Round-optimal Honest-majority MPC in Minicrypt and with Everlasting Security

Benny Applebaum[*] Eliran Kachlon[*] Arpita Patra[†]

Abstract

We study the round complexity of secure multiparty computation (MPC) in the challenging model where full security, including guaranteed output delivery, should be achieved at the presence of an active rushing adversary who corrupts up to half of parties. It is known that 2 rounds are insufficient in this model (Gennaro et al., Crypto 2002), and that 3 round protocols can achieve computational security under public-key assumptions (Gordon et al., Crypto 2015; Ananth et al., Crypto 2018; and Badrinarayanan et al., ASIACRYPT 2020). However, despite much effort, it is unknown whether public-key assumptions are inherently needed for such protocols, and whether one can achieve similar results with security against computationally-unbounded adversaries.

In this paper, we use Minicrypt-type assumptions to realize 3-round MPC with full and active security at the presence of honest-majority. Our protocols come in two flavors: standard computational security and online-computational security with *statistical everlasting security*, i.e., the protocol is secure against adversaries that are computationally unlimited after the protocol execution. Specifically, we prove the following results:

- (Statistical everlasting security) Every NC^1 functionality can be computed in 3 rounds given a hash function that is modeled as a *non-programable* random oracle. The random oracle can be replaced with a common reference string (CRS), a statically-hiding non-interactive commitments (NICOM) and a family of "$\mathbb{R}$-intractable" hash functions for which it is hard to find inputs that are correlated under some explicit sparse algebraically-simple relation $\mathbb{R}$.
- (Computational security) Every efficiently-computable function can be realized in 3 rounds in the plain model assuming computationally-hiding NICOM and $\mathbb{R}$-intractable hash functions.

The assumption of $\mathbb{R}$-intractability can be removed in both settings at the expense of making any one of the following relaxations: (a) increasing the round complexity to 4; (b) restricting the attention to linear functionalities; or (c) restricting the adversary to be non-rushing. In fact, we can support an intermediate notion of *semi-rushing* adversaries that, in each round, can delay their messages till *almost all* honest parties speak. The latter notion may be of independent interest.

[*]Tel-Aviv University, Israel `bennyap@post.tau.ac.il`, `elirn.chalon@gmail.com`
[†]Indian Institute of Science, Bangalore, India `arpita@iisc.ac.in`

1 Introduction

Interaction is a valuable and expensive resource in cryptography and distributed computation. Consequently, a huge amount of research has been devoted towards characterizing the amount of interaction, typically measured via round complexity, that is needed for various distributed tasks (e.g., Byzantine agreement [45, 27, 30], coin flipping [24, 47], and zero-knowledge proofs [35, 22]) under different security models. In this paper, we focus on the problem of general secure-multiparty-computation (MPC) in the challenging setting of *full security* (including guaranteed output delivery) with *maximal resiliency*. That is, even an active (aka Byzantine or malicious) adversary that controls a minority (up to half) of the parties should not be able to violate privacy or to prevent the honest parties from receiving a valid output. In this setting, originally presented in the classical work of Rabin and Ben-Or [52], we assume that each pair of parties is connected by a secure and authenticated point-to-point channel and that all parties have access to a common broadcast channel, which allows each party to send a message to all players and ensures that the received message is identical.

The round complexity of honest-majority fully-secure MPC protocols was extensively studied. The lower-bound of [33, 37] shows that two rounds are insufficient for this task even when the parties are given access to a common reference string (CRS). In [6] a 5-round protocol was constructed based on Threshold Fully-Homomorphic Encryption (TFHE) and Non-Interactive Zero-Knowledge proofs (NIZK). An optimal round complexity of three, was later obtained by [37] in the CRS model by relying on a stronger variant of TFHE that can be based on the learning with errors (LWE) assumption. Later in [1, 11], the CRS was removed and LWE was replaced by weaker public-key primitives like general public-key cryptosystems and two-round witness indistinguishable proofs (ZAPs). (The latter can be based on primitives like trapdoor permutations [28] and indistinguishability obfuscation [17], or on intractability assumptions related to bilinear groups [39] and LWE [10, 38].)

The above results may give the impression that public-key assumptions are essential for honest-majority fully-secure MPC. However, if one puts no restriction on the round complexity, then, as shown by Rabin and Ben-Or [52], one can obtain unconditional results and no assumptions are needed at all! Specifically, every efficiently computable function can be securely computed with statistical security against computationally-unbounded adversaries.[1] Constant-round versions of this protocol are known either with an exponential dependency in the circuit-depth (or space-complexity) of the underlying function [42], or with computational security under the weakest-known cryptographic assumption: the existence of one-way functions [15, 25]. Moreover, for the special case of 3 parties (and single corruption), 3-round protocols were constructed by [51] based on injective one-way functions.

This leaves an intriguing *gap* between general-purpose *optimal-round* protocols to protocols with larger round complexity, both in terms of the underlying assumptions and with respect to the resulting security notion. We therefore ask:

> **Q1:** Are public-key assumptions inherently needed for 3-round fully-secure honest-majority MPC? Is it possible to replace these assumptions with symmetric-key assumptions, possibly in an idealized form (e.g., in the random oracle model)?

[1]Interestingly, perfect security is impossible to achieve in this setting as it requires a strong honest majority of $2n/3$ [16].

Q2: Is it possible to obtain 3-round fully-secure honest-majority MPC with some form of unconditional security against computationally-unbounded adversaries?

We answer these questions to the affirmative. We show that 3-round MPC with full security at the presence of honest-majority can be realized based on Minicrypt-type assumptions and present variants of our protocol that achieve *statistical everlasting security*. To the best of our knowledge, both these results are achieved for the first time: (a) everlasting security for fully-secure honest-majority 3-round protocols *regardless of the underlying assumptions* and (b) computational security for fully-secure honest-majority 3-round protocols that are not based on public-key assumptions. We continue with a detailed description of our results.

1.1 Our Contribution

It will be instructive to start by assuming that we are given an idealized symmetric-key primitive, namely a random oracle.

Theorem 1.1 (Informal)**.** *Given a hash function that is "modelled as a non-programable random oracle", every efficiently-computable functionality can be realized in 3 rounds with full security against an active rushing adversary that corrupts a minority of the parties. Moreover, NC^1 functions can be realized with* statistical everlasting security*.*

We emphasize the importance of this theorem below and then continue to discuss its corollaries obtained by (a) substituting the random oracle assumption with a milder correlation intractability assumption and (b) avoiding it completely riding on different relaxations.

Everlasting security. The notion of statistical everlasting security [48] can be viewed as a hybrid version of statistical and computational security. During the run-time, the adversary is assumed to be computationally-bounded (e.g., cannot find collisions in the hash function) but after the protocol terminates, the adversary hands its view to a computationally-unbounded analyst who can apply arbitrary computations in order to extract information on the inputs of the honest parties.[2] This feature is one of the main advantages of information-theoretic protocols: after-the-fact secrecy holds regardless of technological advances and regardless of the time invested by the adversary. It is important to mention that this is achieved without exploiting the "random oracle" property of the hash function. That is, the computationally-unbounded analyst is allowed to read the whole truth-table of the hash function.

The difference between everlasting and computational security is *fundamental* and is analogous to the difference between statistical commitments and computational commitments or statistical ZK arguments vs. computational ZK arguments (see, e.g., the discussions in [19, 50]). In both the former cases, we get computational security against "online cheating" and statistical security against after-the-fact attacks. We therefore believe that our protocols with everlasting security form an important *feasibility* result regardless of the underlying assumptions. Previous results inherently fail to achieve this due to the use of public-key cryptography.

All our everlasting-security protocols are restricted to NC^1. More generally, the computational complexity of these protocols grows exponentially with the depth or space of the underlying function. This is expected since even for strictly-weaker notions of security (e.g., passive statistical

[2]Technically, in the UC-framework we allow the environment to output its view and require statistical indistinguishability between the real and ideal experiments. For details, refer to Appendix A.2.

security against a single corrupted party), it is unknown how to construct *efficient constant-round* protocols for functions beyond NC^1 and log-space. (In fact, this is a well-known open problem that goes back to [14].)

Are we really in Minicrypt? There are several different interpretations for the notion of Minicrypt ranging from a minimal view (OWFs exist) to a maximal view ("random-oracle" like objects exist). While the choice is subjective we believe that even the maximalist view is meaningful when compared to *public-key-cryptosystems* (on which the alternative solutions rely). Still, we will describe several ways to relax the random oracle assumption, possibly at the expense of weakening the end result. We emphasize that even our strongest implications require only a relatively weak version of random oracle, at the form of a keyed hash function that achieves collision-resistance and correlation intractability with respect to a *specific* simple relation, as we elaborate next. Such hash functions are not subject to random oracle impossibility results [21], and they are likely to be realizable in the standard model. In fact, it will be very surprising if ad-hoc constructions of cryptographic hash-functions (e.g., based on block-ciphers) do not satisfy these assumptions.

1.1.1 Replacing the Random Oracle by Correlation Intractable Hashing

We can replace the random oracle by some simple form of correlation-intractable hash functions [21]. Unlike some of the previous instantiations of these primitive that require intractability against a large (non-explicit) family of relations, we will only consider a *single* concrete relation. Specifically, we need a family H of hash functions such that given a random hash function $h \in H$ it is hard to find a vector of inputs x that satisfies some explicit sparse relation $\mathbb{R}^h$ that has a simple algebraic description and can be computed in fixed polynomial time. This type of assumption is similar in spirit to collision-resistance (see Section 3.4 and A.3) and it seems likely that practical hash functions are $\mathbb{R}$-intractable. We obtain the following result from $\mathbb{R}$-intractibility.

Theorem 1.2. *Assuming a family of $\mathbb{R}$-intractable hash functions and a statistically-hiding NICOM, every NC^1 function can be realized in 3 rounds in the CRS model with full everlasting security against a rushing adversary that corrupts a minority of the parties. Moreover, if the NICOM is replaced with a computationally-hiding NICOM with sub-exponential privacy and security is downgraded to computational, then the theorem holds for any efficiently-computable function and in the plain model (no CRS).*

It is known that statistically-hiding NICOMs necessitates the use of CRS and can be constructed based on collision-resistance hash functions (CRH) [40, 26].[3] Computationally-hiding NICOMs can be constructed in the plain model based on injective one-way functions [18, 53, 36] or even on standard one-way functions assuming worst-case complexity-theoretic derandomization assumptions [49, 12]. The requirement for sub-exponential privacy is needed for technical reasons, and for this purpose it suffices to assume that the underlying one-way function (or injective one-way function) cannot be inverted in polynomial-time except with sub-exponentially small probability. This seems to be a relatively mild assumption (see Remark A.16).

[3]Indeed, this is the only use of CRS in our protocols. Using the CRH based instantiation, one can assume that the CRS consists of a randomly selected hash function. Furthermore, this CRS can be selected in a single round by letting each party select its own, sufficiently shrinking, CRH and by using the trivial CRH-combiner.

A random-oracle like hash function, satisfies correlation-intractibility (see Section A.3) and can be used to implement each of the ingredients in Theorem 1.2. Therefore, Theorem 1.1 follows from Theorem 1.2.

1.1.2 Avoiding Correlation Intractability

We can bypass the requirement of correlation intractable hashing, if we settle for any of the following relaxations: (a) allow an additional round of interaction (i.e., settle for a sub-optimal 4-round protocol) (b) consider only linear functions, or (c) restrict the adversary's control on the network and allow her only to be *semi-rushing*. We elaborate on the latter notion which is new to our work.

Rushing, non-rushing and semi-rushing. A *rushing* adversary may delay sending the messages of the corrupted parties in any given round until the honest parties send their messages in that round; thus, the messages of the corrupted parties in a given round may depend on the messages of the honest parties in the same round. In contrast, a *non-rushing* adversary must decide on what messages the corrupted parties should send in any given round before seeing the honest parties' messages in that round. The rushing model may seem over-pessimistic whereas the non-rushing setting may be viewed as being over-optimistic. We introduce an intermediate model of *semi-rushing adversary* whose $\imath$th round messages may depend on the $\imath$th messages of all but *one* honest party (that can be selected by the adversary adaptively during the protocol and can be updated from round to round). This notion of *semi-rushing* adversary seems reasonable in some natural scenarios and we view it as a conceptual contribution of this work. We postpone more detailed motivation to Appendix A.2. Overall, we get the following theorem.

Theorem 1.3. *Assuming a statistically-hiding NICOM, every NC^1 function f can be realized in the CRS model with everlasting security against an adversary that corrupts a minority of the parties. The protocol can be implemented with (a) 4 rounds if the adversary is rushing; (b) 3 rounds if the adversary is semi-rushing; and (c) 3 rounds if the adversary is rushing and the functionality is linear. Moreover, if the NICOM is replaced with a computationally-hiding NICOM with sub-exponential privacy and security is downgraded to computational, then all three results can be obtained in the plain model (without a CRS) and the first two results apply to any efficiently-computable function.*

To the best of our knowledge, none of the above results were known before. The existence of 3-round MPC even for *non-rushing* adversaries without public-key assumption was open prior to this work. Our results resolve this question under either collision-resistant hashing for everlasting-security (due to statistically-hiding NICOM) or under sub-exponentially-hard injective-OWF for computational-security (due to computationally-hiding NICOM). For linear functions, our 3-round protocols have optimal round complexity, since 2 rounds are insufficient [33] (see Remark 3.11) even with a CRS.

1.1.3 Summary of the results

We summarize our results in Table 1 and compare them to the existing strict honest-majority results. We point out that 3-round (optimal) information-theoretic MPC protocols are known if the corruption threshold is relaxed to one-third [4].

Ref.	Rounds	Threshold	Setup Plain / CRS	Security it / es / cs[†]	Cryptographic Assumptions
[52]	$O(d)$[‡]	$t < n/2$	Plain	it	–
[42][⋆]	> 3 (but constant)	$t < n/2$	Plain	it	–
[15, 25]	> 3 (but constant)	$t < n/2$	Plain	cs	OWF
[51]	3	$t < n/2; t = 1$	Plain	cs	injective OWF
[37]	3	$t < n/2$	CRS	cs	threshold multi-key FHE
[1, 11]	3	$t < n/2$	Plain	cs	PKE, ZAPs
This[⋆]	3	$t < n/2$	CRS	es	$\mathbb{R}$-intractable hash functions[§], statistically-hiding NICOM
This	3	$t < n/2$	Plain	cs	$\mathbb{R}$-intractable hash functions[§], computationally-hiding NICOM

[†] it: information-theoretic, es: everlasting security, cs: computational security.
[‡] d: the depth of the circuit to be computed.
[⋆] For NC^1 circuits
[§] $\mathbb{R}$-intractable hash functions can be avoided if we settle for either (a) 4 rounds or (b) linear functions or (c) semi-rushing adversaries.

Table 1: Comparison of our work with the state-of-the-art relevant results

1.2 Our Techniques

To prove our results, we present a novel 3-round protocol that can be easily adopted to the various settings that are described in the previous section. The protocol strongly relies on several ideas that were introduced by [5, 4] (AKP) in the context of round-optimal perfect/statistical secure MPC at the presence of strong honest majority where the adversary can corrupt up to $t < n/3$ parties. The first and foremost being degree-2 completeness: that is, instead of computing the target functionality g, we use the completeness result of [5] (building on [3]) and compute a degree-2 functionality f. The current setting of $t < n/2$ introduces significant new challenges for which we develop new techniques. We begin by recalling some high-level ideas from the literature of information-theoretic MPC and concretely from [5, 4], that we rely on.

1.2.1 Background

Like many MPC protocols, our protocol will have the following 3-phase structure: (1) *Sharing* of the inputs via verifiable secret sharing (VSS); (2) *Homomorphically computing* the functionality over the shares; and (3) *Opening* the outputs. (This is analogous to FHE-based protocols that follow a similar Encrypt-Compute-Decrypt structure.) Recall that we strive for guaranteed output delivery and so we should generate a valid output even if the adversary cheats.

Tentative outputs and offline rounds. Recall that we have only 3 rounds of communication to implement these 3 phases. However, even VSS alone takes at least 2 rounds [32, 8]. We bypass this problem by running the phases with some overlap. Specifically, we make an extensive use of (1) *tentative-output* protocols that prepare a tentative version of the output in an early round and only later, at the end, approve/reject/correct the tentative output; and (2) *offline-phase* protocols that begin with an *offline*, input-independent, round and only later receive the inputs. Consequently, we can save some rounds by allowing partial overlap between sub-protocols that would have been executed sequentially otherwise. As we will later see, in our setting, this solution introduces several significant challenges, e.g., when tentative outputs turn to be invalid and when the offline information yields selective-opening attacks, due to the higher corruption threshold.[4]

Realizing the homomorphic computation. After the sharing phase, the parties hold a distributed encoding (secret-sharing) of the inputs. Roughly speaking, the encoding is based on bivariate polynomials and may be viewed as a matrix whose ith row is being held ("owned") by the ith party and the jth entry of this row is being held by the jth party who will later use it for consistency checks. Of course, the dealer (the owner of the corresponding input) has a full view of this matrix. The task of homomorphically applying degree-2 computation over inputs that were distributed by different parties is gradually reduced to simpler functionalities, e.g., linear functionalities and single-input functionalities that depend on inputs that arrive from a single party. In [5, 4] these functionalities are further reduced to *guided functionalities* in which the ith party, who owns the ith row of each of the shared values, *guides* a homomorphic computation on her rows. Each other party j *guards* the computation by forcing consistency with the jth entry. In [4], it was shown that, in the statistical setting, such protocols can be based on a *Secure Computation with a Guide* (SCG) primitive that can be implemented in a single round (plus an offline input-independent round). Roughly, this primitive involves a Sender (the guide), a designated Receiver and a Guard, the sender wishes to compute some information on its private input and to send it to the receiver, whereas the guard who has some partial view of the sender's input, wishes to make sure that the computation is consistent with its partial view. The SCG construction of [4] is based on private simultaneous message (PSMs) protocols [29] and it achieves information-theoretic security.

1.2.2 New Challenges and resolutions

Adopting the above framework to handle up to $t < n/2$ corruptions is a non-trivial task. There are several high-level challenges, some of which are as follows. First, noisy polynomial interpolation does not work in this setting and so we cannot use error-correction. Second, the shares generated by VSS in this regime are non-linear (due to the use of non-homomorphic NICOMs), and so we have to devise round-efficient protocols for linear operations. (A task that is trivial when the VSS is linear as in the $t < n/3$ regime.) Third, fully-secure degree-2 computation involves reconstructing a $2t$-degree polynomial with $n \geq 2t + 1$ parties, which is clearly non-trivial because VSS itself requires 2 rounds and t of the corrupt parties may refuse to disclose their shares in round 3. This is again relatively simple in the $t < n/3$ regime, where the honest parties themselves jointly hold enough shares on the $2t$-degree polynomial. At a high level, in order to achieve guaranteed output delivery, we will have to detect misbehaviour of corrupt parties within the *first two* rounds and, in some cases, reveal all the information that is known to honest parties about the shares of such bad parties. This form of detection and revealing is typically expensive in terms of interaction, and one has to carefully implement it in a private way that does not allow for false accusations against honest parties. We continue by highlighting some of the main technical challenges and their resolutions.

The verifiable secret sharing (VSS). We will augment the aforementioned matrix secret sharing (based on bivariate polynomials) with a public commitment to the entire matrix. This commitment

[4]In fact, even the task of formally capturing these worst-case scenarios is fairly complicated and is reflected in the lengthy and somewhat involved definitions of the functionalities that capture our sub-protocols.

will serve as a *ground-truth*. Assuming that the matrix was properly distributed, any disagreement about the value of the (i, j)th entry can be resolved by opening the corresponding commitments (each party receives from the dealer the openings of its entries). Fortunately, such a representation can be distributed within 2 rounds using the VSS protocol of [8]. Moreover, we note that tentative shares can be outputted already after the first round. On the downside, the protocol provides only *weak* consistency guarantees and it does not ensure that the resulting sharing is fully valid. In particular, when the dealer is corrupted, the row of a corrupted party and its corresponding public commitment may be invalid (e.g., does not form a degree-t polynomial). This weakness turns to be problematic and we will have to resolve it later with the help of single-input functionalities. Another challenge with this VSS arises when linear computation is to be performed on shared secrets. While this is free in $t < n/3$ regime, the use of non-homomorphic NICOMs for our case necessitates additional techniques and is resolved via guided linear computation.

General/Guided Linear Computation and publicly-decodable SCG. We will extensively use a protocol for Guided Linear Computation. In such a protocol the ith party, P_i, wishes to guide a linear computation over the ith secondary shares (ith rows) of the inputs. Since she holds the corresponding ith rows, this can be done in a *single online round* (plus an offline round) by running an SCG with each P_j as a guard. In the protocol of [4] this process has to be repeated n times, one time for every potential receiver. Since these copies may be inconsistent (e.g., a corrupted guide may abort some of them), a final error correction is applied to the result. This approach cannot be taken when $t > n/3$. We bypass this problem by constructing a *publicly-decodable* version of SCG (without a designated receiver) under the assumption that the inputs of the sender and the guard are publicly committed. In this setting, one may have to take into account the case where the openings of the sender and the guard are faulty (e.g., due to a bad dealer). We show how to handle such "partial-computation" scenarios in the special case of linear functions. Our new SCG construction makes an additional internal usage of NICOMs. (See Sections 3.1 and 3.3.) Lastly, using n parallel instances of guided linear computation, we can realize any general linear computation on the shared inputs. Though the former is a 2-round protocol, the latter turns out to be a 3-round protocol, which is optimal due to [33].

Triple Secret Sharing (TSS). Following Beaver [13], the key ingredient for moving from linear functionalities to quadratic functionalities is the so-called Triple Secret Sharing functionality in which a dealer secret-shares a random multiplication triple $(a, b, c = ab)$. In the $t < n/3$ setting this is done based on the [16] degree-reduction technique, that again cannot be used here. Instead, we embed the sharing polynomials (A, B, C) inside a vector of high-degree polynomials $Q = (Q_1, \ldots, Q_\ell)$ so that the multiplicative relation holds if and only if the Q polynomials satisfy some quadratic relation. The latter property can be verified by securely evaluating the Q's on a randomly chosen point α that is chosen as a challenge by the verifiers. Crucially, the whole procedure should be terminated after 2 rounds (where tentative shares should be ready after the first round). To achieve this we employ our guided-linear protocol, and let the challenge point α be chosen at the first round. Unfortunately, this raises a security problem: A corrupted *rushing* dealer can break the soundness of the proof by choosing an invalid sharing that is tailored to pass the test induced by α. We present multiple workarounds:

1. (Weakening the network model) First, we observe that the protocol works well for *non-rushing adversaries* who select their messages based on messages that were sent in the pre-

vious round. In fact, by modifying the protocol so that each party sends its own challenge α_i, we can prove that soundness holds as long as the sharing is independent of at least one challenge. We can therefore handle semi-rushing adversaries.

2. (Intractable Hash functions) Moving back to the case of a rushing adversary, we follow the Fiat-Shamir heuristic [31] and generate the challenge α by applying a hash function h to the (public part) of the sharing. Soundness holds, as long as it is hard to find a "bad sharing" that passes the test that is induced by its hashed value. This relation $\mathbb{R}$ has a simple algebraic structure, and, assuming that the NICOM is implemented based on the same hash function (e.g., via [40, 26]) it can be written as a simple algebraic circuit. The key to the hash function can be sampled during the protocol by the verifiers. That is, in the first round, we let each verifier broadcast a random key z_i, and continue as in the previous item with n challenges $\alpha_1, \ldots, \alpha_n$ where α_i is generated locally by applying h_{z_i} to the (public part of the) sharing that is published by the dealer in the first round.

3. (Increasing the round complexity) Alternatively, if we allow 3 rounds for TSS, then α can be chosen in round 2.

See Section 3.5 for full details.

Single-input functionalities, ZK proofs, and strong sharing/VSS. Based on the above protocols, one can realize any degree-2 single-input functionality within 2 rounds, and by using standard tools, get a *general* single-input functionalities. This establishes, for the first time, the feasibility of 2-round protocols for single-input functionalities with optimal security threshold of $t < n/2$ improving over the previous 2-round protocol [33] for such functionalities that achieves a sub-optimal security-threshold of $t < n/6$. As a special case, the resulting protocol yields a distributed zero-knowledge proof that is being used to achieve strongly-consistent VSS. Conveniently, our protocol for single-input functionalities has an important feature: the messages sent at the first round reveal no private information. This allows us to abort the protocol, if needed, without sacrificing privacy. This is useful in case we want to reveal the output of the functionality in round r only if some (public) event occurred in round $r-1$: we can simply execute the first round of the protocol in round $r-1$, and continue the execution in round r only if the event occurred.

Degree-2 computation. Riding on the completeness of degree-2 computation [5], our last goal is to achieve a fully-secure degree-2 computation. Even basing on the tools we have built so far, this task poses several significant challenges. To simplify discussion, we consider the computation for $y = x^{\alpha}x^{\beta} + x^1 + \ldots + x^n$. For simplicity, we denote the party holding x^{α}, x^{β} and x^i by P_α, P_β and P_i. First, building on TSS and guided linear function, we construct guided degree-2 computation. In spirit of guided linear computation, the goal of this primitive is to allow a party P_i to guide a degree-2 computation of the same form as that of y over the ith secondary shares (ith rows) of the inputs and compute a "valid" ith row (that corresponds to degree-t polynomial) for y. (See Section 3.8 for guided degree-2 computation.) We build such a primitive in 2 rounds, with the first round being input-independent. Notably both guided computations (linear and degree-2) offer the feature of identifying the guide as corrupt when it misbehaves.

With guided degree-2 computation, the high-level idea for degree-2 computation is to (1) let each party share all of its inputs in the first round (using strongly-consistent VSS), and (2) let the parties compute the i-th share of $y = x^\alpha \cdot x^\beta + x^1 + \ldots + x^n$ via guided degree-2 computation, guided by P_i. Note that y is shared via a degree-$2t$ polynomial, and since $n \geq 2t+1$, we need to recover *all* of y's shares, including those belonging to corrupt parties. We execute step (1) and input-independent round of step (2) in the first round, while step (2) is executed in the second round over the tentative shares. At this stage, several problems may occur: A corrupt dealer may send a wrong row of its input to P_i in the first round; A corrupt guide P_i might abort its execution of guided degree-2 computation, thus not revealing its share of y; Even an honest guide's result may be only "partially correct", since it may not have received a valid row from a corrupt (yet not discarded) dealer in round 1. To turn the "partially-correct" shares to full shares and to recover the missing shares, we use the third round and a few additional primitives as elaborated below.

In the end of round 2, we identify a set of parties V, which excludes parties who are identified as corrupt so far, e.g. as a part of strongly-consistent VSS or guided degree-2 computation. All the parties' inputs outside V is set to 0. Now we have two cases. First, if at least one of P_α, P_β reside outside V, then our degree-2 computation reduces to a linear computation (as at least one of x^α, x^β is 0), which can be handled through guided linear computation in the third round. However, unaware of the impending situation, we need to execute guided degree-2 computation before we know whether we need to compute a linear function in the third round, so we use a padding trick to ensure that the output of exactly one of these computations is revealed based on whether P_α, P_β belongs to V.

Otherwise, assume that both P_α, P_β reside inside V. In this case, we recover the ith row of y for every corrupt guide P_i not in V, by making all honest parties disclose the complete rows/matrices of P_i. Next, for every P_i that belongs to V, we have a "partial result" from its guided degree-2 computation. Assume that P_i claims in round 1 to not posses a row dealt by a dealer $P_j \in \mathsf{V}$, so this row has not been accounted for in the result. The treatment to account for this missing row differs based on whether the missing row correspond to a term contributing to the degree-2 term (rows of x^α or x^β) or to a linear term (row of x^i). The latter is easier to tackle: the row is ensured to be publicly known as a part of the VSS step and can be added to the partial result. Without going into the details, the former is handled via Beaver's trick accompanied with linear function evaluation and subsequent summing up of the result as in the earlier case.

A complementary situation arises, when a row is accounted for in the result of the guided degree-2 computation and yet its dealer is identified as corrupt and does not belong to V, so we need to eliminate this row from the result. Yet again the treatment for the elimination of the row differs based on whether the concerned row correspond to a term contributing to the degree-2 term (rows of x^α or x^β) or to a linear term (row of x^i). If it is the former case, then either x^α or x^β will be taken as 0 and we would have taken the route of reducing our goal of degree-2 computation to linear computation as discussed earlier. For the latter case, the elimination can be achieved through recovery of the row and subtraction from the partial result. While this sounds simple to achieve, the recovery is a problematic step since a corrupt P_i may refuse to disclose the row and more worse, the dealer of this sharing might be a corrupt party that did not strongly-share in its VSS (i.e. the row of P_i does not correspond to a degree-t polynomial). We introduce another primitive called "very & open" that by the end of round 2 makes sure that P_i inputting an inconsistent row in the guided degree-2 computation and not raising an alert will be identified to be corrupt. (See Section 3.7 for "verify and open".)

There are other minor issues such as rerandomization of the underlying degree-2 polynomial which we postpone to Section 3.9 and conclude the discussion here.

Organization. Minimal technical background is given in Section 2 (other preliminaries appear in Appendix A.6). We describe our protocol in Section 3 while focusing on the setting of Theorem 1.2. The flow of Section 3 is described in Figure 1. Security proofs are deferred to the appendices.

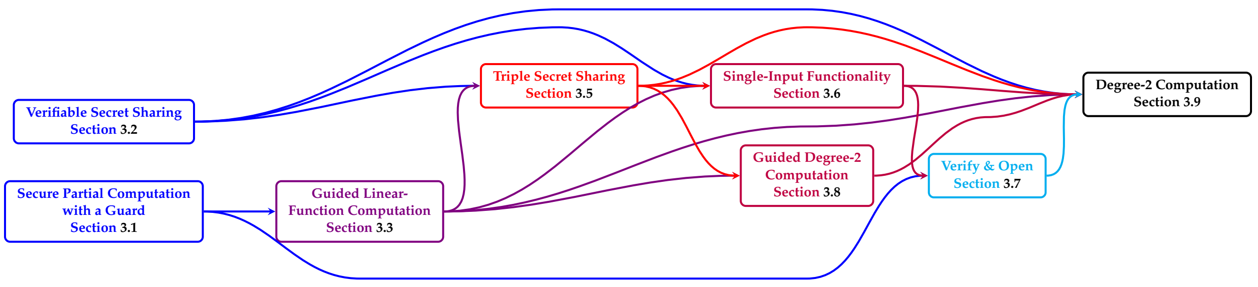
Verifiable Secret Sharing
Section 3.2
Secure Partial Computation with a Guard
Section 3.1
Guided Linear-Function Computation
Section 3.3
Triple Secret Sharing
Section 3.5
Single-Input Functionality
Section 3.6
Guided Degree-2 Computation
Section 3.8
Verify & Open
Section 3.7
Degree-2 Computation
Section 3.9

Figure 1: The protocols constructed in Section 3. A directed edge from a protocol π to a protocol ρ indictees that π is being used as a subprotocol in the protocol ρ.

2 Preliminaries

Security and Setting. We assume there are n parties denoted as $\mathsf{P} = \{P_1, \ldots, P_n\}$, at most $t \leq n/2$ of which are corrupted by a malicious, static, rushing adversary. The parties are connected by pairwise secure channels and additionally a broadcast channel is available. We prove security of our protocols in the UC-framework [20] which is recalled in Appendix A.2.

We identify the set of parties P with $\{1, \ldots, n\}$, and we denote the set of honest parties by $\mathsf{H} \subseteq \mathsf{P}$, and the set of corrupt parties by $\mathsf{C} \subseteq \mathsf{P}$. We denote by κ the security parameter.

Definitions/Notations. Let L be an integer and let $\mathbb{F}$ be a field. We will occasionally make use of a pair $(\delta, \mathbf{b}) \in \{0,1\}^L \times \mathbb{F}^L$ where the vector $\mathbf{b}$ holds "data items" (field elements) and the binary vector δ is a "meta-data" vector that holds a list of flags that indicate which entries of $\mathbf{b}$ are "known" or "certain". We further define the following non-symmetric "disagreement" operator.

Definition 2.1. *Let $A = (\delta^A, \mathbf{b}^A) \in \{0,1\}^L \times \mathbb{F}^L$ and let $B = (\delta^B, \mathbf{b}^B) \in \{0,1\}^L \times \mathbb{F}^L$. We define an operator $\diamond$ as below.*

$$(\delta^A, \mathbf{b}^A) \diamond (\delta^B, \mathbf{b}^B) := \{i \in \{1, \ldots, L\} : (\delta_i^A = 1 \wedge \delta_i^B = 0) \vee (\delta_i^A = \delta_i^B = 1 \wedge b_i^A \neq b_i^B)\}.$$

That is, the operator returns the indices that are "certain" under A but "uncertain" under B and the indices that are certain under both A and B, but A and B disagree upon the value of the corresponding data items. Notably, the operation is not symmetric.

Notation 1. *For a binary indicator vector $\delta = (\delta_1, \ldots, \delta_m)$, we freely treat δ as a subset of $\{1, \ldots, m\}$, where for $i \in \{1, \ldots, m\}$ it holds that $i \in \delta$ if and only if $\delta_i = 1$.*

NICOM. A non-interactive commitment scheme (NICOM) consists of two PPT algorithms $(\mathsf{commit}, \mathsf{open})$ defined as follows. Both algorithms are given a security parameter κ, and a common parameter pp, which is either a common reference string (CRS), or an empty string (when a CRS is not needed). The commit algorithm also takes a message x and random coins r, and outputs a commitment C and a corresponding opening information o. The open algorithm takes a commitment and a corresponding opening information (C, o), and outputs the message x. The algorithms should satisfy the standard properties of correctness, binding (i.e., it must be hard for an adversary to come up with two different openings of any c) and hiding (a commitment must not leak information about the underlying message) properties. The formal definition and its different instantiations are given in Appendix A.6. We denote the commitment scheme with respect to a crs given by $\mathcal{F}_{\mathsf{crs}}$ as $(\mathsf{commit}_{\mathsf{crs}}, \mathsf{open}_{\mathsf{crs}})$.

CRS Sampling for a NICOM. In order to obtain a protocol with everlasting security, we use *statistically-hiding* NICOMs (see Section A.6.2). It is well known that statistically-hiding NICOM cannot be implemented in the plain model, but that it can be implemented with a minimal setup assumption, such as a CRS. Since our functionalities depend on the commitment scheme, it is important that the CRS is uniformly distributed, and not chosen by the environment. Therefore, we assume a global functionality $\mathcal{F}_{\mathsf{crs}}$, to which all parties and all functionalities have access, that samples a uniformly distributed string for CRS at the beginning of the execution, and returns the CRS upon receiving a query. Thus we make sure that all parties have access to the same CRS, and that it is picked at random.

Throughout, we assume that the commitment scheme is instantiated with security parameter κ, and we assume that the hiding property holds conditioned on *any* CRS. Observe that this property indeed holds for the statistically-hiding construction presented in Section A.6.2.

Finally, we mention that all our protocols are secure even when implemented with computationally-hiding NICOM, albeit without everlasting security, assuming sub-exponentially hardness. See Remark 3.4 and Appendix A.7.3. Since, in general, computationally-hiding commitments do not require CRS, in such cases we simply assume that $\mathcal{F}_{\mathsf{crs}}$ does nothing.

3 A Three-Round MPC with Everlasting Security

In this section, we prove Theorem 1.2. That is, our goal is to build a 3-round protocol that can evaluate any n-party functionality in NC^1 with everlasting security (or any functionality with computational-security), even in the presence of $t < n/2$ corrupt parties.

Throughout we denote the security parameter by κ, and we assume that $\kappa = \Omega(n)$. We fix $\mathbb{F}$ to be a finite-field which is an $\mathbb{F}_2$-extension field, and we assume that $|\mathbb{F}| \geq 2n$ and that $\kappa \leq \log |\mathbb{F}| \leq \mathrm{poly}(\kappa)$. By using the completeness result of [5, Prop. 4.5 and Thm. 5.23] (see Proposition A.1), we may assume that the target functionality is a degree-2 functionality over $\mathbb{F}$ whose outputs are of the form $x^\alpha x^\beta + \sum_{j=1}^{n} r^j$, where x^α and x^β are the inputs of party P_α and P_β respectively and r^j is an input of party P_j for $j \in \{1, \dots, n\}$.[5]

3.1 Secure Partial Computation with a Guard

We propose a primitive called *Secure Partial Computation with a Guard (SPCG)*, building on the well-known private simultaneous message (PSM) [29]. Much like PSM's set-up, SPCG has Alice and Bob, each holding an input and trying to deliver an evaluation of a function on their inputs

[5]In fact, all the results of this section hold over an arbitrary finite field of appropriate size, $\kappa \leq \log |\mathbb{F}| \leq \mathrm{poly}(\kappa)$. We focus on fields of characteristic 2 since the completeness result (Proposition A.1) is limited to such fields.

in a *single* round to a receiver Carol who holds no input, while keeping the inputs hidden from Carol. SPCG employs an offline phase (that is independent of the inputs) and it is tailored to the case where some of the inputs of Alice are known to Bob. While PSM achieves a minimal form of privacy against Carol, SPCG provides an additional correctness property when some of the senders are malicious.

SPCG draws a similarity with a primitive having a similar name (SCG: secure computation with a guard) introduced in [4], with the following differences: (a) in SPCG, a subset of the inputs are available in committed form and the openings are available to Alice, (b) the inputs for which Alice holds the opening exclusively contributes to the computation (this explains the term 'partial' in the name of the primitive), (c) it is publicly-decodable and hence, instead of a single party Carol, all can obtain the output from a single execution and (d) finally, as opposed to property based definition, we bring in a functionality that captures the requirement of this primitive and helps in composing this primitive in higher-level protocols.

3.1.1 PSM Protocols

In a PSM protocol, proposed by [29], there are m honest parties, $P_1, \ldots, P_m$, each P_i holding a secret input x_i, and all having access to a common random string r. Each P_i sends a single message to an evaluator E depending on x_i and r. Based on these messages, the evaluator can compute $f(x_1, \ldots, x_m)$, but nothing else. PSM is formally defined as follows:

Definition 3.1 (PSM Protocols)**.** *Let $X_1, \ldots, X_m, Z$ be finite sets, and let $X = X_1 \times \ldots \times X_m$. An m-party PSM protocol* psm*, computing a m-argument function $f : X \to Z$ consists of:*

- *A message computation function $\mathsf{psm}_i : X_i \times R \to M_i$, for every party $i \in \{1, \ldots, m\}$, where R is a finite set of common random inputs and M_i is a finite message domain.*
- *A reconstruction function $\mathsf{rec} : M_1 \times \ldots \times M_m \to Z$ that will be computed by the evaluator E.*

The protocol $\mathsf{psm} = (\mathsf{psm}_1, \ldots, \mathsf{psm}_m, \mathsf{rec})$ should satisfy the following properties.

1. **(Correctness)** *For every $(x_1, \ldots, x_m) \in X$ and $r \in R$, $\mathsf{rec}(\mathsf{psm}_1(x_1, r), \ldots, \mathsf{psm}_m(x_m, r)) = f(x_1, \ldots, x_m)$.*
2. **(Security)** *There exists a simulator $\mathcal{S}_{\mathsf{psm}}$, such that for every $(x_1, \ldots, x_m) \in X$, $\mathcal{S}_{\mathsf{psm}}(f(x_1, \ldots, x_m)) \equiv \mathrm{REAL}_{\mathsf{psm}}(x_1, \ldots, x_m)$, where $\mathrm{REAL}_{\mathsf{psm}}(x_1, \ldots, x_m)$ denotes the distribution of $(\mathsf{psm}_1(x_1, r), \ldots, \mathsf{psm}_m(x_m, r))$ over the choice of r. For computational security, $\equiv$ needs to be $\equiv_c$, whereas for statistical security, $\equiv$ needs to be $\equiv_s$.*

Note that PSM security addresses only the case where the parties $P_1, \ldots, P_m$ are honest.

Lemma 3.2 (Polynomial-time PSM Protocols [43])**.** *For every m-argument functionality f that admits a Boolean NC^1 circuit of size s,there exists a PSM protocol with complexity of $\mathrm{poly}(s)$. In particular, if $s = \mathrm{poly}(m)$, then there exists a PSM protocol with complexity $\mathrm{poly}(m)$.*

3.1.2 SPCG: Functionality and Protocol

Let $A = \mathbb{F}^{L_A}$, $B = \mathbb{F}^{L_B}$ and $C = \mathbb{F}$. Let $f : A \times B \to C$ be a function of the form $f(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{L_B} \alpha_i(\mathbf{a}) b_i$, where $\mathbf{a} = (a_1, \ldots, a_{L_A})$, $\mathbf{b} = (b_1, \ldots, b_{L_B})$ and $\alpha_i : A \to \mathbb{F}$. We assume that L_A and L_B are (at most) polynomial in κ.

Consider the following situation with two distinguished parties, Alice and Bob. Alice holds the input $\mathbf{a}$, and the vector $\mathbf{b}$ is publicly committed by $(C_1, \ldots, C_{L_B})$, such that Alice holds a partial-opening $\{o_i\}_{i:\delta_i^A=1}$, specified by an indicator vector $\delta^A = (\delta_1^A, \ldots, \delta_{L_B}^A) \in \{0,1\}^{L_B}$. The goal of Alice is to let all the parties learn the output of g_{spcg}, for a set I. Looking ahead, I indicates one of the two: the positions where Alice and Bob disagree or the positions in δ^A when Bob is implicated by Alice to be corrupt.

$$g_{\mathsf{spcg}}\big(\mathbf{a}, \delta^A, \{o_i, b_i^A\}_{i=1}^{L_B}, I\big) = \Big(\mathbf{a}, \delta^A, \{i, o_i, b_i^A\}_{i\in I}, \sum_{i:\delta_i^A=1} \alpha_i(\mathbf{a}) \cdot b_i^A\Big) \tag{1}$$

On the other hand, Bob claims to know some of the b_i's, which are specified by an indicator vector δ^B. The goal of Bob is to "guard" the computation, by making sure that the partial-sum is consistent with his knowledge of the b_i's, even when Alice is corrupt.

Let I_1 be the set of indices i such that Alice *exclusively* claims to know b_i i.e. $I_1 := \Big\{i \in \{1, \ldots, L_B\} : \delta_i^A = 1 \wedge \delta_i^B = 0\Big\}$. Let I_2 be the set of indices i such that Alice and Bob do *not* agree on the value of b_i i.e. $I_2 := \Big\{i \in \{1, \ldots, L_B\} : \delta_i^A = \delta_i^B = 1 \wedge b_i^A \neq b_i^B\Big\}$. We refer I, the union of I_1, I_2, as the set of "dissenting" indices, and we allow the corresponding b_i's to be leaked. Observe that the set I is exactly the set $(\delta^A, \mathbf{b}^A)\diamond(\delta^B, \mathbf{b}^B)$, where $\diamond$ is the operator defined in Definition 2.1. Let J be the set of indices i such that Alice and Bob agree on the value of b_i (note that J may not be equal to $\{1, \ldots, m\} \setminus I$). J is referred as the set of "assenting" indices, and we would like to keep the corresponding b_i's secret. Since for every $i \in J$ it holds that $b_i^A = b_i^B$, we define $b_i := b_i^A = b_i^B$. Our goal is to let all parties learn the output of the following function (spcg denotes $\mathsf{s}$ecure $\mathsf{p}$artial $\mathsf{c}$omputation with a $\mathsf{g}$urad):

$$f_{\mathsf{spcg}}\Big(\big(\mathbf{a}, \delta^A, \{o_i, b_i^A\}_{i=1}^{L_B}\big), \big(\delta^B, \{b_i^B\}_{i=1}^{L_B}\big)\Big) = \Big(\mathbf{a}, \delta^A, \delta^B, \{i, o_i, b_i^A\}_{i\in I}, \sum_{i\in J} \alpha_i(\mathbf{a}) b_i\Big). \tag{2}$$

so that using the opening o_i^A of every "dissenting" index i, together with the sum $\sum_{i\in J} \alpha_i(\mathbf{a})\ b_i$ over all "assenting" indices, a party can recover the correct partial-sum $\sum_{i:\delta_i^A=1} \alpha_i(\mathbf{a})\ b_i^A$, as desired for g_{spcg}-output. Note that f_{spcg} ignores all (o_i, b_i^A) for which $\delta_i^A = 0$, and all b_i^B for which $\delta_i^B = 0$.

We have the following security requirements. When both Alice and Bob are honest, then all parties learn only the output of f_{spcg} (Eqn. 2) and nothing else. When at least one of Alice and Bob is corrupt, we allow the adversary to learn the honest party's inputs to f_{spcg}. If exactly one of Alice and Bob is corrupt, all honest parties should either learn the output of f_{spcg} or who the corrupt party is. When both Alice and Bob are corrupt the output can be chosen by the adversary. These are formalized in a functionality $\mathcal{F}_{\mathsf{spcg}}$ given in Fig. 2.

Functionality $\mathcal{F}_{\mathsf{spcg}}$

The functionality receives the set of corrupt parties C, and has access to $\mathcal{F}_{\mathsf{crs}}$.

Honest parties' inputs:

- All honest parties input the same commitments $(C_1, \ldots, C_{L_B})$.

- An honest Alice inputs $\mathbf{a} = (a_1, \dots, a_{L_A}) \in \mathbb{F}^{L_A}$, $\mathbf{b}^A = (b_1^A, \dots, b_{L_B}^A) \in \mathbb{F}^{L_B}$, $\delta^A = (\delta_1^A, \dots, \delta_{L_B}^A) \in \{0,1\}^{L_B}$ and openings $\{o_i\}_{i=1}^{L_B}$. It holds that $\mathsf{open}_{\mathsf{crs}}(C_i, o_i) = b_i^A$ for each $i \in \delta^A$.
- An honest Bob inputs $\mathbf{b}^B = (b_1^B, \dots, b_{L_B}^B) \in \mathbb{F}^{L_B}$ and $\delta^B = (\delta_1^B, \dots, \delta_{L_B}^B) \in \{0,1\}^{L_B}$.

Leakage: The adversary receives $(C_1, \dots, C_{L_B})$. In addition, if only Alice is corrupt, then the adversary receives honest Bob's input; if only Bob is corrupt, then the adversary receives an honest Alice's input.

Adversary's inputs: If Alice is corrupt, then it inputs $(\mathbf{a}, \mathbf{b}^A, \delta^A, \{o_i\}_{i=1}^{L_B})$ (the openings may not be correct). If Bob is corrupt, then it inputs $(\mathbf{b}^B, \delta^B)$. In addition, the adversary inputs a bit flag, a bit reveal and a string z.

Outputs: $\mathcal{F}_{\mathsf{spcg}}$ computes $I = (\delta^A, \mathbf{b}^A) \diamond (\delta^B, \mathbf{b}^B)$, and f_{spcg} and g_{spcg} using Eqn 2 and 1 respectively and acts as below.

- *(Honest Alice and Bob)* Return g_{spcg}-output to all honest parties, and f_{spcg}-output to the adversary.
- *(Honest Alice, Corrupt Bob)* If $\mathsf{flag} = 1$, then return "Bob is corrupt" to all parties. Otherwise, return g_{spcg}-output to all parties.
- *(Corrupt Alice, Honest Bob)* If $\mathsf{reveal} = 1$, then reset $I := \delta^A$. If $\mathsf{flag} = 1$, or there exists $i \in I$ such that $b_i^A \neq \mathsf{open}_{\mathsf{crs}}(C_i, o_i)$, then return "Alice is corrupt" to all. Otherwise, return g_{spcg}-output, recomputed using changed I to all.
- *(Corrupt Alice and Bob)* Return z to all parties.

Figure 2: Functionality $\mathcal{F}_{\mathsf{spcg}}$

In the following, we construct a 2-round protocol that securely implements spcg, such that the first round is an offline (input-independent) round. At a high-level, we think of f_{spcg} as a binary-function, $f_{\mathsf{spcg}} : \{0,1\}^\ell \to \{0,1\}^m$, so that the first ℓ_A bits correspond to Alice's inputs, and the last $\ell_B = \ell - \ell_A$ bits correspond to Bob's inputs. We use an ℓ-party PSM protocol for the computation of f_{spcg}, so that Alice simulates the first ℓ_A senders in the protocol, and Bob simulates the last ℓ_B senders (each sender holds a single input bit). In the offline-round, we let Bob pick the randomness r for the psm protocol and send it to Alice. In order to make sure that both Alice and Bob follow the psm protocol in the online round, we also let Bob commit all possible messages $\mathsf{psm}_i(x, r)$ for $x \in \{0,1\}$ and $i \in \{1, \dots, \ell\}$ and send the openings to Alice. In the online-round, if Alice or Bob sends a PSM message $\mathsf{psm}_i(x, r)$ on behalf of the i-th sender, then they also have to provide the corresponding commitment's opening, so that the parties can verify that a valid message was sent. Every party then acts on the opening and PSM messages to either conclude Alice/Bob to be corrupt or obtain either f_{spcg} or g_{spcg}-output.

Protocol spcg= ($\mathsf{spcg.off}$,$\mathsf{spcg.on}$)

All parties have access to $\mathcal{F}_{\mathsf{crs}}$.

Primitives: A PSM $\mathsf{psm} = (\mathsf{psm}_1, \dots, \mathsf{psm}_\ell, \mathsf{rec})$ for $f_{\mathsf{spcg}} : \{0,1\}^\ell \to \{0,1\}^m$. A NICOM scheme $(\mathsf{commit}_{\mathsf{crs}}, \mathsf{open}_{\mathsf{crs}})$.

$\mathsf{spcg.off}$**(R1):** Bob samples a random string r for psm. For each $i \in \{1, \dots, \ell\}$ and $x \in \{0,1\}$, Bob computes $(C'_{i,x}, o'_{i,x}) \leftarrow \mathsf{commit}_{\mathsf{crs}}(\mathsf{psm}_i(x, r))$ and picks a random shift σ_i of $\{0,1\}$. For each $i \in \{1, \dots, \ell\}$, Bob broadcasts the shifted list of commitments $\{C'_{i,\sigma_i(x)}\}_{x \in \{0,1\}}$. (That is, the commitment with index (i, x)

is moved to index $(i, \sigma_i(x))$.) Bob sends $\Big(r, \{o'_{i,x}\}_{i\in\{1,\ldots,\ell\},x\in\{0,1\}}, \sigma_i\Big)$ to Alice.

spcg.on**(R2):** All parties hold publicly known commitments $C_1, \ldots, C_{L_B}$. Alice holds input $\big(\mathbf{a}, \delta^A, \{o_i, b_i^A\}_{i=1}^{L_B}\big)$ and Bob holds $\big(\delta^B, \{b_i^B\}_{i=1}^{L_B}\big)$.

