \mathsf{poly}(\lambda))$.
- Step 3: The $\mathcal{F}_{\mathsf{2PC}}$ invocation simply samples $O(\mathsf{poly}(\lambda))$ amount of randomness, and thus involves $O(\mathsf{poly}(\lambda))$ amount of communication.
- Step 4: The views of x_1 have size $O(\ell_1 \cdot \mathsf{poly}(\lambda))$ [GIKR01], and thus the decommitments communicated have size $O(\ell_1 \cdot \mathsf{poly}(\lambda))$.
- Step 8(h): The ciphertext of y has size $O(\ell_o \cdot \mathsf{poly}(\lambda))$. Each vector commitment has size $O(\mathsf{poly}(\lambda))$ (due to conciseness; see Definition 7). Each view of X_2 has size $O(K \cdot \mathsf{poly}(\lambda))$, and thus each extractable commitment of an encrypted view of X_2 also has size $O(K \cdot \mathsf{poly}(\lambda))$, where $K = \ell_2^{1+\alpha}$. The total communication in this step is thus $O((\ell_o + \ell_2^{1+\alpha}) \cdot \mathsf{poly}(\lambda))$.
- Step 9: The runtime of $\mathcal{R}_f((u,v),w)$ as defined in Figure 4 is $O(K \cdot \mathsf{polylog}(K))$ (for decoding X_2) + $O(|f|)$ (for evaluating f). Plugging in $K = \ell_2^{1+\alpha}$, this is $O(\ell_2^{1+\alpha} \cdot \mathsf{polylog}(\ell_2) + |f|)$. By Theorem 3, for the PCPP on NP relation $\mathcal{R}_f$, the size of verifier randomness r_M communicated in this step is given by $O(\mathsf{polylog}(\ell_2^{1+\alpha} \cdot \mathsf{polylog}(\ell_2) + |f|)) = O(\mathsf{polylog}(\ell_2 + |f|))$.

- Step 13: By Theorem 3, the PCPP verifier complexity is $O(\mathsf{polylog}(\ell_2+|f|))$ as above. Thus, each view of the MPC-in-the-head execution of the PCPP verifier has size $O(\mathsf{polylog}(\ell_2+|f|)\cdot\mathsf{poly}(\lambda))$, which implies each commitment of such an encrypted view also has size $O(\mathsf{polylog}(\ell_2 + |f|) \cdot \mathsf{poly}(\lambda))$.
- Step 14: This step has $O(\mathsf{poly}(\lambda))$ communication, as in Step 3.
- Step 15(a): Decommitments of encrypted MPC views have size $O(\mathsf{polylog}(\ell_2 + |f|) \cdot \mathsf{poly}(\lambda))$. Decommitments of encrypted views of X_2 have size $O(K \cdot \mathsf{poly}(\lambda))$. The total communication in this step is thus $O((\mathsf{polylog}(|f|) + \ell_2^{1+\alpha}) \cdot \mathsf{poly}(\lambda))$.
- Step 15(b): By Theorem 3, the PCPP query complexity is $O(\mathsf{polylog}(\ell_2 + |f|))$ as above. Thus, the encrypted views of all queried locations of the proof have size $O(\mathsf{polylog}(\ell_2 + |f|) \cdot \mathsf{poly}(\lambda))$. This is also the size of the encrypted proofs of vector openings for all queried locations of the proof (due to conciseness; see Definition 7), and thus the total communication in this step is $O(\mathsf{polylog}(\ell_2 + |f|) \cdot \mathsf{poly}(\lambda))$.
- Step 16(d): The communicated output has size ℓ_o.

Collecting the above calculations, we conclude that the total communication complexity of Π_{2PC} is $O\big((\ell_1 + \ell_2^{1+\alpha} + \ell_o + \mathsf{polylog}(|f|)) \cdot \mathsf{poly}(\lambda)\big)$.