- **Alice's communication.** For each $x \in \{0,1\}$ and $i \in \{1,\ldots,\ell\}$, Alice verifies that $\mathsf{open}_{\mathsf{crs}}(C'_{i,\sigma_i(x)}, o'_{i,x}) = \mathsf{psm}_i(x, r)$. If the verification fails then Alice broadcasts $(\mathbf{a}, \delta^A, \{i, o_i, b_i^A\}_{i:\delta_i^A=1})$. Otherwise, Alice computes the binary string $x^A := (\mathbf{a}, \delta^A, \{o_i, b_i^A\}_{i=1}^{L_B}) \in \{0,1\}^{\ell_A}$, computes $s_i := \mathsf{psm}_i(x_i^A, r)$ for each $i \in \{1,\ldots,\ell_A\}$ and broadcasts the list $\{(\sigma_i(x_i^A), o'_{i,x_i^A}, s_i)\}_{i\in\{1,\ldots,\ell_A\}}$.
- **Bob's communication.** Bob computes the binary string $x^B := (\delta^B, \{b_i^B\}_{i=1}^{L_B}) \in \{0,1\}^{\ell_B}$, computes $s_i := \mathsf{psm}_i(x_{i-\ell_A}^B, r)$ for each $i \in \{\ell_A+1,\ldots,\ell\}$ and broadcasts $\{(\sigma_i(x_{i-\ell_A}^B), o'_{i,x_{i-\ell_A}^B}, s_i)\}_{i\in\{\ell_A+1,\ldots,\ell\}}$.
- **Local Computation.** Each party does the following.
  1. If Alice broadcasts $(\mathbf{a}, \delta^A, \{i, o_i, b_i^A\}_{i:\delta_i^A=1})$, then verify that $b_i^A \overset{?}{=} \mathsf{open}_{\mathsf{crs}}(C_i, o_i)$ for every $i \in \delta^A$, and conclude Alice to be corrupt if the verification fails. Otherwise, output $(\mathbf{a}, \delta^A, \{i, o_i, b_i^A\}_{i:\delta_i^A=1}, \sum_{i:\delta_i^A=1} \alpha_i(\mathbf{a}) \cdot b_i)$.
  2. Else, denote the messages received from Alice by $\{(i_j, o'_j, s_j)\}_{j\in\{1,\ldots,\ell_A\}}$, and the messages received from Bob by $\{(i_j, o'_j, s_j)\}_{j\in\{\ell_A+1,\ldots,\ell\}}$. If some (i_j, o'_j, s_j) with $j \in \{1,\ldots,\ell_A\}$ is not received, or $\mathsf{open}_{\mathsf{crs}}(C'_{j,i_j}, o'_j) \neq s_j$ then output "Alice is corrupt". If some (i_j, o'_j, s_j) with $j \in \{\ell_A+1,\ldots,\ell\}$ is not received, or $\mathsf{open}_{\mathsf{crs}}(C'_{j,i_j}, o'_j) \neq s_j$ then output "Bob is corrupt".
  3. Else, compute $w \leftarrow \mathsf{rec}(s_1,\ldots,s_\ell)$. Parse w to obtain $(\mathbf{a}, \delta^A, \delta^B, \{i, o_i, b_i^A\}_{i\in I}, \sum_{i\in J} \alpha_i(\mathbf{a}) \cdot b_i)$. If there exists $i \in I$ such that $\mathsf{open}_{\mathsf{crs}}(C_i, o_i) \neq b_i^A$ then output "Alice is corrupt". Otherwise, for each $i \in I$ set $b_i := b_i^A$ and output $\big(\mathbf{a}, \delta^A, \{i, o_i, b_i^A\}_{i\in I}, \sum_{i\in\delta^A} \alpha_i(\mathbf{a}) \cdot b_i\big)$.

Figure 3: Protocol spcg= (spcg.off,spcg.on)

Lemma 3.3. *Let κ be a security parameter, let n be the number of parties, let $t < n/2$, and let $\mathbb{F}$ be a field of characteristic 2. Protocol* spcg *is a UC-secure implementation of $\mathcal{F}_{\mathsf{spcg}}$ with everlasting security, against a static, active, rushing adversary corrupting up to t parties. The complexity of the protocol is* $\mathrm{poly}(n, \log|\mathbb{F}|, \kappa, L_A, L_B)$, *assuming each $\alpha_i(\cdot)$ can be represented as a Boolean function in NC^1.*

We prove Lemma 3.3 in Appendix A.7.

Remark 3.4 (Computationally-hiding commitments). *If we are willing to relax the everlasting statistical security to computational security, one can employ* computationally-hiding, *perfectly-binding commitments. However, the proof requires some level of security under selective-opening attack (loosely speaking, when the adversary receives commitments $C_1, \ldots, C_k$ and chooses to receive the openings of a subset of those commitments, the unopened commitments should remain secure). To achieve this property, we assume that the underlying commitments achieve sub-exponential level of hiding (see Section A.6.2 for details). By using "complexity leveraging" arguments, we show that protocol* spcg *is a UC-secure implementation of $\mathcal{F}_{\mathsf{spcg}}$ against a static, active, rushing computationally-bounded adversary corrupting up to t parties. See Appendix A.7.3 for full details.*

3.2 Verifiable Secret Sharing

We begin by defining two closely related sharing semantics. Next, we use our VSS protocol as a means to generate such sharings of a dealer's secret. When the dealer is honest, then the first kind, denoted as *strong double t-sharing*, will be generated, whereas a corrupt dealer can only be enforced to generate the second kind, referred as *weak double t-sharing*. The strong sharing essentially means committing to $F(i, j)$ for $0 \le i, j \le n$, for some symmetric bivariate polynomial $F(x, y)$ of degree t in each variable. The commitments and openings are expected to be symmetric, i.e., $C_{ij} = C_{ji}$ and $o_{ij} = o_{ji}$. Fix some common reference string crs, and consider a pair $(\mathbf{C}, \mathbf{O})$ of commitments $\mathbf{C} = \{C_{ij}\}_{i,j\in\{0,\ldots,n\}}$ and openings $\mathbf{O} = \{o_{ij}\}_{i,j\in\{0,\ldots,n\}}$. For $i \in \{0, \ldots, n\}$, let $\mathbf{C}_i = \{C_{ij}\}_{j\in\{0,\ldots,n\}}$ and $\mathbf{O}_i = \{o_{ij}\}_{j\in\{0,\ldots,n\}}$.

Definition 3.5 (validity). *We say that $(\mathbf{C}, \mathbf{O}_i)$ is* valid *(with respect to crs) if the following conditions hold:*

1. *(Consistent commitments) $C_{jk} = C_{jk}$ for all $j, k \in \{0, \ldots, n\}$.*
2. *(Valid openning) For all $j \in \{0, \ldots, n\}$ the value $f_{ij} := \mathsf{open}_{\mathsf{crs}}(C_{ij}, o_{ij})$ is not $\perp$.*
3. *(Low degree) The polynomial $f_i(x)$, obtained by interpolating $\{f_{ij}\}_{j\in\{0,\ldots,n\}}$, is of degree at most t.*

Definition 3.6 (Strong double t-sharing aka $\langle\!\langle\cdot\rangle\!\rangle$-sharing). *A pair $(\mathbf{C}, \mathbf{O})$ is a* strong double t-sharing *of s (with respect to crs), denoted as $\langle\!\langle s\rangle\!\rangle$, if the following conditions hold:*

1. *(Validity) $(\mathbf{C}, \mathbf{O}_i)$ is valid for every $i \in \{0, \ldots, n\}$.*
2. *(Consistent openning) $o_{ij} = o_{ji}$ for $i, j \in \{0, \ldots, n\}$.*
3. *(Sharing of s) The values $\{f_{i0} := \mathsf{open}_{\mathsf{crs}}(C_{i0}, o_{i0})\}_{i\in\{1,\ldots,n\}}$ correspond to a degree t polynomial $f(x)$ such that $f(i) = f_{i0}$ and $f(0) = s$.*

Overall, when $(\mathbf{C}, \mathbf{O})$ is a strong double t-sharing of s, it holds (by conditions 1 and 2) that each $f_i(x)$ is of degree at most t and that $f_i(j) = f_j(i)$ for all $i, j \in \{0, \ldots, n\}$. It therefore follows (see Fact A.4) that the values $\{f_{ij}\}_{i,j\in\{0,\ldots,n\}}$ correspond to a unique symmetric bivariate polynomial $F(x, y)$ of degree at most t in each variable, such that $F(i, j) = f_{ij}$ for all $i, j \in \{0, \ldots, n\}$.

Next, the weak sharing ensures that the shares of the honest parties are consistent with some symmetric bivariate polynomial $F(x, y)$ of degree t in each variable. However, a share of a corrupt P_i might not be consistent with $F(x, y)$. The following definition is tailored to the case where the decommitment information that "belongs" to a subset of the parties, W, have been published. (The set W may consist both honest and corrupted parties.)

Definition 3.7 (Weak double t-sharing aka $[\![\cdot]\!]$-sharing). *A tuple $(\mathsf{W}, \mathbf{C}, \mathbf{O}_{\mathsf{W}}, \mathbf{O}_{\mathsf{H}\setminus\mathsf{W}})$ of parties $\mathsf{W} \subseteq \{1, \ldots, n\}$, public commitments $\mathbf{C} = \{C_{ij}\}_{i,j\in\{0,\ldots,n\}}$, public openings $\mathbf{O}_{\mathsf{W}} = \{o_{ij}\}_{i\in\mathsf{W},j\in\{0,\ldots,n\}}$, and private openings $\mathbf{O}_{\mathsf{H}\setminus\mathsf{W}} = \{o_{ij}\}_{i\in\mathsf{H}\setminus\mathsf{W},j\in\{0,\ldots,n\}}$ for the set of honest parties H is a* weak double t-sharing *of s (with respect to crs), denoted as $[\![s]\!]$, if the following conditions holds:*

1. *(Partial validity) For every $i \in \mathsf{W} \cup \mathsf{H}$ it holds that $(\mathbf{C}, \mathbf{O}_i)$ is valid.*
2. *(Weakly consistent openning) $\mathsf{open}_{\mathsf{crs}}(C_{ij}, o_{ij}) = \mathsf{open}_{\mathsf{crs}}(C_{ji}, o_{ji})$[6] for $i, j \in \mathsf{W} \cup \mathsf{H}$*
3. *(Weak sharing of s) The values $\{f_{i0} := \mathsf{open}_{\mathsf{crs}}(C_{i0}, o_{i0})\}_{i\in\mathsf{W}\cup\mathsf{H}}$ correspond to a degree t polynomial $f(x)$ such that $f(i) = f_{i0}$ and $f(0) = s$.*

[6]Notice that, we allow $o_{ij} \ne o_{ji}$, and yet require $\mathsf{open}_{\mathsf{crs}}(C_{ij}, o_{ij}) = \mathsf{open}_{\mathsf{crs}}(C_{ji}, o_{ji})$ to hold.

Consider a weak double t-sharing $(\mathsf{W}, \mathbf{C}, \mathbf{O}_{\mathsf{W}}, \mathbf{O}_{\mathsf{H}\setminus\mathsf{W}})$, and let $f_i(x)$ be the polynomial defined by $\{f_{ij} := \mathsf{open}_{\mathsf{crs}}(C_{ij}, o_{ij})\}$, for $i \in \mathsf{W} \cup \mathsf{H}$. By condition (1) it follows that $f_i(x)$ is a degree-t polynomial, and by condition (2) it holds that $f_i(j) = f_j(i)$ for all $i, j \in \mathsf{W} \cup \mathsf{H}$. Therefore, (see Fact A.4) the polynomials $\{f_i(x)\}_{i \in \mathsf{W} \cup \mathsf{H}}$ define a unique symmetric bivariate polynomial $F(x, y)$ of degree at most t in each variable, such that $F(x, i) = f_i(x)$ for all $i \in \mathsf{W} \cup \mathsf{H}$. (Note that $|\mathsf{W} \cup \mathsf{H}| \geq t + 1$ since we always have honest majority, i.e., $t < n/2$.) We conclude that any tuple $(\mathsf{W}, \mathbf{C}, \mathbf{O}_{\mathsf{W}}, \mathbf{O}_{\mathsf{H}\setminus\mathsf{W}})$ that satisfies Conditions (1) and (2), is a weak double t-sharing of a value $s := F(0, 0)$, where $F(x, y)$ is the corresponding sharing polynomial.

For both sharing, we refer to $F(x, y)$ as the *sharing polynomial* and $f_i(x) = F(x, i) = F(i, y)$ as the ith *row polynomial*.

Definition 3.8 (Rows of Sharing). *Let* $(\mathbf{C}, \mathbf{O})$ *and* $(\mathsf{W}, \mathbf{C}, \mathbf{O}_{\mathsf{W}}, \mathbf{O}_{\mathsf{H}\setminus\mathsf{W}})$ *denote a* $\langle\langle s \rangle\rangle$ *and respectively* $[\![s]\!]$. *We refer* $(\mathbf{C}_i, \mathbf{O}_i)$ *as the* ith row *and denote by* $\langle\langle s \rangle\rangle_i$ *and* $[\![s]\!]_i$ *for respective sharing. We refer* $(\mathbf{C}_0, \mathbf{O}_0)$ *as the* main row *of the sharings and denote by* $\langle s \rangle$ *and* $[s]$ *for respective sharing.*

3.2.1 VSS: Functionality and Protocol

We start with the VSS functionality, $\mathcal{F}_{\mathsf{vss}}$ (Fig. 4), which has two phases: sharing and verification. In the sharing phase, an honest dealer will always input a strong sharing of a secret, while for a corrupt dealer, the functionality guarantees output of a weak sharing of some secret at the end of the verification phase (or a disqualification of D). The functionality concludes the sharing phase after sending the shares meant for different parties. In the verification phase, we let each party input an indicator bit. When the VSS is used as a primitive in a bigger protocol, the indicator bit reflects whether a party has identified the dealer as corrupt outside the VSS. The indicator bit signifies that the share of the party who sets it as 1 must be publicly disclosed by the functionality. An honest party's indicator bit is leaked to the adversary, before it commits to the bits of the corrupt parties. In addition, since there is no guarantee for a corrupt dealer to submit even a weak sharing in the sharing phase, in the verification phase $\mathcal{F}_{\mathsf{vss}}$ also reveals the shares of any honest party that got an invalid share in the sharing phase. Finally, while we allow the adversary to disclose the *openings* late during the verification phase on behalf of a corrupt dealer, the *commitments* remain unchanged from the sharing phase.

Functionality $\mathcal{F}_{\mathsf{vss}}$

The functionality $\mathcal{F}_{\mathsf{vss}}$ receives the set of corrupt parties C, and has access to $\mathcal{F}_{\mathsf{crs}}$.

Sharing phase.

- **Inputs:** $\mathcal{F}_{\mathsf{vss}}$ receives from D a pair $(\mathbf{C}, \mathbf{O})$ of commitments $\mathbf{C} = \{C_{ij}\}_{i,j \in \{0,\ldots,n\}}$ and openings $\mathbf{O} = \{o_{ij}\}_{i,j \in \{0,\ldots,n\}}$. If D is honest then $(\mathbf{C}, \mathbf{O})$ is a strong double t-sharing of some value s.
- **Outputs:** For $i \in \{1, \ldots, n\}$, $\mathcal{F}_{\mathsf{vss}}$ returns $(\mathbf{C}, \{o_{ij}\}_{j \in \{0,\ldots,n\}})$ to P_i.

Verification phase.

- **Honest parties' inputs:** Each honest party P_i inputs a bit flag_i, such that if D is honest then $\mathsf{flag}_i = 0$ for every honest P_i.

- **Leakage:** For any honest P_i, the bit flag_i is leaked to the adversary.
- **Adversary's inputs:** Each corrupt P_i inputs a bit flag_i. If D is corrupt, then D has two additional inputs, $\bar{\mathbf{O}} := \{\bar{o}_{ij}\}_{i,j\in\{0,\dots,n\}}$ and a bit flag_D.
- **Outputs:** We split into two cases.
  - **Honest** D**.** Let W be the set of all corrupt parties P_i with $\mathsf{flag}_i = 1$. $\mathcal{F}_{\mathsf{vss}}$ returns $\big(\mathsf{W}, \{o_{ij}\}_{i\in\mathsf{W}, j\in\{0,\dots,n\}}\big)$ to all parties.
  - **Corrupt** D**.** Let W be the set containing all parties P_i with $\mathsf{flag}_i = 1$, together with all honest parties P_i with an invalid pair $(\mathbf{C}, \mathbf{O}_i)$, where $\mathbf{O}_i := \{o_{ij}\}_{j\in\{0,\dots,n\}}$. Let $\bar{\mathbf{O}}_{\mathsf{W}} := \{\bar{o}_{ij}\}_{i\in\mathsf{W}, j\in\{0,\dots,n\}}$. If the tuple $(\mathsf{W}, \mathbf{C}, \bar{\mathbf{O}}_{\mathsf{W}}, \mathbf{O}_{\mathsf{H}\setminus\mathsf{W}})$ is a $[\![s]\!]$ for some s, and $\mathsf{flag}_D = 0$ then $\mathcal{F}_{\mathsf{vss}}$ returns $\big(\mathsf{W}, \bar{\mathbf{O}}_{\mathsf{W}}\big)$ to all parties. Otherwise, $\mathcal{F}_{\mathsf{vss}}$ returns "D is corrupt" to all parties.

Figure 4: Functionality $\mathcal{F}_{\mathsf{vss}}$

We now present a 2-round protocol to realize the functionality $\mathcal{F}_{\mathsf{vss}}$, borrowing the techniques of the 2-round VSS construct of [8]. Similar to the functionality, the protocol is structured in two phases: sharing and verification. D prepares a strong double t-sharing of its secret using a (random) sharing polynomial $F(x, y)$, broadcasts the commitments and delivers the opening corresponding to ith row polynomial $f_i(x) = F(x, i)$ to P_i. This concludes the sharing phase. To ensure that a corrupt dealer either chooses to get disqualified or lends off a weak double t-sharing at the very least, every party gets *unhappy* with the dealer if the public commitments and opening for the ith row polynomial is invalid as per Definition 3.5 (or additionally if its indicator bit flag_i is 1). With the goal to make the opening for the ith row polynomial public for every unhappy party in round 2, every P_i commits to $n+1$ masks (corresponding to openings of $n+1$ points on ith row polynomial) in round 1 and discloses the corresponding openings to the dealer alone. If D finds inconsistency between the these commitments and openings, it publicly discloses the openings for P_i's row polynomial $f_i(x)$. Otherwise, it broadcasts masked openings for $f_i(x)$ blinded with the masks. Now when a P_i is unhappy with D it can open the masks and thereby make the opening of the $n+1$ points on $f_i(x)$ public. If the public commitments for $F(x, y)$ and these openings are invalid as per Definition 3.5, then D gets discarded. Otherwise, it is clear that a weak double t-sharing is established in the end. We present the protocol vss below and prove its security as stated below in Appendix B.1.

Protocol vsh

All parties have access to $\mathcal{F}_{\mathsf{crs}}$.

Primitives: A NICOM scheme $(\mathsf{commit}_{\mathsf{crs}}, \mathsf{open}_{\mathsf{crs}})$.

Sharing Phase (R1):

- *Inputs.* D holds a pair $(\mathbf{C}, \mathbf{O})$, which is a strong double t-sharing.
- D broadcasts $\mathbf{C}$. In addition, for every $i \in \{1, \dots, n\}$, D sends $\mathbf{O}_i$ to P_i.
- Each party P_i picks $n+1$ random values $(g_{i0}, \dots, g_{in})$, computes $\big(G_{ij}, h_{ij}\big) = \mathsf{commit}_{\mathsf{crs}}(g_{ij})$ for $j \in \{0, \dots, n\}$, sends $\{(g_{ij}, h_{ij})\}_{j\in\{0,\dots,n\}}$ to D, and broadcasts $\mathbf{G}_i := \{G_{ij}\}_{j\in\{0,\dots,n\}}$.
- *Output.* Each P_i outputs $\mathbf{C}$ and $\{o_{ij}\}_{j\in\{0,\dots,n\}}$.

Verification Phase (R2):

- *Inputs.* Each P_i holds an input-bit flag_i.
- D does the following for every party P_i:
  - Becomes *unhappy* with P_i if the check $g_{ij} \overset{?}{=} \mathsf{open}_{\mathsf{crs}}(G_{ij}, h_{ij})$ fails for some $j \in \{0, \ldots, n\}$.
  - Broadcasts $\{o_{ij}\}_{j \in \{0,\ldots,n\}}$ when *unhappy*, and $\{\alpha_{ij}\}_{j \in \{0,\ldots,n\}}$ otherwise, where $\alpha_{ij} := o_{ij} + g_{ij}$.
- A party P_i is *unhappy* with D if the pair $(\mathbf{C}, \mathbf{O}_i)$ is invalid, or if $\mathsf{flag}_i = 1$. P_i broadcasts $\{(g_{ij}, h_{ij})\}_{j \in \{0,\ldots,n\}}$ when *unhappy* and no message otherwise.
- *(Local Computation)* The pair (D, P_i) is said to be in *conflict* if P_i broadcasts $\{(g_{ij}, h_{ij})\}_{j \in \{0,\ldots,n\}}$ such that $g_{ij} = \mathsf{open}_{\mathsf{crs}}(G_{ij}, h_{ij})$ for all $j \in \{0, \ldots, n\}$. Let W be the set of parties conflicted with D. For every $i \in \mathsf{W}$ and $j \in \{0, \ldots, n\}$, if D broadcasted o_{ij} in the verification phase, set $\bar{o}_{ij} := o_{ij}$, and otherwise set $\bar{o}_{ij} := \alpha_{ij} - g_{ij}$. Let $\bar{\mathbf{O}}_{\mathsf{W}} := \{\bar{o}_{ij}\}_{i \in \mathsf{W}, j \in \{0,\ldots,n\}}$.
- **Output.** If there exists $i \in \mathsf{W}$ such that the pair $(\mathbf{C}, \bar{\mathbf{O}}_i)$ is invalid then the parties output "D is corrupt". Otherwise, every party outputs $(\mathsf{W}, \bar{\mathbf{O}}_{\mathsf{W}})$.

Figure 5: Protocol vsh

Notation 2. *We say that the dealer* shares *a value s via* vss *if (1) the dealer picks a random symmetric bivariate polynomial $F(x, y)$ of degree at most t in each variable, such that $F(0, 0) = s$, (2) samples $(C_{ij}, o_{ij}) \leftarrow \mathsf{commit}_{\mathsf{crs}}(F(i, j); r_{ij})$ for every $i, j \in \{0, \ldots, n\}$ such that $i \leq j$, where r_{ij} is a fresh random string, (3) sets $C_{ji} := C_{ij}$ and $o_{ji} := o_{ij}$ for every $i < j$, and (4) initiates* vss *with $\mathbf{C} := \{C_{ij}\}_{i,j \in \{0,\ldots,n\}}$ and $\mathbf{O} := \{o_{ij}\}_{i,j \in \{0,\ldots,n\}}$.*

Notation 3 (Tentative Sharing aka $\llbracket \cdot \rrbracket$-sharing)**.** *We refer the sharing of s at the end of the sharing phase by* tentative sharing *and denote it as $\llbracket s \rrbracket$. The ith and the main row of a $\llbracket s \rrbracket$ is denoted as $\llbracket s \rrbracket_i$ and $\lfloor s \rceil$ respectively.*

Theorem 3.9. *Let κ be a security parameter, n be the number of parties with $t < n/2$, and $\mathbb{F}$ be a field. Protocol* vss *is a UC-secure implementation of $\mathcal{F}_{\mathsf{vss}}$ with everlasting security, against a static, active, rushing adversary corrupting up to t parties. The complexity of the protocol is* $\mathrm{poly}(n, \log |\mathbb{F}|, \kappa)$.

3.3 Guided and General Linear Function Computation

A common situation that often occurs in our later constructions is as follows. There are m secrets $s_1, \ldots, s_m$ that are shared by either different parties or the same party (we always assume that m is polynomial in κ). The goal is to reconstruct a linear combination of the m secrets: $\sum_{i=1}^{m} a_i s_i$ for a publicly known linear combiner. We cannot use the reconstruction of our VSS scheme[7] right-away, without leaking the individual secrets, since our commitments are non-homomorphic and do not allow for non-interactive generation of commitments of the sum shares. The problem gets further compounded with our need (e.g. in triple sharing) of packing the sharing and reconstruction in 2 rounds for an overall 2-round construct. This requires us to come up with a reconstruction that can operate, not just over, final shares, but also over the tentative shares, which are available after the first round (the sharing phase) of VSS protocols.

Here we propose guided linear function evaluation protocol, that would enable the kth party P_k, denoted as the guide G, to reconstruct the kth linearly combined rows of the secrets

[7]While we do not recall the reconstruction phase protocol of the VSS in this paper, it is presented in [8].

i.e. $\sum_{i=1}^{m} a_i [\![s_i]\!]_k$. Our protocol requires 2 rounds in total, in which the first round is input-independent, and can be run in parallel to the tentative sharing generation (i.e. along with the sharing phase of VSS instances sharing s_i's), the rows of which acts as its input. This allows us to pack both sharing and reconstruction in an overall 2 round protocol, as needed for our future constructs. Operating over tentative sharing, however, brings additional complication since some of the kth rows may not be valid i.e. the received openings may not be correct or the underlying polynomial may not correspond to degree t (c.f. Definition 3.5). Therefore, we need to adapt our goal to finding a partially linearly combined sum, where only 'valid' rows are taken into account. This needs the guide P_k to use a m-length binary vector indicating the 'valid'/'invalid' rows. Since the remaining rows will be published during the verification of the respective VSS instances, the partial sum can be turned to the full sum outside the guided reconstruction (otherwise the guide will be identified as corrupt). In a dual scenario, it is also possible that a valid row turns into an invalid one (since VSS dealer is identified as corrupt), for which we need to subtract the concerned row from the sum. Later in the paper, we come up with a verify and open primitive to enable opening such rows. Looking ahead, for the reconstruction of the linearly combination of m secrets, we need to run n guided reconstructions, one for every P_k as guide.

With the above background, we design an ideal functionality $\mathcal{F}_{\mathsf{glinear}}$ (Fig. 6) for the guided reconstruction which gets the following inputs and attains the following task. All the parties input the commitments to m tentative rows (which may be available from the m VSS sharing phases). A guide G enters a linear combiner $\mathbf{a}$, the m tentative rows which need to be linearly combined, the openings for them and lastly an indicator vector suggesting 'valid' tentative rows. To ensure that a guide remains as a guide and do not plant any polynomial to be reconstructed, every party P_i provides the tentative shares of the concerned rows and its indicator vector suggesting which values to be used for verification of G's information. For both G and the individual parties, the indicator vectors reflect the respective parties' impression on whether a particular row/share is valid or invalid. All the indicator positions where G and P_i conflict and G indicates a valid row, are disclosed. This allows an honest P_i to make sure that the combined tentative row agrees with its combined value at point i (where the combination takes into account all valid rows as per G). Having an honest majority implies a guide cannot make any polynomial to get reconstructed.

Functionality $\mathcal{F}_{\mathsf{glinear}}$

The functionality receives the set of corrupt parties C, and has access to $\mathcal{F}_{\mathsf{crs}}$.

Honest parties' inputs:

- All honest parties input the same commitments $(C_{ij})_{i\in\{1,\dots,m\},j\in\{0,\dots,n\}}$.
- If G is honest, G inputs (1) a list of coefficients $\mathbf{a} = (a_1,\dots,a_m) \in \mathbb{F}^m$, (2) a list of values $\mathbf{b}^G = (b_{ij}^G)_{i\in\{1,\dots,m\},j\in\{0,\dots,n\}} \in \mathbb{F}^{m(n+1)}$, (3) a list of openings $\{o_{ij}^G\}_{i\in\{1,\dots,m\},j\in\{0,\dots,n\}}$, and (4) an indicator vector $\delta^G \in \{0,1\}^m$.
  For every $i \in \delta^G$ it holds that $\mathsf{open}_{\mathsf{crs}}(C_{ij}, o_{ij}^G) = b_{ij}^G$ for every $j \in \{0,\dots,n\}$, and the values $\{b_{ij}^G\}_{j\in\{0,\dots,n\}}$ correspond to a degree-t polynomial.
- An honest P_i inputs an indicator vector $\delta^i \in \{0,1\}^m$, and values $(b_1^i,\dots,b_m^i) \in \mathbb{F}^m$.

Leakage: The adversary receives $(C_{ij})_{i\in\{1,\dots,m\},j\in\{0,\dots,n\}}$ and the indicator vectors δ^i of all honest P_i. In addition,

- if G is honest, the adversary receives $\mathbf{a}, \{b^G_{ij}, o^G_{ij}\}_{i\in\{1,\dots,m\},j\in\mathsf{C}}$ and δ^G, as well as $L_j := \{(i,j), b^G_{ij}, o^G_{ij}\}_{i\in I_j}$ for any $j \in \mathsf{H}$, where $I_j := ((b^G_{1j},\dots,b^G_{mj}),\delta^G)\diamond(\mathbf{b}^j,\delta^j)$, and the sum $\sum_{i\in\delta^A} a_i b^A_{ij}$, for any $j \in \{1,\dots,n\}$.
- if G is corrupt the adversary also receives $(b^i_1,\dots,b^i_m)$ for every $i \in \mathsf{H}$.

Adversary's inputs:

- A corrupt guide inputs $\mathbf{a}$, $\mathbf{b}^G$, $\{o^G_{ij}\}_{i\in\{1,\dots,m\},j\in\{0,\dots,n\}}$ and δ^G, and an additional bit flag.

Outputs: We split into cases.

- **Honest guide.** The functionality returns $(\mathbf{a}, \delta^G, \sum_{i:\delta^G_i=1} a_i \cdot b^G_{i,0})$ to all parties.
- **Corrupt guide.** The functionality returns "G is corrupt" if either (1) flag $= 1$, or (2) there exists $j \in \mathsf{H}$ and $i \in I_j$ such that $\mathsf{open}_{\mathsf{crs}}(C_{ij}, o_{ij}) \neq b^G_{ij}$, where $I_j := ((b^G_{1j},\dots,b^G_{mj}),\delta^G)\diamond(\mathbf{b}^i,\delta^i)$. Otherwise, for each $j \in \mathsf{H}$ let $v^j := \sum_{i\in\delta^G} a_i \cdot b^G_{ij}$, and let $g(x)$ be the polynomial obtained by interpolating $\{v^j\}_{j\in\mathsf{H}}$. If the degree of $g(x)$ is more than t then the functionality returns "G is corrupt". Otherwise, the functionality returns $(\mathbf{a}, \delta^G, g(0))$ to all parties.

Figure 6: Functionality $\mathcal{F}_{\mathsf{glinear}}$

We now present a 2-round protocol realizing $\mathcal{F}_{\mathsf{glinear}}$. The protocol simply invokes n instances of SPCG with G as Alice and every P_i as Bob to compute the following function, with their respective inputs. Let $\mathbf{a}, \mathbf{b} \in \mathbb{F}^m$, and let $f : \mathbb{F}^m \times \mathbb{F}^m \to \mathbb{F}$ be

$$f(\mathbf{a},\mathbf{b}) := \sum_{i=1}^{m} \alpha_i(\mathbf{a}) \cdot b_i, \text{ where } \alpha_i(\mathbf{a}) = a_i. \tag{3}$$

If (a) G (aka Alice) gets identified as corrupt in any of the n SPCG instances or (b) it inputs different $\mathbf{a}$, indicator vector or (c) the outputs of SPCGs corresponding to instances where Bob is not identified as corrupt lead to a non-degree-t polynomial, then G is identified as corrupt. Otherwise, the SPCGs which speak out an output are used to form a degree-t polynomial, which is taken as the output, the partially combined row taking into account all rows as per the indicator vector of G. The protocol appears in Fig. 7, its security in Lemma 3.10 and the proof in Appendix B.2. In order to simplify the calls to glinear in our later protocols, we use Notation 4.

Protocol $\mathsf{glinear} = (\mathsf{glinear.off}, \mathsf{glinear.on})$

All parties have access to $\mathcal{F}_{\mathsf{crs}}$.

Primitives: SPCG scheme $\mathsf{spcg} = (\mathsf{spcg.off}, \mathsf{spcg.on})$.

$\mathsf{glinear.off}$**(R1):** For every $j \in \{1,\dots,n\}$, the parties execute an instance of the offline phase of spcg, denoted $\mathsf{spcg.off}^j$, with G as Alice, P_j as Bob, and with function f as in Equation (3).

Inputs: The inputs are exactly as specified in the description of $\mathcal{F}_{\mathsf{glinear}}$ under **Honest parties' Input**.

$\mathsf{glinear.on}$**(R2):** For each $j \in \{1,\dots,n\}$, the protocol $\mathsf{spcg.on}^j$ is executed, where

- All parties input $(C_{1j}, \ldots, C_{mj})$.
- G, as Alice, inputs the coefficient vector $\mathbf{a}$, the vector of values $(b_{1j}^G, \ldots, b_{mj}^G)$, the indicator vector δ^G and the openings $\{o_{ij}^G\}_{i=1}^m$;
- P_j, as Bob, inputs the indicator vector δ^i and the values $(b_1^i, \ldots, b_m^i)$.

(Local computation) For each $k \in \{1, \ldots, n\}$ let out^k be the output of spcg^k. If there exists k such that out^k is "Alice is corrupt", then output "G is corrupt". Otherwise, let K be the set of all indices k such that out^k is not "Bob is corrupt", and let $\mathsf{out}^j = (\mathbf{a}^k, \delta^{A,k}, \{i, o_i^{A,k}, b_i^{A,k}\}_{i \in I_k}, v^k)$ for each $k \in K$. If the the list of coefficients $\mathbf{a}^k$ or the indicator vector $\delta^{A,k}$ is not the same among all $\{\mathsf{out}_k\}_{k \in K}$ then output "G is corrupt". Otherwise set $\mathbf{a} := \mathbf{a}^k$ and $\delta^G := \delta^{A,k}$ for some $k \in K$. Let $g(x)$ be the polynomial obtained by interpolating over $\{v^k\}_{k \in K}$. If $g(x)$ is of degree more than t then output "G is corrupt". Otherwise, output $(\mathbf{a}, \delta^G, g(0))$.

Figure 7: Protocol $\mathsf{glinear} = (\mathsf{glinear.off}, \mathsf{glinear.on})$

Lemma 3.10. *Let κ be a security parameter, let n be the number of parties, let $t < n/2$, and let $\mathbb{F}$ be a field of characteristic 2. Protocol* $\mathsf{glinear}$ *is a UC-secure implementation of $\mathcal{F}_{\mathsf{glinear}}$ with everlasting security, against a static, active, rushing adversary corrupting up to t parties. The complexity of the protocol is* $\mathrm{poly}(n, \log|\mathbb{F}|, \kappa, m)$.

Notation 4. *For a set of ℓ tentative rows $\llbracket s_1 \rrbracket_j, \ldots, \llbracket s_\ell \rrbracket_j$, we say that the parties participate in* $\mathsf{glinear}$ *to partially compute $\sum_{k=1}^{\ell} a_k \llbracket s_k \rrbracket_j$, with party P_j as a guide G with input δ^G and every P_i with input δ^i to mean an execution of* $\mathsf{glinear}$ *with the above involved parties and input (bit) vectors. For brevity, we will omit the additional inputs from the parties, (a) $\mathbf{a} = (a_1, \ldots, a_\ell)$, which can be deduced from the sum expression, (b) the openings and opened values corresponding to all the rows (a total of $\ell(n+1)$ openings/opened values) for G and ith opened value corresponding to all the rows (a total of ℓ opened values) for every P_i, both of which can be deduced from the sum expression.*

Remark 3.11 (Round-optimal general linear function computation.)**.** *We mention that we can obtain a 3-round protocol for general linear function computation assuming only the existence of commitments, reducing to n instances of* $\mathsf{glinear}$ *operated on final sharings (and not on tentative sharing). The protocol is round-optimal due to the lower bound of [33]. For simplicity, we focus on the case that each P_i holds an input s_i and the parties want to compute $s_1 + \ldots + s_n$. The general case follows in a similar way, using some standard padding techniques.*

The protocol is as follows. In rounds 1 and 2 each P_i shares s_i via an instance of vss*. In round 2 the parties also execute an instance of* $\mathsf{glinear.off}^i$ *with P_i as a guide, for all $i \in \{1, \ldots, n\}$. The indicator vector for any party is computed as follows: the jth bit of the indicator vector δ_j is 1 if and only if P_j was not disqualified in its* vss *instance. The input of those who are disqualified are taken as 0. In round 3, for any $i \in \{1, \ldots, n\}$ the parties execute* $\mathsf{glinear.on}^i$*, computing $\llbracket s_1 \rrbracket_i + \ldots + \llbracket s_n \rrbracket_i$ with P_i as a guide, over the* final rows *taken from the* vss *instances. For every $i \in \{1, \ldots, n\}$ we say that the output of* $\mathsf{glinear.on}^i$ *is valid if it is of the form $(\mathbf{a}, \delta^G, \sigma_i)$, where $\mathbf{a}$ is the all-one vector and δ^G is as specified earlier. The parties interpolate over all valid σ_i to obtain a degree-t polynomial $f(x)$, and output $f(0)$. We omit the proof of security, which can be completed easily in the $(\mathcal{F}_{\mathsf{vss}}, \mathcal{F}_{\mathsf{glinear}})$-hybrid model.*

3.4 $\mathbb{R}$-intractable Hash Functions

Following [21], we present the following variant of correlation-intractable hash functions.

Definition 3.12 ($\mathbb{R}$-intractable Hash Functions)**.** *Let $H = \{H_\kappa\}$ be a collection of hash functions where $H_\kappa = \{h_z\}_{z \in \{0,1\}^\kappa}$ is a function from some finite domain D_κ to a finite range R_κ.*

Let $\mathbb{R} = \{\mathbb{R}_\kappa\}$ be a (uniform) sequence of oracle-aided polynomial-size circuits (either Boolean or arithmetic) where $\mathbb{R}_\kappa$ makes use of oracle gates that are compatible with H_κ, i.e., they map inputs in D_κ to outputs in R_κ.

We say that H is $\mathbb{R}$-intractable if the following hold:

1. (Efficient Evaluation and sampling) *There exists a pair of efficient algorithms (a) an evaluation algorithm which given $z \in \{0,1\}^\kappa, x \in D_\kappa$ outputs $h_z(x)$; and (b) a key-sampling algorithm K which given 1^κ samples an index $z \in \{0,1\}^\kappa$.*

2. ($\mathbb{R}$-intractability) *For every PPT adversary $\mathcal{A}$ it holds that $\Pr_{z \leftarrow_R K(1^\kappa)} \Big(\mathcal{A}(z) = x \text{ s. t. } \mathbb{R}_\kappa^{h_z}(x) = 0 \Big) \leq \mathsf{negl}(\kappa)$, where $\mathbb{R}_\kappa^{h_z}$ corresponds to the circuit that is obtained by instantiating the oracle gates in $\mathbb{R}_\kappa$ by the hash function h_z.*

Unlike previous formulations who consider relations over tuples of the form $(x_1, \ldots, x_t, h(x_1), \ldots, h(x_t))$ (e.g., [46]), we allow the relation $\mathbb{R}$ to make oracle calls to the hash function. We find this convention useful, although one can always transform the latter case to the former, more standard, case. We remind again that we will be interested in a single explicit and relatively-simple relation, $\mathbb{R}$, that will be specified in Section 3.5 (see also Section A.3).

3.5 Triple Secret Sharing

The goal of triple sharing is to make a dealer D generate $[\![a]\!], [\![b]\!], [\![c]\!]$ such that $c = ab$ in 2 rounds. The sharing are strong, when D is honest. The functionality $\mathcal{F}_{\mathsf{tss}}$ for this task is very much similar to $\mathcal{F}_{\mathsf{vss}}$, except that D now inputs three sharings instead of one and in the verification phase, in addition to checking that D shared a weak sharing, the parties also verify that $c = ab$. We use an additional indicator bit happy to indicate whether each party is happy with the shares received from D. All honest parties are always happy with an honest D, but a corrupt D can choose which parties will be happy, and which will be unhappy. In particular, every honest party that received invalid shares from a corrupt D is unhappy with D.

Functionality $\mathcal{F}_{\mathsf{tss}}$

$\mathcal{F}_{\mathsf{tss}}$ receives the set of corrupt parties C, and has access to $\mathcal{F}_{\mathsf{crs}}$.

Sharing phase.

- **Inputs.** D inputs three pairs $(\mathbf{C}^a, \mathbf{O}^a), (\mathbf{C}^b, \mathbf{O}^b)$ and $(\mathbf{C}^c, \mathbf{O}^c)$. If D is honest, then these correspond to $\langle\!\langle a \rangle\!\rangle, \langle\!\langle b \rangle\!\rangle, \langle\!\langle c \rangle\!\rangle$ such that $ab = c$. In addition, a corrupt D inputs a bit happy_i for each $i \in \{1, \ldots, n\}$.
- **Outputs.** We split into two cases.
  - **Honest** D**.** $\mathcal{F}_{\mathsf{tss}}$ sets $\mathsf{happy}_i = 1$ for every $i \in \{1, \ldots, n\}$, and returns $(\mathbf{C}^a, \mathbf{O}^a_i), (\mathbf{C}^b, \mathbf{O}^b_i), (\mathbf{C}^c, \mathbf{O}^c_i)$ and happy_i to P_i.
  - **Corrupt** D**.** For every $i \in \{1, \ldots, n\}$, if one of the pairs $(\mathbf{C}^a, \mathbf{O}^a_i), (\mathbf{C}^b, \mathbf{O}^b_i), (\mathbf{C}^c, \mathbf{O}^c_i)$ is invalid, then $\mathcal{F}_{\mathsf{tss}}$ sets happy_i to 0 (otherwise, it does not change its value). $\mathcal{F}_{\mathsf{tss}}$ returns $(\mathbf{C}^a, \mathbf{O}^a_i), (\mathbf{C}^b, \mathbf{O}^b_i), (\mathbf{C}^c, \mathbf{O}^c_i)$ and happy_i to P_i.

Verification phase.

- **Honest parties' inputs.** Each honest P_i inputs a bit flag_i, such that if D is honest then $\mathsf{flag}_i = 0$ for every honest P_i.
- **Leakage.** The adversary receives flag_i from the functionality, for every $i \in \{1, \ldots, n\}$.
- **Adversary's inputs.** A corrupt party P_i inputs a bit flag_i. A corrupt dealer has two additional inputs, a bit flag_D, and $(\bar{\mathbf{O}}^a, \bar{\mathbf{O}}^b, \bar{\mathbf{O}}^c)$, where $\mathbf{O}^v = \{\bar{o}^v_{ij}\}_{i,j \in \{0,\ldots,n\}}$ for $v \in \{a, b, c\}$.
- **Outputs.** We split into two cases.
  - **Honest** D**.** Let W be the set of all corrupt parties P_i with $\mathsf{flag}_i = 1$. $\mathcal{F}_{\mathsf{tss}}$ returns $(\mathsf{W}, \{o^a_{ij}, o^b_{ij}, o^c_{ij}\}_{i \in \mathsf{W}, j \in \{0,\ldots,n\}})$ to all parties.
  - **Corrupt** D**.** Let W be the set containing all parties P_i with either $\mathsf{flag}_i = 1$ or $\mathsf{happy}_i = 0$. The parties output "D is corrupt" if either (1) $\mathsf{flag}_D = 1$, (2) the tuple $(\mathsf{W}, \mathbf{C}^v, \bar{\mathbf{O}}^v_{\mathsf{W}}, \mathbf{O}^v_{\mathsf{H} \setminus \mathsf{W}})$ is not a weak double t-sharing, for some $v \in \{a, b, c\}$, or (3) it does not holds that $F^a(0,0) \cdot F^b(0,0) = F^c(0,0)$, where $F^v(x, y)$ is the sharing polynomial of $(\mathsf{W}, \mathbf{C}^v, \bar{\mathbf{O}}^v_{\mathsf{W}}, \mathbf{O}^v_{\mathsf{H} \setminus \mathsf{W}})$, for $v \in \{a, b, c\}$. Otherwise, $\mathcal{F}_{\mathsf{tss}}$ returns $(\mathsf{W}, \{\bar{o}^a_{ij}, \bar{o}^b_{ij}, \bar{o}^c_{ij}\}_{i \in \mathsf{W}, j \in \{0,\ldots,n\}})$ to all parties.

Figure 8: Functionality $\mathcal{F}_{\mathsf{tss}}$

The core idea for our protocol realizing the above functionality is as follows. D picks three polynomials $A(x), B(x), C(x)$ of degree $n, n, 2n$ such that $A(x)B(x) = C(x)$ and $A(0) = a, B(0) = b$ and $C(0) = c$. D generates $[\![\cdot]\!]$-sharing ($\langle\!\langle \cdot \rangle\!\rangle$-sharing when D is honest) of $\{A^k, B^k, C^k\}_{k \in \{0,\ldots,2n\}}$, where A^k, B^k, C^k are the kth coefficients of the respective polynomials, using VSS instances. In order to verify that $c = ab$, it is enough to verify if the polynomials indeed satisfy the product relation or not. For this, we reconstruct $A(\alpha), B(\alpha), C(\alpha)$ for a uniform random α, chosen from "large enough" field (say, $\mathbb{F}_p$ where $p > 2^\kappa$, for a security parameter κ) and check if $C(\alpha) = A(\alpha) \cdot B(\alpha)$. If the check passes then the the sharings (strong or weak) of $\left(A^0, B^0, C^0\right)$ is taken as the output of the protocol.

To realize the above high-level idea, we face two challenges. First, α needs to be chosen after 1st round in order to tackle a rushing corrupt D which commits its secret in the first round. Instead of jointly generating a single α, we allow every party P_i to randomly sample a $\mathbb{R}^h$-intractable hash function h_{z_i} and then compute α_i as the output of h_{z_i} on the first round public (VSS) commitments. Then the verification turns to n checks $C(\alpha_i) = A(\alpha_i) \cdot B(\alpha_i)$ for all $i \in \{1, \ldots, n\}$. The circuit $\mathbb{R}^h$, at a high level, takes all the commitments and honest party's openings from the VSS instances and output 0, in case of either of the two bad events: (1) α is 0; (2) $C(0) \neq A(0)B(0)$ and (yet) $C(\alpha_i) = A(\alpha_i) \cdot B(\alpha_i)$. For full details on the $\mathbb{R}$-intractable hash function, see Appendix A.3.

The second challenge lies in sharing a bunch of secrets and reconstructing a linear function of of them, all in 2 rounds. This is where we use guided linear function evaluation $\mathsf{glinear}$, whose first round, being input-independent can be run in parallel to the sharing phase of VSS instances. In the 2nd round, the input-dependent round of $\mathsf{glinear}$ is applied on the tentative rows. Therefore for every α_i (which corresponds to one linear combination), we would need to run n independent instances (and a total of $3n$ instances) of $\mathsf{glinear}$ for computing $A(\alpha_i), B(\alpha_i)$ and $C(\alpha_i)$. A party who holds an invalid row for any of sharing of secrets $\{A^k, B^k, C^k\}_{k \in \{0,\ldots,2n\}}$, becomes 'unhappy', by setting $\mathsf{happy}_i = 0$ and sets its flag to 1 in all the VSS verification phases, so that all its row polynomials become publicly available. This implies the following for an honest P_j: either P_j holds valid jth rows for all the secrets and $\mathsf{glinear}$, with P_j as the guide, recovers the complete linearly combined row or P_j ensures all the jth rows are public so that the linear combination can be found in the clear. This means $\sum_{k=0}^{2n} \alpha_i^k [\![A^k]\!]_j$ will be known to all in the end of round 2, for an honest P_j. If D is not found to be corrupt, these can be used in the end to reconstruct $A(\alpha_i), B(\alpha_i), C(\alpha_i)$ (again if interpolation fails then D is identified to be bad) and do the test as mentioned earlier. The protocol is given in Fig. 9, its security statement in Theorem 3.13, and its proof in Appendix B.3.

Protocol tss

All parties have access to $\mathcal{F}_{\mathsf{crs}}$.

Primitives: Guided linear function evaluation glinear $=$ (glinear.off, glinear.on); $\mathbb{R}_\kappa^h$-intractable hash function; VSS vss.

Sharing phase (R1):

- *Inputs: D* inputs three pairs $(\mathbf{C}^a, \mathbf{O}^a), (\mathbf{C}^b, \mathbf{O}^b)$ and $(\mathbf{C}^c, \mathbf{O}^c)$, that are strong t-sharing of a, b and c, respectively, such that $ab = c$.
- *(Sharing phase of VSS)*
  - D picks three random polynomials $A(x), B(x)$, and $C(x)$ of degree n, n and $2n$, respectively, such that $A(x) \cdot B(x) = C(x)$, and $A(0) = a$, $B(0) = b$ and $C(0) = c$. Let A^k, B^k, and C^k denote the k-th coefficient of $A(x), B(x)$ and $C(x)$, respectively, for $k \in \{0, \ldots, 2n\}$, where $A^k = B^k = 0$ for $k > n$.
  - For each $k \in \{1, \ldots, 2n\}$, D shares A^k, B^k and C^k via three vss instances, $\mathsf{vss}^{a,k}, \mathsf{vss}^{b,k}$ and $\mathsf{vss}^{c,k}$, respectively, and we denote the corresponding sharing polynomials by $F^{a,k}(x,y), F^{b,k}(x,y)$ and $F^{c,k}(x,y)$.[a]
  - For $k = 0$ the values $A^0 = a, B^0 = b$ and $C^0 = c$ are shared by executing $\mathsf{vss}^{a,0}, \mathsf{vss}^{b,0}$ and $\mathsf{vss}^{c,0}$ with inputs $(\mathbf{C}^a, \mathbf{O}^a), (\mathbf{C}^b, \mathbf{O}^b)$ and $(\mathbf{C}^c, \mathbf{O}^c)$, respectively. We denote the corresponding sharing polynomials by $F^{a,0}(x,y), F^{b,0}(x,y)$ and $F^{c,0}(x,y)$.
- (glinear.off *calls)* For every $i, j \in \{1, \ldots, n\}$ and $v \in \{a, b, c\}$, the parties execute $\mathsf{glinear.off}^{i,j,v}$, with P_j as the guide.
- *(Challenge randomness)* Each P_i picks a random string $z_i \leftarrow \{0,1\}^\kappa$ and broadcasts z^i.
- *(Local computation)*
  - *(VSS sharing phase output)* For every $k \in \{0, \ldots, 2n\}$, the parties hold the tentative shares $[\![A^k]\!], [\![B^k]\!]$ and $[\![C^k]\!]$. Let $\mathbf{C} = \{\mathbf{C}^{v,k}\}_{v \in \{a,b,c\}, k \in \{0,\ldots,2n)\}}$.
  - *(Happiness)* If P_i received an invalid pair $(\mathbf{C}^{v,k}, \mathbf{O}_i^{v,k})$ for some $v \in \{a,b,c\}$ and $k \in \{0, \ldots, 2n\}$, then P_i is "unhappy", and sets $\mathsf{happy}_i := 0$. Otherwise, P_i is "happy", and sets $\mathsf{happy}_i := 1$.
  - *(Challenge generation)* For every $i \in \{1, \ldots, n\}$ the parties compute the challenge $\alpha_i \leftarrow h_{z_i}(\mathbf{C})$, where h_{z_i} is a $\mathbb{R}_\kappa^h$-intractable hash function; for $\mathbb{R}$'s exact description see[b]. If some α_i is 0, then the parties change its value to 1[c].
  - *(Output)* P_i outputs $(\mathbf{C}^{a,0}, \mathbf{O}_i^{a,0}), (\mathbf{C}^{b,0}, \mathbf{O}_i^{b,0}), (\mathbf{C}^{c,0}, \mathbf{O}_i^{c,0})$, and happy_i.

Verification phase (R2):

- *(Inputs)* Each P_i has an input bit flag_i.

- *(Verification phase of VSS)* For each P_i, if $\mathsf{flag}_i = 1$ then P_i sets happy_i to 0. P_i participates in the verification phase of all VSS instances with happy_i.
- *(Challenge computation)* For every $i \in \{1, \ldots, n\}$, the parties participate in $\mathsf{glinear.on}^{i,j,a}$, $\mathsf{glinear.on}^{i,j,b}$ and $\mathsf{glinear.on}^{i,j,c}$ to compute $\sum_{k=0}^{2n} \alpha_i^k [\![A^k]\!]_j$, $\sum_{k=0}^{2n} \alpha_i^k [\![B^k]\!]_j$ and $\sum_{k=0}^{2n} \alpha_i^k [\![C^k]\!]_j$, respectively, with P_j as a guide, with input $\delta^G = (\mathsf{happy}_i, \ldots, \mathsf{happy}_i)$ and every P_m with input $\delta^m = (\mathsf{happy}_m, \ldots, \mathsf{happy}_m)$.
- *(Local computation)*
  - *(VSS verification phase outputs)* The parties compute the output of all vss executions. If the output of some execution is "D is corrupt" then they output "D is corrupt" and terminate. Otherwise, denote the output of $\mathsf{vss}^{v,k}$ by $(\mathsf{W}^{v,k}, \{\bar{o}_{ij}^{v,k}\}_{i\in \mathsf{W}^{v,k}, j\in\{0,\ldots,n\}})$. Let $\mathsf{W} := \bigcap_{v\in\{a,b,c\},k\in\{0,\ldots,2n)\}} \mathsf{W}^{v,k}$.
  - *(Challenge verification)* Denote the output of $\mathsf{glinear.on}^{i,j,v}$ by $\mathsf{out}^{i,j,v}$. Let V be the set of parties P_j such that (1) $j \notin \mathsf{W}^{v,k}$, and for all $v \in \{a, b, c\}, i \in \{1, \ldots, n\}$ (2) $\mathsf{out}^{i,j,v}$ is not "G is corrupt", and (3) $\mathsf{out}^{i,j,v}$ contain the same $\mathbf{a} = (\alpha_i^0, \alpha_i^1, \ldots, \alpha_i^{2n})$ and $\delta^G = (1, \ldots, 1)$.
    If $|\mathsf{V} \cup \mathsf{W}| < n - t$, then output "$D$ is corrupt". Otherwise, for each $i \in \{1, \ldots, n\}$ do the following.
    * For each $j \in \mathsf{V}$ and $v \in \{a, b, c\}$, let $\mathsf{out}^{i,j,v} = (\mathbf{a}, \delta^G, t_{ijv})$.
    * For each $j \in \mathsf{W}$, $k \in \{0, \ldots, 2n\}$, and $v \in \{a, b, c\}$, let $f_{j0}^{v,k} := \mathsf{open}_{\mathsf{crs}}(C_{j0}^{v,k}, \bar{o}_{j0}^{v,k})$, and let $t_{ijv} := \sum_{k=0}^{2n} \alpha_i^k f_{j0}^{vk}$.
    * For each $v \in \{a, b, c\}$, if $\{t_{ijv}\}_{j\in \mathsf{V}\cup\mathsf{W}}$ do not correspond to a degree-t polynomial, then output "D is corrupt" and terminate. Otherwise, let t_{iv} be the free coefficient of this polynomial.
    * If $t_{ia} \cdot t_{ib} \neq t_{ic}$ then output "D is corrupt" and terminate. Otherwise, the i-th verification has succeeded.
  - *(Output)* If all the n the verification succeeded then output $(\mathsf{W}, \{\bar{o}_{ij}^{a,0}, \bar{o}_{ij}^{b,0}, \bar{o}_{ij}^{c,0}\}_{i\in\mathsf{W},j\in\{0,\ldots,n\}})$.

[a] For $k > n$ we simply assume a default strong double t-sharing of $A^k = B^k = 0$, which is locally computed by the parties. In particular, the parties set $F^{a,k}(x, y)$ and $F^{b,k}(x, y)$ to be the zero polynomial, and set $(C_{ij}^{v,k}, o_{ij}^{v,k}) \leftarrow \mathsf{commit}_{\mathsf{crs}}(0; \vec{0})$ for every $v \in \{a, b\}$ and $i, j \in \{0, \ldots, n\}$, where $\vec{0}$ is the all-zero string.

[b] Let $\mathbb{R}_\kappa^h$ be the circuit that takes as an input, commitments $\mathbf{C} := \{\mathbf{C}^{v,k}\}_{v\in\{a,b,c\},k\in\{0,\ldots,2n\}}$ and openings $\mathbf{O} := \{\mathbf{O}_i^{v,k}\}_{v\in\{a,b,c\},k\in\{0,\ldots,2n\},i\in\mathsf{H}}$. If there exists v, k such that $(\varnothing, \mathbf{C}, \varnothing, \mathbf{O}_{\mathsf{H}}^{v,k})$ is not a weak double t-sharing, with a sharing polynomial $F^{v,k}(x, y)$, then $\mathbb{R}$ outputs 1. Otherwise, the circuit computes $\alpha := h(\mathbf{C})$, and outputs 0 if and only if either (1) $\alpha = 0$, or (2) $F^{a,0}(0, 0) \cdot F^{b,0}(0, 0) \neq F^{c,0}(0, 0)$ and $(\sum_{k=0}^{2n} \alpha^k F^{a,k}(0, 0)) \cdot (\sum_{k=0}^{2n} \alpha^k F^{b,k}(0, 0)) = (\sum_{k=0}^{2n} \alpha^k F^{c,k}(0, 0))$. See also Appendix A.3.

[c] If some α_i is 0 then $A(0) = a$, $B(0) = b$ and $C(0) = c$ will be revealed (and we want to keep them secret). Note that a corrupt P_i might choose bad randomness z^i that will force $\alpha_i = 0$, even though the hash-function should be resistant against this event.

Figure 9: Protocol tss

Theorem 3.13. *Let κ be a security parameter, let n be the number of parties, let $t < n/2$, and let $\mathbb{F}$ be a field of characteristic 2. Protocol* tss *is a UC-secure implementation of $\mathcal{F}_{\mathsf{tss}}$ with everlasting security, against a static, active, rushing adversary corrupting up to t parties. The complexity of the protocol is* $\mathrm{poly}(n, \log |\mathbb{F}|, \kappa)$.

Notation 5. *We say that D shares a random triple $\langle\!\langle a\rangle\!\rangle$, $\langle\!\langle b\rangle\!\rangle$, $\langle\!\langle c\rangle\!\rangle$ via* tss, *if (1) D picks random symmetric bivariate polynomials $F^a(x, y)$, $F^b(x, y)$ and $F^c(x, y)$, of degree at most t in each variable, such that $F^a(0, 0) \cdot F^b(0, 0) = F^c(0, 0)$, (2) samples $(C_{ij}^v, o_{ij}^v) \leftarrow \mathsf{commit}_{\mathsf{crs}}(F^v(i, j); r_{ij}^v)$ for every $v \in \{a, b, c\}$ and $i, j \in \{0, \ldots, n\}$ such that $i \leq j$, where r_{ij}^v is a fresh random string, (3) sets $C_{ji}^v := C_{ij}^v$ and $o_{ji}^v := o_{ij}^v$ for every $i < j$ and $v \in \{a, b, c\}$, and (4) initiates* tss *with $\mathbf{C}^a := \{C_{ij}^a\}_{i,j\in\{0,\ldots,n\}}$, $\mathbf{C}^b := \{C_{ij}^a\}_{i,j\in\{0,\ldots,n\}}$, $\mathbf{C}^c := \{C_{ij}^a\}_{i,j\in\{0,\ldots,n\}}$, and $\mathbf{O}^a := \{o_{ij}^a\}_{i,j\in\{0,\ldots,n\}}$, $\mathbf{O}^b := \{o_{ij}^b\}_{i,j\in\{0,\ldots,n\}}$, $\mathbf{O}^c := \{o_{ij}^c\}_{i,j\in\{0,\ldots,n\}}$.*

3.5.1 Triple secret sharing without $\mathbb{R}$-intractable hash functions

We can achieve this goal in three ways: (a) by allowing 3 rounds (b) assuming semi-rushing adversaries. For the latter case, every party broadcasts their α_i in the first round itself and the adversary being semi-rushing commits its sharing via VSS instances before seeing at least one α_i belonging to some honest party, ensuring sound verification of the product relation. We explain the former below.

Protocol tss is the only protocol that uses $\mathbb{R}$-intractable hash-functions, which are used in order to generate the challenges $\alpha_1, \ldots, \alpha_n$ already in the first round. As we have mentioned above, we cannot let the parties pick $\alpha_1, \ldots, \alpha_n$ in the first round by themselves, since a corrupt rushing dealer would first see the challenges, and then would be able to pick $A(x), B(x)$ and $C(x)$ such that $A(\alpha_i)B(\alpha_i) = C(\alpha_i)$ for all $i \in \{1, \ldots, n\}$, while $A(0) \cdot B(0) \neq C(0)$.

However, if we allow tss to have 3 rounds, then we can get rid of the $\mathbb{R}$-intractable hash-functions with a small modification to the above protocol: (1) in the first round the parties execute the sharing phase of the original protocol, except for the sampling of the hash-function, (2) in the second round each P_i broadcasts a random field element α_i, and (3) in the third round the parties execute the verification phase of the original protocol. The same analysis shows that the new protocol securely implements $\mathcal{F}_{\mathsf{tss}}$. Using this 3-round protocol, instead of the 2-round protocol, we can obtain a 4-round protocol for degree-2 computation with everlasting security assuming only statistically-hiding NICOM and CRS, and with computational security assuming only computationally-hiding NICOM (instead of a 3-round protocol assuming also $\mathbb{R}$-intractable hash functions).

3.6 General Single-input Functions

In this section, we obtain a 2-round protocol for every function whose outputs are determined by the input of a single party. The class of single-input functions include important tasks such as distributed ZK which we will utilize in our degree-2 computation. [33] reduces secure computation of a single-input function (SIF) to that of degree-2 polynomials[8] and subsequently show a 2-round construct to evaluate the latter with perfect security and threshold $t < n/6$. In this work, we complement their reduction with optimal threshold $t < n/2$. The functionality $\mathcal{F}_{\mathsf{sif}}$ for SIF evaluation has two phases. In the input phase, an honest dealer gives its inputs to $\mathcal{F}_{\mathsf{sif}}$, and the adversary receives no information at all about the inputs. In the output phase, the parties receive the outputs from $\mathcal{F}_{\mathsf{sif}}$. A corrupt dealer has more power, and he can send his inputs to $\mathcal{F}_{\mathsf{sif}}$ in this phase.

Functionality $\mathcal{F}_{\mathsf{sif}}$

$\mathcal{F}_{\mathsf{sif}}$ receives the set of corrupt parties C, and has access to $\mathcal{F}_{\mathsf{crs}}$.

Input phase. $\mathcal{F}_{\mathsf{tss}}$ receives from an honest D inputs $\mathbf{z} = (z^1, \ldots, z^\ell)$. There are no outputs in this phase.

Output phase. We split into two cases.

- **Honest** D. $\mathcal{F}_{\mathsf{sif}}$ returns $y(\mathbf{z}) = (y^1(\mathbf{z}), \ldots, y^m(\mathbf{z}))$ to all parties, where $y^i(z^1, \ldots, z^\ell) =$

[8] We mention that this reduction works even when the functionality depends on the common reference string, since all parties, and in particular the dealer, have access to the CRS.

$\sum_{p\in\{1,\ldots,\ell\}} \alpha^i_p z^p + \sum_{p,q\in\{1,\ldots,\ell\}} \alpha^i_{pq} z^p z^q$ is a degree-2 polynomial in the variables $z^1,\ldots,z^\ell$.

- **Corrupt** D**.** $\mathcal{F}_{\mathsf{sif}}$ receives from D inputs $\mathbf{z} = (z^1,\ldots,z^\ell)$ and returns $y(\mathbf{z})$ to all parties.

Figure 10: Functionality $\mathcal{F}_{\mathsf{sif}}$

We now present a 2-round protocol sif (Fig 11), realizing $\mathcal{F}_{\mathsf{sif}}$. The idea of the protocol is simple: D shares each input z^i as well as the product $z^i z^j$ for each $i,j \in \{1,\ldots,\ell\}$ using VSS instances. In order to make sure that D shared the correct product, the parties execute TSS with $\langle\langle z^i\rangle\rangle$, $\langle\langle z^j\rangle\rangle$ and $\langle\langle z^i z^j\rangle\rangle$ as the inputs. Now, to compute each output y_i, a linear function of z^i's and $z^i z^j$'s, we will use glinear (n instances). For security, we need to randomize each y_i before reconstruction using a sharing of 0. This can be generated by sharing $\langle\langle 0\rangle\rangle$,$\langle\langle 0\rangle\rangle$ and $\langle\langle 0\rangle\rangle$ via TSS, and then opening first two sharing to prove that the last sharing corresponds to 0. The security statement is given below and the proof appear in Appendix B.4.

Protocol sif

All parties have access to $\mathcal{F}_{\mathsf{crs}}$.

Primitives: Guided linear function evaluation glinear $=$ (glinear.off, glinear.on); TSS tss; VSS vss.

Inputs. D has input $z^1,\ldots,z^\ell$.

Input phase (R1). The parties do the following.

- *(Input sharing)* For every $i \in \{1,\ldots,\ell\}$, D shares z^i via a VSS instance vss^i. Denote the corresponding strong degree-t sharing by $(\mathbf{C}^i, \mathbf{O}^i)$, and the sharing polynomial by $F^i(x,y)$.
- *(Monomials sharing)* For every $i,j \in \{1,\ldots,\ell\}$, D shares $z^{ij} := z^i \cdot z^j$ via a VSS instance vss^{ij}. Denote the corresponding strong degree-t sharing by $(\mathbf{C}^{ij}, \mathbf{O}^{ij})$, and the sharing polynomial by $F^{ij}(x,y)$.
- *(*tss *Input Sharing)* For every $i,j \in \{1,\ldots,\ell\}$ the parties execute an instance of tss, denoted tss^{ij}, where D inputs $(\mathbf{C}^i, \mathbf{O}^i)$, $(\mathbf{C}^j, \mathbf{O}^j)$ and $(\mathbf{C}^{ij}, \mathbf{O}^{ij})$.
- *(0 sharing)* For each $i \in \{1,\ldots,m\}$, the parties execute an instance of tss, denoted tss^i, where D inputs three random strong sharings of zero, denoted $(\mathbf{C}^{\gamma_i}, \mathbf{O}^{\gamma_i})$, $(\mathbf{C}^{\rho_i}, \mathbf{O}^{\rho}_i)$ and $(\mathbf{C}^{\eta_i}, \mathbf{O}^{\eta_i})$. In addition, D broadcasts $(\mathbf{C}^{\gamma_i}, \mathbf{O}^{\gamma_i})$ and $(\mathbf{C}^{\rho_i}, \mathbf{O}^{\rho_i})$.
- *(*glinear.off *calls)* For every $i \in \{1,\ldots,m\}$ and $j \in \{1,\ldots,n\}$, the parties execute $\mathsf{glinear.off}^{i,j}$ with P_j as a guide.
- *(Local computation)* If, for some $i \in \{1,\ldots,m\}$, either $(\mathbf{C}^{\gamma_i}, \mathbf{O}^{\gamma_i})$ or $(\mathbf{C}^{\rho_i}, \mathbf{O}^{\rho_i})$ which were broadcasted by D as part of the 0 sharing, are not a strong sharing of zero, then D is disqualified. Otherwise, denote the output of party P_k in the execution of vss^i by $(\mathbf{C}^i, \mathbf{O}^i_k)$, and in the execution of vss^{ij} by $(\mathbf{C}^{ij}, \mathbf{O}^{ij}_k)$. If (1) P_k received an invalid pair in some vss or tss call, or (2) for some $i,j \in \{1,\ldots,\ell\}$, the output of party P_k in tss^{ij} is not $(\mathbf{C}^i, \mathbf{O}^i_k)$, $(\mathbf{C}^j, \mathbf{O}^j_k)$, $(\mathbf{C}^{ij}, \mathbf{O}^{ij}_k)$ and $\mathsf{happy}_k = 1$, or (3) for some $i \in \{1,\ldots,m\}$, the output of party P_k in tss^i is not $(\mathbf{C}^{\gamma_i}, \mathbf{O}^{\gamma_i}_k)$, $(\mathbf{C}^{\rho_i}, \mathbf{O}^{\rho_i}_k)$, $(\mathbf{C}^{\eta_i}, \mathbf{O}^{\eta_i}_k)$ and $\mathsf{happy}_k = 1$, where $(\mathbf{C}^{\gamma_i}, \mathbf{O}^{\gamma_i})$ and $(\mathbf{C}^{\rho_i}, \mathbf{O}^{\rho_i})$ were broadcasted by D as part of the 0 sharing, then P_k sets $\mathsf{flag}_k = 1$. Otherwise P_k sets $\mathsf{flag}_k = 0$.

Output phase (R2). The parties output $y(0,\ldots,0)$ if D was disqualified and do the following otherwise.

- *(VSS verification phase)* The parties execute the verification phase of vss^i and vss^{ij} for every $i,j \in \{1,\ldots,\ell\}$, where party P_k inputs flag_k in all executions.

- (tss *verification phase)* For each $i, j \in \{1, \ldots, \ell\}$ the parties execute the verification phase of tss^{ij}, where party P_k inputs flag_k in all executions.
- (glinear.on *calls)* For $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, n\}$, in $\mathsf{glinear.on}^{i,j}$ the parties partially compute $\sum_{p=1}^{\ell} \alpha_p^i [\![z^p]\!]_j + \sum_{p,q \le \ell} \alpha_{pq}^i [\![z^{pq}]\!]_j + [\![\eta_i]\!]_j$, with P_j as the guide with inputs $\delta^G = (\overline{\mathsf{flag}_j}, \ldots, \overline{\mathsf{flag}_j})$ and every P_m with $\delta^m = (\overline{\mathsf{flag}_m}, \ldots, \overline{\mathsf{flag}_m})$.[a]
- *(Local computation)*
  - *(VSS verification phase outputs)* The parties compute the output of all vss executions. If the output of some execution is "D is corrupt" then they output $y(0, \ldots, 0)$ and terminate. Otherwise, denote the output of vss^i by $(\mathsf{W}^i, \bar{\mathbf{O}}^i_{\mathsf{W}^i})$, and of vss^{ij} by $(\mathsf{W}^{ij}, \bar{\mathbf{O}}^{ij}_{\mathsf{W}^{ij}})$.
  - *(TSS verification phase outputs)* The parties compute the output of all tss executions. If the output of some execution is "D is corrupt" then they output $y(0, \ldots, 0)$ and terminate. Otherwise, for $i, j \in \{1, \ldots, \ell\}$, denote the output of tss^{ij} by $(\mathsf{W}^{ij}_{\mathsf{tss}}, \tilde{\mathbf{O}}^i_{\mathsf{W}^{ij}_{\mathsf{tss}}}, \tilde{\mathbf{O}}^j_{\mathsf{W}^{ij}_{\mathsf{tss}}}, \tilde{\mathbf{O}}^{ij}_{\mathsf{W}^{ij}_{\mathsf{tss}}})$, and for $i \in \{1, \ldots, m\}$ denote the output of tss^i by $(\mathsf{W}^i_{\mathsf{tss}}, \tilde{\mathbf{O}}^i_{\mathsf{W}^i_{\mathsf{tss}}}, \tilde{\mathbf{O}}^i_{\mathsf{W}^i_{\mathsf{tss}}}, \tilde{\mathbf{O}}^i_{\mathsf{W}^i_{\mathsf{tss}}})$. Let $\mathsf{W} := \left(\bigcap_{i,j \in \{1,\ldots,\ell\}} (\mathsf{W}^i \cap \mathsf{W}^{ij} \cap \mathsf{W}^{ij}_{\mathsf{tss}})\right) \cap \left(\bigcap_{i \in \{1,\ldots,m\}} \mathsf{W}^i_{\mathsf{tss}}\right)$.
  - (glinear *outputs)* Denote the output of $\mathsf{glinear.on}^{i,j}$ by $\mathsf{out}^{i,j}$.
  - *(Output computation)* Let V be the set of all parties P_k such that (1) $k \notin \mathsf{W}^i$, $k \notin \mathsf{W}^{ij}$ and $k \notin \mathsf{W}^{ij}_{\mathsf{tss}}$ for any $i, j \in \{1, \ldots, \ell\}$, and $k \notin \mathsf{W}^i_{\mathsf{tss}}$ for any $i \in \{1, \ldots, m\}$, (2) none of $\{\mathsf{out}^{i,k}\}_{i \in \{1,\ldots,m\}}$ is "G is corrupt", and (3) according to $\{\mathsf{out}^{i,k}\}_{i \in \{1,\ldots,m\}}$ in all executions as a guide, party P_k used $(\alpha_1^i, \ldots, \alpha_\ell^i, \alpha_{11}^i, \ldots, \alpha_{\ell,\ell}^i, 1)$ and $\delta^G = (1, \ldots, 1)$ in the execution of $\mathsf{glinear}^{i,k}$.
    If $|\mathsf{V} \cup \mathsf{W}| < n - t$ then output $y(0, \ldots, 0)$ and terminate. Otherwise, for each $i \in \{1, \ldots, m\}$ do the following.
    - For each $k \in \mathsf{V}$ let $\mathsf{out}^{i,k} = (\mathbf{a}, \delta^G, t_{ik})$.
    - For each $k \in \mathsf{W}$ let $f_{k0}^j := \mathsf{open}_{\mathsf{crs}}(C_{k0}^j, \bar{o}_{k0}^j)$, $f_{k0}^{jr} := \mathsf{open}_{\mathsf{crs}}(C_{k0}^{jr}, \bar{o}_{k0}^{jr})$ for $j, r \in \{1, \ldots, \ell\}$, and let $p_k^i := \mathsf{open}_{\mathsf{crs}}(C_{k0}^{\eta_i}, \tilde{o}_{k0}^{\eta_i})$. Let $t_{ik} := \sum_{p=1}^{\ell} \alpha_p^i f_{k0}^p + \sum_{p,q \le \ell} \alpha_{pq}^i f_{k0}^{pq} + p_k^i$.
    - If $\{t_{ik}\}_{k \in \mathsf{V} \cup \mathsf{W}}$ do not correspond to a degree-t polynomial, then output $y(0, \ldots, 0)$ and terminate. Otherwise set y^i to the free coefficient of the degree-t polynomial.
    - Output $(y^1, \ldots, y^m)$.

[a]Here $\overline{\mathsf{flag}}$ is $1 - \mathsf{flag}$.

Figure 11: Protocol sif

Theorem 3.14. *Let κ be a security parameter, let n be the number of parties, let $t < n/2$, and let $\mathbb{F}$ be a field of characteristic 2. Protocol* sif *is a UC-secure implementation of $\mathcal{F}_{\mathsf{sif}}$ with everlasting security, against a static, active, rushing adversary corrupting up to t parties. The complexity of the protocol is* $\mathrm{poly}(\ell, m, n, \log|\mathbb{F}|, \kappa)$.

3.7 Verify & Open

To achieve robust degree-2 computation in 3 rounds, we need a way to recover any kth row polynomial of a VSS instance in the 3rd round irrespective of the corrupt status of D and P_k. We make a couple of failed attempts before reaching to the final solution. First, asking P_k to disclose the openings for its row polynomial in 3rd round (after VSS gets over) does not work, since a corrupt P_k may refuse. For tackling this single-point failure, distributing trust can be a way-forward. That is, we let every P_i open the ith share of P_k's row polynomial. However, this too does not work: For a VSS instance dealt by a corrupt dealer, it is possible that a corrupt P_k holds an invalid row i.e., the public commitments for this row correspond to a polynomial of degree more than t. This polynomial is pairwise consistent with all the honest parties' polynomials, yet the rest of the points are such that it's degree goes beyond degree t. For such a scenario, a reconstruction, no matter whether distributed or centralised (as mentioned above), will fail to reconstruct the t-degree polynomial defined by the honest parties row polynomials. Notably, while the corrupt status of the parties (P_k in centralized case, and D, P_k in the distributed case) can be detected in round-3, it is too late to settle everything in the same round. Here we introduce a new primitive– verify & open– that allows a special party D (P_k in the above discussion) to show that a set of n commitments correspond to a polynomial of degree-t within 2 rounds and open the same if the check verifies. As follows from the previous discussion, even an honest D's verification may fail, owing to the fact that the dealer of the VSS instance, the row of which is under scrutiny, is corrupt and had dealt invalid rows to honest parties. But, looking ahead, knowing this by 2 rounds, helps to take preemptive action in our 3-round degree-2 computation.

A two-phase primitive verify & open has a verification and an opening phase. In the former phase, a dealer D inputs openings to commitments $C_1, \ldots, C_n$, and each P_i inputs an opening to C_i. When D is honest, then the output of the verification phase is either (1) "verification succeeds", which happens when the openings correspond to a degree-t polynomial $f(x)$, or (2) "verification failed", which happens otherwise. The output of the opening phase is $f(0)$ in the former case and $\perp$ otherwise. When the inputs are bad (i.e., some opening of D is not valid, or the values do not correspond to a degree-t polynomial, or some opening of P_i is not valid), then we leak all of them to the adversary during the verification phase itself. In all cases, the adversary receives all the openings in the opening phase. We allow an additional bit input happy_D from D, which can be used by D to indicate that it is happy to disclose the openings during the verification itself. Looking ahead, this bit can be used by an honest D to indicate that the row that it is verifying comes from a corrupt VSS dealer.

For a corrupt D, an output of "verification succeeds" in the verification phase ensures that the shares of the honest parties correspond to a degree-t polynomial $f(x)$ and later in the opening phase, the output will be $f(0)$. Otherwise, the output of the verification phase may be "verification failed" or "D is corrupt", in which case $\perp$ will be the output of opening phase.

Functionality $\mathcal{F}_{\mathsf{vao}}$

$\mathcal{F}_{\mathsf{vao}}$ receives the set of corrupt parties C, and has access to $\mathcal{F}_{\mathsf{crs}}$.

Verification phase.

- **Honest parties' inputs.** All honest parties input the same commitments $(C_1, \ldots, C_n)$. An honest dealer inputs openings $(o_1^D, \ldots, o_n^D)$ and a bit verify_D. Every honest P_i inputs an opening o_i.
- **Leakage.** The adversary receives $(C_1, \ldots, C_n)$ from $\mathcal{F}_{\mathsf{vao}}$.
  If D is honest, the adversary also receives $\{o_i^D\}_{i \in \mathsf{C}}$ and verify_D. In addition, if (1) $\mathsf{verify}_D = 1$, or (2) there exists an index $k \in \{1, \ldots, n\}$ such that $f_k := \mathsf{open}_{\mathsf{crs}}(C_k, o_k^D)$ is $\perp$, or (3) $\{f_k\}_{k \in \{1,\ldots,n\}}$ do not correspond to a degree-t polynomial $f(x)$, or (4) there exists $k \in \mathsf{H}$ such that $\mathsf{open}_{\mathsf{crs}}(C_k, o_k) \neq f_k$, then the adversary also receives the openings $\{o_k^D\}_{k \in \{1,\ldots,n\}}$ and $\{o_k\}_{k \in \mathsf{H}}$.
  If D is corrupt, the adversary also receives o_i for every honest P_i.

- **Adversary's inputs.** A corrupt dealer inputs $(o_1^D, \ldots, o_n^D)$ and two bits, verify_D and flag_D.
- **Outputs.** We split into two cases.
  - **Honest D.** If either of the conditions (1), (2) or (3) under **Leakage/honest D case** is true, then return "verification failed" to all parties. Otherwise, set $s = f(0)$, where $f(x)$ is the degree t polynomial over $\{f_k\}_{k\in\{1,\ldots,n\}}$ and return "verification succeeded" to all parties.
  - **Corrupt D.** If $\mathsf{flag}_D = 1$ then output "D is corrupt" to all parties. Otherwise, if there exists $i \in \mathsf{H}$ such that $\mathsf{open}_{\mathsf{crs}}(C_i, o_i^D) = \perp$, or $\mathsf{verify}_D = 1$ then return "verification failed" to all parties. Otherwise, for $i \in \mathsf{H}$ let $f_i := \mathsf{open}_{\mathsf{crs}}(C_i, o_i^D)$, and let $f(x)$ be the polynomial obtained by interpolating $\{f_i\}_{i\in\mathsf{H}}$. If the degree of $f(x)$ is more than t then output "D is corrupt". Otherwise, set $s = f(0)$ and return "verification succeeded" to all parties.

Opening phase.

- **Leakage.** The adversary receives $(o_1^D, \ldots, o_n^D)$ and $(o_1, \ldots, o_n)$ from $\mathcal{F}_{\mathsf{vao}}$.
- **Outputs.** If the verification phase ended with "verification failed" or "D is corrupt" then return $\perp$ to all parties. Otherwise, if the verification phase ended with "verification succeeded", return s to all.

Figure 12: Functionality $\mathcal{F}_{\mathsf{vao}}$

We now present a realization, vao (Fig. 13), that takes 2 rounds for verification phase, in which the first round is input-independent and can be run in parallel to generating inputs to vao (via VSS instances), and one round for open phase, totalling to 3 rounds. The input-independent round is used by D to generate a degree-t polynomial, say $h(x)$, commit its n points publicly and give the ith value's opening to P_i. The 2 rounds of verification phase is used to run a SIF instance to prove that the underlying polynomial is of degree t. In order to check whether the polynomial, say $f(x)$, defined by the n commitments, $C_1, \ldots, C_n$, that are input to vao, is of degree at most t, $h(x)$ is then used as a mask for a public recovery of $h(x) + f(x)$, within the verification phase. This is enabled via n instances of spcg, with D acting as Alice and every P_i, acting as Bob. Again, we leverage the fact that the offline of SPCG $\mathsf{spcg.off}$ is one round and is input (f, h here) independent. If D's SIF instance confirms that $h(x)$ is of degree t and the public polynomial $h(x) + f(x)$ is of degree t, then $f(x)$ (as defined by the honest parties' values) is guaranteed to be of degree t. This concludes the verification phase and we define below the exact single-input function that D needs to invoke.

Let $\mathcal{F}_1$ be the single-input functionality that takes from D commitments $(C'_1, \ldots, C'_n)$ and openings $(o'_1, \ldots, o'_n)$, and returns to all parties the commitments $(C'_1, \ldots, C'_n)$ and a bit out such that $\mathsf{out} = 1$ if and only if $h_i := \mathsf{open}_{\mathsf{crs}}(C'_i, o'_i)$ is not $\perp$ for every $i \in \{1, \ldots, n\}$, and $\{h_i\}_{i\in\{1,\ldots,n\}}$ correspond to some degree-t polynomial $h(x)$.

Moving on to the opening phase, no steps are taken when either verification fails or D is identified as corrupt. Otherwise, the open happens via n SIF instances, each carried output by P_i as the dealer. To conclude the opening in one round, the first round of these SIFs takes place during the 2nd round of verification phase. The exact single-input function that we use is given below.

Let $\mathcal{F}_2$ be the single-input functionality that takes from a dealer commitments (C, C') and openings (o, o'), computes $f = \mathsf{open}_{\mathsf{crs}}(C, o)$ and $h = \mathsf{open}_{\mathsf{crs}}(C', o')$, returns "fail" if either $h = \perp$ or $f = \perp$, and otherwise returns $(C, C', h + f, f)$ to all parties. For P_i, the commitments are C_i, C'_i and the openings are o_i, o'_i.

The value $(h + f)$ from ith SIF is patted against the sum computed by $\mathsf{spcg.on}^i$ to prove that

P_i indeed holds openings to C_i and C'_i that sums up to what it allows its SPCG to compute as Bob. Recall that the outputs of SPCGs are checked for degree-t property and the verification phase's success is conditional to this. This step is important because, a corrupt D and P_i can choose the output of $\mathsf{spcg.on}^i$ to be whatever they like. That is, even if C_i's opening with P_i is not consistent with the honest parties shares', D and P_i could make it look as if it is consistent. However, now in SIF, P_i will fail and we will not consider its contribution for reconstructing $f(x)$. So the f component of ith SIF that gives non-$\perp$ output and allows a successful check for $(h+f)$-component as above, is taken for reconstructing $f(x)$. Lastly, while it seems making every P_i reveal the openings on the ith values of $h(x), f(x)$ would allow us to recover $f(x)$, we use this non-trivial way of opening $f(x)$ to make the simulation work when D is honest. In particular, our method via SIF allows to disclose the desired output, while keeping the openings secret, which is exactly what the simulation needs for its success. The security statement is given below and the proof appears in Appendix B.5.

Protocol vao

All parties have access to $\mathcal{F}_{\mathsf{crs}}$.

Primitives: SPCG $\mathsf{spcg} = (\mathsf{spcg.off}, \mathsf{spcg.on})$; SIF sif.

Verification phase (R1 and R2).

- **R1 (Input-independent).** The parties do as follows.
  - D picks a random degree-t polynomial $h(x)$, computes $(C'_i, o'_i) \leftarrow \mathsf{commit}_{\mathsf{crs}}(h(i), r_i)$ for every $i \in \{1, \ldots, n\}$, where r_i is a fresh random string, broadcasts $\{C'_i\}_{i\in\{1,\ldots,n\}}$ and sends o'_i to P_i.
  - ($\mathsf{spcg.off}$ *call)* For every $i \in \{1, \ldots, n\}$, the parties execute $\mathsf{spcg.off}^i$, with D as Alice and P_i as Bob.
  - (sif *input phase for* $\mathcal{F}_1$*)* The parties execute the input phase of sif, denoted sif^D, computing the function $\mathcal{F}_1$, where D acts as the dealer and participates with inputs $\{C'_i, o'_i\}_{i\in\{1,\ldots,n\}}$.
- **Inputs.** All the parties input the same commitments $(C_1, \ldots, C_n)$. D also inputs openings $(o_1^D, \ldots, o_n^D)$ and a bit verify_D. Every P_i inputs an opening o_i.
- **R2.** The parties do as follows.
  - For every i, D computes $f_i := \mathsf{open}_{\mathsf{crs}}(C_i, o_i^D)$. If $\mathsf{verify}_D = 1$, or $f_i = \perp$ for some i, or the polynomial obtained by interpolating $\{f_i\}_{i\in\{1,\ldots,n\}}$ is of degree more than t, then D broadcasts "verification failed" and terminates.
  - ($\mathsf{spcg.on}$ *calls)* Every P_i computes $f'_i := \mathsf{open}_{\mathsf{crs}}(C_i, o_i)$, and $h'_i := \mathsf{open}_{\mathsf{crs}}(C'_i, o'_i)$. If $f'_i = \perp$ or $h'_i = \perp$ then P_i sets $\delta^i = (0,0)$, and $b_1^i = b_2^i = 0$. Otherwise, P_i sets $\delta^i = (1,1)$, $b_1^i := f'_i$ and $b_2^i := h'_i$. For every $i \in \{1, \ldots, n\}$, the parties execute the online phase of $\mathsf{spcg.on}^i$ as follows.
    - All parties input (C_i, C'_i).
    - D, as Alice, inputs $\mathbf{a} := (1,1)$, $\delta^A := (1,1)$, and $((o_i^D, f_i), (o'_i, h(i)))$.[a]
    - P_i, as Bob, inputs δ^i and (b_1, b_2).
  - (sif *output phase for* $\mathcal{F}_1$*)* The parties execute the output phase of sif^D.
  - (sif *input phase for* $\mathcal{F}_2$*)* For every $i \in \{1, \ldots, n\}$ the parties execute an instance of sif, denoted sif^i, computing the function $\mathcal{F}_2$ with P_i as the dealer, where P_i inputs (C_i, C'_i) and (o_i, o'_i).
  - *(Local computation)* If D has broadcasted "verification failed" then all parties output the same and terminate. The parties output "D is corrupt" and terminate if one of the following is true:

1. The output of sif^D is not $(C'_1, \ldots, C'_n)$ and $\mathsf{out} = 1$.
2. For spcg^i denote the output by out_i. Some out_i is "Alice is corrupt".
3. Let V be the set of all indices i such that out_i is not "Bob is corrupt", and for some i, the out_i is not in format $(\mathbf{a}^i, \delta^{A,i}, \{j, o_j^{A,i}, b_j^{A,i}\}_{j \in I_i}, \sigma_i)$, where I_i is the set I corresponding to out_i, and σ_i is the partial-sum.
4. There exists $i \in \mathsf{V}$ such that (1) $\mathbf{a}^i \neq (1,1)$, or (2) $\delta^{A,i} \neq (1,1)$, or (3) $1 \in I_i$ and $\mathsf{open}_{\mathsf{crs}}(C_1, o_1^{A,i}) \neq b_1^{A,i}$, or (4) $2 \in I_i$ and $\mathsf{open}_{\mathsf{crs}}(C_2, o_2^{A,i}) \neq b_2^{A,i}$, or (5) $1, 2 \in I_i$ and $b_1^{A,i} + b_2^{A,i} \neq \sigma_i$.
5. The interpolation over $\{\sigma_i\}_{i \in \mathsf{V}}$ results in a polynomial of degree larger than t.

Otherwise, parties output "verification succeeded".

Opening phase (R3). If the verification phase ended with "verification succeeded", the parties execute the output phase of sif^i for every $i \in \{1, \ldots, n\}$. Otherwise, the parties do nothing.

- *(Local Computation)* If the verification phase ended with "D is corrupt" or "verification failed", then output $\perp$. Otherwise, let V' be the set of all P_i in V such either (1) $I_i = \{1,2\}$, or (2) $I_i = \varnothing$, and the output of sif^i is $(C_i, C'_i, \sigma_i, f_i)$, where σ_i is the output of $\mathsf{spcg.on}^i$. For every $i \in \mathsf{V}'$ such that $I_i = \{1,2\}$ set $s_i := b_1^{A,i}$, and for every $i \in \mathsf{V}'$ such that $I_i = \varnothing$, let $s_i := f_i$. Let $S(x)$ be the polynomial obtained by interpolating $\{s_i\}_{i \in \mathsf{V}'}$. Output $S(0)$.

[a]We use $\alpha_i(\mathbf{a}) = a_i$ for $i \in \{1,2\}$ in the expression of function f to be computed by SPCG.

Figure 13: Protocol vao

Lemma 3.15. *Let κ be a security parameter, let n be the number of parties, let $t < n/2$, and let $\mathbb{F}$ be a field of characteristic 2. Protocol* vao *is a UC-secure implementation of $\mathcal{F}_{\mathsf{vao}}$ with everlasting security, against a static, active, rushing adversary corrupting up to t parties. The complexity of the protocol is* $\mathrm{poly}(n, \log|\mathbb{F}|, \kappa)$.

3.8 Guided Degree-2 Computation

Building on and extending the notion of guided linear function evaluation, we introduce here guided degree-2 computation, the final stop before arriving our destination of general degree-2 computation. Jumping directly in the problem, here the guide G has $m+2$ tentative rows, say $\lfloor b_\alpha \rceil, \lfloor b_\beta \rceil, \lfloor b_1 \rceil, \ldots, \lfloor b_m \rceil$, each of which can be either kth row or main row of a corresponding sets of double sharing dealt by various parties.[9] The goal is to compute $a_0 b_\alpha b_\beta + \sum_{i=1}^m a_i b_i$ for a publicly known linear combiner $\mathbf{a} = (a_0, a_1, \ldots, a_m)$, which includes a degree-2 term. Since we will operate over tentative shares, G as well as every P_k provide $m+2$ length binary indicator vectors. Similar to $\mathsf{glinear}$, we are pressed with the need to complete the current task in 2 rounds and the first round needs to be input-independent so that it can run in parallel to the generation of the tentative sharing which act as the inputs to this task. When either of the indicators corresponding to b_α, b_β is 0, the degree-2 computation turns into a linear computation. Looking ahead, for a degree-2 computation of $m+2$ secrets, we need to run n guided degree-2 computation, one for every P_k as guide. The ideal functionality $\mathcal{F}_{\mathsf{gdtc}}$ (Fig. 14) for this task looks similar to that of guided linear function evaluation, and reflects the above discussion.

[9]We assume that m is (at most) polynomial in κ.

Functionality $\mathcal{F}_{\mathsf{gdtc}}$

$\mathcal{F}_{\mathsf{gdtc}}$ receives the set of corrupt parties C, and has access to $\mathcal{F}_{\mathsf{crs}}$.

Honest parties' inputs:

- All honest parties input the same commitments $(C_{ij})_{i\in\{\alpha,\beta,1,\ldots,m\},j\in\{0,\ldots,n\}}$.
- If G is honest, G inputs (1) a list of coefficients $\mathbf{a} = (a_0,\ldots,a_m) \in \mathbb{F}^{m+1}$, (2) a list of values $\mathbf{b}^G = (b^G_{ij})_{i\in\{\alpha,\beta,1,\ldots,m\},j\in\{0,\ldots,n\}} \in \mathbb{F}^{(m+2)(n+1)}$, (3) a list of openings $\{o^G_{ij}\}_{i\in\{\alpha,\beta,1,\ldots,m\},j\in\{0,\ldots,n\}}$, and (4) an indicator vector $\delta^G \in \{0,1\}^{m+2}$.
  For every $i \in \delta^G$ it holds that $\mathsf{open}_{\mathsf{crs}}(C_{ij}, o^G_{ij}) = b^G_{ij}$ for every $j \in \{0,\ldots,n\}$, and the values $\{b_{ij}\}_{j\in\{0,\ldots,n\}}$ correspond to a degree-t polynomial. In addition, $a_0 \neq 0$.
- An honest P_i inputs a list $\delta^i \in \{0,1\}^{m+2}$, and values $(b^i_\alpha, b^i_\beta, b^i_1,\ldots,b^i_m) \in \mathbb{F}^{m+2}$.

Leakage: The adversary receives $(C_{ij})_{i\in\{\alpha,\beta,1,\ldots,m\},j\in\{0,\ldots,n\}}$ and δ^i of every honest P_i. In addition,

- If G is honest, the adversary receives $\mathbf{a}, \{b^G_{ij}, o^G_{ij}\}_{i\in\{\alpha,\beta,1,\ldots,m\},j\in\mathsf{C}}$ and δ^G, as well as $L_j := \{(i,j), b^G_{ij}, o^G_{ij}\}_{i\in I_j}$, for any $j \in \mathsf{H}$, where $I_j := (\mathbf{b}^G, \delta^G)\diamond(\mathbf{b}^i, \delta^i)$. In addition, if $\delta^G_\alpha = 0$ or $\delta^G_\beta = 0$, then the adversary receives the sum $\sum_{i\in\delta^G\setminus\{\alpha,\beta\}} a_i b^G_{ij}$, for any $j \in \{1,\ldots,n\}$.
- If G is corrupt the adversary receives $(b^i_\alpha, b^i_\beta, b^i_1,\ldots,b^i_m)$ for every $i \in \mathsf{H}$.

Adversary's inputs: A corrupt guide inputs $\mathbf{a}$, $\mathbf{b}^G$, $\{o^G_{ij}\}_{i\in\{\alpha,\beta,1,\ldots,m\},j\in\{0,\ldots,n\}}$, δ^G and an input bit flag.

Outputs: We split into two cases.

- **Honest guide.** For each $i \in \{\alpha,\beta,1,\ldots,m\}$ such that $\delta^G_i = 1$, let $b^G_i := b^G_{i,0}$, and otherwise let $b^G_i = 0$. $\mathcal{F}_{\mathsf{gdtc}}$ returns $(\mathbf{a}, \delta^G, a_0 \cdot b^G_\alpha b^G_\beta + \sum_{i\in\delta^G\setminus\{\alpha,\beta\}} a_i \cdot b^G_i)$ to all parties.
- **Corrupt guide.** Let $g(x)$ be the polynomial obtained by interpolating $\{\sum_{i\in\delta^G\setminus\{\alpha,\beta\}} a_i b^G_{ij}\}_{j\in\mathsf{H}}$. The parties output "$G$ is corrupt" if either (1) $\mathsf{flag} = 1$ or (2) $a_0 = 0$, or (3) there exists $j \in \mathsf{H}$ and $i \in I_j$ such that $\mathsf{open}_{\mathsf{crs}}(C_{ij}, o^G_{ij}) \neq b^G_{ij}$, or (4) $g(x)$ is of degree more than t. Otherwise, we split into cases.
  - If $\delta^G_\alpha \wedge \delta^G_\beta = 1$, let $f^\alpha(x)$ be the polynomial obtained by interpolating $\{b^G_{\alpha,j}\}_{j\in\mathsf{H}}$, and let $f^\beta(x)$ be the polynomial obtained by interpolating $\{b^G_{\beta,j}\}_{j\in\mathsf{H}}$. If $f^\alpha(x)$ or $f^\beta(x)$ are of degree more than t then return "G is corrupt" to all parties. Otherwise, $\mathcal{F}_{\mathsf{gdtc}}$ returns $(\mathbf{a}, \delta^G, a_0 \cdot f^\alpha(0) \cdot f^\beta(0) + g(0))$ to all parties.
  - Otherwise $\delta^G_\alpha \wedge \delta^G_\beta = 0$, and $\mathcal{F}_{\mathsf{gdtc}}$ returns $(\mathbf{a}, \delta^G, g(0))$ to all parties.

Figure 14: Functionality $\mathcal{F}_{\mathsf{gdtc}}$

We design protocol gdtc (Fig. 15) to the realize $\mathcal{F}_{\mathsf{gdtc}}$. We reduce the task of degree-2 computation to three instances of glinear via Beaver's trick that demonstrates how given sharing of a triple $\gamma, \rho, \eta = \gamma\rho$ and sharing of b_α, b_β, the sharing of the product $b_\alpha b_\beta$ can be computed using a linear function $\rho u + \gamma v + \eta + uv$, where $u = b_\alpha - \gamma$, $v = b_\alpha - \rho$ are public constants. So in gdtc, G shares a triple using an instance of tss and using this Beaver's trick makes the rest of the computation linear. The offline/online of glinear is run in the offline/online of gdtc. The security statement appears below and the proof in Appendix B.6.

Protocol $\mathsf{gdtc} = (\mathsf{gdtc.off}, \mathsf{gdtc.on})$

All parties have access to $\mathcal{F}_{\mathsf{crs}}$.

Primitives: Guided Linear function evaluation $\mathsf{glinear}$; TSS tss.

- $\mathsf{gdtc.off}$**(R1)**
  - *(Sharing Phase of* tss*)* G shares a random triple $\langle\langle\gamma\rangle\rangle, \langle\langle\rho\rangle\rangle, \langle\langle\eta\rangle\rangle$ so that $\eta = \gamma\rho$ using an instance of tss (c.f. Notation 5).
  - *(*$\mathsf{glinear.off}$ *calls)* The parties execute $\mathsf{glinear.off}^{\mathsf{out}}$, $\mathsf{glinear.off}^{u}$ and $\mathsf{glinear.off}^{v}$, led by G .
  - *(Local computation)* We denote the output of P_i in the sharing phase of tss by $(\mathbf{C}^{\gamma}, \mathbf{O}_i^{\gamma}), (\mathbf{C}^{\rho}, \mathbf{O}_i^{\rho}), (\mathbf{C}^{\eta}, \mathbf{O}_i^{\eta})$ and happy_i.
- **Inputs:** This is exactly same as specified under **Honest Parties' input** in the description of $\mathcal{F}_{\mathsf{gdtc}}$.
- $\mathsf{gdtc.on}$**(R2)** The parties do the following.
  - G broadcasts $\mathbf{a}$ and δ^G.
  - *(Verification phase of* tss*)* The parties execute the verification phase of tss, where P_i sets $\mathsf{flag}_i := 1$ if $\mathsf{happy}_i = 0$ (computed during the sharing phase of tss), and otherwise $\mathsf{flag}_i := 0$.
  - *(*$\mathsf{glinear}^{\mathsf{out}}$ *call)* For $i \in \{\alpha, \beta, 1, \ldots, m\}$, let $\mathbf{C}^i := (C_{ij})_{j\in\{0,\ldots,n\}}$ and $\mathbf{O}^i := (o_{ij}^G)_{j\in\{0,\ldots,n\}}$, and think of $(\mathbf{C}^i, \mathbf{O}^i)$ as a main-row of a sharing $\lfloor w^i \rceil$. The parties compute $\sum_{i=1}^m a_i \lfloor w^i \rceil + a_0 \delta_\alpha^G \delta_\beta^G \cdot ((b_{\beta 0}^G - \rho)\lfloor\gamma\rceil + (b_{\alpha 0}^G - \gamma)\lfloor\rho\rceil + \lfloor\eta\rceil + (b_{\beta 0}^G - \rho)\cdot(b_{\alpha 0}^G - \gamma)\langle 1\rangle)$, guided by G with input $\delta^{\mathsf{out}} = (\delta_1^G, \ldots, \delta_m^G, 1, 1, 1, 1)$ and every P_i with input $[\delta_1^i, \ldots, \delta_m^i, \mathsf{happy}_i, \mathsf{happy}_i, \mathsf{happy}_i, 1]$.[a]
  - *(*$\mathsf{glinear}^{u}$ *and* $\mathsf{glinear}^{v}$ *calls)* The parties compute $\lfloor w^\alpha \rceil - \lfloor\gamma\rceil$ in $\mathsf{glinear}^u$, and $\lfloor w^\beta \rceil - \lfloor\rho\rceil$ in $\mathsf{glinear}^v$, guided by G, with respective inputs $\delta^u = (\delta_\alpha^G, 1), \delta^v = (\delta_\beta^G, 1)$ and every P_i with respective inputs $(\delta_\alpha^i, \mathsf{happy}_i)$ and $(\delta_\beta^i, \mathsf{happy}_i)$.
  - *(Local computation)* The parties output "G is corrupt" if:
    1. the output of the verification phase of the tss call is "D is corrupt";
    2. if $a_0 = 0$;
    3. the output of $\mathsf{glinear}^{\mathsf{out}}$, $\mathsf{glinear}^u$ or $\mathsf{glinear}^v$ is "G is corrupt"; otherwise, denote the output by $(\mathbf{a}^{\mathsf{out}}, \delta^{\mathsf{out}}, \mathsf{out})$, $(\mathbf{a}^u, \delta^u, u)$ and $(\mathbf{a}^v, \delta^v, v)$, respectively,
    4. if one of the following holds: (1) $\mathbf{a}^{\mathsf{out}} \neq (a_1, \ldots, a_m, \delta_\alpha^G\delta_\beta^G a_0 v, \delta_\alpha^G\delta_\beta^G a_0 u, \delta_\alpha^G\delta_\beta^G a_0, \delta_\alpha^G\delta_\beta^G a_0 uv)$, (2) $\mathbf{a}^u \neq (1, -1)$, (3) $\mathbf{a}^v \neq (1, -1)$, (4) $\delta^{\mathsf{out}} \neq (\delta_1^G, \ldots, \delta_m^G, 1, 1, 1, 1)$, (5) $\delta^u \neq (\delta_\alpha^G, 1)$, or (6) $\delta^v \neq (\delta_\beta^G, 1)$.

    Otherwise, parties output $(\mathbf{a}, \delta^G, \mathsf{out})$.

[a] Here $\langle 1\rangle$ is a main row of a default sharing of 1, denoted $(C_0', \ldots, C_n')$ and $(o_0', \ldots, o_n')$, which is locally computed by the parties by setting $(C_i', o_i') = \mathsf{commit}_{\mathsf{crs}}(1; \vec{0})$, for each $i \in \{0, \ldots, n\}$, where $\vec{0}$ is the all-zero string.

Figure 15: Protocol $\mathsf{gdtc} = (\mathsf{gdtc.off}, \mathsf{gdtc.on})$

Lemma 3.16. *Let κ be a security parameter, let n be the number of parties, let $t < n/2$, and let $\mathbb{F}$ be a field of characteristic 2. Protocol* gdtc *is a UC-secure implementation of $\mathcal{F}_{\mathsf{gdtc}}$ with everlasting security, against a static, active, rushing adversary corrupting up to t parties. The complexity of the protocol is* $\mathrm{poly}(m, n, \log|\mathbb{F}|, \kappa)$.

3.9 General Degree-2 Computation

Here we present our degree-2 computation protocol. We start with the functionality $\mathcal{F}_{\mathsf{dtc}}$.[10]

Functionality $\mathcal{F}_{\mathsf{dtc}}$

The functionality receives the set of corrupt parties C, and has access to $\mathcal{F}_{\mathsf{crs}}$.

Inputs. Party P_i holds a vector of ℓ_i inputs to the functionality, denoted $(w^{L_{i-1}+1}, \ldots, w^{L_{i-1}+\ell_i})$, where $L_{i-1} = \sum_{j=1}^{i-1} \ell_j$ and $L_0 = 0$.

Outputs. All parties receive $y(\mathbf{w}) = (y^1(\mathbf{w}), \ldots, y^m(\mathbf{w}))$, where each y^k is of the form $y^k(x^\alpha, x^\beta, x_1, \ldots, x_n) = x^\alpha x^\beta + x^1 + \ldots + x^n$, where each of $x^\alpha, x^\beta, x^1, \ldots, x^n$ is either equal to some w_i, or a constant specified by the functionality. That is, there exists public mapping $\mathsf{map}^k(\mathbf{w})$ that takes $\mathbf{w}$ and maps the correct w to each x term in y^k expression.

Figure 16: Functionality $\mathcal{F}_{\mathsf{dtc}}$

To simplify discussion, we consider the computation for $y = x^\alpha x^\beta + x^1 + \ldots + x^n$. For simplicity, we denote the party holding x^α, x^β and x^i by P_α, P_β and P_i. The high-level idea is to (1) let each party share all of its inputs in the first round via vss sharing phase, and (2) let the parties compute the i-th share of $x^\alpha \cdot x^\beta + x^1 + \ldots + x^n$ via gdtc, guided by P_i. Step (1) and input-independent round of step (2) are executed in the first round, while step (2) is executed in the second round over the tentative shares.

The above optimistic 2-round construction works perfectly fine for a semi-honest adversary. A malicious adversary challenges us with issues in each of the above two steps. First, the corrupt dealers might send corrupt shares of their inputs to P_i in the first round. Second, a corrupt guide P_i might abort the execution of gdtc. The first issue throws many complications arising out of operating on unsettled/tentative (non-final) shares. The second issue needs us to find a way to recover *all* shares. Notice that, the shares of $x^\alpha \cdot x^\beta + x^1 + \ldots + x^n$ correspond to a degree-$2t$ polynomial, and so we must recover *all* the shares in order to recover the correct value $x^\alpha \cdot x^\beta + x^1 + \ldots + x^n$. Therefore, we add another round which is used for *correcting* the shares of $x^\alpha \cdot x^\beta + x^1 + \ldots + x^n$, and *recovering* the shares of all parties who aborted the computation. More concretely, the protocol consists of the following three phases, where each phase requires one round.

Sharing Phase: A sharing phase, in which the parties share their inputs using the sharing phase of the vss protocol. Observe that, the output of this phase is only tentative shares. Along with the sharing, every dealer for each of its input sharing initiates a proof in zero-knowledge, via sif, that it holds openings to the corresponding vss commitments defining a strong double t-sharing. By the binding property of the commitment scheme, this means that whenever the zero-knowledge proof succeeded, any opening of the corrupt parties will be consistent with the honest parties' shares. The exact single-input function that we use is given below.

Let $\mathcal{F}^{\mathsf{sif}}_{\langle\!\langle\cdot\rangle\!\rangle}$ be the single-input functionality that takes commitments $\mathbf{C} = \{C_{ij}\}_{i,j\in\{0,\ldots,n\}}$ and openings $\mathbf{O} = \{o_{ij}\}_{i,j\in\{0,\ldots,n\}}$, and returns $\mathbf{C}$ and a bit out to all parties, such that out $= 1$ if and

[10]Observe that $\mathcal{F}_{\mathsf{dtc}}$ is a *fictitiously corruption aware* functionality, that is, $\mathcal{F}_{\mathsf{dtc}}$ does not depend on the set of corrupt parties C. For more information, see Appendix A.2.

only if $(\mathbf{C}, \mathbf{O})$ is a strong double t-sharing.

Partial-computation phase: In this, each P_i uses all of its available shares for the partial computation of $[\![x^\alpha]\!]_i[\![x^\beta]\!]_i + [\![x^1]\!]_i + \ldots + [\![x^n]\!]_i$ via the online round of gdtc (the offline round can be executed in parallel to the sharing phase). In parallel, the parties also execute the verification phase of all vss instances, where P_i raises a flag in each vss instance whose dealer was identified by P_i as corrupt. This means, for every corrupt dealer P_j, either P_j is disqualified by the vss functionality, or the shares that correspond to P_i become public. At the end of this phase, the parties identify those who are necessarily corrupt, and compute a set V that consists of all parties except them. In particular, a party P_i is necessarily corrupt if (1) it was disqualified by a vss or gdtc instance, or (2) according to the gdtc instance in which P_i is the guide, the shares of some x^j are not known to P_i, but P_i did not raise a flag in the corresponding vss instance.

Correction phase: We set the inputs of all parties outside V to be 0, while for all parties inside V the input values are successfully shared via vss. There are two main cases to consider.

- *(Exiting through linear functions)* If P_α or P_β are outside V, then either $x^\alpha = 0$ or $x^\beta = 0$, so it is enough to compute the linear function $x^1 + \ldots + x^n$. Since there are $n - t \geq t + 1$ honest parties, it is enough to make sure that we recover the corresponding shares of all honest parties, and that a recovered share of a corrupt party is either consistent with those shares, or is $\perp$. This can be done by letting each P_i guide the execution of the online round of glinear on the i-th rows of the inputs of parties in V (the offline round can be executed in the partial computation phase). Consequently, this glinear will be run on final shares, as opposed to tentative shares.

- *(Correcting shares)* Otherwise, both P_α and P_β are in V. In this case we need to recover the shares of *all* parties. In order to recover the shares of the parties outside V, which are known to be corrupt, we simply let the parties in V reveal all the openings that correspond to the corrupt parties' shares. This allows to recover these shares in the clear.

  For a party P_i in V, denote by y_i^k the sum in the output of the gdtc execution guided by P_i. We may need to correct y_i^k, if the following cases happen.

  1. *(Adding missing linear terms)* For $j \in \{1, \ldots, n\}$, consider an input x^j with $P_j \in \mathsf{V}$. However, according to gdtc, P_i did not receive a valid row of x^j in the sharing phase. Since both P_i and P_j are in V, the i-th row is public at the end of the corresponding vss verification. Therefore, we let all parties locally add the share to y_i^k.
  2. *(Removing unnecessary terms)* We deal with the dual of the above scenario. For $j \in \{1, \ldots, n\}$, consider an input x^j with $P_j \notin \mathsf{V}$. However, according to the gdtc, the value y_i^k consists of a share of x^j. In this case, we have to subtract this share from y_i^k, as the inputs of all parties outside V are set to 0. For this, we can let P_i open the i-th row of x^j, as received in the sharing phase. However, a corrupt P_i might refuse to open this row, so in order to make sure that the i-th row is revealed, we use the vao protocol in the following way. We require from all parties to execute a vao instance for each valid row they receive in the sharing phase. The offline round of the verification phase can be executed in round 1, while the online round of the verification phase can be executed in round 2. Now, whenever P_i is required to open the i-th row of some x^j, the parties can simply execute the opening phase of the corresponding vao instance.

3. *(Adding $x^\alpha x^\beta$)* If, according to gdtc, P_i did not receive the i-th row of x^α or x^β, then the corresponding share of $x^\alpha x^\beta$ is computed using Beaver's trick (see Fact A.8) in the following way. Already in the sharing phase, we let each P_i share a random triple via tss. Using this triple, the parties can compute the i-th share of $x^\alpha x^\beta$ by computing a linear function via glinear. Finally, the parties simply add the corresponding share to y_i^k.

The above protocol still suffers from several problems, stated below along with the way-outs.

- *(Randomization)* Given two random degree-t polynomials, $f(x)$ and $g(x)$, it is well known that the polynomial $f(x) \cdot g(x)$ is not a random polynomial. In particular, in our case, whenever both P_α and P_β are in V, the shares that the parties recover do not correspond to a random degree-$2t$ polynomial, so it may leak information about the inputs of the honest parties. In order to solve this, we let each party P_i share a random polynomial $F^{z_i}(x,y)$ in the sharing phase, which will be used for zero-sharing. At a high-level, this means that there exist coefficients $\{\lambda_j^i\}_{i,j\in\{1,\ldots,n\}}$ such that the degree-$2t$ polynomial $g(x)$, obtained by interpolating $\{\sum_{j=1}^n \lambda_i^j F^{z_j}(0,i)\}_{i\in\{1,\ldots,n\}}$ is a random degree-$2t$ polynomial conditioned on $g(0) = 0$. (See Fact A.10 for a formal statement.) Therefore, instead of computing $\llbracket x^\alpha \rrbracket_i \llbracket x^\beta \rrbracket_i + \llbracket x^1 \rrbracket_i + \ldots + \llbracket x^n \rrbracket_i$ in the gdtc execution guided by P_i, we let the parties compute $\llbracket x^\alpha \rrbracket_i \llbracket x^\beta \rrbracket_i + \llbracket x^1 \rrbracket_i + \ldots + \llbracket x^n \rrbracket_i + \lambda_i^1 \llbracket z_1 \rrbracket_i + \ldots + \lambda_i^n \llbracket z_n \rrbracket_i$, where z_i is the free coefficient of $F^{z_i}(x,y)$.

  Similarly, in order to simplify the simulation, we also require from the parties to randomize the output when exiting through linear functions. We do so by letting P_i share a random polynomial $F^{0_i}(x,y)$, whose free coefficient is 0, in the sharing phase. We also require from P_i to prove in zero-knowledge, via sif, that she holds openings to the corresponding commitments, that correspond to a strong double t-sharing of 0. Finally, in the correction phase, we let the parties compute $\sum_{j=1}^n (\langle\!\langle x^j \rangle\!\rangle_i + \langle\!\langle 0_j \rangle\!\rangle_i)$ in the glinear execution guided by P_i. The exact single-input function, we refer to here, is given below.

  Let $\mathcal{F}^{\mathsf{sif}}_{\langle\!\langle 0 \rangle\!\rangle}$ be the single-input functionality that takes commitments $\mathbf{C} = \{C_{ij}\}_{i,j\in\{0,\ldots,n\}}$ and openings $\mathbf{O} = \{o_{ij}\}_{i,j\in\{0,\ldots,n\}}$, and returns $\mathbf{C}$ and a bit out to all parties, such that out $= 1$ if and only if $(\mathbf{C},\mathbf{O})$ is a strong double t-sharing of 0.

- *(Padding)* The above protocol is not secure yet. A corrupt P_α can learn the value of x^β with probability 1. Consider a corrupt P_α that shares the value $x^\alpha = 1$ honestly in the sharing phase, but is then discovered to be outside V. In this case, in the partial-computation phase P_α learns all the shares of $1 \cdot x^\beta + x^1 + \ldots + x^n$. However, in the correction phase, the parties recover all the shares of the sum $x^1 + \ldots + x^n$. Therefore, P_α can recover both $x^\beta + x^1 + \ldots + x^n$ and $x^1 + \ldots + x^n$, and by subtracting the two values P_α learns x^β. In order to solve this problem, each party P_i uses a random pad p_i in the computation of gdtc. Therefore, the final form of computation via gdtc is as follows:

$$\left(\llbracket x^\alpha \rrbracket_i \cdot \llbracket x^\beta \rrbracket_i + \llbracket x^1 \rrbracket_i + \ldots + \llbracket x^n \rrbracket_i\right) + \left(\lambda_i^1 \llbracket z_1 \rrbracket_i + \ldots + \lambda_i^n \llbracket z_n \rrbracket_i\right) + \lfloor p_i \rceil,$$

  When both P_α, P_β are found to be in V, this random pad is revealed and subtracted from y_i^k in the correction phase.

- (Opening pads via sif evaluation) Opening the above pad sounds as simple as revealing the openings in round 3. However, due to simulation issues, as mentioned in the discussion of vao protocol, we would need to hide the openings and yet publish the shares of the pad. We use sif for the below function, which exactly does this.

  Let $\mathcal{F}^{\mathsf{sif}}_{\mathsf{co}}$ be the single-input functionality that takes a commitment C and opening o, and outputs $(C, \mathsf{open}_{\mathsf{crs}}(C, o))$.

Notation 6. *For a secret s, we define* $\mathsf{Dof}(s)$ *as the identity of the party that belongs it and shares it using a VSS instance.*

For clear notation, for degree-2 term, y^k (as specified in $\mathcal{F}_{\mathsf{dtc}}$), the element in an indicator vector δ corresponding to an element $x \in \{x^\alpha, x^\beta, x^i, z_i^k, p_i^k\}$ will be referred as δ_x.

Protocol dtc

All parties have access to $\mathcal{F}_{\mathsf{crs}}$.

Primitives: Guided degree-2 computation gdtc = (gdtc.off, gdtc.on); Guided linear function evaluation glinear = (glinear.off, glinear.on); Verify & Open vao, TSS tss, SIF sif, VSS vss.

Inputs. P_i holds a vector of ℓ_i inputs $(w^{L_{i-1}+1}, \ldots, w^{L_{i-1}+\ell_i})$, where $L_{i-1} = \sum_{j=1}^{i-1} \ell_j$ and $L_0 = 0$. The parties know $\mathsf{map}^k(\mathbf{w})$ for every x-term in the y^k expression.

(R1) The parties do the following.

- *(Input sharing)* Every party P_i shares each of its inputs $\{w^j\}_{j=L_{i-1}+1}^{L_i}$ via an instance of vss, denoted vss^{w_j}. Denote the corresponding sharing by $(\mathbf{C}^{w^j}, \mathbf{O}^{w^j})$.
  In addition, for every constant w used by the function $y(\mathbf{x})$ (see $\mathcal{F}_{\mathsf{dtc}}$) for the computation of some y^k, the parties locally compute a default $\langle\langle w \rangle\rangle$, denoted $(\mathbf{C}^w, \mathbf{O}^w)$, by setting $(C_{ij}, o_{ij}) \leftarrow \mathsf{commit}_{\mathsf{crs}}(w; \vec{0})$ for all $i, j \in \{0, \ldots, n\}$, where $\vec{0}$ is the all-zero vector.
- *(Pads and zero sharing)* For every $k \in \{1, \ldots, m\}$, P_i shares (a) a random pad p_i^k using $\mathsf{vss}^{p_i^k}$, (b) a random value z_i^k, which will be used for zero-sharing, using $\mathsf{vss}^{z_i^k}$ and (c) 0 using $\mathsf{vss}^{0_i^k}$. Denote the corresponding sharings by $(\mathbf{C}^{p_i^k}, \mathbf{O}^{p_i^k})$ and $(\mathbf{C}^{z_i^k}, \mathbf{O}^{z_i^k})$ and $(\mathbf{C}^{0_i^k}, \mathbf{O}^{0_i^k})$
- *(Triple sharing)* For every $k \in \{1, \ldots, m\}$, every P_i shares a random triple $\langle\langle \gamma_i^k \rangle\rangle, \langle\langle \rho_i^k \rangle\rangle$ and $\langle\langle \eta_i^k \rangle\rangle$, such that $\gamma_i^k \cdot \rho_i^k = \eta_i^k$, via an instance of tss, denoted tss_i^k. Denote the corresponding sharings by $(\mathbf{C}^{\gamma_i^k}, \mathbf{O}^{\gamma_i^k})$, $(\mathbf{C}^{\rho_i^k}, \mathbf{O}^{\rho_i^k})$ and $(\mathbf{C}^{\eta_i^k}, \mathbf{O}^{\eta_i^k})$.
- (sif *call)* For every $i \in \{1, \ldots, n\}$, for every value s with $\mathsf{Dof}(s) = i$, where s is either an input x^j, or one of the pads p_i^k, z_i^k, or a triple-sharing element γ_i^k, ρ_i^k or η_i^k, the parties execute sif^s, computing functionality $\mathcal{F}^{\mathsf{sif}}_{\langle\langle \cdot \rangle\rangle}$ with P_i as the dealer, with inputs $(\mathbf{C}^s, \mathbf{O}^s)$.
  In addition, for $i \in \{1, \ldots, n\}$ and every $k \in \{1, \ldots, m\}$, the parties execute an instance of sif, denoted $\mathsf{sif}^{0_i^k}$, computing functionality $\mathcal{F}^{\mathsf{sif}}_{\langle\langle 0 \rangle\rangle}$ with P_i as the dealer, with inputs $(\mathbf{C}^{0_i^k}, \mathbf{O}^{0_i^k})$.
- (vao *call)* For every value s with $\mathsf{Dof}(s) \in \{1, \ldots, n\}$, and for every $i \in \{1, \ldots, n\}$, the parties execute round 1 (input-independent) of verification phase of vao, denoted $\mathsf{vao}^{s,i}$, with P_i as D.
- (gdtc.off *calls)* For every $k \in \{1, \ldots, m\}$ and every $i \in \{1, \ldots, n\}$, the parties execute $\mathsf{gdtc.off}^{k,i}$, with P_i as G.

- *(*glinear.off *calls)* For every $k \in \{1, \ldots, m\}$ and every $i \in \{1, \ldots, n\}$ the parties execute two instances $\mathsf{glinear.off}^{k,i,u}$ and $\mathsf{glinear.off}^{k,i,v}$.
- *(Local computation)* If P_i received an invalid row from a vss or a tss execution in which P_j is the dealer, or $\mathsf{happy}_i = 0$ in any such execution, then P_i sets $\mathsf{flag}_{ij} = 1$, and turns all openings received from P_j to $\perp$. Otherwise, P_i sets $\mathsf{flag}_{ij} = 0$.

(R2) The parties do the following.

- Each party P_i broadcasts $\{\mathsf{flag}_{ij}\}_{j \in \{1,\ldots,n\}}$.
- *(VSS and TSS verification)* The parties execute the verification phase of all vss and tss instances, so that P_i has input flag_{ij} in every instance of vss and tss in which P_j is the dealer.
- *(*sif *output phase)* The parties compute the output phase of every sif instance.
- *(*vao *call)* The parties execute the online round of the verification phase of every instance of vao. In every $\mathsf{vao}^{s,i}$ where $\mathsf{Dof}(s) = j$, the dealer P_i inputs $(\mathbf{C}_i^s, \mathbf{O}_i^s)$ and $\mathsf{flag}_D := \mathsf{flag}_{ij}$, and every honest P_k inputs $\mathbf{C}_i^s$ and o_{ki}^s.
- *(*gdtc.on *calls)* For every $k \in \{1, \ldots, m\}$, let $y^k = x^\alpha x^\beta + x^1 + \ldots + x^m$. For every $i \in \{1, \ldots, n\}$ the parties partially compute Equation (4) with P_i as G with $(2n+3)$-bit indicator vector $[\overline{\mathsf{flag}}_{i\mathsf{Dof}(x^\alpha)}, \overline{\mathsf{flag}}_{i\mathsf{Dof}(x^\beta)}, \overline{\mathsf{flag}}_{i\mathsf{Dof}(x^1)} \ldots, \overline{\mathsf{flag}}_{i\mathsf{Dof}(x^n)}, \overline{\mathsf{flag}}_{i1}, \ldots, \overline{\mathsf{flag}}_{in}, \overline{\mathsf{flag}}_{i1}, \ldots, \overline{\mathsf{flag}}_{in}]$.[a] Every other party's indicator vector is determined similarly using their flags.

$$\left(\llbracket x^\alpha \rrbracket_i \cdot \llbracket x^\beta \rrbracket_i + \llbracket x^1 \rrbracket_i + \ldots + \llbracket x^n \rrbracket_i\right) + \left(\lambda_i^1 \llbracket z_1^k \rrbracket_i + \ldots + \lambda_i^n \llbracket z_n^k \rrbracket_i\right) + \lfloor p_i^k \rceil, \tag{4}$$

- *(*glinear.on *calls)* For every $k \in \{1, \ldots, m\}$ and $i \in \{1, \ldots, n\}$, the parties partially compute $\llbracket x^\alpha \rrbracket_i - \lfloor \gamma_i^k \rceil$ and $\llbracket x^\beta \rrbracket_i - \lfloor \rho_i^k \rceil$, via $\mathsf{glinear.on}^{k,i,u}$ and $\mathsf{glinear.on}^{k,i,v}$, guided by P_i with the respective indicators $[\overline{\mathsf{flag}}_{i\mathsf{Dof}(x^\alpha)}, 1]$ and $[\overline{\mathsf{flag}}_{i\mathsf{Dof}(x^\beta)}, 1]$, and every party P_r with indicators $[\overline{\mathsf{flag}}_{r\mathsf{Dof}(x^\alpha)}, \overline{\mathsf{flag}}_{ri}]$ and $[\overline{\mathsf{flag}}_{i\mathsf{Dof}(x^\beta)}, \overline{\mathsf{flag}}_{ri}]$.
- *(*glinear.off *calls for linear exit)* For every $k \in \{1, \ldots, m\}$ and $i \in \{1, \ldots, n\}$, the parties execute $\mathsf{glinear.off}_{\mathsf{lin}}^{k,i}$, with P_i as G.
- *(*glinear.off *calls for correcting shares)* For every $k \in \{1, \ldots, m\}$ and $i, j \in \{1, \ldots, n\}$ the parties execute $\mathsf{glinear.off}_{\mathsf{correct}}^{k,i,j}$, with P_j as G .
- *(*sif *for opening pads)* For every $k \in \{1, \ldots, m\}$ and $i, j \in \{1, \ldots, n\}$ the parties execute an instance of sif, denoted $\mathsf{sif}^{k,j,i}$, computing functionality $\mathcal{F}_{\mathsf{co}}^{\mathsf{sif}}$ with P_j as the dealer with inputs $(C_{j,0}^{p_i^k}, o_{j,0}^{p_i^k})$.
- *(Local computation)* Let V be the set of parties P_i such that
  1. the output of any verification phase of vss/tss where P_i enacts D is not "D is corrupt",
  2. for every s shared by P_i, the output of sif^s is $\mathbf{C}^s$ and $\mathsf{out} = 1$, and for every $k \in \{1, \ldots, m\}$, the output of $\mathsf{sif}^{0_i^k}$ is $\mathbf{C}^{0_i^k}$ and $\mathsf{out} = 1$,
  3. the verification phase of all vao instances in which P_i is D did not end with "D is corrupt",
  4. for every $k \in \{1, \ldots, m\}$, the output of gdtc_i^k is $(\mathbf{a}, \delta^G, y_i^k)$, where $\mathbf{a}$ is consistent with Eqn. (4),
  5. for every $k \in \{1, \ldots, m\}$, the output of $\mathsf{glinear}^{k,i,u}$ and of $\mathsf{glinear}^{k,i,v}$ is $(\mathbf{a}^u, \delta^u, u)$ and $(\mathbf{a}^v, \delta^v, v)$, respectively, such that $\mathbf{a}^u = \mathbf{a}^v = (1, -1)$, and
  6. for all $j \in \{1, \ldots, n\}$, flag_{ij} is consistent among all vss, tss, glinear, gdtc and vao instances.

  At this stage, each party P_i turns all openings received from a party not in V to $\perp$.

(R3) For every $k \in \{1, \ldots, m\}$, in order to compute y^k the parties do as follows.

- *(Exit through* $\mathsf{glinear}_{\mathsf{lin}}$*)* If P_α or P_β is not in V, then for every P_i in V, the parties partially compute $\langle\!\langle x^1 \rangle\!\rangle_i + \ldots + \langle\!\langle x^n \rangle\!\rangle_i + \langle\!\langle 0^k_1 \rangle\!\rangle_i + \ldots + \langle\!\langle 0^k_n \rangle\!\rangle_i$ via $\mathsf{glinear}^i_{\mathsf{lin}}$, guided by P_i with input $2n$-bit indicator vector $\delta || \delta'$ such that $\delta_j = 1$ if $\mathsf{Dof}(x^j) \in \mathsf{V}$ and $\delta'_j = 1$ if $\mathsf{Dof}(0^k_j) \in \mathsf{V}$ and 0 otherwise for every $j \in \{1, \ldots, n\}$. Every other party has the same indicator vector.

  Let V^k be the set of all parties $i \in \mathsf{V}$ such that the output of $\mathsf{glinear}^i_{\mathsf{lin}}$ is $(\mathbf{a}, \delta^G, y^k_i)$, $\mathbf{a}$ is the all-ones vector, and $\delta^G = \delta \parallel \delta'$ is of the above form (i.e., $\delta_j = 1$ if $\mathsf{Dof}(x^j) \in \mathsf{V}$ and $\delta'_j = 1$ if $\mathsf{Dof}(0^k_j) \in \mathsf{V}$ and 0 otherwise for every $j \in \{1, \ldots, n\}$). Let $f^k(x)$ be the degree-t polynomial obtained by interpolating $\{y^k_i\}_{i \in \mathsf{V}^k}$. Set $y^k := f^k(0)$.
- *(Output through* gdtc*)* Otherwise both P_α and P_β are in V. For every $i \in \{1, \ldots, n\}$, the computation of the i-th share of y^k, denoted $\bar{y}^k_i$, is done as follows.
  - *(Public opening for $P_i \notin \mathsf{V}$)* If $P_i \notin \mathsf{V}$, then for every value s such that $\mathsf{Dof}(s) \in \mathsf{V}$, every P_j in V broadcasts o^s_{ji} if $\mathsf{flag}_{j\mathsf{Dof}(s)} = 0$ (otherwise the value o^s_{ji} is public). The parties set $f^{s_i}_j := \mathsf{open}_{\mathsf{crs}}(C^s_{ji}, o^s_{ji})$, interpolate over all non-$\perp$ shares in $\{f^{s_i}_j\}_{j \in \mathsf{V}}$ to obtain a degree t polynomial $f^{s_i}(x)$ and set $s_i := f^{s_i}(0)$. Finally, the parties compute $\bar{y}^k_i := x^\alpha_i x^\beta_i + \sum_{\mathsf{Dof}(x^j) \in \mathsf{V}} x^j_i + \sum_{j \in \mathsf{V}} \lambda^j_i z^k_{ji}$.
  - *(Correcting* gdtc *outputs for $P_i \in \mathsf{V}$)* If $P_i \in \mathsf{V}$, let $(\mathbf{a}, \delta^G, y^k_i)$ be the output of gdtc^k_i.
    * *(Recovering missing linear terms)* For every $j \in \{1, \ldots, n\}$, (a) with $\delta^G_{x^j} = 0$ and $\mathsf{Dof}(x^j) \in \mathsf{V}$, the opening $o^{x^j}_{i0}$ is public, and the parties set $x^j_i := \mathsf{open}_{\mathsf{crs}}(C^{x^j}_{i0}, o^{x^j}_{i0})$; (b) with $\delta^G_{z^k_j} = 0$ and $\mathsf{Dof}(z^k_j) \in \mathsf{V}$, the opening $o^{z^k_j}_{i0}$ is public, and the parties set $z^k_{ji} := \mathsf{open}_{\mathsf{crs}}(C^{z^k_j}_{i0}, o^{z^k_j}_{i0})$.
    * *(Recovering linear terms to be eliminated)* For every x^j with $\delta^G_{x^j} = 1$ and $\mathsf{Dof}(x^j) \notin \mathsf{V}$, the parties execute the opening phase of $\mathsf{vao}^{x^j, i}$ that ended with "verification succeeded", in order to recover the value x^j_i.
    * *(Recovering pad p^k_i when $\delta^G_{x^\alpha} = \delta^G_{x^\beta} = 1$)* For each party P_j in V with $\mathsf{flag}_{ji} = 0$ the parties execute the output phase of $\mathsf{sif}^{k,j,i}$, and we denote the output by $(C_j, f^{p^k_i}_j)$. We say that the j-th share is valid if $C_j = C^{p^k_i}_{j0}$ and $f^{p^k_i}_j \neq \perp$. For each P_j in V with $\mathsf{flag}_{ji} = 1$ the opening $o^{p^k_i}_{j0}$ is public, so the parties set $f^{p^k_i}_j := \mathsf{open}_{\mathsf{crs}}(C^{p^k_i}_{j0}, o^{p^k_i}_{j0})$. We consider each such share as valid. The parties interpolate over all valid shares in $\{f^{p^k_i}_{j0}\}_{j \in \mathsf{V}}$ to obtain a degree t polynomial $f^{p^k_i}(x)$ and set $p^k_i := f^{p^k_i}(0)$.
    * *(Recovering share of $x^\alpha x^\beta$ when $\delta^G_{x^\alpha} \wedge \delta^G_{x^\beta} = 0$)* Denote the output of $\mathsf{glinear}^{k,i,u}$ by $(\mathbf{a}^u, \delta^u, u^k_i)$ and the output of $\mathsf{glinear}^{k,i,v}$ by $(\mathbf{a}^v, \delta^v, v^k_i)$. If $\delta^G_{x^\alpha} = 0$ then $o^{x^\alpha}_{i0}$ is public, and the parties set $u := u^k_i + \mathsf{open}_{\mathsf{crs}}(C^{x^\alpha}_{i0}, o^{x^\alpha}_{i0})$. Otherwise, if $\delta^G_{x^\alpha} = 1$, the parties set $u := u^k_i$. Similarly, if $\delta^G_{x^\beta} = 0$ then $o^{x^\beta}_{i0}$ is public, and the parties set $v := v^k_i + \mathsf{open}_{\mathsf{crs}}(C^{x^\beta}_{i0}, o^{x^\beta}_{i0})$. Otherwise, if $\delta^G_{x^\beta} = 1$, the parties set $v := v^k_i$. For every $j \in \mathsf{V}$ the parties partially compute $v[\![\gamma^k_i]\!]_j + u[\![\rho^k_i]\!]_j + [\![\eta^k_i]\!]_j + uv\langle 1 \rangle - [\![p^k_i]\!]_j$ via $\mathsf{glinear}^{k,i,j}_{\mathsf{correct}}$. Let V^k_i be the set of all $j \in \mathsf{V}$ for which the output is $(\mathbf{a}, \delta^G, c^k_{ij})$ where $\mathbf{a} = (v, u, 1, uv, -1)$ and $\delta^G = (1, 1, 1, 1, 1)$. Let $f^k_i(x)$ be the polynomial obtained by interpolating $\{c^k_{ij}\}_{j \in \mathsf{V}^k_i}$, and let $c^k_i := f^k_i(0)$.
  - *(Computing share of y^k)* The value $\bar{y}^k_i$ is below when $\delta^G_{x^\alpha} = \delta^G_{x^\beta} = 1$:

$$\bar{y}^k_i := y^k_i - p^k_i + \sum_{\mathsf{Dof}(x^j) \in \mathsf{V} : \delta^G_{x^j} = 0} x^j_i + \sum_{j \in \mathsf{V} : \delta^G_{z^k_j} = 0} \lambda^j_i z^k_{ji} - \Bigg(\sum_{\mathsf{Dof}(x^j) \notin \mathsf{V} : \delta^G_{x^j} = 1} x^j_i + \sum_{j \notin \mathsf{V} : \delta^G_{z_j} = 1} \lambda^j_i z^k_{ji} \Bigg).$$

and as below, when $\delta^G_{x^\alpha} \wedge \delta^G_{x^\beta} = 0$:

$$\bar{y}^k_i := y^k_i + c^k_i + \sum_{\mathsf{Dof}(x^j) \in \mathsf{V} : \delta^G_{x^j} = 0} x^j_i + \sum_{j \in \mathsf{V} : \delta^G_{z^k_j} = 0} \lambda^j_0 z^k_{ji} - \left(\sum_{\mathsf{Dof}(x^j) \notin \mathsf{V} : \delta^G_{x^j} = 1} x^j_i + \sum_{j \notin \mathsf{V} : \delta^G_{z_j} = 1} \lambda^j_i z^k_{ji} \right).$$

Finally, the parties interpolate over $\{\bar{y}^k_i\}_{i \in \{1,\ldots,n\}}$ to obtain $f^k(x)$ and set y^k to be $f^k(0)$.

[a]Here, $\overline{\mathsf{flag}}$ is $1 - \mathsf{flag}$.

Figure 17: Protocol dtc

Theorem 3.17. *Let κ be a security parameter, let n be the number of parties, let $t < n/2$, and let $\mathbb{F}$ be a field of characteristic 2. Let L_{n+1} and m be the number of inputs and the number of outputs of $\mathcal{F}_{\mathsf{dtc}}$, respectively, and assume that they are (at most) polynomial in κ. Protocol* dtc *is a UC-secure implementation of $\mathcal{F}_{\mathsf{dtc}}$ with everlasting security, against a static, active, rushing adversary corrupting up to t parties. The complexity of the protocol is* $\mathrm{poly}(m, L_{n+1}, n, \log|\mathbb{F}|, \kappa)$.

References

[1] Prabhanjan Ananth, Arka Rai Choudhuri, Aarushi Goel, and Abhishek Jain. Round-optimal secure multiparty computation with honest majority. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, pages 395–424, 2018.

[2] Benny Applebaum. *Cryptography in Constant Parallel Time*. Information Security and Cryptography. Springer, 2014.

[3] Benny Applebaum, Zvika Brakerski, and Rotem Tsabary. Degree 2 is complete for the round-complexity of malicious MPC. In *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part II*, pages 504–531, 2019.

[4] Benny Applebaum, Eliran Kachlon, and Arpita Patra. The resiliency of MPC with low interaction: The benefit of making errors (extended abstract). In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part II*, volume 12551 of *Lecture Notes in Computer Science*, pages 562–594. Springer, 2020.

[5] Benny Applebaum, Eliran Kachlon, and Arpita Patra. The round complexity of perfect MPC with active security and optimal resiliency. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1277–1284. IEEE, 2020.

[6] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, pages 483–501, 2012.

[7] Gilad Asharov and Yehuda Lindell. A full proof of the BGW protocol for perfectly secure multiparty computation. *J. Cryptology*, 30(1):58–151, 2017.

[8] Michael Backes, Aniket Kate, and Arpita Patra. Computational verifiable secret sharing revisited. In *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, pages 590–609, 2011.

[9] Christian Badertscher, Ran Canetti, Julia Hesse, Björn Tackmann, and Vassilis Zikas. Universal composition with global subroutines: Capturing global setup within plain uc. In *Theory of Cryptography Conference*, pages 1–30. Springer, 2020.

[10] Saikrishna Badrinarayanan, Rex Fernando, Aayush Jain, Dakshita Khurana, and Amit Sahai. Statistical ZAP arguments. In *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part III*, pages 642–667, 2020.

[11] Saikrishna Badrinarayanan, Aayush Jain, Nathan Manohar, and Amit Sahai. Secure MPC: laziness leads to GOD. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part III*, volume 12493 of *Lecture Notes in Computer Science*, pages 120–150. Springer, 2020.

[12] Boaz Barak, Shien Jin Ong, and Salil P. Vadhan. Derandomization in cryptography. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 299–315. Springer, 2003.

[13] D. Beaver. Efficient Multiparty Protocols Using Circuit Randomization. In J. Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15*, volume 576 of *Lecture Notes in Computer Science*, pages 420–432. Springer Verlag, 1991.

[14] Donald Beaver, Joan Feigenbaum, Joe Kilian, and Phillip Rogaway. Security with low communication overhead. In *Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings*, pages 62–76, 1990.

[15] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 503–513, 1990.

[16] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 1–10, 1988.

[17] Nir Bitansky and Omer Paneth. Zaps and non-interactive witness indistinguishability from indistinguishability obfuscation. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *Theory*

of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II, volume 9015 of *Lecture Notes in Computer Science*, pages 401–427. Springer, 2015.

[18] Manuel Blum. Coin flipping by telephone. In *Advances in Cryptology: A Report on CRYPTO 81, CRYPTO 81, IEEE Workshop on Communications Security, Santa Barbara, California, USA, August 24-26, 1981.*, pages 11–15, 1981.

[19] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.

[20] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 136–145, 2001.

[21] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.

[22] Ran Canetti, Joe Kilian, Erez Petrank, and Alon Rosen. Black-box concurrent zero-knowledge requires omega~(log n) rounds. In *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 570–579, 2001.

[23] J Lawrence Carter and Mark N Wegman. Universal classes of hash functions. *Journal of computer and system sciences*, 18(2):143–154, 1979.

[24] Richard Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 364–369, 1986.

[25] Ivan Damgård and Yuval Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, pages 378–394, 2005.

[26] Ivan Damgård, Torben P. Pedersen, and Birgit Pfitzmann. Statistical secrecy and multibit commitments. *IEEE Trans. Inf. Theory*, 44(3):1143–1151, 1998.

[27] Danny Dolev and Rüdiger Reischuk. Bounds on information exchange for byzantine agreement. *J. ACM*, 32(1):191–204, 1985.

[28] Cynthia Dwork and Moni Naor. Zaps and their applications. *SIAM J. Comput.*, 36(6):1513–1543, 2007.

[29] Uriel Feige, Joe Kilian, and Moni Naor. A minimal model for secure computation (extended abstract). In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 554–563, 1994.

[30] Paul Feldman and Silvio Micali. Byzantine agreement in constant expected time (and trusting no one). In *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*, pages 267–276, 1985.

[31] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[32] 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*, pages 580–589. ACM, 2001.

[33] Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. On 2-round secure multiparty computation. In *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, pages 178–193, 2002.

[34] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[35] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM J. Comput.*, 25(1):169–192, 1996.

[36] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, pages 25–32, 1989.

[37] S. Dov Gordon, Feng-Hao Liu, and Elaine Shi. Constant-round MPC with fairness and guarantee of output delivery. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 63–82, 2015.

[38] Vipul Goyal, Abhishek Jain, Zhengzhong Jin, and Giulio Malavolta. Statistical zaps and new oblivious transfer protocols. In *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part III*, pages 668–699, 2020.

[39] Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for noninteractive zero-knowledge. *J. ACM*, 59(3):11:1–11:35, 2012.

[40] Shai Halevi and Silvio Micali. Practical and provably-secure commitment schemes from collision-free hashing. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 201–215. Springer, 1996.

[41] Alexander Healy and Emanuele Viola. Constant-depth circuits for arithmetic in finite fields of characteristic two. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 672–683. Springer, 2006.

[42] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 294–304, 2000.

[43] Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In *Automata, Languages and Programming, 29th International Colloquium, ICALP 2002, Malaga, Spain, July 8-13, 2002, Proceedings*, pages 244–256, 2002.

[44] Jonathan Katz and Ji Sun Shin. Modeling insider attacks on group key-exchange protocols. In *Proceedings of the 12th ACM Conference on Computer and Communications Security, CCS 2005, Alexandria, VA, USA, November 7-11, 2005*, pages 180–189, 2005.

[45] Leslie Lamport and Michael Fischer. Byzantine generals and transaction commit protocols. Technical report, Technical Report 62, SRI International, 1982.

[46] Alex Lombardi and Vinod Vaikuntanathan. Multi-input correlation-intractable hash functions via shift-hiding. *IACR Cryptol. ePrint Arch.*, 2020:1378, 2020.

[47] Tal Moran, Moni Naor, and Gil Segev. An optimally fair coin toss. *J. Cryptology*, 29(3):491–513, 2016.

[48] Jörn Müller-Quade and Dominique Unruh. Long-term security and universal composability. *J. Cryptol.*, 23(4):594–671, 2010.

[49] Moni Naor. Bit commitment using pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991.

[50] Moni Naor, Rafail Ostrovsky, Ramarathnam Venkatesan, and Moti Yung. Perfect zero-knowledge arguments for *NP* using any one-way permutation. *J. Cryptol.*, 11(2):87–108, 1998.

[51] Arpita Patra and Divya Ravi. On the exact round complexity of secure three-party computation. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, pages 425–458, 2018.

[52] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, pages 73–85, 1989.

[53] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 80–91, 1982.

A Appendix: Security Model, Useful Facts and Standard Primitives

A.1 Completeness of Degree-2 Functionalities

The following completeness theorem was proved in [5, Prop. 4.5 and Thm. 5.23] (building on [3]).

Proposition A.1 ([5]). *Let $\mathcal{F}$ be an n-party functionality that can be computed by a Boolean circuit of size S and depth D and let $\mathbb{F}$ be an arbitrary extension field of the binary field $\mathbb{F}_2$. Then, the task of securely-computing $\mathcal{F}$ non-interactively reduces to the task of securely-computing the degree-2 n-party functionality f over $\mathbb{F}$ that each of its outputs is of the form*

$$x^{\alpha}x^{\beta} + \sum_{j=1}^{n} r^{j}, \tag{5}$$

where x^α and x^β are the inputs of party P_α and P_β respectively and r^j is an input of party P_j for $j \in \{1,\dots,n\}$.

The reduction preserves active statistical-security with resiliency threshold of $\lfloor \frac{n-1}{2} \rfloor$, and the complexity of the function f and the overhead of the reduction is $\mathrm{poly}(n, S, 2^D, \log|\mathbb{F}|)$. Furthermore, assuming one-way functions, one can get a similar reduction that preserves computational-security with resiliency threshold of $\lfloor \frac{n-1}{2} \rfloor$ and complexity/security-loss of $\mathrm{poly}(n, S, \log|\mathbb{F}|)$.

A.2 Security model

In this section we give a high-level description of the UC-framework, due to [20]. For more details, the reader is referred to [20]. We begin with a short description of the standard model, and then explain how the UC-framework augments it. At a high level, in the standard model, security of a protocol is argued by comparing the real-world execution to an ideal-world execution. In an ideal-world execution, the inputs of the parties are transferred to a trusted party $\mathcal{F}$ (called the *ideal functionality*) over a perfectly secure channel, the trusted party computes the function based on these inputs and sends to each party its respective output. Informally, a protocol π securely implements $\mathcal{F}$ if for any real-world adversary $\mathcal{A}$, there exists an ideal-world adversary $\mathcal{S}$ (called the *simulator*), that controls the same parties as $\mathcal{A}$, so that the global output of an execution of π with $\mathcal{A}$ (consisting of the honest parties' outputs and the output of $\mathcal{A}$), is indistinguishable from the global output of the ideal-world execution with $\mathcal{F}$ and $\mathcal{S}$ (consisting of the honest parties' outputs and the output of $\mathcal{S}$).

The UC-framework augments the standard model by adding an additional entity, called the *environment* $\mathcal{Z}$. In the real-world, $\mathcal{Z}$ arbitrarily interacts with the adversary $\mathcal{A}$, and, in addition, $\mathcal{Z}$ generates the inputs of the honest parties at the beginning of the execution, and receives their outputs at the end of the execution. In the ideal world, *the same* environment $\mathcal{Z}$ arbitrarily interacts with the simulator $\mathcal{S}$, and, in addition, $\mathcal{Z}$ communicates with dummy parties, that receive the honest parties' inputs from $\mathcal{Z}$ and immediately transfer them to $\mathcal{F}$, and later receive the honest parties' outputs from $\mathcal{F}$ and immediately transfer them to $\mathcal{Z}$. In both worlds, at the end of the execution the environment $\mathcal{Z}$ outputs a single bit.

For a security parameter κ and input ζ to $\mathcal{Z}$, we denote the distribution of the output bit of $\mathcal{Z}(\zeta)$ in a real-world execution of π with adversary $\mathcal{A}$ by $\mathrm{REAL}_{\pi,\mathcal{Z}(\zeta),\mathcal{A}}(\kappa)$. We denote the distribution of the output bit of $\mathcal{Z}(\zeta)$ in an ideal-world execution with ideal-functionality $\mathcal{F}$, simulator $\mathcal{S}$ by $\mathrm{IDEAL}_{\mathcal{F},\mathcal{Z}(\zeta),\mathcal{S}}(\kappa)$. Intuitively, we say that a protocol π *UC-emulates* an ideal-functionality $\mathcal{F}$ if for every real-world polynomial-time adversary $\mathcal{A}$ there exists an ideal-world polynomial-time simulator $\mathcal{S}$, so that for any environment $\mathcal{Z}$ and any input ζ to $\mathcal{Z}$, it holds that$\{\mathrm{REAL}_{\pi,\mathcal{Z}(\zeta),\mathcal{A}}(\kappa)\}_\kappa$ is computationally indistinguishable from $\{\mathrm{IDEAL}_{\mathcal{F},\mathcal{Z}(\zeta),\mathcal{S}}(\kappa)\}_\kappa$.

The dummy-adversary. Since the above definition quantifies over all environments, we can merge the adversary $\mathcal{A}$ with the environment $\mathcal{Z}$. That is, it is enough to require that the simulator $\mathcal{S}$ will be able to simulate, for any environment $\mathcal{Z}$, the *dummy adversary* that simply delivers messages from $\mathcal{Z}$ to the protocol machines. For more information, see [20].

The hybrid model. The UC-framework is appealing because it has strong composability properties. Consider a protocol ρ that securely implements an ideal functionality $\mathcal{G}$ in the $\mathcal{F}$-hybrid model (which means that the parties in ρ have access to an ideal functionality $\mathcal{F}$), and let π be a protocol that securely implements $\mathcal{F}$. The composition theorem guarantees that if we replace in ρ each call to $\mathcal{F}$ with an execution of π we obtain a secure protocol. This means that it is enough to prove the security of a protocol in the hybrid model, where the analysis is much simpler.

Corruption-aware functionalities. Throughout, we assume that our functionalities are *corruption-aware*, which means that they might depend on the identities of the corrupt parties C. The notion of corruption aware functionalities was first introduced by [20], and the reader is referred to [20] for more information (see also [7, Section 6.2]).

We mention that our final functionality for degree-2 computation, $\mathcal{F}_{\mathsf{dtc}}$, is a *fictitiously corruption aware* functionality, which means that $\mathcal{F}_{\mathsf{dtc}}$ receives the set of corrupt parties C, but does not depend on it. It is known (see, e.g., [7, Section 6.2]) that if a protocol securely computes a fictitiously corruption aware functionality, then it also securely computes it in the standard model, where the functionality does not receive the set of corrupt parties C. Therefore, our final functionality $\mathcal{F}_{\mathsf{dtc}}$ is also secure in the standard model.

Everlasting security. We also consider a hybrid version of statistical and computational security. Intuitively, we require that an environment which is polynomially-bounded *during* the execution and is allowed to be unbounded *after* the execution, cannot distinguish the real-world from the ideal-world. We refer to this notion as everlasting security. Observe that this security notion lies between computational-security (where we consider only environments that are *always* polynomially-bounded) and statistical-security (where we also consider environments that are unbounded during the execution of the protocol).

The idea of everlasting security was formalized in the UC-framework by [48]. In a nutshell, instead of considering environments that are unbounded after the execution, it is enough to consider only environments that are always polynomially-bounded, but are not limited to a single bit output. In particular, such environments can output their whole view. Using the same notation as before, $\text{REAL}_{\pi,\mathcal{Z}(\zeta),\mathcal{A}}(\kappa)$ and $\text{IDEAL}_{\mathcal{F},\mathcal{Z}(\zeta),\mathcal{S}}(\kappa)$, to denote the output distribution of $\mathcal{Z}$ in the real-world and in the ideal-world (where now the output may contain more than one bit), we say that a protocol π UC-emulates an ideal functionality $\mathcal{F}$ with everlasting security, if for every polynomial-time real-world adversary $\mathcal{A}$ there exists an ideal-world polynomial-time simulator $\mathcal{S}$ such that for any polynomial-time environment $\mathcal{Z}$ and any input ζ to $\mathcal{Z}$, the random variables $\{\text{REAL}_{\pi,\mathcal{Z}(\zeta),\mathcal{A}}(\kappa)\}_{\kappa}$, and $\{\text{IDEAL}_{\mathcal{F},\mathcal{Z}(\zeta),\mathcal{S}}(\kappa)\}_{\kappa}$ are *statistically* indistinguishable. Therefore, in general, in order to prove security it is enough to show that the view of the environment in the real-world is statistically-close to the view of the environment in the ideal-world.

We mention that the composition theorems of UC-security hold for protocols with everlasting security (i.e., the composition of two protocols with everlasting security results in a protocol with everlasting security). For a formal definition and statement of the composition theorem, the reader is referred to [48].

Global setup. In order to obtain protocols with everlasting security, we use non-interactive commitments which are statistically-hiding. It is well known that such commitments cannot be implemented in the plain model, and that some global setup is required . In our case, the global setup is a common reference string (CRS). Throughout the paper, we assume that all functionalities and parties have access to the same global functionality $\mathcal{F}_{\mathsf{crs}}$, that, upon receiving a query, returns the common reference string. We mention that, since all our protocols are static systems, where all identities and connectivity is fixed beforehand, the composition theorems in this model follow immediately from the composition theorems guaranteed by UC-security, even when we consider everlasting security (see, e.g., [9, Section 1]).

Semi-rushing adversary. We believe that there are natural network settings in which semi-rushing adversary suffices to capture an adversarial capability and full rushing is an overkill, especially when the number of parties is large. Imagine that a message arrives with probability 0.5 within 1 “hour” and with probability 0.5 within 2 “hours”. Accordingly, each round of messages can begin at an even hour and if messages do not arrive after 2 hours, we replace them with some default value. Now, even if the adversary controls the delay of her own message, she cannot wait to see all the honest parties messages before sending her message, because she is likely to reach a time-out. More generally, assume that the delay of each message m is a random variable r that takes values in $\{1, \cdots, T\}$ for some constant T. Let us further assume that if a message does not reach after T time units the receiving party states ”TIME-OUT” and reads the message as $\perp$. Let $\epsilon = \Pr[$that all the delays of the honest parties are smaller than T$]$. Then, even if the adversary can control his own delays, semi-rushing security prevents any cheating except with probability ϵR, where R denotes the number of rounds. This may be sufficiently good, especially when R is small (e.g., constant in our case) and ϵ is tiny (e.g., negligible due to a large number of parties).

A.3 On the $\mathbb{R}$-Intractable Hash Functions in Protocol tss

In this section we discuss the $\mathbb{R}$-intractable hash function used in protocol tss. We begin by defining $\mathbb{R}$ and showing that it can be represented by a circuit with a simple structure. Then, we show that a the hash function can be modelled as a random oracle. That is, we show that a random oracle is $\mathbb{R}$-intractable. Throughout, we assume that the commitments and the opening correspond to a NICOM with security parameter κ.[11] We also assume that the hash-function h_z takes commitments $\mathbf{C} := \{\mathbf{C}^{v,k}\}_{v\in\{a,b,c\},k\in\{0,\ldots,2n\}}$ and returns a single field element α.

The circuit $\mathbb{R}^{h_z}_{\kappa}$. Let $\mathbb{R}^{h_z}_{\kappa}$ be the circuit that takes as an input commitments $\mathbf{C} := \{\mathbf{C}^{v,k}\}_{v\in\{a,b,c\},k\in\{0,\ldots,2n\}}$, and openings $\mathbf{O}_{\mathsf{H}} := \{\mathbf{O}^{v,k}_i\}_{v\in\{a,b,c\},k\in\{0,\ldots,2n\},i\in\mathsf{H}}$. If there exist $v \in \{a, b, c\}$ and $k \in \{0, \ldots, 2n\}$ such that $(\varnothing, \mathbf{C}^{v,k}, \varnothing, \mathbf{O}^{v,k}_{\mathsf{H}})$ is not a weak double t-sharing with sharing polynomial $F^{v,k}(x, y)$ then $\mathbb{R}$ outputs 1. Otherwise, the circuit computes $\alpha := h(\mathbf{C})$, and outputs 0 if and only if either $\alpha = 0$, or

$$F^{a,0}(0,0)\cdot F^{b,0}(0,0) \neq F^{c,0}(0,0) \quad \text{and} \quad \left(\sum_{k=0}^{2n}\alpha^k F^{a,k}(0,0)\right)\cdot\left(\sum_{k=0}^{2n}\alpha^k F^{b,k}(0,0)\right) = \left(\sum_{k=0}^{2n}\alpha^k F^{c,k}(0,0)\right).$$

Observe that, using known reduction techniques (see, e.g., [33, Theorem 1]) $\mathbb{R}^{h_z}_{\kappa}$ can be implemented by circuit with the following simple properties: (1) The first layer of the circuit consists of h_z gates, AND gates, OR gates, and NOT gates, (2) the second layer consists of non-equal gates (note that each gate can be implemented as $(\neg x \vee \neg y) \wedge (x \vee y)$), and (3) the rest of the layers consists of an OR-tree whose inputs are the outputs of the second layer.

[11] If the commitment scheme depends on a CRS, we extend the definition of $\mathbb{R}$-intractability in a natural way, by considering a distribution over circuits that receive a random CRS as part of their inputs, and we assume that the CRS is also given to the adversary.

Random oracle as $\mathbb{R}$-intractable hash function. We continue by showing that a random oracle $\mathcal{O}$ is a $\mathbb{R}$-intractable hash function. We begin by analysing the following game, played with an adversary $\mathcal{A}$. First, $\mathcal{A}$ broadcasts commitments $\mathbf{C} := \{\mathbf{C}^{v,k}\}_{v\in\{a,b,c\},k\in\{0,\ldots,2n\}}$. Then a random field element α is sampled by a trusted party and given to $\mathcal{A}$. Finally, $\mathcal{A}$ broadcasts $\mathbf{O}_{\mathsf{H}} := \{\mathbf{O}_i^{v,k}\}_{v\in\{a,b,c\},k\in\{0,\ldots,2n\},i\in\mathsf{H}}$. We say that $\mathcal{A}$ wins if for all $v \in \{a,b,c\}$ and $k \in \{0,\ldots,2n\}$ it holds that $(\varnothing, \mathbf{C}^{v,k}, \varnothing, \mathbf{O}_{\mathsf{H}}^{v,k})$ is a weak double t-sharing with sharing polynomial $F^{v,k}(x,y)$, and either $\alpha = 0$ or

$$F^{a,0}(0,0)\cdot F^{b,0}(0,0) \neq F^{c,0}(0,0) \quad\text{and}\quad \left(\sum_{k=0}^{2n}\alpha^k F^{a,k}(0,0)\right)\cdot\left(\sum_{k=0}^{2n}\alpha^k F^{b,k}(0,0)\right) = \left(\sum_{k=0}^{2n}\alpha^k F^{c,k}(0,0)\right).$$

Lemma A.2. *Every PPT wins the game with probability at most* $\mathsf{negl}(\kappa)$*, where* κ *is the security parameter of the commitments.*

It is not hard to see that, when the underlying commitment scheme is perfectly-biding, then the probability of every adversary to win is at most $2n/|\mathbb{F}| = 2^{-\Omega(\kappa)}$. We omit the proof of Lemma A.2 for the case of computationally-biding commitment scheme.

We continue by showing that if there exists a PPT adversary that violates the $\mathbb{R}$-intractability property of the random oracle, then there exists a PPT adversary that wins the above game with non-negligible probability. This will imply that no adversary can violate the $\mathbb{R}$-intractability property of the random oracle.

Consider a PPT adversary $\mathcal{A}$ that violates $\mathbb{R}$-intractability with probability $p(\kappa)$ for some polynomial $p(\cdot)$. We may assume without loss of generality that (1) the number of queries that $\mathcal{A}$ makes to the random oracle is always the same, and we denote it by $T = \mathrm{poly}(\kappa)$, (2) we denote the queries by $Q_1,\ldots,Q_T$, and always assume that the commitments $\mathbf{C}$ in the last broadcast of $\mathcal{A}$ is in the set $\{Q_1,\ldots,Q_T\}$, and (3) all queries are distinct.

Observe that there exists an index $i^* \in \{1,\ldots,T\}$ such that $\Pr[\mathcal{A}^{\mathcal{O}} = (\mathbf{C},\mathbf{O}_{\mathsf{H}})$ s.t. $\mathbb{R}(\mathbf{C},\mathbf{O}_{\mathsf{H}}) = 0 \wedge \mathbf{C} = Q_{i^*}] \geq 1/p(\kappa)T$. Consider an adversary $\mathcal{B}$ that simulates $\mathcal{A}$, and at the end of the simulation computes the output $(\mathbf{C},\mathbf{O}_{\mathsf{H}})$ of $\mathcal{A}$ and outputs $(Q_{i^*},\mathbf{O}_{\mathsf{H}})$ instead. Then clearly, $\Pr[\mathcal{B}^{\mathcal{O}} = (\mathbf{C},\mathbf{O}_{\mathsf{H}})$ s.t. $\mathbb{R}(\mathbf{C},\mathbf{O}_{\mathsf{H}}) = 0] \geq 1/p(\kappa)T$. We now define an adversary $\mathcal{B}'$ against our game. $\mathcal{B}'$ simulates an execution of $\mathcal{B}$ by giving a random field element for each query (recall that the queries are distinct) up to query i^*, where $\mathcal{B}'$ obtains Q_{i^*}. At this step, $\mathcal{B}'$ broadcasts Q_{i^*} in the game, obtains α, and returns α to $\mathcal{B}$ as the answer from the oracle. $\mathcal{B}'$ continues to simulate $\mathcal{B}$ by giving a random field element for each query and obtains the output $(\mathbf{C},\mathbf{O}_{\mathsf{H}})$ of $\mathcal{B}$. $\mathcal{B}'$ broadcasts $\mathbf{O}_{\mathsf{H}}$ in the game and terminates.

It is not hard to see that $\mathcal{B}'$ wins the game whenever $\mathcal{B}$ provides $(\mathbf{C},\mathbf{O}_{\mathsf{H}})$ that violate the intractability, and since $\mathcal{B}$ is perfectly simulated, this happens with probability at least $1/p(\kappa)T \geq 1/\mathrm{poly}(\kappa)$. This concludes the proof.

□

A.4 Polynomials: Useful Facts

Let $n > 0$ be a natural number, and let $t < n$. In the following, unless stated otherwise, $\mathbb{F}$ is a field of size greater than n. We start with basic facts about polynomials (see., e.g., [7]).

Fact A.3. *Let $s \in \mathbb{F}$ and let $p(x)$ be a random degree-d polynomial, conditioned on $p(0) = s$. Let $\alpha_1, \ldots, \alpha_d \in \mathbb{F}$ be distinct nonzero field elements. Then the random variables*

$$p(\alpha_1), \ldots, p(\alpha_d)$$

are uniformly distributed over $\mathbb{F}^d$.

Fact A.4. *Let $K \subseteq \{1, \ldots, n\}$ be a set of size at least $t + 1$, and let $\{f_k(x)\}_{k \in K}$ be a set of degree-t polynomials. If for every $i, j \in K$ it holds that $f_i(j) = f_j(i)$ then there exists a unique symmetric bivariate polynomials $F(x, y)$ of degree at most t in each variable such that $f_k(x) = F(x, k) = F(k, x)$ for every $k \in K$.*

We denote by $\mathcal{P}^{s,t}$ the uniform distribution over symmetric bivariate polynomials $F(x, y)$ of degree at most t in each variable, conditioned on $F(0, 0) = s$.

Fact A.5. *For any $s, s' \in \mathbb{F}$ and $\mathsf{C} \subseteq \{1, \ldots, n\}$ of size at most t, it holds that*

$$\{(i, F(x, i))\}_{i \in \mathsf{C}} \equiv \{(i, F'(x, i))\}_{i \in \mathsf{C}},$$

where F is sampled from $\mathcal{P}^{s,t}$ and F' is sampled from $\mathcal{P}^{s',t}$.

Fact A.6. *For every $s, s' \in \mathbb{F}$, $\mathsf{C} \subseteq \{1, \ldots, n\}$ of size at most t, and two sets of degree-t polynomials $\{f_i(x)\}_{i \in \mathsf{C}}$ and $\{f'_i(x)\}_{i \in \mathsf{C}}$ such that $f_i(j) = f_j(i)$ and $f'_i(j) = f'_j(i)$ for all $i, j \in \mathsf{C}$, it holds that the support size of $F(x, y)$ is equal to the support size of $F'(x, y)$, where $F(x, y)$ is sampled from $\mathcal{P}^{s,t}$ conditioned on $F(x, i) = f_i(x)$ for every $i \in \mathsf{C}$, and $F'(x, y)$ is sampled from $\mathcal{P}^{s',t}$ conditioned on $F'(x, i) = f'_i(x)$ for every $i \in \mathsf{C}$.*

The following fact, for which we provide a proof, will be used in the security proof of protocol tss.

Fact A.7. *Let $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$ be distinct non-zero elements and let $\mathsf{C} \subseteq \{1, \ldots, n\}$ be a set of size at most t. Let $P(x)$ be a degree-$2n$ polynomial, and let $\bar{F}^0(x, y)$ and $\bar{F}^{n+1}(x, y), \ldots, \bar{F}^{2n}(x, y)$ be symmetric bivariate polynomials of degree at most t in each variable, such that $\bar{F}^k(0, 0) = P^k$, where P^k is the k-th coefficient of $P(x)$. For $k = 0$ and $k = n + 1, \ldots, 2n$, and $i \in \mathsf{C}$ let $f_i^k(x) := \bar{F}^k(x, i)$ and let $\{\bar{f}_i^k(x)\}_{k \in \{1, \ldots, n\}, i \in \mathsf{C}}$ be the set of degree-t polynomials, such that $f_i^k(j) = f_j^k(i)$ for all $i, j \in \mathsf{C}$.*

Let $F^0(x, y), \ldots, F^{2n}(x, y)$ be uniformly distributed symmetric bivariate polynomials of degree at most t in each variable, conditioned on (1) $F^0(x, y) = \bar{F}^0(x, y)$ and $F^k(x, y) = \bar{F}^k(x, y)$ for every $k \in \{n + 1, \ldots, 2n\}$, (2) $F^k(x, i) = f_i^k(x)$ for all $k \in \{1, \ldots, n\}$ and $i \in \mathsf{C}$, and (3) $F^k(0, 0) = P^k$, where P^k is the k-th coefficient of $P(x)$, for every $k \in \{1, \ldots, n\}$.

Then the random variables $\{G^j(x, y)\}_{j \in \{1, \ldots, n\}}$, where

$$G^j(x, y) := \sum_{k=0}^{2n} \alpha_j^k \cdot F^k(x, y),$$

are uniformly distributed symmetric bivariate polynomials of degree at most t in each variable, conditioned on

$$G^j(x, i) = \sum_{k=0}^{2n} \alpha_j^k \cdot f_i^k(x) \quad \textit{and} \quad G^j(0, 0) = P(\alpha_i),$$

for all $j \in \{1, \ldots, n\}$ and $i \in \mathsf{C}$.

Proof. Fix any symmetric bivariate polynomials $H^1(x,y),\ldots,H^n(x,y)$. It is not hard to see that if $H^j(x,i) \neq \sum_{k=0}^{2n} \alpha_j^k f_i^k(x)$ or $H^j(0,0) \neq P(\alpha_j)$, for some $j \in \{1,\ldots,n\}$ and $i \in \mathsf{C}$, then the probability that $(G^1,\ldots,G^n)$ is equal to $(H^1,\ldots,H^n)$ is 0. Therefore, we assume that $H^1(x,y),\ldots,H^n(x,y)$ are in the support, i.e., $H^j(0,0) = P(\alpha_j)$ and $H^j(x,i) = \sum_{k=0}^{2n} \alpha_j^k f_i^k(x)$ for all $j \in \{1,\ldots,n\}$ and $i \in \mathsf{C}$. Observe that $(G^1,\ldots,G^n) = (H^1,\ldots,H^n)$ if and only if

$$\begin{bmatrix} \alpha_1^0 & \ldots & \alpha_1^{2n} \\ \vdots & \ddots & \vdots \\ \alpha_n^0 & \ldots & \alpha_n^{2n} \end{bmatrix} \begin{bmatrix} F^0(x,y) \\ \vdots \\ F^{2n}(x,y) \end{bmatrix} = \begin{bmatrix} H^1(x,y) \\ \vdots \\ H^n(x,y) \end{bmatrix},$$

which occurs if and only if

$$\begin{bmatrix} F^1(x,y) \\ \vdots \\ F^n(x,y) \end{bmatrix} = V^{-1} \begin{bmatrix} H^1(x,y) - F^0(x,y) - \alpha_1^{n+1}F^{n+1}(x,y) - \ldots - \alpha_1^{2n}F^{2n}(x,y) \\ \vdots \\ H^n(x,y) - F^0(x,y) - \alpha_n^{n+1}F^{n+1}(x,y) - \ldots - \alpha_n^{2n}F^{2n}(x,y) \end{bmatrix},$$

where V is the $n \times n$ invertible matrix whose i-th row is $(\alpha_i,\ldots,\alpha_i^n)$. Observe that when assigning $y = i \in \mathsf{C}$, the RHS is equal to the LHS. Indeed, the RHS is equal to

$$\begin{aligned} V^{-1} \begin{bmatrix} H^1(x,i) - F^0(x,i) - \alpha_1^{n+1}F^{n+1}(x,i) - \ldots - \alpha_1^{2n}F^{2n}(x,i) \\ \vdots \\ H^n(x,i) - F^0(x,i) - \alpha_n^{n+1}F^{n+1}(x,i) - \ldots - \alpha_n^{2n}F^{2n}(x,i) \end{bmatrix} &= V^{-1} \begin{bmatrix} \sum_{k=1}^n \alpha_1^k f_i^k(x) \\ \vdots \\ \sum_{k=1}^n \alpha_n^k f_i^k(x) \end{bmatrix} \\ &= \begin{bmatrix} f_i^1(x) \\ \vdots \\ f_i^n(x) \end{bmatrix}. \end{aligned}$$

In addition, when assigning $x = y = 0$, the RHS is also equal to the LHS,

$$\begin{aligned} V^{-1} \begin{bmatrix} H^1(0,0) - F^0(0,0) - \alpha_1^{n+1}F^{n+1}(0,0) - \ldots - \alpha_1^{2n}F^{2n}(0,0) \\ \vdots \\ H^n(0,0) - F^0(0,0) - \alpha_n^{n+1}F^{n+1}(0,0) - \ldots - \alpha_n^{2n}F^{2n}(0,0) \end{bmatrix} &= V^{-1} \begin{bmatrix} \sum_{k=1}^n \alpha_1^k P^k \\ \vdots \\ \sum_{k=1}^n \alpha_n^k P^k \end{bmatrix} \\ &= \begin{bmatrix} P^1 \\ \vdots \\ P^n \end{bmatrix}. \end{aligned}$$

We conclude that the RHS is in the support of the random variables in the LHS. By Fact A.6 the random variables $F^1(x,y),\ldots,F^n(x,y)$ have the same support size, which we denote by S, and we conclude that $(G^1,\ldots,G^n) = (H^1,\ldots,H^n)$ with probability $1/S^n$. Since this is true for every $(H_1,\ldots,H_n)$ in the support, the claim follows. □

The following facts are due to Beaver [13].

Fact A.8 (Beaver's Trick over Univariate Polynomials). *Let $\mathsf{C} \subseteq \{1,\ldots,n\}$ be a set of size at most t, and let $f^\alpha(x)$ and $f^\beta(x)$ be any degree-t polynomials. Then the following two experiments have the same distribution.*

- Experiment 1. *Sample three random degree-t polynomials $f^a(x), f^b(x)$ and $f^c(x)$ conditioned on $f^a(0) \cdot f^b(0) = f^c(0)$. Set $u := f^\alpha(0) - f^a(0)$ and $v := f^\beta(0) - f^b(0)$. Output*
$$\left(\{f^a(i), f^b(i), f^c(i)\}_{i \in \mathsf{C}}, f^\alpha(x) - f^a(x), f^\beta(x) - f^b(x), uf^b(x) + vf^a(x) + f^c(x) + uv\right).$$
- Experiment 2. *Sample uniform triples $\{a_i, b_i, c_i\}_{i \in \mathsf{C}}$. Sample a degree-$t$ polynomials $f^u(x)$ and $f^v(x)$ conditioned on $\{f^u(i) = f^\alpha(i) - a_i\}_{i \in \mathsf{C}}$ and $\{f^v(i) = f^\beta(i) - b_i\}_{i \in \mathsf{C}}$. Set $u := f^u(0)$ and $v := f^v(0)$ and sample a degree-t polynomial $f(x)$ conditioned on $f(0) = f^\alpha(0) \cdot f^\beta(0)$ and $\{f(i) = ub_i + va_i + c_i + uv\}_{i \in \mathsf{C}}$. Output*
$$\left(\{a_i, b_i, c_i\}_{i \in \mathsf{C}}, f^u(x), f^v(x), f(x)\right).$$

Fact A.9 (Beaver's Trick over Bivariate Polynomials). *Let $\mathsf{C} \subseteq \{1, \ldots, n\}$ be a set of size at most t, and let $F^\alpha(x, y)$ and $F^\beta(x, y)$ be any symmetric bivariate polynomials of degree-t in each variable. Then the following two experiments have the same distribution.*

- Experiment 1. *Sample three random symmetric bivariate polynomials $F^a(x, y), F^b(x, y)$ and $F^c(x, y)$ conditioned on $F^a(0, 0) \cdot F^b(0, 0) = F^c(0, 0)$. Set $u := F^\alpha(0, 0) - F^a(0, 0)$ and $v := F^\beta(0, 0) - F^b(0, 0)$. Output*
$$\left(\{F^a(x, i), F^b(x, i), F^c(x, i)\}_{i \in \mathsf{C}}, F^\alpha(x, y) - F^a(x, y), F^\beta(x, y) - F^b(x, y), uF^b(x, y) + vF^a(x, y) + F^c(x, y) + uv\right).$$
- Experiment 2. *Sample uniform degree-t polynomials $\{f_i^a(x), f_i^b(x), f_i^c(x)\}_{i \in \mathsf{C}}$ conditioned on $f_i^a(j) = f_j^a(i)$, $f_i^b(j) = f_j^b(i)$ and $f_i^c(j) = f_j^c(i)$ for all $i, j \in \mathsf{C}$. Sample symmetric bivariate polynomials $F^u(x, y)$ and $F^v(x, y)$ of degree at most t in each variable, conditioned on $\{F^u(x, i) = F^\alpha(x, i) - f_i^a(x)\}_{i \in \mathsf{C}}$ and $\{F^v(x, i) = F^\beta(x, i) - f_i^b(x)\}_{i \in \mathsf{C}}$. Set $u := F^u(0, 0)$ and $v := F^v(0, 0)$ and sample a symmetric bivariate polynomial $F(x, y)$ of degree at most t in each variable, conditioned on $F(0, 0) = F^\alpha(0, 0) \cdot F^\beta(0, 0)$ and $\{F(x, i) = uf_i^b(x) + vf_i^a(x) + f_i^c(x) + uv\}_{i \in \mathsf{C}}$. Output*
$$\left(\{f_i^a(x), f_i^b(x), f_i^c(x)\}_{i \in \mathsf{C}}, F^u(x, y), F^v(x, y), F(x, y)\right).$$

In order to obtain a random degree-$2t$ polynomial that shares the value 0, we need the following fact, which was previously used in [5].

Fact A.10 (Zero-sharing). *Let $|\mathbb{F}| \geq 2n$. There exist coefficients $\{\lambda_j^i\}_{i,j \in \{1,\ldots,n\}}$, where $\lambda_j^i \in \mathbb{F}$, such that for any $t < n/2$ the following holds.*

Let $\mathsf{C} \subseteq \{1, \ldots, n\}$ be a set of size at most t, let $\{F_i(x, y)\}_{i \in \mathsf{C}}$ be fixed symmetric bivariate polynomials of degree at most t in each variable, and define $f_{ij}(x) := F_i(x, j)$ for all $i \in \mathsf{C}$ and $j \in \{1, \ldots, n\}$. Let $\{f_{ij}(x)\}_{i \in \{1,\ldots,n\} \setminus \mathsf{C}, j \in \mathsf{C}}$ be fixed degree-t polynomials such that $f_{ij}(k) = f_{ik}(j)$ for all $i \in \{1, \ldots, n\} \setminus \mathsf{C}$ and $j, k \in \mathsf{C}$, and let $\{F_i(x, y)\}_{i \in \{1,\ldots,n\} \setminus \mathsf{C}}$ be random symmetric bivariate polynomials of degree at most t in each variable, conditioned on $F_i(x, j) = f_{ij}(x)$ for all $i \in \{1, \ldots, n\} \setminus \mathsf{C}$ and $j \in \mathsf{C}$.

For $i \in \{1, \ldots, n\}$, let
$$g_i(x) := \sum_{j=1}^{n} \lambda_i^j F_j(x, i),$$
and let $g(x)$ be the degree-$2t$ polynomial obtained by interpolating $\{g_i(0)\}_{i \in \{1,\ldots,n\}}$. Then the polynomial $g(x)$ is uniformly distributed conditioned on $g(0) = 0$ and $g(i) = \sum_{j=1}^{n} \lambda_i^j f_{ji}(0)$ for all $i \in \mathsf{C}$.

A.5 Statistical Distance

We start with a basic fact, which can be found, e.g., in [2].

Fact A.11. *Let $\mathcal{W}$ be a set, and let $\{X_w\}_{w\in\mathcal{W}}$, $\{Y_w\}_{w\in\mathcal{W}}$ be distribution ensembles. Then, for every distribution W over $\mathcal{W}$, we have $\Delta((W, X_W), (W, Y_W)) = \mathbb{E}_{w\leftarrow W}[\Delta(X_w, Y_w)]$.*

Fact A.12. *Let $X = (X_1, X_2)$ and $Y = (Y_1, Y_2)$ be probability distributions on a set $A \times B$ such that $\Delta(X, Y) \leq \epsilon$. Let $\delta = \sqrt{2\epsilon}$. Then, with probability at least $1 - \delta$ over $z \leftarrow X_1$ it holds that*

$$\Delta(X_2 \mid_{X_1=z}, Y_2 \mid_{Y_1=z}) \leq \delta.$$

Proof. Consider the random variable $Z = (Z_1, Z_2)$ distributed over $A \times B$, that is sampled in the following way. First, Z_1 is sampled, and it has the same marginal distribution as X_1. Then, Z_2 is sampled according to the conditional distribution $Y_2 \mid_{Y_1=Z_1}$.

Since $\Delta(X, Y) \leq \epsilon$ then $\Delta(Z, Y) \leq \epsilon$. Indeed, Z_1 and Y_1 are ϵ-close in statistical distance, and conditioned on any value $Z_1 = Y_1 = z$, the random variables $Z_2 \mid_{Z_1=z}$ and $Y_2 \mid_{Y_1=z}$ have the same distribution. By the triangle-inequality it follows that $\Delta(X, Z) \leq \Delta(X, Y) + \Delta(Y, Z) \leq 2\epsilon$.

By Fact A.12 it follows that

$$2\epsilon \geq \Delta(X, Z) = \mathbb{E}_{z\leftarrow X_1}[\Delta(X_2 \mid_{X_1=z}, Y_2 \mid_{Y_1=z})].$$

Finally, by Markov's inequality we conclude that

$$\Pr_{z\leftarrow X_1}[\Delta(X_2 \mid_{X_1=z}, Y_2 \mid_{Y_1=z}) \geq \delta] \leq 2\epsilon/\delta = \delta.$$

This concludes the proof. □

A.6 Preliminaries

A.6.1 Hashing

We recall two hash functions: universal and collision-resistant. Both of these are used for statistically-hiding and computationally binding NICOM of [26, 40] as recalled later.

Universal Hashing. Universal Hashing was introduced by Carter and Wegman [23].

Definition A.13 (Universal Hash Function)**.** *A collection of functions*

$$U = \left\{u_z \ : \ \{0,1\}^\kappa \to \{0,1\}^{m(\kappa)}\right\}_{z\in\{0,1\}^{s(\kappa)}}$$

is a universal hash function if the following hold for any two different inputs s_1, s_2 and for any two output elements t_1, t_2, we have $\Pr_{u_z\in U}\left[u_z(s_1) = t_1 \wedge u_z(s_2) = t_2\right] = \frac{1}{m^2}$.

For input space $\{0,1\}^\kappa$ and output space $\{0,1\}^m$ with $m \leq \kappa$, we can have $U = \{u_{A,b} : A \in \{0,1\}^{m\times\kappa}\ b \in \{0,1\}^\kappa\}$, where $u_{A,b}(x) = Ax + b$ (all operations take place over $GF(2)$). Picking a function uniformly from set U simply requires picking A, b uniformly at random from the respective spaces.

Collision-resistant Hashing. We now define collision-resistant (CR) hash functions.

Definition A.14 (Collision-resistant Hash Function). *A collection of functions*

$$H = \left\{ h_z \ : \ \{0,1\}^\kappa \to \{0,1\}^{m(\kappa)} \right\}_{z \in \{0,1\}^{s(\kappa)}}$$

is collision-resistant (CR) hash function if the following hold:

1. *(Shrinkage) The output length is smaller than the input length i.e. $m(\kappa) < \kappa$ for every κ.*
2. *(Efficient Evaluation and sampling) There exists a pair of efficient algorithms (a) an evaluation algorithm which given $z \in \{0,1\}^s, x \in \{0,1\}^\kappa$ outputs $h_z(x)$; and (b) a key-sampling algorithm K which given 1^κ samples an index $z \in \{0,1\}^{s(\kappa)}$.*
3. *(Collision Resistance) For every PPT adversary $\mathcal{A}$ it holds that*

$$\Pr_{z \leftarrow_R K(1^\kappa)} \left[\mathcal{A}(z) = (x, x') \text{ s.t. } x' \neq x \text{ and } h_z(x) = h_z(x') \right] \leq \mathsf{negl}(\kappa).$$

A.6.2 Non-Interactive Commitment Schemes (NICOM)

Definition A.15 (NICOM). *An ϵ-secure NICOM* $(\mathsf{commit}, \mathsf{open})$ *satisfies the following.*

- *Correctness: For all* pp*, $x \in M$ and randomness r, if $(C, o) \leftarrow \mathsf{commit}(x; r)$ then $\mathsf{open}(C, o) = x$.*
- *Binding: For all* **PPT** *adversaries $\mathcal{A}$, it is with probability at most ϵ (over a uniform choice of* pp *and the random coins of $\mathcal{A}$) that $\mathcal{A}(\mathsf{pp})$ outputs (C, o, o') such that $\mathsf{open}(C, o) \neq \mathsf{open}(C, o')$ and $\perp \notin \{\mathsf{open}(C, o), \mathsf{open}(C, o')\}$.*
- *Hiding: For all* **PPT** *adversaries $\mathcal{A}$ and all* pp*, the following difference at most ϵ for all $x, x' \in M$:*

$$\left| \mathsf{Pr}_{(C,o) \leftarrow C(x)}[\mathcal{A}(C) = 1] - \mathsf{Pr}_{(C,o) \leftarrow C(x')}[\mathcal{A}(C) = 1] \right|$$

For a security parameter κ, we say that $(\mathsf{commit}, \mathsf{open})$ *is secure, if it is ϵ-secure for some $\epsilon(\kappa)$ negligible in κ. We say that a NICOM is statistically binding, if that the binding property holds against and unbounded adversary $\mathcal{A}$, and we say that it is statistically hiding, if the hiding property holds against an unbounded adversary $\mathcal{A}$.*

Instantiations. Here we present three instantiations of NICOM, all based on symmetric key primitives. Operating in the plain model, the first one provides cryptographic hiding and perfect binding. The remaining two work in the CRS setting. One of these offers statistical binding and computational hiding hiding. The last one offers computational binding and statistical hiding, the latter property is instrumental in achieving everlasting security of our protocols.

NICOM from injective OWF [18, 53, 36]. The first construction relies on any injective one-way function and works in the plain model. The scheme offers computational hiding and perfect binding. Let $f : \{0,1\}^\kappa \to \{0,1\}^\kappa$ be a one-way permutation and $h : \{0,1\}^\kappa \to \{0,1\}$ a hard-core predicate for $f(\cdot)$. Then the commitment scheme for a single bit x is:

- The public parameter pp is f, g. This can be selected by the committer and so this construction works in plain model.
- $\mathsf{commit}(x;r)$: set $C = (f(r), x \oplus h(r))$, where $r \in_R \{0,1\}^\kappa$; set $o = (r, x)$.
- $\mathsf{open}(C, o = (r,x))$: return x if $C = (f(r), x \oplus h(r))$; otherwise return $\perp$.

For commitment of multi-bit string, the Goldreich-Goldwasser-Micali [34] construction from a one-way permutation f can be used. Recall the GGM construction: given one-way permutation $f : \{0,1\}^\kappa \to \{0,1\}^\kappa$ with hard-core predicate $h : \{0,1\}^\kappa \to \{0,1\}$, first construct a length-doubling pseudorandom generator $G : \{0,1\}^\kappa \to \{0,1\}^{2\kappa}$ via: $G(s) = f^\kappa(s)\ h(f^{\kappa-1}(s)) \dots h(s)$, where f^κ denotes the κth application of f. Let $G_0(s)$ denote the first κ bits of $G(s)$, and let $G_1(s)$ denote the last k bits of $G(s)$. For a binary string s, the commitment C can be defined as $C = G_0(\dots(G_0(G_0(s)))\dots)$ with $o = s$. The commitment function can be shown to be a permutation and so binding holds unconditionally. Hiding follows from the property of PRF F [44].

Remark A.16 (Sub-exponential hardness)**.** *Assuming injective OWF over m-bit inputs that cannot be inverted by a PPT adversary with probability better than 2^{-m^δ} for some positive constant $\delta > 0$, we can use the above construction to provide a perfectly-binding computationally-hiding NICOM with error $2^{-\kappa}$ where κ is the security parameter. Moreover, under worst-case derandomization assumptions [12], the above holds for general (not necessarily injective) OWFs.*

NICOM from PRG [49]. The second construction relies on PRG security and works in the common random string (CRS) setup. The scheme is computationally hiding and statistically binding and operates for single bit message. Let $G : \{0,1\}^\kappa \to \{0,1\}^{4\kappa}$ be a PRG.

- The CRS consists of σ which is a uniformly random string chosen from $\{0,1\}^{4\kappa}$.
- $\mathsf{commit}(x;r)$: set $C = G(r)$ if $x = 0$, else $C = G(r) \oplus \sigma$; set $o = (r, x)$
- $\mathsf{open}(C, o = (r,x))$: return x if $C = G(r) \oplus x \cdot \sigma$ (where '$\cdot$' denotes multiplication by constant); otherwise return $\perp$.

NICOM from CR hash function [26, 40]. The third construction relies on CR hash function security and works in the CRS setup. The scheme is statistically hiding and computationally binding. This construction works for long (or multi-bit) messages. Let the message space be $\{0,1\}^m$. Let $H : \{0,1\}^{4\kappa+2m+4} \to \{0,1\}^\kappa$ be a collection of CR hash functions (see Definition A.14).

- The CRS consists of $z \leftarrow_R K(1^\kappa)$ where K is the key-sampling algorithm of CR hash functions (see Definition A.14).
- $\mathsf{commit}(x;r)$ where $r \leftarrow_R \{0,1\}^{4\kappa+2m+4}$: set $C = (u, h_z(r), u(r) \oplus x)$, where u is a function chosen uniformly at random from the collection of universal functions $U : \{0,1\}^{4\kappa+2m+4} \to \{0,1\}^m$. Set $o = r$
- $\mathsf{open}(C = (C_1, C_2, C_3), o)$: return x where $x = C_1(o) \oplus C_3$ if $C_2 = h_z(o)$; otherwise return $\perp$.

A.7 Secure Partial Computation with a Guard

Proof of Lemma 3.3. In this section we prove that protocol spcg UC-emulates $\mathcal{F}_{\mathsf{spcg}}$ (with everlasting security when the underlying commitment scheme is statistically-hiding). Let $\mathcal{A}$ be an efficient adversary against spcg. We define the simulator $\mathcal{S}$ as follows. The simulator $\mathcal{S}$ uses $\mathcal{A}$ in a black-box manner, and forwards all messages between $\mathcal{Z}$ and $\mathcal{A}$. The simulator first receives the set of corrupt C parties from $\mathcal{Z}$.

We split into cases, and begin by considering the case of honest Alice and Bob. In this case, we provide a single simulator (Section A.7.1), but provide two different analyses: one for the the case where the underlying commitment scheme is statistically-hiding and everlasting security is required (Section A.7.2), and one for the case where the underlying commitment scheme is only computationally-hiding (Section A.7.3). Then, we present the rest of the cases (where at least one of Alice and Bob is corrupt), together with their analysis, which applies both to the case of statistically-hiding commitments and computationally-hiding commitments.

A.7.1 Honest Alice and Bob – Simulator

Offline round. $\mathcal{S}$ simulates the actions of an honest Bob. That is, $\mathcal{S}$ queries $\mathcal{F}_{\mathsf{crs}}$ to obtain crs, samples a random string r for the psm protocol, commitments and openings $(C'_{i,x}, o'_{i,x}) \leftarrow \mathsf{commit}_{\mathsf{crs}}(\mathsf{psm}_i(x, r))$ for each $x \in \{0,1\}$ and $i \in \{1, \dots, \ell\}$, and random shifts σ_i for any $i \in \{1, \dots, \ell\}$, like an honest Bob. $\mathcal{S}$ sends the shifted commitments $\{C'_{i,\sigma_i(x)}\}_{i\in\{1,\dots,\ell\},x\in\{0,1\}}$ to the adversary, as the broadcast of Bob.

Online round. $\mathcal{S}$ receives

$$\left(\mathbf{a}, \delta^A, \delta^B, \{i, o_i, b_i^A\}_{i\in I}, \chi\right) \quad \text{and} \quad (C_1, \dots, C_{L_B})$$

from $\mathcal{F}_{\mathsf{spcg}}$, where χ is the partial sum $\sum_{i\in J} \alpha_i(\mathbf{a}) b_i^A$. $\mathcal{S}$ extracts the set I from the output, and computes the set $J := \delta^A \setminus I$. $\mathcal{S}$ then finds $\{b'_i\}_{i\in J}$ such that $\sum_{i\in J} \alpha_i(\mathbf{a}) \cdot b'_i = \chi$. This can be done by finding the first $i^* \in J$ such that $\alpha_{i^*}(\mathbf{a}) \neq 0$, and setting b'_{i^*} to $\alpha_{i^*}(\mathbf{a})^{-1} \cdot \chi$ while the rest of the b'_i's are set to 0 (if no such $\alpha_i(\mathbf{a})$ exists then $\chi = 0$, and then any assignment would work).

For each $i \in J$ set $\bar{b}_i^A := b'_i$ and $\bar{b}_i^B := b'_i$; for each $i \in I$ set $\bar{b}_i^A := b_i^A$ and $\bar{b}_i^B := b_i^A + 1$; and for every $i \notin I \cup J$ set $\bar{b}_i^A := 0$ and $\bar{b}_i^B := 0$. In addition, for $i \in I$ set $\bar{o}_i := o_i$ and for $i \notin I$ set $\bar{o}_i := 0$. $\mathcal{S}$ then computes the binary string $x_A := (\mathbf{a}, \delta_A, \{\bar{o}_i, \bar{b}_i^A\}_{i=1}^{L_B})$, computes $\bar{s}_i := \mathsf{psm}_i(x_i^A, r)$ for each $i \in \{1, \dots, \ell_A\}$ and broadcasts $\{(\sigma_i(x_i^A), o'_{i,x_i^A}, \bar{s}_i)\}_{i\in\{1,\dots,\ell_A\}}$ on behalf of Alice. $\mathcal{S}$ also computes the binary string $x_B := (\delta_B, \{\bar{b}_i^B\}_{i=1}^{L_B})$, computes $\bar{s}_i := \mathsf{psm}_i(x_{i-\ell_A}^B, r)$ for each $i \in \{\ell_A + 1, \dots, \ell\}$ and broadcasts $\{(\sigma_i(x_{i-\ell_A}^B), o'_{i,x_{i-\ell_A}^B}, \bar{s}_i)\}_{i\in\{\ell_A+1,\dots,\ell\}}$ on behalf of Bob.[12]

A.7.2 Honest Alice and Bob – Statistically-Hiding Commitment Scheme

Fix any polynomial time environment $\mathcal{Z}$ and input ζ to the environment. We show that the view of $\mathcal{Z}$ in the real world is statistically close to the view of $\mathcal{Z}$ in the ideal world.

[12] Notice that the simulation here is done by computing some pre-image that gives the desired output and using it as the input of the PSM. Looking ahead, in the indistinguishability proof, we use PSM privacy that guarantees that those messages have (almost) the same distribution as in the real-world as long as the output remains the same in both cases.

$\mathcal{Z}$'s view. In the real-world, since Alice and Bob are honest, then the verification of Alice at the beginning of the online phase always succeeds. Therefore the adversary's view consists of (1) the crs and the first-round broadcast of Bob $\{C'_{i,\sigma_i(x)}\}_{x\in\{0,1\}}$ for each $i \in \{1,\ldots,\ell\}$, (2) the inputs of Alice and Bob, (3) the second-round broadcast of Alice $\{(\sigma_i(x_i^A), o'_{i,x_i^A}, s_i)\}_{i\in\{1,\ldots,\ell_A\}}$, and (4) the second-round broadcast of Bob $\{(\sigma_i(x_{i-\ell_A}^B), o'_{i,x_{i-\ell_A}^B}, s_i)\}_{i\in\{\ell_A+1,\ldots,\ell\}}$.

We begin by showing that (1) has same distribution in both worlds. (this would imply that (2) has the same distribution in both worlds). Then, we consider the random variables $(r, \{\sigma_i\}_{i\in\{1,\ldots,\ell\}})$, where where r is the PSM randomness sampled by Bob, and $\{\sigma_i\}_{i\in\{1,\ldots,\ell\}}$ are the random shifts sampled by Bob. We show that even conditioned on (1), i.e $(\mathsf{crs}, \{C'_{i,\sigma_i(x)}\}_{i\in\{1,\ldots,\ell\},x\in\{0,1\}})$, in both worlds the random variables $(r, \{\sigma_i\}_{i\in\{1,\ldots,\ell\}})$ are $O((\ell\epsilon)^{1/2})$-close to uniform, and there are overwhelmingly many such $(\mathsf{crs}, \{C'_{i,\sigma_i(x)}\}_{i\in\{1,\ldots,\ell\},x\in\{0,1\}})$ for which the above is true. This will allow us to extend the argument from $O((\ell\epsilon)^{1/2})$-closeness between distributions $(r, \{\sigma_i\}_{i\in\{1,\ldots,\ell\}})$ and uniform to $O((\ell\epsilon)^{1/2})$-closeness between real and ideal world distributions of (3) and (4).

Consider the random variables $(\mathsf{crs}, \{C'_{i,\sigma_i(x)}\}_{i\in\{1,\ldots,\ell\},x\in\{0,1\}}, r, \{\sigma_i\}_{i\in\{1,\ldots,\ell\}})$. It is not hard to see that, since $\mathcal{S}$ follows the steps of an honest Bob in the offline phase, those random variables have the same distribution in both worlds. In the following, we denote the length of r by $|r|$, and note that each σ_i can be represented by a single bit.

First, we observe that the random variables $(\mathsf{crs}, \{C'_{i,\sigma_i(x)}\}_{i\in\{1,\ldots,\ell\},x\in\{0,1\}}, r, \{\sigma_i\}_{i\in\{1,\ldots,\ell\}})$ are $O(\ell\epsilon)$-close in statistical distance to the random variables $(\mathsf{crs}, \{C''_{i,\sigma_i(x)}\}_{i\in\{1,\ldots,\ell\},x\in\{0,1\}}, U_{|r|}, U_\ell)$, where each commitment in $\{C''_{i,\sigma_i(x)}\}_{i\in\{1,\ldots,\ell\},x\in\{0,1\}}$ is a commitment of the all-zero string. Indeed, $(r, \{\sigma_i\}_{i\in\{1,\ldots,\ell\}})$ have the same distribution as $(U_{|r|}, U_\ell)$. Conditioned on those values, and by the hiding property of the commitment scheme, it follows that $(\mathsf{crs}, \{C'_{i,\sigma_i(x)}\}_{i\in\{1,\ldots,\ell\},x\in\{0,1\}})$ is $O(\ell\epsilon)$-close in statistical distance to $(\mathsf{crs}, \{C''_{i,\sigma_i(x)}\}_{i\in\{1,\ldots,\ell\},x\in\{0,1\}})$, as required.

We say that a fixing of $(\mathsf{crs}, \{C'_{i,\sigma_i(x)}\}_{i\in\{1,\ldots,\ell\},x\in\{0,1\}})$ is "good", if conditioned on this fixing, the random variables $(r, \{\sigma_i\}_{i\in\{1,\ldots,\ell\}})$ are $O((\ell\epsilon)^{1/2})$-close to $(U_{|r|}, U_\ell)$. By Fact A.12 it follows that a fixing is good with probability at least $1 - O((\ell\epsilon)^{1/2})$. Condition on any good fixing of $\{C'_{i,\sigma_i(x)}\}_{i\in\{1,\ldots,\ell\},x\in\{0,1\}}$. At this stage the honest parties inputs are picked by $\mathcal{Z}$, and so they have the same distribution in both worlds. Fix those inputs as well.

It remains to show that

$$(i_1,\ldots,i_\ell), (o'_{1,i_1},\ldots,o'_{\ell,i_\ell}), (s_1,\ldots,s_\ell)$$

have the same distribution in both worlds, where i_j, o'_{j,i_j} and s_j are the indices, openings and PSM messages broadcasted by Alice and Bob in the online round. Since in both worlds $(r, \{\sigma_i\}_{i\in\{1,\ldots,\ell\}})$ is $O((\ell\epsilon)^{1/2})$-close to $(U_{|r|}, U_\ell)$, and by the security of the PSM protocol, we conclude that in both worlds $((s_1,\ldots,s_\ell),(i_1,\ldots,i_\ell))$ are $O((\ell\epsilon)^{1/2})$-close to $(\mathcal{S}_{\mathsf{psm}}(\mathsf{val}), U_\ell)$, where $\mathcal{S}_{\mathsf{psm}}$ is the simulator of the PSM protocol, and $\mathsf{val} := \left(\mathbf{a}, \delta^A, \delta^B, \{i, o_i, b_i^A\}_{i\in I}, \sum_{i\in J}\alpha_i(\mathbf{a})\cdot b_i\right)$ is the output of the f_{spcg} function, which is fixed (as we've already fixed the honest parties inputs). Finally, fix $((s_1,\ldots,s_\ell),(i_1,\ldots,i_\ell))$ as well, and observe that the random variables $(o'_{1,i_1},\ldots,o'_{\ell,i_\ell})$ have the same distribution in both worlds. We conclude that the real-world view is $O((\ell\epsilon)^{1/2})$-close to the ideal-world view.

Honest parties' outputs. Fix any view View of the adversary. In the ideal world, the output of the honest parties is $(\mathbf{a}, \delta^A, \{i, o_i, b_i^A\}_{i\in I}, \sum_{i:\delta_i^A=1} \alpha_i(\mathbf{a}) \cdot b_i^A)$ with probability 1. In the real world, since Bob is honest, the verification of Alice succeeds, and so both Alice and Bob broadcast $\{(i_j, o'_j, s_j)\}_{j\in\{1,\dots,\ell\}}$. It is not hard to see that honest Alice and Bob are never discarded, and the perfect correctness of the PSM protocol implies that all honest parties correctly $(\mathbf{a}, \delta^A, \delta^B, \{i, o_i, b_i^A\}_{i\in I}, \sum_{i\in J} \alpha_i(\mathbf{a}) \cdot b_i)$, and so they output $(\mathbf{a}, \delta^A, \{i, o_i, b_i^A\}_{i\in I}, \sum_{i:\delta_i^A=1} \alpha_i(\mathbf{a}) \cdot b_i^A)$ with probability 1. This concludes the case of honest Alice and Bob with statistically-hiding commitments.

A.7.3 Honest Alice and Bob - Computationally-Hiding Commitment Scheme

We continue by proving the security of our protocol when the underlying commitment scheme is merely computationally-hiding. In the following, we assume without loss of generality that $\mathcal{A}$ is the dummy adversary, that simply delivers messages from $\mathcal{Z}$ to the protocol machines. We show that for the simulator described in Section A.7, no environment can distinguish the real-world from the ideal-world.

Under sub-exponential hardness assumption (see Section A.6.2), we may assume that for every security parameter μ, there exists an efficient commitment scheme $(\mathsf{commit}_{\mathsf{crs}}, \mathsf{open}_{\mathsf{crs}})$ which is perfectly-binding and computationally-hiding, and has error $2^{-\mu}$. We assume that the commitments used for the inputs of Alice and Bob have security parameter κ, which is the security parameter of the protocol. Observe that there exists a constant c such that the total bit-length of the honest inputs, which we've denoted by ℓ, is at most $c(L_A + L_B)(\kappa \log |\mathbb{F}|)^c$. In the protocol itself, we let Bob use a commitment scheme with security parameter κ', and require that $\kappa' = \ell + \log \ell + \kappa$. In this section we set $\epsilon := 2^{-\kappa}$ and $\epsilon' := 2^{-\kappa'}$.

Fix any environment $\mathcal{Z}$ and an advice string z, and assume without loss of generality that $\mathcal{Z}$ is deterministic. We start by proving that any degenerate environment $\mathcal{Z}'$, that always gives the same inputs to Alice and Bob, cannot distinguish real-world from ideal-world with advantage more than $\ell\epsilon'$. We then show that if a (general) environment $\mathcal{Z}$ distinguishes real-world from ideal-world with advantage δ, then there exists a degenerate environment that distinguishes with advantage $\delta/2^\ell$. We conclude that a general environment distinguishes real-world from ideal-world with advantage at most $2^\ell \cdot \ell\epsilon' = 2^{-\kappa}$.

We begin with the following claim.

Claim A.17. *Let $\mathcal{Z}'$ be an environment, and let z' be an advice string. Assume that $\mathcal{Z}'(z')$ always corrupts Alice and Bob, and always gives the honest parties the same inputs at the beginning of the online phase. Then,*

$$\mathrm{REAL}_{\mathsf{spcg},\mathcal{Z}'(z'),\mathcal{A}} \approx_{\ell\epsilon'} \mathrm{IDEAL}_{\mathcal{F}_{\mathsf{spcg}},\mathcal{Z}'(z'),\mathcal{S}}.$$

Proof. Let x_A and x_B be the binary strings that Alice and Bob compute as the psm input, and note that those strings are fixed. Let $\mathcal{X} := x_A \circ x_B$ be the concatenation of those strings. Similarly, let $\bar{x}_A$ and $\bar{x}_B$ be the binary strings that the simulator computes as the psm input, and note that those strings are fixed as well. Let $\bar{\mathcal{X}} := \bar{x}_A \circ \bar{x}_B$.

Consider the real-world random variables

$$\left(\{C'_{i,\sigma_i(x)}\}_{i\in\{1,\dots,\ell\},x\in\{0,1\}}, \{o'_{i,\mathcal{X}_i}\}_{i\in\{1,\dots,\ell\}}, \{\sigma_i(\mathcal{X}_i)\}_{i\in\{1,\dots,\ell\}}\right),$$

and the ideal-world random variables

$$\left(\{\bar{C}'_{i,\sigma_i(x)}\}_{i\in\{1,\dots,\ell\},x\in\{0,1\}}, \{\bar{o}'_{i,\bar{\mathcal{X}}_i}\}_{i\in\{1,\dots,\ell\}}, \{\bar{\sigma}_i(\bar{\mathcal{X}}_i)\}_{i\in\{1,\dots,\ell\}}\right),$$

and observe that they are $\ell\epsilon'$-indistinguishable. Indeed, by the perfect-security of the psm scheme, and since $f_{\mathsf{spcg}}(\mathcal{X}) = f_{\mathsf{spcg}}(\bar{\mathcal{X}})$, it follows that $s_1, \dots, s_\ell$ have the same distribution in both worlds. In addition, it is not hard to see that $\{\sigma_i(\mathcal{X}_i)\}_{i\in\{1,\dots,\ell\}}$ and $\{\bar{\sigma}_i(\bar{\mathcal{X}}_i)\}_{i\in\{1,\dots,\ell\}}$ have the same distribution. Conditioned on those values, fix any r in the real-world and $\bar{r}$ in the ideal-world. The random variables $\{C'_{i,\sigma_i(\mathcal{X}_i)}, o'_{i,\mathcal{X}_i}\}_{i\in\{1,\dots,\ell\}}$ and $\{C'_{i,\bar{\sigma}_i(\bar{\mathcal{X}}_i)}, o'_{i,\bar{\mathcal{X}}_i}\}_{i\in\{1,\dots,\ell\}}$ have the same distribution, and the random variables $\{C'_{i,\sigma_i(1-\mathcal{X}_i)}\}_{i\in\{1,\dots,\ell\}}$ and $\{C'_{i,\bar{\sigma}_i(1-\bar{\mathcal{X}}_i)}\}_{i\in\{1,\dots,\ell\}}$ are $\ell\epsilon'$-indistinguishable.

Finally, since both Alice and Bob are honest and by the perfect correctness of the psm scheme, it follows that, in both worlds, the output of the honest parties is always $g_{\mathsf{spcg}}(\mathbf{a}, \mathbf{b}^A, \delta^A, \{o_i\}_{i=1}^{L_B}, I)$, for $I = (\mathbf{b}^A, \delta^A)\diamond(\mathbf{b}^B, \delta^B)$. □

Denote by $\mathcal{X}_1, \dots, \mathcal{X}_{2^\ell}$ all the length-ℓ binary strings. Observe that each input $(\mathcal{X}_A, \mathcal{X}_B)$ to Alice and Bob corresponds to a unique binary string $\mathcal{X}_i = \mathcal{X}_A \circ \mathcal{X}_B$.

For $i \in \{1, \dots, 2^\ell\}$, we turn $\mathcal{Z}$ into a new environment $\mathcal{Z}_i$ with advice string $z_i := (z, \mathcal{X}_i)$ in the following way. The environment $\mathcal{Z}(z_i)$ acts in the offline phase exactly like $\mathcal{Z}(z)$. At the beginning of the online-phase the environment computes the inputs $(\mathcal{X}_A, \mathcal{X}_B)$ to Alice and Bob that $\mathcal{Z}(z)$ would produce. If $(\mathcal{X}_A, \mathcal{X}_B)$ correspond to $\mathcal{X}_i$ then $\mathcal{Z}_i$ continues the execution by acting like $\mathcal{Z}$, and outputs the same output as $\mathcal{Z}$. Otherwise, the environment outputs a random bit.

We continue by showing that if $\mathcal{Z}(z)$ distinguishes the real-world from the ideal-world with advantage δ, then some $\mathcal{Z}_i$ distinguishes the real-world from the ideal-world with advantage $\delta/2^\ell$.

Claim A.18. *Assume that* $|\Pr[\mathrm{REAL}_{\mathsf{spcg},\mathcal{Z}(z),\mathcal{A}} = 1] - \Pr[\mathrm{IDEAL}_{\mathcal{F}_{\mathsf{spcg}},\mathcal{Z}(z),\mathcal{S}} = 1]| = \delta$. *Then there exists* $i \in \{1, \dots, 2^\ell\}$ *such that* $|\Pr[\mathrm{REAL}_{\mathsf{spcg},\mathcal{Z}_i(z_i),\mathcal{A}} = 1] - \Pr[\mathrm{IDEAL}_{\mathcal{F}_{\mathsf{spcg}},\mathcal{Z}_i(z_i),\mathcal{S}} = 1]| \geq \delta/2^\ell$.

Proof. Assume without loss of generality that

$$\Pr[\mathrm{REAL}_{\mathsf{spcg},\mathcal{Z}(z),\mathcal{A}} = 1] - \Pr[\mathrm{IDEAL}_{\mathcal{F}_{\mathsf{spcg}},\mathcal{Z}(z),\mathcal{S}}] = \delta.$$

For $i \in \{1, \dots, 2^\ell\}$, let E_i^{REAL} be the event that, in a real-world execution of the protocol with $\mathcal{Z}(z)$ and $\mathcal{A}$, $\mathcal{Z}$ picked inputs that correspond to $\mathcal{X}_i$. Similarly, let E_i^{IDEAL} be the event that, in an ideal-world execution with $\mathcal{Z}(z)$ and $\mathcal{S}$, $\mathcal{Z}$ picked inputs that correspond to $\mathcal{X}_i$. Since the simulator acts like an honest Bob in the offline-round, it follows that $\Pr[E_i^{\mathrm{REAL}}] = \Pr[E_i^{\mathrm{IDEAL}}]$ for every $i \in \{1, \dots, 2^\ell\}$, and so we omit the superscript.

For $i \in \{1, \dots, 2^\ell\}$, let $\bar{E}_i^{\mathrm{REAL}}$ be the event that, in a real-world execution of the protocol with $\mathcal{Z}_i(z_i)$ and $\mathcal{A}$, $\mathcal{Z}_i$ picked inputs that correspond to $\mathcal{X}_i$. Similarly, let $\bar{E}_i^{\mathrm{IDEAL}}$ be the event that, in an ideal-world execution with $\mathcal{Z}_i(z_i)$ and $\mathcal{S}$, $\mathcal{Z}_i$ picked inputs that correspond to $\mathcal{X}_i$. As before, it holds that $\Pr[\bar{E}_i^{\mathrm{REAL}}] = \Pr[\bar{E}_i^{\mathrm{IDEAL}}]$ for every $i \in \{1, \dots, 2^\ell\}$, so we omit the superscript.

It follows that

$$\begin{aligned}\delta &= \Pr[\mathrm{REAL}_{\mathsf{spcg},\mathcal{Z}(z),\mathcal{A}} = 1] - \Pr[\mathrm{IDEAL}_{\mathcal{F}_{\mathsf{spcg}},\mathcal{Z}(z),\mathcal{S}} = 1] \\ &= \sum_{i=1}^{2^\ell} \Pr[E_i]\big(\Pr[\mathrm{REAL}_{\mathsf{spcg},\mathcal{Z}(z),\mathcal{A}} = 1 \mid E_i] - \Pr[\mathrm{IDEAL}_{\mathcal{F}_{\mathsf{spcg}},\mathcal{Z}(z),\mathcal{S}} = 1 \mid E_i]\big) \\ &= \sum_{i=1}^{2^\ell} \Pr[\bar{E}_i]\big(\Pr[\mathrm{REAL}_{\mathsf{spcg},\mathcal{Z}_i(z_i),\mathcal{A}} = 1 \mid \bar{E}_i] - \Pr[\mathrm{IDEAL}_{\mathcal{F}_{\mathsf{spcg}},\mathcal{Z}_i(z_i),\mathcal{S}} = 1 \mid \bar{E}_i]\big) \\ &= \sum_{i=1}^{2^\ell} \big(\Pr[\mathrm{REAL}_{\mathsf{spcg},\mathcal{Z}_i(z_i),\mathcal{A}} = 1] - \Pr[\mathrm{IDEAL}_{\mathcal{F}_{\mathsf{spcg}},\mathcal{Z}_i(z_i),\mathcal{S}} = 1]\big)\end{aligned}$$

where (1) in the third equality we used the fact that $\Pr[E_i] = \Pr[\bar{E}_i]$ for all $i \in \{1, \ldots, 2^\ell\}$, and that $\Pr[\mathrm{REAL}_{\mathsf{spcg},\mathcal{Z}(z),\mathcal{A}} = 1 \mid E_i] = \Pr[\mathrm{REAL}_{\mathsf{spcg},\mathcal{Z}_i(z_i),\mathcal{A}} = 1 \mid \bar{E}_i]$, and $\Pr[\mathrm{IDEAL}_{\mathcal{F}_{\mathsf{spcg}},\mathcal{Z}(z),\mathcal{S}} = 1 \mid E_i] = \Pr[\mathrm{IDEAL}_{\mathcal{F}_{\mathsf{spcg}},\mathcal{Z}_i(z_i),\mathcal{S}} = 1 \mid \bar{E}_i]$, which follow since $\mathcal{Z}_i$ acts exactly like $\mathcal{Z}$ in the offline round and the generation of the honest parties inputs, and conditioned on E_i and $\bar{E}_i$, $\mathcal{Z}_i$ continues to act like $\mathcal{Z}$ in the online-round, and (2) the fourth equality follows since for every $i \in \{1, \ldots, 2^\ell\}$,

$$\begin{aligned}&\Pr[\mathrm{REAL}_{\mathsf{spcg},\mathcal{Z}_i(z_i),\mathcal{A}} = 1] - \Pr[\mathrm{IDEAL}_{\mathcal{F}_{\mathsf{spcg}},\mathcal{Z}_i(z_i),\mathcal{S}} = 1] \\ &\quad = \Pr[\bar{E}_i]\big(\Pr[\mathrm{REAL}_{\mathsf{spcg},\mathcal{Z}_i(z_i),\mathcal{A}} = 1 \mid \bar{E}_i] - \Pr[\mathrm{IDEAL}_{\mathcal{F}_{\mathsf{spcg}},\mathcal{Z}_i(z_i),\mathcal{S}} = 1 \mid \bar{E}_i]\big) \\ &\quad\quad + \Pr[\neg\bar{E}_i]\big(\Pr[\mathrm{REAL}_{\mathsf{spcg},\mathcal{Z}_i(z_i),\mathcal{A}} = 1 \mid \neg\bar{E}_i] - \Pr[\mathrm{IDEAL}_{\mathcal{F}_{\mathsf{spcg}},\mathcal{Z}_i(z_i),\mathcal{S}} = 1 \mid \neg\bar{E}_i]\big),\end{aligned}$$

and the second term is equal to zero, since whenever $\bar{E}_i$ does not occur, $\mathcal{Z}_i$ outputs a random bit.

Hence, by an averaging argument, there exists $i \in \{1, \ldots, 2^\ell\}$ such that $\Pr[\mathrm{REAL}_{\mathsf{spcg},\mathcal{Z}_i(z_i),\mathcal{A}} = 1] - \Pr[\mathrm{IDEAL}_{\mathcal{F}_{\mathsf{spcg}},\mathcal{Z}_i(z_i),\mathcal{S}} = 1] \geq \delta/2^\ell$. □

Finally, assume that $|\Pr[\mathrm{REAL}_{\mathsf{spcg},\mathcal{Z}(z),\mathcal{A}} = 1] - \Pr[\mathrm{IDEAL}_{\mathcal{F}_{\mathsf{spcg}},\mathcal{Z}(z),\mathcal{S}} = 1]| = \delta$. By Claim A.18 there exists $i \in \{1, \ldots, 2^\ell\}$ such that $|\Pr[\mathrm{REAL}_{\mathsf{spcg},\mathcal{Z}_i(z_i),\mathcal{A}} = 1] - \Pr[\mathrm{IDEAL}_{\mathcal{F}_{\mathsf{spcg}},\mathcal{Z}_i(z_i),\mathcal{S}} = 1]| \geq \delta/2^\ell$. By Claim A.17 it follows that $|\Pr[\mathrm{REAL}_{\mathsf{spcg},\mathcal{Z}_i(z_i),\mathcal{A}} = 1] - \Pr[\mathrm{IDEAL}_{\mathcal{F}_{\mathsf{spcg}},\mathcal{Z}_i(z_i),\mathcal{S}} = 1]| < \ell\epsilon'$. We conclude that $|\Pr[\mathrm{REAL}_{\mathsf{spcg},\mathcal{Z}(z),\mathcal{A}} = 1] - \Pr[\mathrm{IDEAL}_{\mathcal{F}_{\mathsf{spcg}},\mathcal{Z}(z),\mathcal{S}} = 1]| < 2^\ell \cdot \ell\epsilon'$.

A.7.4 Corrupt Alice, Honest Bob

Offline round. $\mathcal{S}$ simulates the actions of an honest Bob. That is, $\mathcal{S}$ queries crs from $\mathcal{F}_{\mathsf{crs}}$, samples a random string r for the psm protocol, commitments and openings $(C'_{i,x}, o'_{i,x}) \leftarrow \mathsf{commit}_{\mathsf{crs}}(\mathsf{psm}_i(x, r))$ for each $x \in \{0,1\}$ and $i \in \{1, \ldots, \ell\}$, and random shifts σ_i for any $i \in \{1, \ldots, \ell\}$, like an honest Bob. $\mathcal{S}$ sends the commitments $\{C'_{i,\sigma_i(x)}\}_{i\in\{1,\ldots,\ell\},x\in\{0,1\}}$ to the adversary on behalf of Bob.

Online round. $\mathcal{S}$ receives

$$\big(\mathbf{b}^B, \delta^B\big) \quad \text{and} \quad (C_1, \ldots, C_{L_B})$$

from the leakage of the ideal functionality $\mathcal{F}_{\mathsf{spcg}}$. $\mathcal{S}$ holds the input of Bob, and simulates the messages that Bob sends, and then receives from $\mathcal{A}$ the massages that the corrupt Alice sends. At the end of the simulation, $\mathcal{S}$ computes the honest parties output, denoted v. If v is "Alice is corrupt", then $\mathcal{S}$ inputs $\mathsf{flag} := 1$ to $\mathcal{F}_{\mathsf{spcg}}$ (the rest of the inputs do not matter). Otherwise, when v is not "Alice is corrupt", we split into cases.

1. If the corrupt Alice broadcasted $(\mathbf{a}, \delta^A, \{i, o_i, b_i^A\}_{i:\delta_i^A=1})$, let $\bar{o}_i := o_i$ and $\bar{b}_i^A := b_i^A$ for each $i \in \delta^A$, and note that $\bar{b}_i^A = \mathsf{open}_{\mathsf{crs}}(C_i, \bar{o}_i)$ (or otherwise the honest parties would output "Alice is corrupt"). For each $i \notin \delta^A$, set $\bar{b}_i^A := 0$ and $\bar{o}_i := 0$. $\mathcal{S}$ inputs $(\mathbf{a}, \delta^A, \{\bar{o}_i, \bar{b}_i^A\}_{i=1}^{L_B})$, $\mathsf{flag} := 0$ and $\mathsf{reveal} := 1$ to $\mathcal{F}_{\mathsf{spcg}}$.

2. Otherwise, the corrupt Alice has broadcasted $\{(i_j, o'_j, s_j)\}_{j \in \{1,\ldots,\ell_A\}}$. For each $j \in \{1, \ldots, \ell_A\}$, $\mathcal{S}$ takes the index i_j broadcasted by the corrupt Alice, and sets $x_j := \sigma_j^{-1}(i_j)$. $\mathcal{S}$ then sets $x^A := (x_1, \ldots, x_{\ell_A}) \in \{0,1\}^{\ell_A}$, and parses $x^A = (\mathbf{a}, \delta^A, \{o_i, b_i^A\}_{i=1}^{L_B})$. $\mathcal{S}$ inputs $(\mathbf{a}, \delta^A, \{o_i, b_i^A\}_{i=1}^{L_B})$, $\mathsf{flag} := 0$ and $\mathsf{reveal} := 0$ to the ideal functionality.

Fix any polynomial time environment $\mathcal{Z}$ and input ζ to the environment. We show that the view of $\mathcal{Z}$ in the real world is close to the view of $\mathcal{Z}$ in the ideal world.

$\mathcal{Z}$'s view. Since $\mathcal{S}$ receives the inputs of the honest parties, and acts like an honest Bob in both rounds, it perfectly simulate the view of $\mathcal{Z}$. It remains to show that the output of the honest parties has the same distribution in both worlds.

Honest parties' outputs. We say that a view View is "good" if for any commitment C from the view, and for any two openings o, and o' that appear in the view, it holds that either $\mathsf{open}_{\mathsf{crs}}(C, o) = \perp$ or $\mathsf{open}_{\mathsf{crs}}(C, o') = \perp$ or $\mathsf{open}_{\mathsf{crs}}(C, o) = \mathsf{open}_{\mathsf{crs}}(C, o)$. By the binding property, a view View is good with probability at least $1 - \epsilon$.

For every good View, the outputs of the honest parties can be extracted from View by the efficient deterministic process, that simply outputs what the honest parties would output given the broadcasts of Alice and Bob from View. The process clearly works for a good View obtained from the real-world. For a good View obtained from the ideal-world, we split into cases.

- If the output according to View is "Alice is corrupt", then $\mathcal{S}$ sets $\mathsf{flag} = 1$, so the ideal-world output is also "Alice is corrupt".
- Otherwise, if Alice broadcasts $(\mathbf{a}, \delta^A, \{i, o_i, b_i^A\}_{i:\delta_i^A=1})$ and the output according to View is $(\mathbf{a}, \delta^A, \{i, o_i, b_i^A\}_{i \in \delta^A}, \sum_{i \in \delta^A} \alpha_i(\mathbf{a}) \cdot b_i^A)$, then it must hold that $b_i^A = \mathsf{open}_{\mathsf{crs}}(C_i, o_i)$ for all $i \in \delta^A$ (or otherwise Alice would be discarded). It is not hard to see that the ideal-world output is also $(\mathbf{a}, \delta^A, \{i, o_i, b_i^A\}_{i \in \delta^A}, \sum_{i \in \delta^A} \alpha_i(\mathbf{a}) \cdot b_i^A)$.
- Otherwise, if Alice broadcasts $\{(i_j, o'_j, s_j)\}_{j \in \{1,\ldots,\ell_A\}}$ and the output according to View is $\big(\mathbf{a}, \delta^A, \delta^B, \{i, o_i, b_i^A\}_{i \in I}, \sum_{i \in J} \alpha_i(\mathbf{a}) \cdot b_i^A\big)$, where $I := (\mathbf{b}^A, \delta^A) \diamond (\mathbf{b}^B, \delta^B)$, then it must hold that $b_i^A = \mathsf{open}_{\mathsf{crs}}(C_i, o_i)$ for all $i \in I$, and $\mathsf{open}_{\mathsf{crs}}(C'_{j,i_j}, o'_j) = s_j$ for any $j \in \{1, \ldots, \ell_A\}$ (or otherwise Alice would be discarded). Since View is good, and by the correctness of the underlying PSM protocol, we conclude that the output in the ideal-world is also $\big(\mathbf{a}, \delta^A, \delta^B, \{i, o_i, b_i^A\}_{i \in I}, \sum_{i \in J} \alpha_i(\mathbf{a}) \cdot b_i^A\big)$.

This concludes the case of corrupt Alice and honest Bob.

A.7.5 Honest Alice, Corrupt Bob

Offline round. In the offline round only the corrupt Bob communicates, and so $\mathcal{S}$ receives the messages that Bob sends to the honest parties from $\mathcal{A}$.

Online round. The adversary receives the inputs of the honest Alice, $(\mathbf{a}, \delta^A, \{o_i, b_i^A\}_{i=1}^{L_B})$, and so can simulate the online round broadcast of the honest Alice. $\mathcal{S}$ then receives from $\mathcal{A}$ the online round broadcast of the corrupt Bob.

At the end of the simulation $\mathcal{S}$ computes the output of the honest parties in the simulation. If the output is "Bob is corrupt" $\mathcal{S}$ inputs $\mathsf{flag} = 1$ to $\mathcal{F}_{\mathsf{spcg}}$ (the rest of the inputs do not matter). Otherwise, the output is not "Bob is corrupt", and $\mathcal{S}$ extracts the set I from the output. Let $J := \delta^A \setminus I$. For every $i \in J$, $\mathcal{S}$ sets $\delta_i^B := 1$ and $b_i^B := b_i^A$. For $i \notin J$, $\mathcal{S}$ sets and $\delta_i^B := 0$ and $b_i^B := 0$. $\mathcal{S}$ inputs $\delta^B, \mathbf{b}^B$ and $\mathsf{flag} := 0$ to $\mathcal{F}_{\mathsf{spcg}}$.

Fix any polynomial time environment $\mathcal{Z}$ and input ζ to the environment. We show that the view of $\mathcal{Z}$ in the real world is close to the view of $\mathcal{Z}$ in the ideal world.

$\mathcal{Z}$'s view. Since $\mathcal{S}$ receives the inputs of the honest parties, and acts like an honest Alice in both rounds, it perfectly simulate the view of $\mathcal{A}$. It remains to show that the output of the honest parties has the same distribution in both worlds.

Honest parties' output. We say that a view View is "good" if for any commitment C from the view, and for any two openings o, and o' that appear in the view, it holds that either $\mathsf{open}_{\mathsf{crs}}(C, o) = \bot$ or $\mathsf{open}_{\mathsf{crs}}(C, o') = \bot$ or $\mathsf{open}_{\mathsf{crs}}(C, o) = \mathsf{open}_{\mathsf{crs}}(C, o)$. By the binding property, a view View is good with probability at least $1 - \epsilon$. Whenever a view is good, the output in the ideal-world is equal to the output of the honest parties according to View. This concludes the case of honest Alice and corrupt Bob.

A.7.6 Corrupt Alice and Bob

Since the only parties that hold inputs and communicate are Alice and Bob, $\mathcal{S}$ can trivially simulate the view of the corrupt parties. At the end of the simulation $\mathcal{S}$ computes the output of the honest parties in the simulation, denoted z. $\mathcal{S}$ inputs z to $\mathcal{F}_{\mathsf{spcg}}$ (the rest of the inputs do not matter). It is not hard to see that the above simulator perfectly simulates spcg. This completes the case of corrupt Alice and Bob.

A.7.7 Complexity Analysis

Here we analyse the complexity of protocol spcg. In order to show that the complexity of spcg is $\mathrm{poly}(n, \log|\mathbb{F}|, \kappa, L_A, L_B)$, it is enough to show that the complexity of the underlying PSM protocol is $\mathrm{poly}(n, \log|\mathbb{F}|, \kappa, L_A, L_B)$. For this, according to Theorem 3.2 it is enough to show that f_{spcg} has a circuit with size polynomial in $(n, \log|\mathbb{F}|, \kappa, L_A, L_B)$ and depth logarithmic in $(n, \log|\mathbb{F}|, \kappa, L_A, L_B)$.

First, observe that $\ell = \mathrm{poly}(\log|\mathbb{F}|, L_A, L_B, \kappa)$. It is not hard to see that the outputs $\mathbf{a}, \delta^A, \delta^B$ can be computed by a constant depth circuit. We represent the list $\{i, o_i^A, b_i^A\}_{i \in I}$ as a concatenation of L_B binary strings, $\chi_1, \ldots, \chi_{L_B}$, each of length $\mathrm{poly}(\kappa, \log|\mathbb{F}|)$, such that the first bit of χ_{L_i}, denoted β_i, indicates whether $i \in I$. When $\beta_i = 1$ the rest of the string encodes (o_i^A, b_i^A), and otherwise, when $\beta_i = 0$ it encodes the all-zero string. It is not hard to see that computing the output can be reduced to computing the β_i's, which are computed in the following way. For each i we use a comparator to check whether $b_i^A = b_i^B$ and get a bit $\gamma_{i,1}$, we check whether $\delta_i^A = \delta_i^B = 1$ and get a bit $\gamma_{i,2}$, we check whether $\delta_i^A = 1, \delta_i^B = 0$ and get a bit $\gamma_{i,3}$, and finally we set $\beta_i = (\gamma_{i,1} \wedge \gamma_{i,2}) \vee \gamma_{i,3}$.

As the comparator can be computed in depth $\log\log|\mathbb{F}|$, and $\gamma_{i,2}, \gamma_{i,3}$ can be computed in constant depth, then each β_i can be computed in depth $\log\log|\mathbb{F}|$, as required.

It remains to show how to compute the sum $\sum_{i\in J}\alpha_i(\mathbf{a})\cdot b_i$. First, it is not hard to see that given δ^A and the β_i's we can compute an indicator bit η_i to whether $i\in J$ or not in constant depth. Since computing each $\alpha_i(\mathbf{a})$ is in NC^1, and multiplying field elements is in NC^1 (see [41]), we can compute the field element $c_i := \eta_i\cdot\alpha_i(\mathbf{a})\cdot b_i$ for each $i\in L_B$ with logarithmic depth. Finally, computing the sum of all the c_i's requires depth logarithmic in L_B, as required. This completes the complexity analysis, and completes the proof. $\square$

B Appendix: 3-round MPC with Everlasting Security

In the following sections we prove the security of our protocols. In all proofs except that of spcg, the same proof that shows that a protocol π securely implements a functionality $\mathcal{F}$ when the underlying commitment scheme is computationally-hiding, also shows that π securely implements $\mathcal{F}$ *with everlasting security* when the underlying commitment scheme is statistically-hiding, simply by changing computational-indistinguishability to statistical-distance throughout the proof. Thus, we unify notation and say that random variables X and Y are ϵ-close, which means that X and Y are ϵ-indistinguishable when the underlying commitment scheme is computationally-hiding, or that X and Y are ϵ-close in statistical distance, when the underlying commitment scheme is statistically-hiding. For protocol spcg we provide a different proof for each case.

Throughout, we denote by View the tuple consists of the randomness of the environment, the messages that the corrupt parties sent and received, and the inputs of the honest parties (which are picked by the environment). We denote by ϵ the error term of the commitment scheme, where $\epsilon = \mathsf{negl}(\kappa)$.

B.1 Cryptographic VSS

Proof of Theorem 3.9. In this section we prove that protocol vss UC-emulates $\mathcal{F}_{\mathsf{vss}}$ (with everlasting security when the underlying commitment scheme is statistically-hiding). Let $\mathcal{A}$ be an efficient adversary against vss. We define the simulator $\mathcal{S}$ as follows. $\mathcal{S}$ uses $\mathcal{A}$ in a black-box manner, and forwards all messages between $\mathcal{Z}$ and $\mathcal{A}$. $\mathcal{S}$ first receives the set of corrupt C parties from $\mathcal{Z}$. We split into two cases.

B.1.1 Honest Dealer

Sharing phase. $\mathcal{S}$ receives $(\mathbf{C}, \mathbf{O}_i)$, for every $i\in\mathsf{C}$, from $\mathcal{F}_{\mathsf{vss}}$. $\mathcal{S}$ sends $\mathbf{C}$ to $\mathcal{A}$ as the broadcast of D, and $\mathbf{O}_i$ as the private message from D to P_i, for every $i\in\mathsf{C}$. In addition, on behalf of every honest P_i, $\mathcal{S}$ sets $\{(G_{ij}, h_{ij}) := \mathsf{commit}_{\mathsf{crs}}(g_{ij}, r_{ij})\}_{j\in\{0,\ldots,n\}}$, where r_{ij} is a fresh random string and g_{ij} is a random field element, and sends $\mathbf{G}_i := \{G_{ij}\}_{j\in\{0,\ldots,n\}}$ to $\mathcal{A}$ as the broadcast of P_i. This completes the communication from honest parties to corrupt parties in the first round. At this stage, $\mathcal{S}$ receives from $\mathcal{A}$ the messages that every corrupt party P_i sends, that is, the broadcast $\mathbf{G}_i$ and the private messages $\{(g_{ij}, h_{ij})\}_{j\in\{0,\ldots,n\}}$ to D.

Verification phase. $\mathcal{S}$ receives the input bits of the honest parties, $\{\mathsf{flag}_i\}_{i\in\mathsf{H}}$ as a leakage from $\mathcal{F}_{\mathsf{vss}}$. Since D is honest, we are promised that all those bits are 0. $\mathcal{S}$ simulates the honest parties (except the dealer D) by not sending any message. $\mathcal{S}$ simulates D in the following way.

- For every honest P_i, $\mathcal{S}$ broadcasts $\{\alpha_{ij}\}_{j\in\{0,\ldots,n\}}$, where each α_{ij} is a random string.
- For every corrupt P_i, $\mathcal{S}$ first verifies if $g_{ij} \overset{?}{=} \mathsf{open}_{\mathsf{crs}}(G_{ij}, h_{ij})$ for $j \in \{0,\ldots,n\}$ and becomes *unhappy* with P_i if the check fails for some j. $\mathcal{S}$ broadcasts $\{o_{ij}\}_{j\in\{0,\ldots,n\}}$ on behalf of D when *unhappy*, and otherwise broadcasts $\{\alpha_{ij}\}_{j\in\{0,\ldots,n\}}$, such that $\alpha_{ij} := o_{ij} + g_{ij}$.

At this stage, $\mathcal{S}$ receives from $\mathcal{A}$ the message that every corrupt party P_i sends. Some corrupt parties might send no message, while others broadcast some values $\{(g'_{ij}, h'_{ij})\}_{j\in\{0,\ldots,n\}}$. For every corrupt P_i such that $g'_{ij} = \mathsf{open}_{\mathsf{crs}}(G_{ij}, h'_{ij})$ for all $j \in \{0,\ldots,n\}$, $\mathcal{S}$ sets $\mathsf{flag}_i := 1$, and for all other corrupt parties $\mathcal{S}$ sets $\mathsf{flag}_i := 0$. Finally, $\mathcal{S}$ inputs $\{\mathsf{flag}_i\}_{i\in\mathsf{C}}$ to $\mathcal{F}_{\mathsf{vss}}$.

We assume without loss of generality that $\mathcal{A}$ is the dummy adversary (see Section A.2), and we fix a polynomial-time environment $\mathcal{Z}$ with input ζ. We may assume without loss of generality that $\mathcal{Z}$ is deterministic. We begin by showing that the view of $\mathcal{Z}$ in the real-world is close to the view of $\mathcal{Z}$ in the ideal world.

$\mathcal{Z}$'s view. $\mathcal{Z}$'s view consists of (1) the common reference string crs, (2) the inputs to the honest dealer, (3) broadcast $\mathbf{C}$ and messages $\{\mathbf{O}_i\}_{i\in\mathsf{C}}$ from D, (4) the broadcasts $\{\mathbf{G}_i\}_{i\in\{1,\ldots,n\}}$, (5) verification phases' inputs of the honest parties, and (6) the verification phase broadcasts the parties. In order to prove that the real-world view is close to the ideal-world view, we define the following hybrid-worlds, where we assume that the honest parties know the set H.

- In Hybrid 1, the honest parties act like in the real-world, except that in round 2, (a) an honest party P_i never sends a broadcast message, (b) for every $i \in \mathsf{H}$, D does not verify that $\mathsf{open}_{\mathsf{crs}}(G_{ij}, h_{ij}) \overset{?}{=} g_{ij}$ for $j \in \{0,\ldots,n\}$, but D is always happy with P_i, and (c) for every $i \in \mathsf{H}$, the computation of $\alpha_{i0},\ldots,\alpha_{in}$ by D is done using the values $g_{i0},\ldots,g_{in}$ that P_i sends to D (and not the values extracted from the commitments).
- In Hybrid 2, the honest parties act like in Hybrid 1, except that in the first round, an honest P_i samples random $(g_{i0},\ldots,g_{in})$ and $(g'_{i0},\ldots,g'_{in})$, samples $(G_{ij}, h_{ij}) \leftarrow \mathsf{commit}_{\mathsf{crs}}(g'_{ij}; r_{ij})$, where r_{ij} is a fresh random string, broadcasts $G_{i0},\ldots,G_{in}$, and sends $\{(g_{ij}, h_{ij})\}_{j\in\{0,\ldots,n\}}$ to D (so that value g_{ij} will be used in the computation of α_{ij}).

Real-world vs. Hybrid 1. We claim that the real-world view has the same distribution as in Hybrid 1. This follows by noting that in the real-world (a) an honest P_i is always happy with an honest D, and we are promised that $\mathsf{flag}_i = 0$, so P_i never sends a broadcast message in round 2, (b) D is always happy with an honest P_i, and (c) for every honest P_i it holds that $g_{ij} = \mathsf{open}_{\mathsf{crs}}(G_{ij}, h_{ij})$, so α_{ij} is computed in the same way in both worlds.

Hybrid 1 vs. Hybrid 2. We claim that the view in Hybrid 1 is $O(n^2\epsilon)$-close to the view in Hybrid 2. Indeed, the random variables $(\mathsf{crs}, \{(g_{i0},\ldots,g_{in})\}_{i\in\mathsf{H}})$ have the same distribution in both hybrids, where g_{ij}'s are the values sent from P_i to D in the first round. Fix those values, and note that the Hybrid 1 random variables $\{\mathbf{G}_i\}_{i\in\mathsf{H}}$ are $O(n^2\epsilon)$-close to the corresponding Hybrid 2 random variables. Finally, one can verify that in both hybrids the rest of view can be obtained from $(\mathsf{crs}, \{\mathbf{G}_i\}_{i\in\mathsf{H}}, \{(g_{i0},\ldots,g_{in})\}_{i\in\mathsf{H}})$ by the same efficient process. We conclude that the view in Hybrid 1 is $O(n^2\epsilon)$-close to the view in Hybrid 2.

Hybrid 2 vs. ideal-world. We claim that the view in Hybrid 2 has the same distribution as the ideal-world view. This follows by noting that the random variables $(\mathsf{crs}, \{\mathbf{G}_i\}_{i \in \mathsf{H}})$ have the same distribution in both worlds, and that the rest of the view can be obtained by the same efficient process.

We conclude that the real-world view is $O(n^2\epsilon)$-close to the ideal-world view. This completes the analysis of $\mathcal{Z}$'s view. We continue by analysing the honest parties' outputs.

Honest parties' outputs. We say that a view View is "good" if for any commitment C from the view, and for any two openings o, and o' that appear in the view, it holds that either $\mathsf{open}_{\mathsf{crs}}(C, o) = \bot$ or $\mathsf{open}_{\mathsf{crs}}(C, o') = \bot$ or $\mathsf{open}_{\mathsf{crs}}(C, o) = \mathsf{open}_{\mathsf{crs}}(C, o)$. By the binding property, a view View is good with probability at least $1 - \epsilon$.

First, note that whenever View is good then D is not discarded in the real-world. Now, for every good View, it is not hard to see that both in the real-world and the ideal-world, the outputs of the honest parties can be extracted from View by the following efficient deterministic process: (1) in the sharing phase each honest P_i outputs $(\mathbf{C}, \mathbf{O}_i)$, where $(\mathbf{C}, \mathbf{O})$ are the inputs of V according to View, (2) in the verification phase W is the set of all corrupt parties that are in conflict with D according to View, and each honest party outputs $(\mathsf{W}, \mathbf{O}_{\mathsf{W}})$. This completes the case of an honest dealer.

B.1.2 Corrupt Dealer

Sharing phase. $\mathcal{S}$ begins the simulation of the first round by taking the role of the honest parties, computing their messages in the first round, and transferring messages from honest parties to corrupt parties to $\mathcal{A}$. Then, the dealer receives from $\mathcal{A}$ the messages from the corrupt parties to the honest parties, and gives them to the simulated honest parties.

Let $\mathbf{C}$ be the corrupt dealer's broadcast, and let $\mathbf{O}_i$ be the openings that the dealer sent to an honest P_i. $\mathcal{S}$ sets $\mathbf{O}_i := (\bot, \ldots, \bot)$ for any $i \in \{0, \ldots, n\} \setminus \mathsf{H}$, sets $\mathbf{O} := \{\mathbf{O}_i\}_{i \in \{0,\ldots,n\}}$, and inputs $(\mathbf{C}, \mathbf{O})$ to $\mathcal{F}_{\mathsf{vss}}$.

Verification phase. The honest parties' inputs $\{\mathsf{flag}_i\}_{i \in \mathsf{H}}$ are leaked to $\mathcal{S}$ from $\mathcal{F}_{\mathsf{vss}}$. $\mathcal{S}$ continues to simulate the honest parties using the leaked inputs $\{\mathsf{flag}_i\}_{i \in \mathsf{H}}$. That is, $\mathcal{S}$ computes the messages sent by the honest parties, and transfers messages from honest parties to corrupt parties to $\mathcal{A}$. Then, $\mathcal{S}$ receives from $\mathcal{A}$ the messages from the corrupt parties to the honest parties, and gives them to the simulated honest parties. Finally, the dealer computes the output of the honest parties in the simulation.

At the end of the simulation, if the output of the honest parties is "D is corrupt" then $\mathcal{S}$ inputs $\mathsf{flag}_D := 1$ to the functionality (the other inputs do not matter). Otherwise $\mathcal{S}$ sets $\mathsf{flag}_D := 0$, and computes the set W of parties conflicted with the dealer in the simulation. For every $i \in \mathsf{W} \cap \mathsf{C}$ $\mathcal{S}$ sets $\mathsf{flag}_i := 1$, and for every other corrupt P_i the dealer sets $\mathsf{flag}_i := 0$. For every $i \in \mathsf{W}$ and $j \in \{0, \ldots, n\}$, $\mathcal{S}$ computes $\bar{o}_{ij}$ like an honest party in the simulation, and for $i \in \{0, \ldots, n\} \setminus \mathsf{W}$ and $j \in \{0, \ldots, n\}$ $\mathcal{S}$ sets $\bar{o}_{ij} := \bot$. $\mathcal{S}$ sets $\bar{\mathbf{O}} := \{o_{ij}\}_{i,j \in \{0,\ldots,n\}}$ and inputs $\{\mathsf{flag}_i\}_{i \in \mathsf{C}}$, flag_D and $\bar{\mathbf{O}}$ to $\mathcal{F}_{\mathsf{vss}}$.

Fix any polynomial-time environment $\mathcal{Z}$ and input ζ to the environment. We show that the view of $\mathcal{Z}$ in the real-world is close to the view of $\mathcal{Z}$ in the ideal-world.

$\mathcal{Z}$'s view. Since $\mathcal{S}$ always holds all the honest parties inputs, and perfectly emulates the honest parties in an execution of vss, then the corrupt parties' view has the same distribution as the corrupt parties' view in the real-world. This concludes the analysis of the $\mathcal{Z}$'s view.

Honest parties' outputs. We say that a view View is "good" if for any commitment C from the view, and for any two openings o, and o' that appear in the view, it holds that either $\mathsf{open}_{\mathsf{crs}}(C, o) = \perp$ or $\mathsf{open}_{\mathsf{crs}}(C, o') = \perp$ or $\mathsf{open}_{\mathsf{crs}}(C, o) = \mathsf{open}_{\mathsf{crs}}(C, o)$. By the binding property, a view View is good with probability at least $1 - \epsilon$.

We claim that for every every good View, both in the real-world and the ideal-world the outputs of the honest parties can be extracted from View by the following efficient deterministic process: (1) in the sharing phase each honest P_i outputs $(\mathbf{C}, \mathbf{O}_i)$, where $\mathbf{C}$ is the broadcast of D in the first round, and $\mathbf{O}_i$ was sent from D to P_i in the first round, (2) in the verification phase, if D is discarded according to View then every honest party outputs "D is corrupt"; otherwise, W is the set of all parties that are in conflict with D according to View, and each honest party outputs $(\mathsf{W}, \bar{\mathbf{O}}_{\mathsf{W}})$, where $\bar{\mathbf{O}}_{\mathsf{W}}$ is computed from View as defined by the protocol.

This process clearly works for a good View obtained from the real-world. For a good View obtained from the ideal-world, note that the output of each honest P_i in the sharing phase is indeed $(\mathbf{C}, \mathbf{O}_i)$. In addition, whenever D is discarded according to View then $\mathcal{S}$ sets $\mathsf{flag}_D = 1$, so all honest parties output "D is corrupt". It remains to show that whenever D is not discarded, the tuple $(\mathsf{W}, \mathbf{C}, \bar{\mathbf{O}}_{\mathsf{W}}, \mathbf{O}_{\mathsf{H}\setminus\mathsf{W}})$ is a weak double t-sharing of some value s, and the parties indeed output $(\mathsf{W}, \bar{\mathbf{O}}_{\mathsf{W}})$ as defined by the above procedure.

Observe that for every $i \in \mathsf{W}$ the pair $(\mathbf{C}, \bar{\mathbf{O}}_i)$ is valid, or otherwise the honest parties would output "D is corrupt". In addition, for any $i \in \mathsf{H} \setminus \mathsf{W}$ the pair $(\mathbf{C}, \mathbf{O}_i)$ is also valid, or otherwise P_i would be unhappy with D, and so P_i will be in W. In addition, since View is good, and D is not discarded, then $\mathsf{open}_{\mathsf{crs}}(C_{ij}, o'_{ij}) = \mathsf{open}_{\mathsf{crs}}(C_{ji}, o'_{ji})$ for every $i, j \in \mathsf{W} \cup \mathsf{H}$, where $o'_{ij} = \bar{o}_{ij}$ if $i \in \mathsf{W}$, and $o'_{ij} = o_{ij}$ otherwise. This completes the proof.

□

B.2 Guided Linear Function Computation

Proof of Lemma 3.10. In this section we prove that protocol $\mathsf{glinear}$ *perfectly* UC-emulates $\mathcal{F}_{\mathsf{glinear}}$ in the $\mathcal{F}_{\mathsf{spcg}}$-hybrid model. From the composition properties of UC-security, this implies that protocol $\mathsf{glinear}$ UC-emulates $\mathcal{F}_{\mathsf{glinear}}$ (with everlasting security if the underlying commitment scheme is statistically-hiding). Let $\mathcal{A}$ be an efficient adversary against $\mathsf{glinear}$. We define the simulator $\mathcal{S}$ as follows. The simulator $\mathcal{S}$ uses $\mathcal{A}$ in a black-box manner, and forwards all messages between $\mathcal{Z}$ and $\mathcal{A}$. The simulator first receives the set of corrupt C parties from $\mathcal{Z}$. We split into cases.

B.2.1 Honest Guide

Offline round. In the $\mathcal{F}_{\mathsf{spcg}}$-hybrid model there is no communication in the offline round.

Online round. The simulator receives the following leakage from $\mathcal{F}_{\mathsf{glinear}}$: (1) the commitments $(C_{ij})_{i\in\{1,\ldots,m\},j\in\{0,\ldots,n\}}$, (2) the honest parties' indicator vectors $\{\delta^i\}_{i\in\mathsf{H}}$ and δ^G, (3) parts of the dealer's inputs $\mathbf{a}$, $\{b^G_{i,j}, o^G_{i,j}\}_{i\in\{1,\ldots,m\},j\in\mathsf{C}}$, (4) the values and openings $L_j = \{(i,j), b^G_{ij}, o^G_{ij}\}_{i\in I_j}$ for

$j \in \mathsf{H}$, and (5) $v_j := \sum_{i\in\delta^G} a_i b^A_{ij}$ for any $j \in \{1,\ldots,n\}$. In addition, the simulator receives the output of $\mathcal{F}_{\mathsf{glinear}}$, $v := \sum_{i\in\delta^G} a_i b^A_{i0}$.

For each $j \in \{1,\ldots,n\}$, the simulator simulates the call to $\mathcal{F}^j_{\mathsf{spcg}}$ as follows.

- If $j \in \mathsf{H}$ then the simulator gives $(C_{1j},\ldots,C_{mj})$ to $\mathcal{A}$ as the leakage from $\mathcal{F}^j_{\mathsf{spcg}}$, and $(\mathbf{a}, \delta^G, \delta^j, \{i, o^G_{ij}, b^G_{ij}\}_{i\in I_j}, v_j - \sum_{i\in I_j} a_i b^G_{ij})$ as the output from $\mathcal{F}^j_{\mathsf{spcg}}$.
- If $j \in \mathsf{C}$ then the simulator gives the adversary the honest parties' inputs $(C_{1j},\ldots,C_{mj})$, $\mathbf{a}$, $\{b^G_{ij}\}_{i\in\{1,\ldots,m\}}$, δ^G, and $\{o^G_{ij}\}_{i\in\{1,\ldots,m\}}$ as leakage from $\mathcal{F}^j_{\mathsf{spcg}}$. Later, the simulator receives Bob's inputs $\mathbf{b}^B$, δ^B, flag, reveal and z from $\mathcal{A}$. Upon receiving Bob's inputs, the simulator, that holds all inputs to $\mathcal{F}^j_{\mathsf{spcg}}$, computes the output of the functionality, and returns it to $\mathcal{A}$.

This completes the simulation.

Fix any polytime environment $\mathcal{Z}$ and input ζ to the environment. We show that the view of $\mathcal{Z}$ in the real world has the same distribution as the view of $\mathcal{Z}$ in the ideal world.

$\mathcal{Z}$'s view. Fix the crs. The honest parties' inputs are picked by $\mathcal{Z}$, and so they have the same distribution in both worlds. Fix those inputs. The only messages that the adversary receives are the leakage and the output from the various $\mathcal{F}_{\mathsf{spcg}}$ calls. It is not hard to see that, since we've fixed the honest parties' inputs, those messages are fixed for any $\mathcal{F}^j_{\mathsf{spcg}}$ such that $j \in \mathsf{H}$, and are the same in both worlds. Similarly, for $j \in \mathsf{C}$, the leakage is fixed, and is the same in both worlds. After receiving all leakage from the various calls to $\mathcal{F}_{\mathsf{spcg}}$, and the outputs of $\mathcal{F}^j_{\mathsf{spcg}}$ for $j \in \mathsf{H}$, the adversary picks the inputs to $\mathcal{F}^j_{\mathsf{spcg}}$ for $j \in \mathsf{C}$ in the same way in both worlds, and the simulator computes the output of $\mathcal{F}^j_{\mathsf{spcg}}$ like in the real-world, so the output of $\mathcal{F}^j_{\mathsf{spcg}}$ has the same distribution in both worlds. We conclude that the $\mathcal{Z}$'s view has the same distribution in both worlds.

Honest parties' output. Fix any view View. In the ideal world the output of the honest parties is $(\mathbf{a}, \delta^G, \sum_{i\in\delta^G} a_i b^G_{i0})$ with probability 1. We show that this is also the case in the real world. Indeed, for every $j \in \mathsf{H}$, the functionality $\mathcal{F}^j_{\mathsf{spcg}}$ returns $(\mathbf{a}, \delta^G, \{i, o^G_{ij}, b^G_{ij}\}_{i\in I_j}, \sum_{i\in\delta^G} a_i b^G_{ij})$ to all honest parties. In addition, for $j \in \mathsf{C}$, the output of $\mathcal{F}^j_{\mathsf{spcg}}$ is either "Bob is corrupt" or $(\mathbf{a}, \delta^G, \{i, o^G_{ij}, b^G_{ij}\}_{i\in I_j}, \sum_{i\in\delta^G} a_i b^G_{ij})$. Let K be the set of all indices j such that $\mathcal{F}^j_{\mathsf{spcg}}$ returns $(\mathbf{a}, \delta^G, \{i, o^G_{ij}, b^G_{ij}\}_{i\in I_j}, \sum_{i\in\delta^G} a_i b^G_{ij})$. In particular, K includes all $n - t \geq t + 1$ honest parties. For $k \in K$, let $v_k := \sum_{i\in\delta^G} a_i b^G_{ik}$. As we are promised that for every $i \in \delta^G$ the shares $b^G_{i0},\ldots,b^G_{in}$ correspond to a degree t polynomial $g_i(x)$, it follows that the shares $\{v_k\}_{k\in K}$ correspond to a degree-t polynomial $g(x) := \sum_{i\in\delta^G} a_i g_i(x)$, whose free coefficient is $g(0) = \sum_{i\in\delta^G} a_i g_i(0) = \sum_{i\in\delta^G} a_i b^G_{i0}$. Therefore, all honest parties output $(\mathbf{a}, \delta^G, \sum_{i\in\delta^G} a_i b^G_{i0})$ with probability 1. This concludes the case of an honest guide.

B.2.2 Corrupt Guide

Offline round. In the $\mathcal{F}_{\mathsf{spcg}}$-hybrid model there is no communication in the offline round.

Online round. The simulator receives the following leakage from $\mathcal{F}_{\mathsf{glinear}}$: (1) the commitments $(C_{ij})_{i\in\{1,\dots,m\},j\in\{0,\dots,n\}}$ and (2) the honest parties' inputs $\{\mathbf{b}^i, \delta^i\}_{i\in\mathsf{H}}$. The simulator, that holds the inputs of all honest parties, takes the role of the honest parties and simulates their call to $\mathcal{F}_{\mathsf{spcg}}$. In particular, for every $j \in \{1,\dots,n\}$ the simulator sends $(C_{1j},\dots,C_{mj})$, as the leakage from $\mathcal{F}^j_{\mathsf{spcg}}$, and in addition, for $j \in \mathsf{H}$, the the simulator sends $\mathbf{b}^j$ and δ^j as the additional leakage. Later, at an order which is determined by $\mathcal{A}$, it holds that (1) for every j the simulator receives Alice's input to $\mathcal{F}^j_{\mathsf{spcg}}$ from $\mathcal{A}$, denoted $(\mathbf{a}^{A,j}, \mathbf{b}^{A,j}, \delta^{A,j}, \{o_i^{A,j}\}_{i\in\{0,\dots,m\}}, \mathsf{flag}^{A,j}, \mathsf{reveal}^{A,j}, z^{A,j})$, and (2) for $j \in \mathsf{C}$, the simulator receives Bob's input to $\mathcal{F}^j_{\mathsf{spcg}}$ from $\mathcal{A}$, denoted $\mathbf{b}^{B,j}, \delta^{B,j}$. Upon receiving both Alice and Bob's input to $\mathcal{F}^j_{\mathsf{spcg}}$, the simulator computes the output of $\mathcal{F}^j_{\mathsf{spcg}}$ and gives it to $\mathcal{A}$. After all calls to $\mathcal{F}_{\mathsf{spcg}}$ were concluded, the simulator continues to simulate the honest parties, and computes their output.

If the output of the honest parties is "G is corrupt" then the simulator inputs $\mathsf{flag} = 1$ to $\mathcal{F}_{\mathsf{glinear}}$ (the rest of the inputs don't matter). Otherwise, the simulator sets $\mathsf{flag} := 0$. Observe that in this case, for every $j \in \mathsf{H}$ the vectors $\mathbf{a}^{A,j}$ and $\delta^{A,j}$ are the same (or otherwise the guide would be discarded), and we denote them by $\mathbf{a}$ and δ^G, respectively. The simulator sets $b^G_{ij} := b_i^{A,j}$ and $o^G_{ij} := o_i^{A,j}$ for every $i \in \{1,\dots,m\}$ and $j \in \{0,\dots,n\}$. The simulator inputs $(\mathbf{a}, \mathbf{b}^G, \delta^G, \{o^G_{ij}\}_{i\in\{1,\dots,m\},j\in\{0,\dots,n\}})$ and flag to $\mathcal{F}_{\mathsf{glinear}}$.

Fix any polynomial-time environment $\mathcal{Z}$ and input ζ to the environment. We show that the view of $\mathcal{Z}$ in the real world is close to the view of $\mathcal{Z}$ in the ideal world.

$\mathcal{Z}$'s view. In both worlds the honest parties' inputs are picked by $\mathcal{Z}$, so they have the same distribution and we fix them. The only messages that the adversary receives are the leakage and the output from the various $\mathcal{F}_{\mathsf{spcg}}$ calls. Since the simulator holds the honest parties inputs, and since the simulator simulates the honest parties exactly like in the real-world, those messages have the same distribution as in the real-world. This concludes the analysis of the $\mathcal{Z}$'s view.

Honest parties' output. Fix any view View of the adversary. If the output of the honest parties in the real-world according to View is "G is corrupt" then the simulator sets $\mathsf{flag} := 1$, and so the output of the honest parties in the ideal-world is "G is corrupt". Therefore, from now on we assume that the output of the honest parties according to View is not "G is corrupt".

Denote by K the set of all indices $k \in \{1,\dots,n\}$ such that the output of $\mathcal{F}^k_{\mathsf{spcg}}$ is not "Bob is corrupt", and observe that $\mathsf{H} \subseteq K$. Since the guide is not discarded then (1) there is no $\mathcal{F}^j_{\mathsf{spcg}}$ whose output is "Alice is corrupt", (2) the dealer sent the same $\mathbf{a}$ and δ^G to any $\mathcal{F}^k_{\mathsf{spcg}}$ for $k \in H$, (3) for every $k \in H$ and $i \in I_k$ it holds that $\mathsf{open}_{\mathsf{crs}}(C_{ik}, o_i^{A,k}) = b_i^{A,k}$, and (4) the shares $\{v^k\}_{k\in K}$ correspond to a degree-t polynomial, denoted $g(x)$, where v^k is the partial-sum obtained from the output of $\mathcal{F}^k_{\mathsf{spcg}}$. We conclude that in both worlds the output of the honest parties is $(\mathbf{a}, \delta^G, g(0))$, as required.

□

B.3 Triple Secret Sharing

Proof of Theorem 3.13. In this section we prove that protocol tss UC-emulates $\mathcal{F}_{\mathsf{tss}}$ (with everlasting security when the underlying commitment scheme is statistically-hiding). From the composition properties of UC-security, it is enough to prove security in the $(\mathcal{F}_{\mathsf{vss}}, \mathcal{F}_{\mathsf{glinear}})$-hybrid model. Let $\mathcal{A}$ be an efficient adversary against tss. We define the simulator $\mathcal{S}$ as follows. The simulator $\mathcal{S}$ uses $\mathcal{A}$ in a black-box manner, and forwards all messages between $\mathcal{Z}$ and $\mathcal{A}$. The simulator first receives the set of corrupt C parties from $\mathcal{Z}$. We split into cases.

B.3.1 Honest Dealer

Sharing phase. The simulator receives $(\bar{\mathbf{C}}^a, \bar{\mathbf{O}}_i^a)$, $(\bar{\mathbf{C}}^b, \bar{\mathbf{O}}_i^b)$, $(\bar{\mathbf{C}}^c, \bar{\mathbf{O}}_i^c)$, and $\mathsf{happy}_i = 1$, for every $i \in \mathsf{C}$, from $\mathcal{F}_{\mathsf{tss}}$. We sometimes denote the pairs by $(\bar{\mathbf{C}}^{a,0}, \bar{\mathbf{O}}_i^{a,0})$, $(\bar{\mathbf{C}}^{b,0}, \bar{\mathbf{O}}_i^{b,0})$, and $(\bar{\mathbf{C}}^{c,0}, \bar{\mathbf{O}}_i^{c,0})$, respectively.

For $v \in \{a, b\}$ the simulator picks (1) random strong double t-sharings $\langle\langle 0 \rangle\rangle$, denoted $(\bar{\mathbf{C}}^{v,k}, \bar{\mathbf{O}}^{v,k})$, for $k \in \{1, \ldots, n\}$, and (2) fixed sharings $(\bar{\mathbf{C}}^{v,k}, \bar{\mathbf{O}}^{v,k})$ defined by $(\bar{C}_{ij}^{v,k}, \bar{o}_{ij}^{v,k}) \leftarrow \mathsf{commit}_{\mathsf{crs}}(0; \vec{0})$ for every $v \in \{a, b\}$, $i, j \in \{0, \ldots, n\}$, and $k \in \{n+1, \ldots, 2n\}$, where $\vec{0}$ is the all-zero string. In addition, for $v = c$, the simulator picks random strong double t-sharings $\langle\langle 0 \rangle\rangle$, denoted $(\bar{\mathbf{C}}^{c,k}, \bar{\mathbf{O}}^{c,k})$ for any $k \in \{1, \ldots, 2n\}$. For $k \in \{0, \ldots, 2n\}$, the simulator simulates the calls to $\mathcal{F}_{\mathsf{vss}}^{a,k}, \mathcal{F}_{\mathsf{vss}}^{b,k}$ and $\mathcal{F}_{\mathsf{vss}}^{c,k}$ by giving the adversary the outputs that correspond to the corrupt parties, i.e., $\{(\bar{\mathbf{C}}^{a,k}, \bar{\mathbf{O}}_i^{a,k}), (\bar{\mathbf{C}}^{b,k}, \bar{\mathbf{O}}_i^{b,k}), (\bar{\mathbf{C}}^{c,k}, \bar{\mathbf{O}}_i^{c,k})\}_{k \in \{0, \ldots, 2n\}, i \in \mathsf{C}}$. For $v \in \{a, b, c\}$, $i \in \mathsf{C}$ and $k \in \{0, \ldots, 2n\}$ we denote by $\bar{f}_i^{v,k}(x)$ the degree-t polynomial that corresponds to $(\bar{\mathbf{C}}_i^{v,k}, \bar{\mathbf{O}}_i^{v,k})$.

The simulator picks random polynomials $\bar{A}(x), \bar{B}(x)$ and $\bar{C}(x)$ of degree n, n and $2n$, respectively, such that $\bar{A}(x) \cdot \bar{B}(x) = \bar{C}(x)$. For $k \in \{0, \ldots, 2n\}$, We denote their k-th coefficient by $\bar{A}^k$, $\bar{B}^k$ and $\bar{C}^k$, respectively, where $\bar{A}^k = \bar{B}^k = 0$ for $k > n$. For each $k \in \{0, \ldots, 2n\}$, the simulator picks random symmetric bivariate polynomials $\bar{F}^{a,k}(x, y)$, $\bar{F}^{b,k}(x, y)$ and $\bar{F}^{c,k}(x, y)$ of degree at most t in each variable, such that for $k \in \{0, \ldots, n\}$ we condition on

- $\bar{F}^{v,k}(x, i) = \bar{f}_i^{v,k}(x)$ for every $v \in \{a, b, c\}$ and $i \in \mathsf{C}$, and
- $\bar{F}^{a,k}(0, 0) = \bar{A}^k$, $\bar{F}^{b,k}(0, 0) = \bar{B}^k$, and $\bar{F}^{c,k}(0, 0) = \bar{C}^k$,

and for $k \in \{n+1, \ldots, 2n\}$ we condition on

- $\bar{F}^{a,k}(x, y) = 0$,
- $\bar{F}^{b,k}(x, y) = 0$, and
- $\bar{F}^{c,k}(x, i) = \bar{f}_i^{c,k}(x)$ for every $i \in \mathsf{C}$, and $\bar{F}^{c,k}(0, 0) = \bar{C}^k$.

In addition, for every $i \in \mathsf{H}$, the simulator samples a random string $\bar{z}_i$ and gives it to $\mathcal{A}$ as the broadcast of P_i. At this stage, the simulator receives from $\mathcal{A}$ the broadcasts $\bar{z}_i$ of the corrupt P_i's. Let $\bar{\mathbf{C}} := \{\bar{\mathbf{C}}^{v,k}\}_{v \in \{a,b,c\}, k \in \{0, \ldots, 2n\}}$, and let $\bar{\alpha}_i := h_{\bar{z}_i}(\mathbf{C})$ for $i \in \{1, \ldots, n\}$. If some $\bar{\alpha}_i$ is 0 then the simulator changes it to 1.

Verification phase. For any $v \in \{a, b, c\}$, $k \in \{0, \ldots, 2n\}$ and $i \in \mathsf{H}$ the simulator sets $\mathsf{flag}_i^{v,k} := 0$. The simulator sends $\{\mathsf{flag}_i^{v,k}\}_{i \in \mathsf{H}}$ to $\mathcal{A}$, as the leakage from $\mathcal{F}_{\mathsf{vss}}^{v,k}$.

For each $i \in \{1, \ldots, n\}$ and $v \in \{a, b, c\}$ the simulator sets $\bar{G}^{v,i}(x, y) := \sum_{k=0}^{2n} \alpha_i^k \cdot \bar{F}^{v,k}(x, y)$. For $i \in \{1, \ldots, n\}$, $j \in \mathsf{H}$ and $v \in \{a, b, c\}$, the leakage of $\mathcal{F}_{\mathsf{glinear}}^{i,j,v}$ is simulated in the following way. The adversary receives (1) the j-th rows $\bar{\mathbf{C}}_j^{v,0}, \ldots, \bar{\mathbf{C}}_j^{v,2n}$, (2) the indicator vectors $\{\delta^\ell\}_{\ell \in \mathsf{H}}$ such that $\delta^\ell = (1, \ldots, 1)$ for each $\ell \in \mathsf{H}$, (3) the coefficient vector $\mathbf{a} := (\bar{\alpha}_i^0, \ldots, \bar{\alpha}_i^{2n})$, (4) the openings

$\bar{o}^{v,k}_{\ell j}$ and values $\mathsf{open}_{\mathsf{crs}}(\bar{C}^{v,k}_{j\ell}, \bar{o}^{v,k}_{\ell j})$ for every $k \in \{0, \ldots, 2n\}$ and $\ell \in \mathsf{C}$, (5) the indicator vector $\delta^G := (1, \ldots, 1)$, (6) an empty set $L_\ell = \varnothing$ for every $\ell \in \mathsf{H}$, and (7) the values $\{\bar{G}^{v,i}(\ell, j)\}_{\ell\in\{1,\ldots,n\}}$ as the partial-sums. The simulator also gives $\mathcal{A}$ the vector $(\mathbf{a}, \delta^G, \bar{G}^{v,i}(0, j))$ as the output of $\mathcal{F}^{i,j,v}_{\mathsf{glinear}}$.

For $i \in \{1, \ldots, n\}$, $j \in \mathsf{C}$ and $v \in \{a, b, c\}$ the simulator simulates the leakage of $\mathcal{F}^{i,j,v}_{\mathsf{glinear}}$ by giving the adversary the corresponding inputs of the honest parties, i.e., (1) the j-th rows $\bar{\mathbf{C}}^{v,0}_j, \ldots, \bar{\mathbf{C}}^{v,2n}_j$, (2) the indicator vectors $\{\delta^\ell\}_{\ell\in\mathsf{H}}$ such that $\delta_\ell = (1, \ldots, 1)$ for each $\ell \in \mathsf{H}$, and (3) the vector $(\bar{f}^{v,0}_j(\ell), \ldots, \bar{f}^{v,2n}_j(\ell))$ for every $\ell \in \mathsf{H}$. This completes the communication from the honest parties to the corrupt parties.

Later, the simulator receives the corrupt parties' inputs to the functionalities $\mathcal{F}^{i,j,v}_{\mathsf{glinear}}$ for every $i \in \{1, \ldots, n\}$, $j \in \mathsf{C}$, and $v \in \{a, b, c\}$, as well as the inputs to the verification phase of $\mathcal{F}^{v,k}_{\mathsf{vss}}$. Upon receiving the inputs of the corrupt guide to $\mathcal{F}^{i,j,v}_{\mathsf{glinear}}$, the simulator holds all inputs to the functionality, and can compute the output of the functionality and give it to $\mathcal{A}$. Upon receiving all inputs to the verification phase of $\mathcal{F}^{v,k}_{\mathsf{vss}}$, which are $\{\mathsf{flag}^{v,k}_i\}_{i\in\mathsf{C}}$, the simulator sets $\mathsf{W}^{v,k}$ to be the set of all corrupt P_i with $\mathsf{flag}^{v,k}_i = 1$, and returns $(\mathsf{W}^{v,k}, \{\bar{o}^{v,k}_{ij}\}_{i\in\mathsf{W},j\in\{0,\ldots,n\}})$ to $\mathcal{A}$.

At the end of the simulation, the simulator sets $\mathsf{W} = \bigcap_{v\in\{a,b,c\},k\in\{0,\ldots,2n\}} \mathsf{W}^{v,k}$. For every $j \in \mathsf{W}$ the simulator sets $\mathsf{flag}_j := 1$, and otherwise, $\mathsf{flag}_j := 0$. The simulator inputs $\{\mathsf{flag}_j\}_{j\in\mathsf{C}}$ to $\mathcal{F}_{\mathsf{tss}}$.

Fix any polynomial-time environment $\mathcal{Z}$ and input ζ to the environment. We show that the view of $\mathcal{Z}$ in the real world is close to the view of $\mathcal{Z}$ in the ideal world.

$\mathcal{Z}$'s view. The adversary's view consists of (1) the crs string, (2) the inputs to the honest dealer, (3) the outputs $\{(\mathbf{C}^{v,k}, \mathbf{O}^{v,k}_i)\}_{v\in\{a,b,c\},k\in\{0,\ldots,2n\},i\in\mathsf{C}}$ from the sharing phase of $\mathcal{F}_{\mathsf{vss}}$, (4) the randomness $\{z_i\}_{i\in\mathsf{H}}$ and $\{z_i\}_{i\in\mathsf{C}}$, (5) the inputs to the honest parties in the verification phase, (6) the leakage and output of the verification phase of the $\mathcal{F}_{\mathsf{vss}}$ calls, and (7) the leakage and output of the $\mathcal{F}_{\mathsf{glinear}}$ calls. In order to prove that the real-world view is close to the ideal-world view, we define the following hybrid-worlds, where we assume that the honest parties know the set H.

- In Hybrid 1, the honest parties act like in the real-world, except that (1) at the end of the first round, every honest P_i does not check the validity of the rows received from D, but P_i is always happy with D, (2) in the first round, the dealer also sends $F^{v,k}(x, i)$ to an honest P_i, for every $v \in \{a, b, c\}$ and $k \in \{0, \ldots, 2n\}$, and (3) for $i \in \{1, \ldots, n\}$, $j \in \mathsf{H}$ and $v \in \{a, b, c\}$, an honest P_j inputs to $\mathcal{F}^{i,j,v}_{\mathsf{glinear}}$ the vector $\mathbf{a} = (\alpha^0_i, \ldots, \alpha^{2n}_i)$, the values $\{F^{v,k}(r, j)\}_{k\in\{0,\ldots,2n\},r\in\{0,\ldots,n\}}$ where each $F^{v,k}(x, j)$ was received from D as described in (2), the openings $\{o^{v,k}_{r,j}\}_{k\in\{0,\ldots,2n\},r\in\{0,\ldots,n\}}$ and the indicator vector $\delta^G = (1, \ldots, 1)$; In addition, for $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, n\}$ and $v \in \{a, b, c\}$, an honest P_r inputs $(F^{v,0}(r, j), \ldots, F^{v,2n}(r, j))$ and $\delta^r = (1, \ldots, 1)$ to $\mathcal{F}^{i,j,v}_{\mathsf{glinear}}$.
- In Hybrid 2 the honest parties act like in the real-world, except for the following changes.
  - At the end of the first round, every honest P_i does not check the validity of the rows received from D, but P_i is always happy with D.
  - In the first round, instead of sharing the coefficients of random polynomials $A(x)$, $B(x)$ and $C(x)$, the dealer does as follows. For every $k \in \{1, \ldots, n\}$ (resp., $k \in \{1, \ldots, 2n\}$) the dealer sets $A^k = B^k = 0$ (resp., $C^k = 0$), and shares A^k and B^k (resp., C^k) via $\mathcal{F}_{\mathsf{vss}}$.

The dealer also shares its inputs $(\mathbf{C}^{a,0}, \mathbf{O}^{a,0})$, $(\mathbf{C}^{b,0}, \mathbf{O}^{b,0})$ and $(\mathbf{C}^{c,0}, \mathbf{O}^{c,0})$, as A^0, B^0 and C^0, respectively.

For $v \in \{a, b, c\}$ and $k \in \{0, \ldots, 2n\}$ denote the corresponding sharing polynomial by $F^{v,k}(x, y)$. The dealer then samples three random polynomials $\bar{A}(x), \bar{B}(x)$ and $\bar{C}(x)$ such that $\bar{A}(x) \cdot \bar{B}(x) = \bar{C}(x)$, and $\bar{A}(0) = a$, $\bar{B}(0) = b$ and $\bar{C}(0) = c$.

For each $k \in \{0, \ldots, 2n\}$, the simulator picks random symmetric bivariate polynomials $\bar{F}^{a,k}(x, y)$, $\bar{F}^{b,k}(x, y)$ and $\bar{F}^{c,k}(x, y)$ of degree at most t in each variable, such that (a) for $k = 0$ we set $\bar{F}^{v,k}(x, y) = F^{v,k}(x, y)$ for $v \in \{a, b, c\}$, (b) for $k \in \{1, \ldots, n\}$ we condition on (i) $\bar{F}^{v,k}(x, i) = F^{v,k}(x, i)$ for every $v \in \{a, b, c\}$ and $i \in \mathsf{C}$, and (ii) $\bar{F}^{a,k}(0, 0) = \bar{A}^k$, $\bar{F}^{b,k}(0, 0) = \bar{B}^k$, and $\bar{F}^{c,k}(0, 0) = \bar{C}^k$, and (c) for $k \in \{n + 1, \ldots, 2n\}$ we condition on (i) $\bar{F}^{a,k}(x, y) = 0$, (ii) $\bar{F}^{b,k}(x, y) = 0$, and (iii) $\bar{F}^{c,k}(x, i) = F^{c,k}(x, i)$ for every $i \in \mathsf{C}$, and $\bar{F}^{c,k}(0, 0) = \bar{C}^k$.

For every $v \in \{a, b, c\}$ and $k \in \{0, \ldots, 2n\}$ the dealer sends $\bar{F}^{v,k}(x, i)$ to P_i.

– For $i \in \{1, \ldots, n\}$, $j \in \mathsf{H}$ and $v \in \{a, b, c\}$, in the second round an honest P_j inputs to $\mathcal{F}^{i,j,v}_{\mathsf{glinear}}$ the vector $\mathbf{a} = (\alpha_i^0, \ldots, \alpha_i^{2n})$, the values $\{\bar{F}^{v,k}(r, j)\}_{k\in\{0,\ldots,2n\}, r\in\{0,\ldots,n\}}$, the openings $\{o^{v,k}_{r,j}\}_{k\in\{0,\ldots,2n\}, r\in\{0,\ldots,n\}}$ and the indicator vector $\delta^G = (1, \ldots, 1)$; In addition, for $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, n\}$ and $v \in \{a, b, c\}$, an honest P_r inputs to $\mathcal{F}^{i,j,v}_{\mathsf{glinear}}$ the values $(\bar{F}^{v,0}(r, j), \ldots, \bar{F}^{v,2n}(r, j))$ and $\delta^r = (1, \ldots, 1)$.
– Note that for an honest P_j the commitments and openings $\{(C^{v,k}_{r,j}, o^{v,k}_{r,j})\}_{k\in\{0,\ldots,2n\}, r\in\{0,\ldots,n\}}$ are not necessarily consistent with the values $\{\bar{F}^{v,k}(r, j)\}_{k\in\{0,\ldots,2n\}, r\in\{0,\ldots,n\}}$. Therefore, we change the functionality $\mathcal{F}_{\mathsf{glinear}}$ with an honest guide, so that it would have the exact same leakage and output as the original functionality, even if some of the guide's values $(b^G_{i,0}, \ldots, b^G_{i,n})$ are not consistent with the guide's openings $(o^G_{i,0}, \ldots, o^G_{i,n})$ (but they are consistent with some degree-t polynomial).

Real-world vs. Hybrid 1. We claim that the real-world view is distributed exactly like the ideal-world view. This follows immediately by noting that (1) in the real-world an honest P_i is always happy with an honest D, and (2) in Hybrid 1, an honest P_i inputs to $\mathcal{F}_{\mathsf{glinear}}$ the same values she would input to $\mathcal{F}_{\mathsf{glinear}}$ in the real-world.

Hybrid 1 vs. Hybrid 2. We claim that the view in Hybrid 1 is $O(n^3\epsilon)$-close to the view in Hybrid 2. Note that the Hybrid 1 random variables

$$(\mathsf{crs}, \{\mathbf{C}^{v,k}, \mathbf{O}^{v,k}_i\}_{v\in\{a,b,c\}, k\in\{1,\ldots,2n\}, i\in\mathsf{C}}, \{F^{v,k}(x, y)\}_{v\in\{a,b,c\}, k\in\{1,\ldots,2n\}})$$

are $O(n^3\epsilon)$-close to the Hybrid 2 random variables

$$(\mathsf{crs}, \{\mathbf{C}^{v,k}, \mathbf{O}^{v,k}_i\}_{v\in\{a,b,c\}, k\in\{1,\ldots,2n\}, i\in\mathsf{C}}, \{\bar{F}^{v,k}(x, y)\}_{v\in\{a,b,c\}, k\in\{1,\ldots,2n\}}).$$

Finally, in both hybrids the rest of the view can be obtained from the above random variables by the same efficient process, that executes the protocol with $\mathcal{Z}$ and $\mathcal{A}$ in the following way: (1) given crs, obtain the dealer's inputs $\{(\mathbf{C}^{v,0}, \mathbf{O}^{v,0})\}_{v\in\{a,b,c\}}$ from $\mathcal{Z}$, (2) simulate the $\mathcal{F}_{\mathsf{vss}}$ calls using the values $\{\mathbf{C}^{v,k}, \mathbf{O}^{v,k}_i\}_{v\in\{a,b,c\}, k\in\{1,\ldots,2n\}, i\in\mathsf{C}}$, (3) simulate the broadcasts $\{z_i\}_{i\in\mathsf{H}}$ by sampling each

z_i at random, (4) execute $\mathcal{Z}$ and $\mathcal{A}$ to obtain $\{z_i\}_{i\in\mathsf{C}}$, compute the corresponding α_i's like in the protocol, and obtain the honest parties' inputs to the second round, $\{\mathsf{flag}_i = 0\}_{i\in\mathsf{H}}$, from $\mathcal{Z}$ (5) the verification phase leakage of the $\mathcal{F}_{\mathsf{vss}}$ calls is always $\{\mathsf{flag}_i = 0\}_{i\in\mathsf{H}}$, (6) simulate a call to $\mathcal{F}^{i,j,v}_{\mathsf{glinear}}$ for $i \in \{1,\dots,n\}$, $j \in \mathsf{H}$ and $v \in \{a,b,c\}$ by leaking (a) the j-th rows $\mathbf{C}^{v,0}_j,\dots,\mathbf{C}^{v,2n}_j$, (b) the indicator vectors $\{\delta^\ell\}_{\ell\in\mathsf{H}}$ such that, $\delta^\ell = (1,\dots,1)$ for each $\ell \in \mathsf{H}$, (c) the coefficient vector $\mathbf{a} = (\alpha_i^0,\dots,\alpha_i^{2n})$, (d) the openings $o^{v,k}_{j\ell}$ and values $\mathsf{open}_{\mathsf{crs}}(C^{v,k}_{j\ell}, o^{v,k}_{j\ell})$ for every $k \in \{0,\dots,2n\}$ and $\ell \in \mathsf{C}$, (e) the indicator vector $\delta^G := (1,\dots,1)$, (f) a set $L_\ell = \varnothing$, for every $\ell \in \mathsf{H}$, (g) partial sums $\sum_{k=0}^{2n} \alpha_i^k \cdot F^{v,k}(\ell,j)$ for all $\ell \in \{1,\dots,n\}$ and (h) the output $(\mathbf{a}, \delta^G, \sum_{k=0}^{2n} \alpha_i^k \cdot F^{v,k}(0,j))$, (7) simulate $\mathcal{F}^{i,j,v}_{\mathsf{glinear}}$ for $i \in \{1,\dots,n\}$, $v \in \{a,b,c\}$ and $j \in \mathsf{C}$ by leaking (a) the j-th rows $\mathbf{C}^{v,0}_j,\dots,\mathbf{C}^{v,2n}_j$, (b) the indicator vectors $\{\delta^\ell\}_{\ell\in\mathsf{H}}$ such that $\delta^\ell = (1,\dots,1)$ for each $\ell \in \mathsf{H}$, and (c) the vector $(F^{v,0}(\ell,j),\dots,F^{v,2n}(\ell,j))$ for every $\ell \in \mathsf{H}$, (7) at this stage, obtain from $\mathcal{Z}$ and $\mathcal{A}$ the inputs of the corrupt parties to the $\mathcal{F}_{\mathsf{vss}}$ calls and remaining $\mathcal{F}_{\mathsf{glinear}}$ calls (where the guide is corrupt), and compute the outputs according to the functionality (note that in each $\mathcal{F}_{\mathsf{vss}}$ call only corrupt parties are in W, and that for $\mathcal{F}_{\mathsf{glinear}}$ with a corrupt guide, the leakage together with the corrupt guide's inputs determine the output of the functionality). This completes the case of Hybrid 1 vs. Hybrid 2.

Hybrid 2 vs. ideal-world. We claim that the view in Hybrid 2 has the same distribution as the view in the ideal-world. Note that the random variables

$$(\mathsf{crs}, \{\mathbf{C}^{v,k}, \mathbf{O}^{v,k}_i\}_{v\in\{a,b,c\},k\in\{1,\dots,2n\},i\in\mathsf{C}}, \{\bar{f}^{v,k}_i(x)\}_{v\in\{a,b,c\},k\in\{1,\dots,2n\},i\in\mathsf{C}}), \tag{6}$$

have the same distribution in both worlds, where $\bar{f}^{v,k}_i(i)$ is the polynomial corresponding to $(\mathbf{C}^{v,k}, \mathbf{O}^{v,k}_i)$. We claim that in both worlds, the rest of the view can be obtained from the above random variables by the following efficient process: (1) given crs, obtain the dealer's inputs $\{(\mathbf{C}^{v,0}, \mathbf{O}^{v,0})\}_{v\in\{a,b,c\}}$ from $\mathcal{Z}$, (2) picks random polynomials $\bar{A}(x), \bar{B}(x)$ and $\bar{C}(x)$ of degree n, n and $2n$, respectively, such that $\bar{A}(x) \cdot \bar{B}(x) = \bar{C}(x)$, (3) sample polynomials $\{\bar{F}^{v,k}(x,y)\}_{v\in\{a,b,c\},k\in\{0,\dots,2n\}}$ just like the sharing phase of the simulator, using the fixed values $\{\bar{f}^{v,k}_i(x)\}_{v\in\{a,b,c\},k\in\{1,\dots,2n\},i\in\mathsf{C}}$ and $\bar{A}(x), \bar{B}(x)$ and $\bar{C}(x)$, (4) simulate the sharing phase and the verification phase just like the simulator, using the polynomials $\{\bar{F}^{v,k}(x,y)\}_{v\in\{a,b,c\},k\in\{0,\dots,2n\}}$.

Clearly, when the random variables in Equation 6 are taken from the ideal-world, the output of the above process is distributed exactly like the view of $\mathcal{Z}$ in the ideal-world. To see that this is also true for Hybrid 2, note that by Fact A.3 the random variables

$$\{\bar{A}(\alpha_i), \bar{B}(\alpha_i), \bar{C}(\alpha_i)\}_{i\in\{1,\dots,n\}}$$

generated by the process have the same distribution as the corresponding random variables generated in Hybrid 2. Conditioned on those values, and by Fact A.7, the random variables

$$\{\bar{G}^{v,i}(x,y) := \sum_{k=0}^{2n} \alpha_j^k \cdot \bar{F}^{v,k}(x,y)\}_{v\in\{a,b,c\},i\in\{1,\dots,n\}}$$

generated by the process have the same distribution as the corresponding random variables in Hybrid 2. Conditioned on those values it is not hard to verify that the view generated by the process has the same distribution as the view in Hybrid 2. This completes the analysis of $\mathcal{Z}$'s view.

Honest parties' output. We say that a view View is "good" if for any commitment C from the view, and for any two openings o, and o' that appear in the view, it holds that either $\mathsf{open}_{\mathsf{crs}}(C, o) = \perp$ or $\mathsf{open}_{\mathsf{crs}}(C, o') = \perp$ or $\mathsf{open}_{\mathsf{crs}}(C, o) = \mathsf{open}_{\mathsf{crs}}(C, o)$. By the binding property, a view View is good with probability at least $1 - \epsilon$.

We claim that both in the real-world and the ideal-world, the outputs of the honest parties can be extracted from a good View by the following efficient deterministic process: (1) the output of an honest P_i in the sharing phase is $(\mathbf{C}^a, \mathbf{O}^a_i)$, $(\mathbf{C}^b, \mathbf{O}^b_i)$, $(\mathbf{C}^c, \mathbf{O}^c_i)$ and $\mathsf{happy}_i = 1$, where $(\mathbf{C}^a, \mathbf{O}^a)$, $(\mathbf{C}^b, \mathbf{O}^b)$, $(\mathbf{C}^c, \mathbf{O}^c)$ are the inputs of the dealer according to View, (2) in the verification phase, compute the set W according to View and let each honest party output $(\mathsf{W}, \{\mathbf{O}^a_i\}_{i\in\mathsf{W}}, \{\mathbf{O}^b_i\}_{i\in\mathsf{W}}, \{\mathbf{O}^c_i\}_{i\in\mathsf{W}})$.

It is not hard to see that when View is taken from the ideal-world then those are indeed the honest parties' inputs. In addition, when View is taken from the real-world, the sharing-phase outputs are the same as in the process. Therefore, it remains to analyse the verification-phase of the real-world.

First observe that in the real-world the honest parties never output "D is corrupt" according to a good View. This follows since (1) no call to $\mathcal{F}_{\mathsf{vss}}$ ends with "D is corrupt" (2) all honest parties are happy, and $\mathsf{flag}_i = 0$ for any honest P_i, (3) all honest parties are in V, so $|\mathsf{V} \cup \mathsf{W}| \geq n - t$, and (4) since View is good, for each $i \in \{1, \ldots, n\}$ the i-th challenge succeeds with probability 1. Indeed, fix $i \in \{1, \ldots, n\}$ and $v \in \{a, b, c\}$, and let $G^{v,i}(x, y) := \sum_{k=0}^{2n} \alpha^k_j \cdot F^{v,k}(x, y)$.

- Since the dealer is honest then clearly for $j \in \mathsf{W}$ it holds that $t_{ijv} = G^{v,i}(0, j)$.
- For every $j \in \mathsf{H}$ the output of $\mathcal{F}^{i,j,v}_{\mathsf{glinear}}$ is $((\alpha^0_i, \ldots, \alpha^{2n}_i), (1, \ldots, 1), t_{ijv})$, where $t_{ijv} = G^{v,i}(0, j)$.
- For $j \in \mathsf{V} \setminus \mathsf{H}$, the corrupt guide P_j must input $\delta^G = (1, \ldots, 1)$ (or otherwise $j \notin \mathsf{V}$), and all honest parties P_ℓ input $\delta^\ell = (1, \ldots, 1)$ and the values $(F^{v,0}(j, \ell), \ldots, F^{v,2n}(j, \ell))$.

  Consider $\ell \in \mathsf{H}$, and denote the corresponding input values of the corrupt guide by $(b^G_{0\ell}, \ldots, b^G_{(2n)\ell})$. We claim that $b^G_{k\ell} = F^{v,k}(j, \ell)$ for all $k \in \{0, \ldots, 2n\}$. Indeed, if this is not the case then I_ℓ is not empty, and let k be an element of I_ℓ. Since $j \in \mathsf{V}$, the output of $\mathcal{F}^{i,j,v}_{\mathsf{glinear}}$ is not "G is corrupt", so the corrupt guide sent to the functionality an opening o such that $\mathsf{open}_{\mathsf{crs}}(C^{v,k}_{j\ell}, o) = b^G_{k\ell}$. But since View is good we must have $b^G_{k,\ell} = F^{v,k}(\ell, j) = F^{v,k}(j, \ell)$, in contradiction. We conclude that $b^G_{k\ell} = F^{v,k}(j, \ell)$ for all $k \in \{0, \ldots, 2n\}$, so the output of $\mathcal{F}^{i,j,v}_{\mathsf{glinear}}$ is $((\alpha^0_i, \ldots, \alpha^{2n}_i), (1, \ldots, 1), t_{ijv})$, where $t_{ijv} = \sum_{k=0}^{2n} \alpha^k_i F^{v,k}(0, j) = G^{v,i}(0, j)$.

It follows that, $t_{ia} = G^{a,i}(0, 0) = A(\alpha_i)$, $t_{ib} = G^{b,i}(0, 0) = B(\alpha_i)$ and $t_{ic} = G^{c,i}(0, 0) = C(\alpha_i)$, and so, since D is honest and $A(x)B(x) = C(x)$, it follows that $t_{ia} \cdot t_{ib} = t_{ic}$. Therefore the dealer is not disqualified, and the output of the honest parties is $(\mathsf{W}, \{\mathbf{O}^a_i\}_{i\in\mathsf{W}}, \{\mathbf{O}^b_i\}_{i\in\mathsf{W}}, \{\mathbf{O}^c_i\}_{i\in\mathsf{W}})$, as required. This concludes the analysis of the honest parties' output.

B.3.2 Corrupt Dealer

Sharing phase. The simulator takes the role of the honest parties, that have no inputs, in order to simulate an execution of the protocol. In the sharing phase, this includes only broadcasting z_i on behalf of every honest P_i. Then, the simulator receives from $\mathcal{A}$ the inputs to the various $\mathcal{F}_{\mathsf{vss}}$ calls. Denote those inputs by $\{(\mathbf{C}^{v,k}, \mathbf{O}^{v,k})\}_{v\in\{a,b,c\},k\in\{0,\ldots,2n\}}$. The simulator gives the simulated honest party P_i the shares $\{(\mathbf{C}^{v,k}, \mathbf{O}_i^{v,k})\}_{v\in\{a,b,c\},k\in\{0,\ldots,2n\}}$, and computes the output bit happy_i of the sharing phase. The simulator inputs $(\mathbf{C}^a, \mathbf{O}^a)$,$(\mathbf{C}^b, \mathbf{O}^b)$, $(\mathbf{C}^c, \mathbf{O}^c)$ and $\{\mathsf{happy}_i\}_{i\in\mathsf{H}}$ to $\mathcal{F}_{\mathsf{tss}}$.

Verification phase. The simulator receives the inputs of the honest parties $\{\mathsf{flag}_i\}_{i\in\mathsf{H}}$ as leakage from $\mathcal{F}_{\mathsf{tss}}$. The simulator now holds the honest parties inputs, and continues to simulate the honest parties, by following the protocol in the verification phase in $\mathcal{F}_{\mathsf{vss}}$ and in the calls to $\mathcal{F}_{\mathsf{glinear}}$. Observe that since the dealer holds all inputs, the dealer can compute the leakage of $\mathcal{F}_{\mathsf{glinear}}$ as well. At the end of the simulation the simulator computes the output of the simulated honest parties. If the output is "D is corrupt" then the simulator inputs $\mathsf{flag}_D = 1$ to $\mathcal{F}_{\mathsf{tss}}$ (the rest of the inputs do not matter). Otherwise, the simulator computes the set W and the openings $\{\bar{o}_{ij}^a, \bar{o}_{ij}^b, \bar{o}_{ij}^c\}_{i\in\mathsf{W},j\in\{0,\ldots,n\}}$ according to the protocol, and sets $\bar{o}_{ij}^v := 0$ for any $v \in \{a,b,c\}$, $i \notin \mathsf{W}$ and $j \in \{0,\ldots,n\}$. The simulator sets $\mathsf{flag}_i = 1$ for any corrupt P_i in W, and $\mathsf{flag}_i = 0$ for any corrupt P_i not in W. The simulator inputs $(\bar{\mathbf{O}}^a, \bar{\mathbf{O}}^b, \bar{\mathbf{O}}^c)$. $\mathsf{flag}_D = 0$, and $\{\mathsf{flag}_i\}_{i\in\mathsf{C}}$ to $\mathcal{F}_{\mathsf{tss}}$.

Fix any polynomial time environment $\mathcal{Z}$ and input ζ to the environment. We show that the view of $\mathcal{Z}$ in the real world is close to the view of $\mathcal{Z}$ in the ideal world.

$\mathcal{Z}$'s view. At the sharing phase the honest parties hold no inputs, and so they are perfectly simulated by the simulator. Therefore the view of the adversary in the sharing phase is the same in both worlds. Fix any such view, and note that this implies that the honest parties' inputs in the verification phase have the same distribution in both worlds. Fix those inputs as well, and note that the simulator holds all the honest parties' inputs, and so they are perfectly simulated in the verification phase as well. We conclude that the $\mathcal{Z}$'s view has the same distribution in both worlds.

Honest parties' inputs. We say that a view View is "good" if either (1) the output of the honest parties is "D is corrupt", or (2) the output is not "D is corrupt", and it holds that $F^{a,0}(0,0) \cdot F^{b,0}(0,0) = F^{c,0}(0,0)$, where $F^{v,0}$ is the sharing polynomial of the weak double t-sharing produced by $\mathcal{F}_{\mathsf{vss}}^{v,0}$, for $v \in \{a,b,c\}$ (observe that whenever the output is not "D is corrupt" then all $\mathcal{F}_{\mathsf{vss}}$ calls define weak double t-sharing). We claim that a view View is good with probability at least $1-\epsilon$.

Indeed, a view View is not good if and only if the output of the honest parties is not "D is corrupt", so the output of each $\mathcal{F}_{\mathsf{vss}}$ call at the end of the verification phase is a weak double t-sharing, and $F^{a,0}(0,0) \cdot F^{b,0}(0,0) \neq F^{c,0}(0,0)$. Let E be the event that a view is (1) not good, and (2) for every $i \in \mathsf{H}$ either (a) $h_{z_i}(\mathbf{C}) = 0$ or, (b) for $\alpha_i := h_{z_i}(\mathbf{C})$,

$$\Big(\sum_{k=0}^{2n} \alpha_i^k F^{a,k}(0,0)\Big) \cdot \Big(\sum_{k=0}^{2n} \alpha_i^k F^{b,k}(0,0)\Big) = \Big(\sum_{k=0}^{2n} \alpha_i^k F^{c,k}(0,0)\Big),$$

where $F^{v,k}$ is the sharing polynomial of the weak double t-sharing produced by $\mathcal{F}_{\mathsf{vss}}^{v,k}$. By the properties of the hash functions, E occurs with probability at most ϵ. Therefore, with probability at least $1-\epsilon$ either (1) View is good, or (2) the output is not "D is corrupt", so the $\mathcal{F}_{\mathsf{vss}}$ calls define weak double t-sharings, and there exists $i \in \mathsf{H}$ such that $\alpha_i \neq 0$ and

$$\Big(\sum_{k=0}^{2n} \alpha_i^k F^{a,k}(0,0)\Big) \cdot \Big(\sum_{k=0}^{2n} \alpha_i^k F^{b,k}(0,0)\Big) \neq \Big(\sum_{k=0}^{2n} \alpha_i^k F^{c,k}(0,0)\Big).$$

We continue by showing that (2) cannot occur, which means that the probability that View is good is at least $1-\epsilon$. Indeed, assume that (2) occurs, and observe that there are $n-t \geq t+1$ honest parties, each of them is either in V or in W, and that for any honest P_j the parties obtain the shares $\{t_{i,j,v} = \sum_{k=0}^{2n} \alpha_i^k F^{v,k}(0,j)\}_{v\in\{a,b,c\}}$, and so the shares $\{t_{i,j,a}\}_{j\in \mathsf{V}\cup\mathsf{W}}$, $\{t_{i,j,b}\}_{j\in \mathsf{V}\cup\mathsf{W}}$ and $\{t_{i,j,c}\}_{j\in \mathsf{V}\cup\mathsf{W}}$ either define polynomials of degree larger than t, in which case D is disqualified (in contradiction to (2)), or they define the degree-t polynomials $\sum_{k=0}^{2n} \alpha_i^k F^{a,k}(x,0)$, $\sum_{k=0}^{2n} \alpha_i^k F^{b,k}(x,0)$ and $\sum_{k=0}^{2n} \alpha_i^k F^{c,k}(x,0)$, whose free-coefficients are $t_{i,a} := \sum_{k=0}^{2n} \alpha_i^k F^{a,k}(0,0)$, $t_{i,b} := \sum_{k=0}^{2n} \alpha_i^k F^{b,k}(0,0)$ and $t_{i,c} := \sum_{k=0}^{2n} \alpha_i^k F^{c,k}(0,0)$, and $t_{i,a} \cdot t_{i,b} \neq t_{i,c}$, so D is disqualified, again in contradiction to (2). We conclude that a view is good with probability at least $1-\epsilon$.

Conditioned on a good view View it is not hard to see that the output of the honest parties in the real world is fixed and equal to the output of the honest parties in the ideal world. This concludes the proof. □

B.4 Single-Input Functionality

Proof of Lemma 3.14. In this section we prove that protocol sif UC-emulates $\mathcal{F}_{\mathsf{sif}}$ (with everlasting security if the underlying commitment-scheme is statistically-hiding). From the composition properties of UC-security, it is enough to prove security in the $(\mathcal{F}_{\mathsf{vss}}, \mathcal{F}_{\mathsf{glinear}}, \mathcal{F}_{\mathsf{tss}})$-hybrid model. Let $\mathcal{A}$ be an efficient adversary against sif. We define the simulator $\mathcal{S}$ as follows. The simulator $\mathcal{S}$ uses $\mathcal{A}$ in a black-box manner, and forwards all messages between $\mathcal{Z}$ and $\mathcal{A}$. The simulator first receives the set of corrupt C parties from $\mathcal{Z}$. We split into cases.

B.4.1 Honest Dealer

Input phase. For every $i \in \{1,\dots,\ell\}$ the simulator samples a strong double t-sharing $\langle\langle 0\rangle\rangle$, denoted $(\mathbf{C}^i, \mathbf{O}^i)$, and simulates the sharing phase of $\mathcal{F}^i_{\mathsf{vss}}$ by giving $(\mathbf{C}^i, \mathbf{O}^i_j)$ to $\mathcal{A}$ as the output of a corrupt P_j. We denote the corresponding sharing polynomials by $\bar{F}^i(x,y)$. Similarly, for each $i,j \in \{1,\dots,\ell\}$, the simulator samples a strong double t-sharing $\langle\langle 0\rangle\rangle$, denoted $(\mathbf{C}^{ij}, \mathbf{O}^{ij})$, and simulates the sharing phase of $\mathcal{F}^{ij}_{\mathsf{vss}}$ by giving $(\mathbf{C}^{ij}, \mathbf{O}^{ij}_k)$ to $\mathcal{A}$ as the output of a corrupt P_k. We denote the corresponding sharing polynomials by $\bar{F}^{ij}(x,y)$.

In addition, for each $i \in \{1,\dots,m\}$ the simulator samples three strong double t-sharings of zero, denoted $(\mathbf{C}^{\gamma_i}, \mathbf{O}^{\gamma_i})$,$(\mathbf{C}^{\rho_i}, \mathbf{O}^{\rho_i})$ and $(\mathbf{C}^{\eta_i}, \mathbf{O}^{\eta_i})$, and we denote the corresponding sharing polynomials by $\bar{F}^{\gamma_i}(x,y)$, $\bar{F}^{\rho_i}(x,y)$ and $\bar{F}^{\eta_i}(x,y)$. The simulator simulates the sharing phase of $\mathcal{F}^i_{\mathsf{tss}}$ by giving $(\mathbf{C}^{\gamma_i}, \mathbf{O}^{\gamma_i}_k)$,$(\mathbf{C}^{\rho_i}, \mathbf{O}^{\rho_i}_k)$ and $(\mathbf{C}^{\eta_i}, \mathbf{O}^{\eta_i}_k)$ and $\mathsf{happy}_k = 1$ to $\mathcal{A}$ as the output of a corrupt P_k. The simulator also gives $(\mathbf{C}^{\gamma_i}, \mathbf{O}^{\gamma_i})$, and $(\mathbf{C}^{\rho_i}, \mathbf{O}^{\rho_i})$ to $\mathcal{A}$ as the dealer's broadcast.

Finally, for each $i,j \in \{1,\dots,\ell\}$, the simulator simulates the sharing phase of $\mathcal{F}^{ij}_{\mathsf{tss}}$ by giving $(\mathbf{C}^i, \mathbf{O}^i_k)$, $(\mathbf{C}^j, \mathbf{O}^j_k)$, $(\mathbf{C}^{ij}, \mathbf{O}^{ij}_k)$ and $\mathsf{happy}_k = 1$ to $\mathcal{A}$ as the output of a corrupt P_k.

Output phase. The simulator receives the output $(y^1,\dots,y^m)$ from $\mathcal{F}_{\mathsf{sif}}$. For each $i \in \{1,\dots,m\}$, the simulator picks a random symmetric bivariate polynomial $\bar{F}^{y^i}(x,y)$, of degree at most t in each variable, conditioned on $\bar{F}^{y^i}(0,0) = y^i$, and

$$\bar{F}^{y^i}(x,j) = \sum_{p\in\{1,\dots,\ell\}} \alpha^i_p \bar{F}^p(x,j) + \sum_{p,q\in\{1,\dots,\ell\}} \alpha^i_{pq} \bar{F}^{pq}(x,j) + \bar{F}^{\eta_i}(x,j),$$

for all $j \in \mathsf{C}$.

The leakage of the verification phases of each $\mathcal{F}_{\mathsf{vss}}$ and $\mathcal{F}_{\mathsf{tss}}$ call is simulated by giving $\mathcal{A}$ the flags of the honest parties, all of them set to be 0.

For every $i \in \{1,\dots,m\}$ and $j \in \mathsf{H}$ the leakage of $\mathcal{F}^{i,j}_{\mathsf{glinear}}$ is simulated as follows. The simulator sends $\mathcal{A}$, (1) the commitments $(\mathbf{C}^1_j,\dots,\mathbf{C}^\ell_j,\mathbf{C}^{1,1}_j,\dots,\mathbf{C}^{\ell,\ell}_j,\mathbf{C}^{\eta_i}_j)$, (2) the indicator vectors $\{\delta^i := (1,\dots,1)\}_{k\in\mathsf{H}}$ and $\delta^G := (1,\dots,1)$ of the honest parties, (3) the vector of coefficients $\mathbf{a} := (\alpha^i_1,\dots,\alpha^i_\ell,\alpha^i_{11},\dots,\alpha^i_{\ell\ell},1)$, (4) the values and openings $\{\bar{F}^p(k,j),o^p_{kj}\}_{p\in\{1,\dots,\ell\},k\in\mathsf{C}}$, $\{\bar{F}^{pq}(k,j),o^{pq}_{kj}\}_{p,q\in\{1,\dots,\ell\},k\in\mathsf{C}}$, and $\{\bar{F}^{\eta_i}(k,j),o^{\eta_i}_{kj}\}_{k\in\mathsf{C}}$, (5) an empty set $L_k = \varnothing$ for every $k \in \mathsf{H}$, and (6) the values $\bar{F}^{y^i}(k,j)$ for $k \in \{1,\dots,n\}$. The simulator also sends $\bar{F}^{y^i}(0,j)$ as the output of the functionality.

For every $i \in \{1,\dots,\ell\}$ and $j \in \mathsf{C}$ the leakage of $\mathcal{F}^{i,j}_{\mathsf{glinear}}$ is simulated as follows. The simulator sends to $\mathcal{A}$ the corresponding input of the honest parties, i.e., (1) the commitments $(\mathbf{C}^1_j,\dots,\mathbf{C}^\ell_j,\mathbf{C}^{1,1}_j,\dots,\mathbf{C}^{\ell,\ell}_j,\mathbf{C}^{\eta_i}_j)$, (2) the indicator vectors $\{\delta^i := (1,\dots,1)\}_{k\in\mathsf{H}}$ of the honest parties, and (3) the values $(\bar{F}^1(k,j),\dots,\bar{F}^\ell(k,j),\bar{F}^{11}(k,j),\dots,\bar{F}^{\ell\ell}(k,j),\bar{F}^{\eta_i}(k,j))$ for every $k \in \mathsf{H}$.

At this stage the adversary $\mathcal{A}$ sends the inputs to the ideal functionalities. The simulator, that now holds all inputs to those functionalities, computes the outputs and gives them to $\mathcal{A}$. This concludes the simulation.

We assume without loss of generality that $\mathcal{A}$ is the dummy adversary (see Section A.2), and we fix a polynomial-time environment $\mathcal{Z}$ with input ζ. We may assume without loss of generality that $\mathcal{Z}$ is deterministic. We begin by showing that the view of $\mathcal{Z}$ in the real-world is close to the view of $\mathcal{Z}$ in the ideal world.

$\mathcal{Z}$'s view. The adversary's view consists of (1) the crs, (2) the inputs to the honest dealer, (3) the output of the sharing phase of $\mathcal{F}_{\mathsf{vss}}$ and $\mathcal{F}_{\mathsf{tss}}$, and the dealer's broadcast, (4) the leakage and output of the verification phase of $\mathcal{F}_{\mathsf{vss}}$ and $\mathcal{F}_{\mathsf{tss}}$, and (5) the leakage and output of the $\mathcal{F}_{\mathsf{glinear}}$ calls. We consider the following two hybrid worlds, where we assume that the honest parties know the set H.

- In Hybrid 1, the honest parties act like in the real-world, except that (1) at the end of the first round, an honest P_i does not check the validity of the rows received from D, but is always happy with D, (2) in the first round, for every $i,j \in \{1,\dots,\ell\}$ and $k \in \mathsf{H}$, the dealer also sends $F^i(x,k)$ and $F^{ij}(x,k)$ to P_k, where $F^i(x,y)$ and $F^{ij}(x,y)$ are the sharing polynomials of z^i and z^{ij}, (3) in the first round, for every $i \in \{1,\dots,m\}$ and $k \in \mathsf{H}$, the dealer also sends $F^{\eta_i}(x,k)$ to P_k, where $F^{\eta_i}(x,y)$ is the sharing polynomial of η_i, and (4) in the second round, for every $i \in \{1,\dots,m\}$ and $j \in \mathsf{H}$, the honest guide inputs to $\mathsf{glinear}^{ij}$ the vector $\mathbf{a} = (\alpha^i_1,\dots,\alpha^i_\ell,\alpha^i_{11},\dots,\alpha^i_{\ell,\ell},1)$, the values $\{F^p(k,j),F^{pq}(k,j)\}_{p,q\in\{1,\dots,\ell\},k\in\{0,\dots,n\}}$ and $\{F^{\eta_i}(k,j)\}_{k\in\{0,\dots,n\}}$ using the polynomials received from the dealer, the openings $\{o^p_{k,j},o^{pq}_{k,j}\}_{p,q\in\{1,\dots,\ell\},k\in\{0,\dots,n\}}$ and $\{o^{\eta_i}_{k,j}\}_{k\in\{0,\dots,n\}}$ received from the $\mathcal{F}_{\mathsf{vss}}$ and $\mathcal{F}_{\mathsf{tss}}$ functionalities, and the indicator vector $\delta^G = (1,\dots,1)$; In addition, for every $i \in \{1,\dots,m\}$ and $j \in \{1,\dots,n\}$, an honest P_k inputs to $\mathcal{F}^{ij}_{\mathsf{glinear}}$ the values $\{F^p(k,j),F^{pq}(k,j)\}_{p,q\in\{1,\dots,\ell\}}$ and $F^{\eta_i}(k,j)$, and the indicator vector $\delta^k = (1,\dots,1)$.

- In Hybrid 2, honest parties act like in the real-world, except that
  - At the end of the first round, an honest P_i does not check the validity of the rows received from D, but is always happy with D.

- In the first round, for every $p, q \in \{1, \ldots, \ell\}$ the dealer shares 0 instead of z_p and z_{pq} in all $\mathcal{F}_{\mathsf{vss}}$ and $\mathcal{F}_{\mathsf{tss}}$ calls, and we denote the sharing polynomials by $F^p(x, y)$ and $F^{pq}(x, y)$.
- In the first round, for each $p, q \in \{1, \ldots, \ell\}$ the dealer picks random polynomials $\bar{F}^p(x, y)$ and $\bar{F}^{pq}(x, y)$ conditioned on (a) $\bar{F}^p(x, i) = F^p(x, i)$ for $i \in \mathsf{C}$ and $\bar{F}^p(0, 0) = z_p$, and (b) $\bar{F}^{pq}(x, i) = F^{pq}(x, i)$ for $i \in \mathsf{C}$ and $\bar{F}^{pq}(0, 0) = z_{pq}$.
- In the first round, for each $i \in \{1, \ldots, m\}$ the dealer picks a random polynomial $\bar{F}^{\eta_i}(x, y)$ conditioned on $\bar{F}^{\eta_i}(x, k) = F^{\eta_i}(x, k)$ for every $k \in \mathsf{C}$, and $F^{\eta_i}(0, 0) = 0$.
- In the first round, for any $k \in \mathsf{H}$ the dealer sends $\bar{F}^i(x, k)$, $\bar{F}^{ij}(x, k)$, and $\bar{F}^{\eta_i}(x, k)$ to P_k.
- In the second round, for every $i \in \{1, \ldots, m\}$ and $j \in \mathsf{H}$, the honest guide inputs to $\mathcal{F}^{ij}_{\mathsf{glinear}}$ the vector $\mathbf{a} = (\alpha^i_1, \ldots, \alpha^i_\ell, \alpha^i_{11}, \ldots, \alpha^i_{\ell,\ell}, 1)$, the values $\{\bar{F}^p(k, j), \bar{F}^{pq}(k, j)\}_{p,q \in \{1,\ldots,\ell\}, k \in \{0,\ldots,n\}}$ and $\{\bar{F}^{\eta_i}(k, j)\}_{k \in \{0,\ldots,n\}}$ using the polynomials received from the dealer, the openings $\{o^p_{k,j}, o^{pq}_{k,j}\}_{p,q \in \{1,\ldots,\ell\}, k \in \{0,\ldots,n\}}$ and $\{o^{\eta_i}_{k,j}\}_{k \in \{0,\ldots,n\}}$ received from the $\mathcal{F}_{\mathsf{vss}}$ and $\mathcal{F}_{\mathsf{tss}}$ functionalities, and the indicator vector $\delta^G = (1, \ldots, 1)$; In addition, for every $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, n\}$, an honest P_k inputs to $\mathcal{F}^{ij}_{\mathsf{glinear}}$ the values $\{\bar{F}^p(k, j), \bar{F}^{pq}(k, j)\}_{p,q \in \{1,\ldots,\ell\}}$ and $\bar{F}^{\eta_i}(k, j)$, and the indicator vector $\delta^k = (1, \ldots, 1)$.
- Note that for an honest guide P_j the commitments and openings that P_j inputs to $\mathcal{F}^{ij}_{\mathsf{glinear}}$ are not necessarily consistent with the values $\{\bar{F}^p(k, j), \bar{F}^{pq}(k, j)\}_{p,q \in \{1,\ldots,\ell\}, k \in \{0,\ldots,n\}}$ and $\{\bar{F}^{\eta_i}(k, j)\}_{k \in \{0,\ldots,n\}}$ received from the dealer. Therefore, we change the functionality $\mathcal{F}_{\mathsf{glinear}}$ with an honest guide, so that it would have the exact same leakage and output as the original functionality, even if some of the guide's values $(b^G_{i,0}, \ldots, b^G_{i,n})$ are not consistent with the guide's openings $(o^G_{i,0}, \ldots, o^G_{i,n})$ (but they are consistent with some degree-t polynomial).

Real-world vs. Hybrid 1. We claim that the real-world view is distributed exactly like the ideal-world view. This follows immediately by noting that (1) in the real-world an honest P_i is always happy with an honest D, and (2) in Hybrid 1, an honest P_i inputs to $\mathcal{F}_{\mathsf{glinear}}$ the same values she would input to $\mathcal{F}_{\mathsf{glinear}}$ in the real-world.

Hybrid 1 vs. Hybrid 2. We claim that Hybrid 1 is $O((m + \ell^2)n^2\epsilon)$-close to Hybrid 2. First, note that the CRS string crs and the inputs to the honest dealer $\{z^1, \ldots, z^\ell\}$ have the same distribution in both worlds, so we fix them. In addition, the random variables $\{(\mathbf{C}^{\gamma_i}, \mathbf{O}^{\gamma_i}), (\mathbf{C}^{\rho_i}, \mathbf{O}^{\rho_i})\}_{i \in \{1,\ldots,m\}}$ have the same distribution in both worlds, and we fix them as well. Consider the Hybrid 1 random variables

$$(\{(\mathbf{C}^p, \mathbf{O}^p_i), (\mathbf{C}^{pq}, \mathbf{O}^{pq}_i), F^p(x, y), F^{pq}(x, y)\}_{p,q \in \{1,\ldots,\ell\}, i \in \mathsf{C}}, \{(\mathbf{C}^{\eta_i}, \mathbf{O}^{\eta_i}_k), F^{\eta_i}(x, y)\}_{i \in \{1,\ldots,m\}, k \in \mathsf{C}})$$

and the Hybrid 2 random variables

$$(\{(\mathbf{C}^p, \mathbf{O}^p_i), (\mathbf{C}^{pq}, \mathbf{O}^{pq}_i), \bar{F}^p(x, y), \bar{F}^{pq}(x, y)\}_{p,q \in \{1,\ldots,\ell\}, i \in \mathsf{C}}, \{(\mathbf{C}^{\eta_i}, \mathbf{O}^{\eta_i}_k), \bar{F}^{\eta_i}(x, y)\}_{i \in \{1,\ldots,m\}, k \in \mathsf{C}})$$

and note that they are $O((m + \ell^2)n^2\epsilon)$-close, and that in both worlds the view can be obtained from those random variables by the same efficient process. This concludes the case of Hybrid 1 vs. Hybrid 2.

Hybrid 2 vs. Ideal-world. We claim that Hybrid 2 has the same distribution as the ideal-world. First, note that the CRS string crs and the inputs to the honest dealer $\{z^1, \dots, z^\ell\}$ have the same distribution in both worlds, so we fix them. In addition, the random variables $\{(\mathbf{C}^{\gamma_i}, \mathbf{O}^{\gamma_i}), (\mathbf{C}^{\rho_i}, \mathbf{O}^{\rho_i})\}_{i \in \{1,\dots,m\}}$ have the same distribution in both worlds, and we fix them as well. Consider the random variables

$$(\{(\mathbf{C}^p, \mathbf{O}^p_i), (\mathbf{C}^{pq}, \mathbf{O}^{pq}_i), \bar{F}^p(x,i), \bar{F}^{pq}(x,i)\}_{p,q \in \{1,\dots,\ell\}, i \in \mathsf{C}}, \{(\mathbf{C}^{\eta_i}, \mathbf{O}^{\eta_i}_k), \bar{F}^{\eta_i}(x,k)\}_{i \in \{1,\dots,m\}, k \in \mathsf{C}})$$

note that they have the same distribution, and that in both worlds the rest of the view can be obtained from them by the same efficient process. This concludes the case of Hybrid 2 vs. ideal-world.

Honest parties' outputs. We say that a view View is "good" if for any commitment C that appears in the view, and for any two openings o, and o' that appear in the view, it holds that either $\mathsf{open}_{\mathsf{crs}}(C, o) = \perp$ or $\mathsf{open}_{\mathsf{crs}}(C, o') = \perp$ or $\mathsf{open}_{\mathsf{crs}}(C, o) = \mathsf{open}_{\mathsf{crs}}(C, o)$. Observe that, by the binding property of the commitment scheme, a view View is good with probability at least $1 - \epsilon$.

We claim that both in the real-world and the ideal-world, the outputs of the honest parties in the output phase can be extracted from a good View by the following efficient deterministic process: Extract the inputs $z^1, \dots, z^\ell$ from View and output $y(\mathbf{z})$. Clearly, in the ideal-world the output of the honest parties is $y(\mathbf{z})$ with probability 1. We show that this also occurs in the real-world. Indeed, since D is honest, then non of the $\mathcal{F}_{\mathsf{vss}}$ and $\mathcal{F}_{\mathsf{tss}}$ calls end with "D is corrupt", and, in addition, all honest parties are in V, so $|\mathsf{V}| \geq n - t$ and the dealer is not discarded. For every $i \in \{1, \dots, m\}$ and $k \in \mathsf{H} \cup \mathsf{W}$ it holds that $t_{ik} = \sum_{p \in \{1,\dots,\ell\}} \alpha^i_p F^p(0,k) + \sum_{p,q \in \{1,\dots,\ell\}} \alpha^i_{pq} F^{pq}(0,k) + F^{\eta_i}(0,k)$, where F^p, F^{pq} and F^{η_i} are the sharing polynomials picked by the dealer in the first round.

It remains to consider $\mathcal{F}^{i,k}_{\mathsf{glinear}}$, for $i \in \{1, \dots, m\}$ and corrupt P_k in V. Since P_k is in V, then P_k must input $\delta^G = (1, \dots, 1)$ to $\mathcal{F}^{i,k}_{\mathsf{glinear}}$. In addition, all honest parties P_j input $\delta^j = (1, \dots, 1)$ and the values $(F^1(k,j), \dots, F^\ell(k,j), F^{11}(k,j), \dots, F^{\ell\ell}(k,j))$. For an honest P_j, denote the corresponding input values of the corrupt guide P_k by $(b^G_{1,j}, \dots, b^G_{\ell,j}, b^G_{1,1,j}, \dots, b^G_{\ell,\ell,j}, b^G_{\eta_i,j})$, and we claim that $b^G_{p,j} = F^p(k,j)$, $b^G_{p,q,j} = F^{pq}(k,j)$ and $b^G_{\eta_i,j} = F^{\eta_i}(k,j)$, for all $p, q \in \{1, \dots, \ell\}$. Indeed, assume towards contradiction that $b^G_{p,j} \neq F^p(k,j)$ for some $p \in \{1, \dots, \ell\}$ (the other cases are similar). Then $p \in I_j$, and let o be the corresponding opening of the guide. Since P_k is in V then the output of $\mathcal{F}^{i,k}_{\mathsf{glinear}}$ is not "the guide is corrupt", so necessarily $\mathsf{open}_{\mathsf{crs}}(C^p_{kj}, o) = b^G_{p,k} \neq F^p(k,j)$, in contradiction to View being good. We conclude that $t_{ik} = \sum_{p \in \{1,\dots,\ell\}} \alpha^i_p F^p(0,k) + \sum_{p,q \in \{1,\dots,\ell\}} \alpha^i_{pq} F^{pq}(0,k) + F^{p_i}(x,k)$. Therefore, the for every $i \in \{1, \dots, m\}$ the shares $\{t_{ik}\}_{k \in \mathsf{V} \cup \mathsf{W}}$ correspond to the degree-t polynomial $\sum_{p \in \{1,\dots,\ell\}} \alpha^i_p F^p(0,x) + \sum_{p,q \in \{1,\dots,\ell\}} \alpha^i_{pq} F^{pq}(0,x) + F^{\eta_i}(0,x)$, whose free coefficient is $\sum_{p \in \{1,\dots,\ell\}} \alpha^i_p F^p(0,0) + \sum_{p,q \in \{1,\dots,\ell\}} \alpha^i_{pq} F^{pq}(0,0) = \sum_{p \in \{1,\dots,\ell\}} \alpha^i_p z^p + \sum_{p,q \in \{1,\dots,\ell\}} \alpha^i_{pq} z^{pq} = y^i(\mathbf{z})$, as required. This concludes the case of an honest dealer.

B.4.2 Corrupt Dealer

Input phase. The simulator takes the role of the honest parties, that have no inputs, and executes an instance of sif with $\mathcal{A}$. In the input phase this only includes receiving from $\mathcal{A}$ commitments and openings for the $\mathcal{F}_{\mathsf{vss}}$ and $\mathcal{F}_{\mathsf{tss}}$ calls, as well as to the dealer's broadcast, and giving the corresponding shares to the simulated honest parties.

Output phase. The simulator continues the simulation of the honest parties. The simulator computes the messages from the honest parties to the corrupt parties, as well as the leakage from the various functionalities, and gives them to $\mathcal{A}$. Then the simulator receives the messages from the corrupt parties to the honest parties, as well as the output of the various functionalities.

At the end of the simulation, if D was disqualified and the output was set to $y(0, \dots, 0)$, then the simulator inputs $(0, \dots, 0)$ to $\mathcal{F}_{\mathsf{sif}}$ and terminates. Otherwise, D was not disqualified in any $\mathcal{F}_{\mathsf{vss}}$ call, and let $F^i(x, y)$ be the sharing polynomial of the weak double t-sharing of $\mathcal{F}^i_{\mathsf{vss}}$. The simulator sets $z^i := F^i(0, 0)$ and inputs $(z^1, \dots, z^\ell)$ to $\mathcal{F}_{\mathsf{sif}}$.

Fix any polynomial-time environment $\mathcal{Z}$ and input ζ to the environment. We show that the view of $\mathcal{Z}$ (including the adversary's view and the honest parties' outputs) in the real world is close to the view of $\mathcal{Z}$ in the ideal world.

$\mathcal{Z}$'s view. Since the honest parties hold no inputs, it is not hard to see that the simulator perfectly simulates an execution of $\mathcal{F}_{\mathsf{sif}}$. It remains to show that the output of the honest parties has the same distribution in both worlds.

Honest parties' outputs. We say that a view View is "good" if for any commitment C that appears in the view, and for any two openings o, and o' that appear in the view, it holds that either $\mathsf{open}_{\mathsf{crs}}(C, o) = \bot$ or $\mathsf{open}_{\mathsf{crs}}(C, o') = \bot$ or $\mathsf{open}_{\mathsf{crs}}(C, o) = \mathsf{open}_{\mathsf{crs}}(C, o)$. Observe that, by the binding property of the commitment scheme, a view View is good with probability at least $1 - \epsilon$.

Fix any good view View. If D is disqualified according to View, then the output in both worlds is $y(0, \dots, 0)$. Therefore, we focus on the case where D is not disqualified according to View. Since D is not disqualified in View, then each $\mathcal{F}_{\mathsf{vss}}$ call defines a weak double t-sharing, and we denote by $F^p(x, y)$ (resp., $F^{pq}(x, y)$) the sharing polynomial of $\mathcal{F}^p_{\mathsf{vss}}$ (resp., $\mathcal{F}^{pq}_{\mathsf{vss}}$). In addition, for every $p, q \in \{1, \dots, \ell\}$ the output of $\mathcal{F}^{pq}_{\mathsf{tss}}$ defines three weak double t-sharings, and we denote the corresponding sharing polynomials by $\tilde{F}^p(x, y)$, $\tilde{F}^q(x, y)$, and $\tilde{F}^{pq}(x, y)$. Similarly, each tss^i call defines a weak double t-sharing, and we denote by $\tilde{F}^{\gamma_i}(x, y)$, $\tilde{F}^{\rho_i}(x, y)$ and $\tilde{F}^{\eta_i}(x, y)$ the corresponding sharing polynomials. We also denote by $F^{\gamma_i}(x, y)$, $F^{\rho_i}(x, y)$ the polynomials that correspond to the dealer's broadcast $(\mathbf{C}^{\gamma_i}, \mathbf{O}^{\gamma_i})$, and $(\mathbf{C}^{\rho_i}, \mathbf{O}^{\rho_i})$.

Since View is good, and there are $n - t \geq t + 1$ honest parties, by Fact A.4 it follows that that $F^p(x, y) = \tilde{F}^p(x, y)$ and $F^{pq}(x, y) = \tilde{F}^{pq}(x, y)$ for all $p, q \in \{1, \dots, \ell\}$. Similarly, it holds that $F^{\gamma_i}(x, y) = \tilde{F}^{\gamma_i}(x, y)$ and $F^{\rho_i}(x, y) = \tilde{F}^{\rho_i}(x, y)$ for every $i \in \{1, \dots, m\}$. Therefore, by the $\mathcal{F}_{\mathsf{tss}}$ functionality, we conclude that $F^p(0, 0) \cdot F^q(0, 0) = F^{pq}(0, 0)$ for all $p, q \in \{1, \dots, \ell\}$. We also conclude that $\tilde{F}^{\eta_i}(0, 0) = \tilde{F}^{\gamma_i}(0, 0) \cdot \tilde{F}^{\rho_i}(0, 0) = 0$.

Observe that each honest party is either in V or in W. We conclude that for every $i \in \{1, \dots, m\}$ and $j \in \mathsf{H}$ it holds that $t_{ij} = \sum_{p \in \{1, \dots, \ell\}} \alpha^i_p F^p(0, j) + \sum_{p,q \in \{1, \dots, \ell\}} \alpha^i_{pq} F^{pq}(0, j) + \tilde{F}^{\eta_i}(0, j)$. Since there are $n - t \geq t + 1$ honest parties, we conclude that for every $i \in \{1, \dots, m\}$ the shares $\{t_{ij}\}_{j \in \mathsf{V} \cup \mathsf{W}}$ must be consistent with the degree-t polynomial $\sum_{p \in \{1, \dots, \ell\}} \alpha^i_p F^p(0, x) + \sum_{p,q \in \{1, \dots, \ell\}} \alpha^i_{pq} F^{pq}(0, x) + \tilde{F}^{\eta_i}(x, 0)$, or otherwise they are inconsistent with any degree-t polynomial and D is disqualified. Therefore, the real-world honest parties output $(y^1, \dots, y^m)$, where $y^i = \sum_{p \in \{1, \dots, \ell\}} \alpha^i_p F^p(0, 0) + \sum_{p,q \in \{1, \dots, \ell\}} \alpha^i_{pq} F^{pq}(0, 0)$. In the ideal-world, the simulator picks $z^i := F^i(0, 0)$ for all $i \in \{1, \dots, \ell\}$, so the ideal-world honest parties output $(y^1, \dots, y^m)$, where $y^i = \sum_{p \in \{1, \dots, \ell\}} \alpha^i_p F^p(0, 0) + \sum_{p,q \in \{1, \dots, \ell\}} \alpha^i_{pq} F^{pq}(0, 0)$ as well. This concludes the proof of security of protocol sif. □

B.5 Verify & Open

Proof of Lemma 3.15. In this section we prove that protocol vao UC-emulates $\mathcal{F}_{\mathsf{vao}}$ (with everlasting security if the underlying commitment-scheme is statistically-hiding). From the composition properties of UC-security, it is enough to prove security in the $(\mathcal{F}_{\mathsf{sif}}, \mathcal{F}_{\mathsf{spcg}})$-hybrid model. Let $\mathcal{A}$ be an efficient adversary against vao. We define the simulator $\mathcal{S}$ as follows. The simulator $\mathcal{S}$ uses $\mathcal{A}$ in a black-box manner, and forwards all messages between $\mathcal{Z}$ and $\mathcal{A}$. The simulator first receives the set of corrupt C parties from $\mathcal{Z}$. We split into cases.

B.5.1 Honest Dealer

Verification phase, offline round. The simulator takes the role of an honest dealer in the offline phase. That is, the simulator picks a random degree-t polynomial $h(x)$ and computes $(C'_i, o'_i) \leftarrow \mathsf{commit}_{\mathsf{crs}}(h(i), r_i)$ for every $i \in \{1, \ldots, n\}$, where r_i is a fresh random string. The simulator broadcasts $\{C'_i\}_{i \in \{1,\ldots,n\}}$ on behalf of the dealer, and sends o'_i to every corrupt P_i.

Verification phase, online round. The simulator receives commitments $(C_1, \ldots, C_n)$, openings $\{o_i^D\}_{i \in \mathsf{C}}$ and verify_D as a leakage from $\mathcal{F}_{\mathsf{vao}}$. We split into cases.

- *(Case I)* If, in addition, the simulator receives the openings $\{o_k^D\}_{k \in \{1,\ldots,n\}}$ and $\{o_k\}_{k \in \mathsf{H}}$ as a leakage from $\mathcal{F}_{\mathsf{vao}}$, then the simulator holds all the inputs of the honest parties. The simulator takes the role of the honest parties, and continues the execution of vao.
- *(Case II)* Otherwise, the adversary does not receive any additional leakage, which in particular means that $\mathsf{open}_{\mathsf{crs}}(C_i, o_i^D) \neq \perp$ for every $i \in \{1, \ldots, n\}$. The simulator picks a random polynomial $\bar{g}(x)$ conditioned on $\bar{g}(i) = h(i) + f_i$, for all $i \in \mathsf{C}$, where $f_i := \mathsf{open}_{\mathsf{crs}}(C_i, o_i^D)$.

  For $i \in \mathsf{H}$ the simulator simulates $\mathcal{F}^i_{\mathsf{spcg}}$ in the following way. The adversary receives the commitments (C_i, C'_i) as leakage, and $(\mathbf{a}, \delta^A, \delta^B, \bar{g}(i))$ as an output, where $\mathbf{a} := (1, 1)$ and $\delta^A = \delta^B = (1, 1)$.

  For $i \in \mathsf{C}$ the simulator simulates $\mathcal{F}^i_{\mathsf{spcg}}$ in the following way. The adversary receives (C_i, C'_i), and Alice's inputs to $\mathcal{F}^i_{\mathsf{spcg}}$, i.e., $\mathbf{a} := (1, 1)$, the values $\mathbf{b}^A := (f_i, h(i))$, the indicator vector $\delta^A := (1, 1)$ and $\{o_i^D, o'_i\}$ as leakage from $\mathcal{F}_{\mathsf{spcg}}$. At this stage, for each $i \in \mathsf{C}$ the adversary sends the corrupt Bob's inputs to $\mathcal{F}^i_{\mathsf{spcg}}$ Upon receiving Bob's inputs to $\mathcal{F}^i_{\mathsf{spcg}}$ the simulator, that holds all the inputs, computes the output of $\mathcal{F}^i_{\mathsf{spcg}}$ and gives it to $\mathcal{A}$.

Opening phase. The simulator receives $(o_1^D, \ldots, o_n^D)$ and $(o_1, \ldots, o_n)$ from the functionality. If the verification phase ended with "verification failed" then the simulator send no messages on behalf of the honest parties, and terminates. Otherwise, the verification phase ended with "verification succeeded". If Case I occurred, then the dealer holds all the inputs of the honest parties, and simply continues the execution of vao until termination. If Case II occurred, the dealer computes $f_i := \mathsf{open}_{\mathsf{crs}}(C_i, o_i)$ and returns $(C_i, C'_i, \bar{g}(i), f_i)$ as the output of sif^i, for every $i \in \mathsf{H}$. At this stage the dealer receives the inputs of the adversary to $\mathcal{F}^i_{\mathsf{sif}}$ for $i \in \mathsf{C}$, computes the outputs and returns it to $\mathcal{A}$. This completes the simulation.

We assume without loss of generality that $\mathcal{A}$ is the dummy adversary (see Section A.2), and we fix a polynomial-time environment $\mathcal{Z}$ with input ζ. We say that the honest parties' inputs are *good*, if it holds that (1) $\mathsf{verify}_D = 0$, (2) for every $k \in \{1,\ldots,n\}$, the value $f_k := \mathsf{open}_{\mathsf{crs}}(C_k, o_k^D)$ is not $\perp$, (3) $\{f_k\}_{k\in\{1,\ldots,n\}}$ correspond to a degree t polynomial, and (4) for every $k \in \mathsf{H}$ it holds that $\mathsf{open}_{\mathsf{crs}}(C_k, o_k) = f_k$. Otherwise, we say that the honest parties' inputs are *bad*. We say that an environment $\mathcal{Z}'$ and an advice string z' are *degenerate*, if $\mathcal{Z}'(z')$ never corrupt the dealer, and either the inputs to the honest parties are good with probability 1, or the inputs to the honest parties are bad with probability 1.

Note that if there is an environment $\mathcal{Z}$ thad does not corrupt the dealer and distinguishes the real-world from the ideal-world with advantage δ, then there exists a degenerate environment $\mathcal{Z}'$ that distinguishes the real-world from the ideal-world with advantage $\delta/2$. Therefore, we assume without loss of generality that $\mathcal{Z}$ is degenerate. We show that $\text{REAL}_{\mathsf{vao},\mathcal{Z}(\zeta),\mathcal{A}}$ is $O(n\epsilon)$-close to $\text{IDEAL}_{\mathcal{F}_{\mathsf{vao}},\mathcal{Z}(z),\mathcal{S}}$.

Good inputs. Assume that $\mathcal{Z}$ always provides good inputs to the honest parties. We begin by analysing the real-world. We say that a real-world execution is "good" if for any commitment C that appears in the execution, and for any two openings o, and o' that appear in the view, it holds that either $\mathsf{open}_{\mathsf{crs}}(C, o) = \perp$ or $\mathsf{open}_{\mathsf{crs}}(C, o') = \perp$ or $\mathsf{open}_{\mathsf{crs}}(C, o) = \mathsf{open}_{\mathsf{crs}}(C, o)$. Observe that, by the binding property of the commitment scheme, an execution is good with probability at least $1 - \epsilon$.

We claim that in a good execution the output of the honest parties in the verification phase is always "verification succeeded", and the output in the opening phase is $f(0)$, where $f(x)$ is the polynomial defined by the commitments and openings $((C_1, o_1^D), \ldots, (C_n, o_n^D))$ which are generated by $\mathcal{Z}$. Indeed, since the inputs are good then the dealer does not broadcast "verification failed". In addition, the output of $\mathcal{F}_{\mathsf{sif}}^D$ is $(C'_1, \ldots, C'_n)$ and out $= 1$, and the output of each $\mathcal{F}_{\mathsf{spcg}}^i$ according to View is $(\mathbf{a}, \delta^G, f(i) + h(i))$, where $\mathbf{a} = (1,1)$ and $\delta^G = (1,1)$. Moreover, for $i \in \mathsf{C}$ the output of $\mathcal{F}_{\mathsf{spcg}}^i$ is either "Bob is corrupt" or $(\mathbf{a}, \delta^G, f(i)+h(i))$, or $(\mathbf{a}, \delta^G, \{(o_i^D, f(i)), (o'_i, h(i))\}, f(i)+h(i))$, or $(\mathbf{a}, \delta^G, \{(o_i^D, f(i))\}, f(i) + h(i))$ or $(\mathbf{a}, \delta^G, \{(o'_i, h(i))\}, f(i) + h(i))$, where $\mathbf{a} = (1,1)$ and $\delta^G = (1,1)$. Therefore, the verification phase ends with "verification succeeded". In the opening phase, it is not hard to see that all honest parties are in V' and that $s_i = f(i)$ for every honest P_i. For a corrupt P_i in V', if $I_i = \varnothing$ then, since the execution is good, $s_i = f(i)$, and otherwise $I_i = \{1, 2\}$ so $s_i = b_1^{i,A} = f(i)$ as well. Since there are $n - t \geq t + 1$ honest parties, we conclude that $S(x) = f(x)$, so all parties output $f(0)$, as required.

Consider a hybrid-world where the execution is done exactly like in the real-world, except that the output of the honest parties is always set to "verification succeeds" in the verification phase, and $f(0)$ in the output phase. Denote by $\text{REAL}'_{\mathsf{vao},\mathcal{Z}(\zeta),\mathcal{A}}$ the distribution of $\mathcal{Z}$'s view together with the honest parties' inputs. By the above analysis $\text{REAL}_{\mathsf{vao},\mathcal{Z}(\zeta),\mathcal{A}}$ is ϵ-close to $\text{REAL}'_{\mathsf{vao},\mathcal{Z}(\zeta),\mathcal{A}}$. We continue by showing that $\text{IDEAL}_{\mathcal{F}_{\mathsf{vao}},\mathcal{Z}(z),\mathcal{S}}$ is $O(n\epsilon)$-close to $\text{REAL}'_{\mathsf{vao},\mathcal{Z}(\zeta),\mathcal{A}}$.

The hybrid-world distribution

$$(\mathsf{crs}, (C'_1, \ldots, C'_n), \{o'_i\}_{i\in\mathsf{C}}, h(x)),$$

is $O(n\epsilon)$-close to the ideal-world distribution

$$(\mathsf{crs}, (C'_1, \ldots, C'_n), \{o'_i\}_{i\in\mathsf{C}}, \bar{h}(x)),$$

where $\bar{h}(x)$ is a random degree-t polynomial, sampled conditioned on $\bar{h}(i) = \mathsf{open}_{\mathsf{crs}}(C'_i, o'_i)$ for all $i \in \mathsf{C}$. Since in both worlds the rest of the adversary's view and the output of the honest parties can be obtained by the following efficient process: (1) Use crs, $(C'_1, \ldots, C'_n)$ and $\{o'_i\}_{i \in \mathsf{C}}$ to simulate the offline phase of the verification phase. (2) Obtain from $\mathcal{Z}$ the good inputs of the verification phase, $(C_1, \ldots, C_n)$, $(o^D_1, \ldots, o^D_n)$, $\mathsf{verify}_D = 0$ and $(o_i)_{i \in \mathsf{H}}$. We denote by $f(x)$ the degree-t polynomial corresponding to $((C_1, o^D_1), \ldots, (C_n, o^D_n))$. (3) simulate the $\mathcal{F}^i_{\mathsf{spcg}}$ calls for honest P_i by leaking (C_i, C'_i) and giving the output $\mathbf{a} = (1, 1)$, $\delta^A = \delta^B = (1, 1)$, and $h(i) + f(i)$ to $\mathcal{A}$. (4) For $i \in \mathsf{C}$ we hold all inputs of D to $\mathcal{F}^i_{\mathsf{spcg}}$, so only need to receive the output of the corrupt Bob from $\mathcal{A}$. (5) Set the output of $\mathcal{F}^D_{\mathsf{sif}}$ to be $(C'_1, \ldots, C'_n)$ and $\mathsf{out} = 1$. (6) Set the outputs of the honest parties in the verification phase to be "verification succeeded". (7) For any honest P_i, the output of $\mathcal{F}^i_{\mathsf{sif}}$ is $(C_i, C'_i, h(i) + f(i), f(i))$. For corrupt P_i receive the inputs from $\mathcal{F}^i_{\mathsf{sif}}$ from $\mathcal{A}$ and compute the corresponding output. (8) Set the output of the honest parties in the open phase to be $f(0)$.

We conclude that $\text{REAL}'_{\mathsf{vao}, \mathcal{Z}(\zeta), \mathcal{A}}$ is $O(n\epsilon)$-close to $\text{IDEAL}_{\mathcal{F}_{\mathsf{vao}}, \mathcal{Z}(z), \mathcal{S}}$. This concludes the case of good inputs.

Bad inputs. Assume that $\mathcal{Z}$ always provides bad inputs to the honest parties. In this case the simulator holds the honest parties' inputs, and the simulation simply consists of an execution of the protocol.

We say that an execution is "good" if for any commitment C that appears in the execution, and for any two openings o, and o' that appear in the view, it holds that either $\mathsf{open}_{\mathsf{crs}}(C, o) = \perp$ or $\mathsf{open}_{\mathsf{crs}}(C, o') = \perp$ or $\mathsf{open}_{\mathsf{crs}}(C, o) = \mathsf{open}_{\mathsf{crs}}(C, o)$. Observe that, by the binding property of the commitment scheme, an execution is good with probability at least $1 - \epsilon$. A similar analysis to the one in the good inputs case shows that if an execution is good then (1) if the dealer has broadcasted "verification failed" then the output of the verification phase is "verification failed", and the output of the opening phase is $\perp$, (2) if the dealer did not broadcast "verification failed" then the output of the verification phase is "verification succeeded" and the output in the opening phase is $f(0)$, where $f(x)$ is the polynomial defined by the commitments and openings $((C_1, o^D_1), \ldots, (C_n, o^D_n))$ which are generated by $\mathcal{Z}$.

Consider an hybrid-world where the execution is done exactly like in the real-world, except that whenever the dealer does not broadcast "verification failed" then the output of the honest parties is always set to "verification succeeds" in the verification phase, and $f(0)$ in the output phase. Denote by $\text{REAL}'_{\mathsf{vao}, \mathcal{Z}(\zeta), \mathcal{A}}$ the distribution of $\mathcal{Z}$'s view together with the honest parties' inputs. By the above analysis $\text{REAL}_{\mathsf{vao}, \mathcal{Z}(\zeta), \mathcal{A}}$ is ϵ-close to $\text{REAL}'_{\mathsf{vao}, \mathcal{Z}(\zeta), \mathcal{A}}$. In addition, it is not hard to see that $\text{REAL}'_{\mathsf{vao}, \mathcal{Z}(\zeta), \mathcal{A}}$ is identically distributed as $\text{IDEAL}_{\mathcal{F}_{\mathsf{vao}}, \mathcal{Z}(z), \mathcal{S}}$. Therefore, $\text{REAL}_{\mathsf{vao}, \mathcal{Z}(\zeta), \mathcal{A}}$ is ϵ-close to $\text{IDEAL}_{\mathcal{F}_{\mathsf{vao}}, \mathcal{Z}(z), \mathcal{S}}$. This concludes the case of an honest dealer.

B.5.2 Corrupt Dealer

Offline round. The simulator takes the role of the honest parties. The simulator receives from the corrupt dealer the commitments $(C'_1, \ldots, C'_n)$ and openings $\{o'_i\}_{i \in \mathsf{H}}$ and gives them to the simulated honest parties.

Online round. The simulator receives from $\mathcal{F}_{\mathsf{vao}}$ the inputs of the honest parties, which consists of the openings $\{o_i\}_{i \in \mathsf{H}}$. The simulator continues to simulate the honest parties using o_i as the input of an honest P_i, and gives the adversary all messages from honest parties to corrupt parties, including the leakage from the $\mathcal{F}_{\mathsf{spcg}}$ calls. At this stage, the simulator receives from the adversary the inputs $(\hat{C}'_1, \ldots, \hat{C}'_n)$ and $(\hat{o}'_1, \ldots, \hat{o}'_n)$ to $\mathcal{F}_{\mathsf{sif}}$, and gives to all parties the corresponding output of $\mathcal{F}_{\mathsf{sif}}$. The simulator also receives from the adversary the inputs of the corrupt Alice to $\mathcal{F}^{i}_{\mathsf{spcg}}$ for $i \in \{1, \ldots, n\}$, as well as the inputs of the corrupt Bob to $\mathcal{F}^{i}_{\mathsf{spcg}}$ for $i \in \mathsf{C}$. The simulator computes the output of each $\mathcal{F}^{i}_{\mathsf{spcg}}$ and gives it to the simulated honest parties and to the adversary. The simulator continues to simulate the honest parties, by computing their output.

If the output is "D is corrupt", then the simulator inputs $\mathsf{flag}_D = 1$ to $\mathcal{F}_{\mathsf{vao}}$ (the rest of the inputs do not matter). Otherwise, if the output is "verification failed", the simulator inputs $\mathsf{verify}_D = 1$ and $\mathsf{flag}_D = 0$ (the rest of the inputs do not matter). Otherwise the honest parties output "verification succeeded". For $i \in \mathsf{H}$, recall that an honest P_i holds o_i (resp., o'_i) as the opening of C_i (resp., C'_i). We also denote by $\bar{o}_i$ (resp., $\bar{o}'_i$) the input that the corrupt dealer sent to $\mathcal{F}^{i}_{\mathsf{spcg}}$ as the opening of C^i (resp., C'_i). If it holds that $\mathsf{open}_{\mathsf{crs}}(C_i, o_i) \neq \bot$, then set $o_i^D = o_i$. Otherwise, we set $o_i^D = \bar{o}_i$. For $i \in \mathsf{C}$ set $o_i^D = \bot$. The simulator inputs $\mathsf{flag}_D = 0$, $\mathsf{verify}_D = 0$ and $(o_1^D, \ldots, o_n^D)$ to $\mathcal{F}_{\mathsf{vao}}$.

Opening phase. The simulator, that holds all the honest parties inputs, simply continues the simulation of the honest parties, computing the outputs from $\mathcal{F}^{i}_{\mathsf{sif}}$ for every $i \in \mathsf{H}$ and delivering them to $\mathcal{A}$, and then receiving from the adversary the inputs to $\mathcal{F}^{i}_{\mathsf{sif}}$ for $i \in \mathsf{C}$, and delivering the corresponding outputs to the adversary as well. This concludes the simulation.

We assume without loss of generality that $\mathcal{A}$ is the dummy adversary (see Section A.2), and we fix a polynomial-time environment $\mathcal{Z}$ with input ζ, which we assume without loss of generality to be deterministic. We show that the view of $\mathcal{Z}$ together with the honest parties' outputs are *statistically close* in both worlds, even when the underlying commitment scheme is only computationally-hiding.

We say that a View is "good" if for any commitment C that appears in the view, and for any two openings o, and o' that appear in the view, it holds that either $\mathsf{open}_{\mathsf{crs}}(C, o) = \bot$ or $\mathsf{open}_{\mathsf{crs}}(C, o') = \bot$ or $\mathsf{open}_{\mathsf{crs}}(C, o) = \mathsf{open}_{\mathsf{crs}}(C, o)$. Observe that, by the binding property of the commitment scheme, a view View is good with probability at least $1 - \epsilon$. We continue by analysing the verification phase.

Verification phase. First, it is not hard to see that the view of $\mathcal{Z}$ in the verification phase is perfectly simulated by $\mathcal{S}$. Fix any good view of the verification phase. We continue by showing that the output of the honest parties is the same in both worlds.

We denote by o_i the input of an honest P_i. Observe that for each commitment C_i the adversary produces various openings. Indeed, when $i \in \mathsf{H}$ the adversary sends an opening $\bar{o}_i$ as an input to $\mathcal{F}^{i}_{\mathsf{spcg}}$. When $i \in \mathsf{C}$ the adversary sends an opening $\tilde{o}_i$ as an input to $\mathcal{F}^{i}_{\mathsf{sif}}$, and an opening $\bar{o}_i$ might appear in the output of $\mathcal{F}^{i}_{\mathsf{spcg}}$ (if it does not appear then we set $\bar{o}_i := \bot$).

Similarly, for each commitment C'_i the adversary produces various openings. Indeed, when $i \in \mathsf{H}$ the adversary sends an opening o'_i to P_i in the first round, an opening $\hat{o}'_i$ as an input to $\mathcal{F}^{D}_{\mathsf{sif}}$, and an opening $\bar{o}'_i$ as an input to $\mathcal{F}^{i}_{\mathsf{spcg}}$. When $i \in \mathsf{C}$ the adversary sends an opening $\tilde{o}'_i$ as an input to $\mathcal{F}^{i}_{\mathsf{sif}}$, an opening $\hat{o}'_i$ as an input to $\mathcal{F}^{D}_{\mathsf{sif}}$, and an additional opening $\bar{o}'_i$ might appear in the output of $\mathcal{F}^{i}_{\mathsf{spcg}}$ (if it does not appear then we set $\bar{o}'_i := \bot$).

Observe that whenever the output of the honest parties according to View is "D is corrupt" or "verification failed" then this is the output both in the real-world and the ideal-world. Therefore, we focus on the case where the output in the real-world is "verification succeeded".

Whenever the dealer is not disqualified and View is good, one of the following holds for every $i \in \mathsf{H}$.

- If the openings o_i and o'_i are valid openings to C_i and C'_i, respectively, let $f_i := \mathsf{open}_{\mathsf{crs}}(C_i, o_i)$ and $h_i := \mathsf{open}_{\mathsf{crs}}(C'_i, o'_i)$. In this case the output of $\mathcal{F}^i_{\mathsf{spcg}}$ is either $(\mathbf{a}, \delta^{A,i}, f_i + h_i)$ or $(\mathbf{a}, \delta^{A,i}, \{(\bar{o}_i, f_i), (\bar{o}'_i, h_i)\}, f_i + h_i)$, where $\mathbf{a} = (1,1)$, $\delta^{A,i} = (1,1)$ (the latter occurs when D sets $\mathsf{reveal} = 1$ as an input to $\mathcal{F}_{\mathsf{spcg}}$). Otherwise,
- At least one of the openings o_i and o'_i is not a valid opening to C_i or C'_i. In this case the output of $\mathcal{F}_{\mathsf{spcg}}$ is $(\mathbf{a}, \delta^{A,i}, \{(\bar{o}_i, f_i), (\bar{o}'_i, h_i)\}, f_i + h_i)$, where $\mathbf{a} = (1,1)$, $\delta^{A,i} = (1,1)$, $f_i := \mathsf{open}_{\mathsf{crs}}(C_i, \bar{o}_i)$ and $h_i := \mathsf{open}_{\mathsf{crs}}(C'_i, \bar{o}'_i)$.

Since D is not disqualified, the shares $\{f_i + h_i\}_{i \in \mathsf{H}}$ correspond to a degree-t polynomial, and the output of $\mathcal{F}^D_{\mathsf{sif}}$ is $(C'_1, \ldots, C'_n)$ and $\mathsf{out} = 1$. Since View is good, the shares $\{h_i\}_{i \in \mathsf{H}}$ also correspond to a degree-t polynomial $h(x)$, so the shares $\{f_i\}_{i \in \mathsf{H}}$ correspond to a degree-t polynomial $f(x)$. We conclude that, in the ideal-world, $\mathsf{open}_{\mathsf{crs}}(C_i, o^D_i) = f_i \neq \perp$ for all $i \in \mathsf{H}$, so in the ideal-world the output is "verification succeeded" and $s = f(0)$.

Opening phase. We continue by analysing the opening phase. Conditioned on a good view in the verification phase, and the honest parties' outputs, it is not hard to see that the view in the opening phase has the same distribution. We continue by showing that conditioned on the opening-phase view being good as well, the output in the real-world is the same as the output in the ideal-world.

If the verification phase ended with "verification failed" or "D is corrupt" then in both worlds the output is $\perp$. Hence we focus on the case where the output in the verification phase is "verification succeeded". In this case, the output in the ideal world is s. We continue by showing that this is also the output in the real-world. Observe that every honest party P_i is in V', and it holds that $s_i = f_i$. For a corrupt P_i in V', we split into cases.

- Consider a corrupt P_i in V' with $I_i = \varnothing$. Since D was not disqualified, it follows that the sum in the output of $\mathcal{F}^i_{\mathsf{spcg}}$ is $f(i) + h(i)$. Since the view is good it holds that $\mathsf{open}_{\mathsf{crs}}(C'_i, \hat{o}'_i) = h(i)$, so the value f_i in the output of $\mathcal{F}^i_{\mathsf{sif}}$ must be equal to $f(i)$, as required. Therefore $s_i = f(i)$.
- Consider a corrupt P_i in V' with $I_i = \{1, 2\}$. Let $f_i := \mathsf{open}_{\mathsf{crs}}(C_i, \bar{o}_i)$ and $h_i := \mathsf{open}_{\mathsf{crs}}(C'_i, \bar{o}'_i)$. Since D was not disqualified it follows that $h_i + f_i$ is the sum in the output of $\mathcal{F}^i_{\mathsf{spcg}}$, so $h_i + f_i = h(i) + f(i)$. Since the view is good, we conclude that $h_i = h(i)$. Therefore $f_i = f(i)$, so $s_i = f(i)$, as required.

Since there are $n - t \geq t + 1$ honest parties, and each one of them is in V', we conclude that $\{s_i\}_{i \in \mathsf{V}'}$ correspond to the degree-t polynomial $f(x)$, so the parties output $f(0) = s$. This concludes the proof of security. □

B.6 Guided Degree-2 Computation

Proof of Lemma 3.16. In this section we prove that protocol gdtc UC-emulates $\mathcal{F}_{\mathsf{gdtc}}$ (with everlasting security when the underlying commitment-scheme is statistically-hiding). From the composition properties of UC-security, it is enough to prove security in the $(\mathcal{F}_{\mathsf{glinear}}, \mathcal{F}_{\mathsf{tss}})$-hybrid model. Let $\mathcal{A}$ be an efficient adversary against gdtc. We define the simulator $\mathcal{S}$ as follows. The simulator $\mathcal{S}$ uses $\mathcal{A}$ in a black-box manner, and forwards all messages between $\mathcal{Z}$ and $\mathcal{A}$. The simulator first receives the set of corrupt C parties from $\mathcal{Z}$. We split into cases.

B.6.1 Honest Guide

Offline round. The simulator samples three strong double t-sharings of random values γ, ρ, η such that $\eta = \gamma\rho$, denoted $(\mathbf{C}^\gamma, \mathbf{O}^\gamma)$, $(\mathbf{C}^\rho, \mathbf{O}^\rho)$ and $(\mathbf{C}^\eta, \mathbf{O}^\eta)$. The simulator sends $(\mathbf{C}^\gamma, \mathbf{O}^\gamma_i)$, $(\mathbf{C}^\rho, \mathbf{O}^\rho_i)$, $(\mathbf{C}^\eta, \mathbf{O}^\eta_i)$ and $\mathsf{happy}_i := 1$ as the output of the sharing phase of $\mathcal{F}_{\mathsf{tss}}$. For $i \in \mathsf{C}$, denote by $\bar{f}^\gamma_i(x), \bar{f}^\rho_i(x)$ and $\bar{f}^\eta_i(x)$ the degree-t polynomials that correspond to $(\mathbf{C}^\gamma_i, \mathbf{O}^\gamma_i)$, $(\mathbf{C}^\rho_i, \mathbf{O}^\rho_i)$ and $(\mathbf{C}^\eta_i, \mathbf{O}^\eta_i)$, respectively.

Online round. The simulator receives the following leakage from $\mathcal{F}_{\mathsf{gdtc}}$: (1) the commitments $(C_{ij})_{i\in\{\alpha,\beta,1,\ldots,m\}, j\in\{0,\ldots,n\}}$, (2) the honest parties' indicator vectors $\{\delta^i\}_{i\in\mathsf{H}}$ and δ^G, (3) parts of the dealer's inputs $\mathbf{a}$, $\{b^G_{i,j}, o^G_{i,j}\}_{i\in\{\alpha,\beta,1,\ldots,m\}, j\in\mathsf{C}}$, (4) the values and openings $L_j = \{(i,j), b^G_{ij}, o^G_{ij}\}_{i\in I_j}$ for $j \in \mathsf{H}$. In addition, if $\delta^G_\alpha = 0$ or $\delta^G_\beta = 0$ then the simulator also receives (5) $z_j := \sum_{i\in\delta^G\setminus\{\alpha,\beta\}} a_i b^G_{ij}$ for any $j \in \{1,\ldots,n\}$. The simulator also receives the output of $\mathcal{F}_{\mathsf{gdtc}}$, denoted $(\mathbf{a}, \delta^G, z)$.

First, the simulator picks two random degree-t polynomials, $\bar{f}^u(x)$ and $\bar{f}^v(x)$ conditioned on

$$\bar{f}^u(j) = \delta_\alpha \cdot b^G_{\alpha j} - \bar{f}^\gamma_j(0) \quad \text{and} \quad \bar{f}^v(j) = \delta_\beta \cdot b^G_{\beta j} - \bar{f}^\rho_j(0)$$

for every $j \in \mathsf{C}$. The simulator sets $\bar{u} := \bar{f}^u(0)$ and $\bar{v} := \bar{f}^v(0)$.

Let $\delta := \delta^G_\alpha \wedge \delta^G_\beta$. If $\delta = 1$, the simulator picks a random degree-t polynomial $g(x)$ conditioned on

$$g(0) = z \quad \text{and} \quad g(j) = a_0(\bar{v}\bar{f}^\gamma_j(0) + \bar{u}\bar{f}^\rho_j(0) + \bar{f}^\eta_j(0) + \bar{u}\bar{v}) + \sum_{i\in\delta^G\setminus\{\alpha,\beta\}} a_i b^G_{ij}$$

for every $j \in \mathsf{C}$. Otherwise, when $\delta = 0$, the simulator sets $g(x)$ to be the degree-t polynomials obtained by interpolating $g(0) = z$ and $g(i) = z_i$ for $i \in \{1,\ldots,n\}$. (Observe that, since $\delta = 0$, the simulator holds the values $\{z_i\}_{i\in\{1,\ldots,n\}}$.)

The simulator sends $\mathbf{a}$ and δ^G to $\mathcal{A}$ as the broadcasts of the guide. The simulator sends $\{\mathsf{flag}_i := 0\}_{i\in\mathsf{H}}$ to $\mathcal{A}$ as the leakage of $\mathcal{F}_{\mathsf{tss}}$.

For the leakage of $\mathcal{F}^{\mathsf{out}}_{\mathsf{glinear}}$ the simulator sends to $\mathcal{A}$, (1) the commitments $(C_{ij})_{i\in\{1,\ldots,m\}, j\in\{0,\ldots,n\}}$ and $\mathbf{C}^\gamma_0$, $\mathbf{C}^\rho_0$, $\mathbf{C}^\eta_0$, as well as the commitments of the $\langle 1\rangle$ sharing $(C'_0,\ldots,C'_n)$, (2) the honest parties' indicator vectors $\{\delta^{i,\mathsf{out}} := (\delta^i_1,\ldots,\delta^i_m,1,1,1,1)\}_{i\in\mathsf{H}}$ and $\delta^{G,\mathsf{out}} := (\delta^G_1,\ldots,\delta^G_m,1,1,1,1)$, (3) parts of the dealer's inputs, the vector $\mathbf{a}^{\mathsf{out}} := (a_1,\ldots,a_m,\delta a_0\bar{v},\delta a_0\bar{u},\delta a_0,\delta a_0\bar{u}\bar{v})$, and the values $\{b^G_{i,j}, o^G_{i,j}\}_{i\in\{1,\ldots,m\}, j\in\mathsf{C}}$ and $\{\bar{f}^\gamma_j(0), \bar{f}^\rho_j(0), \bar{f}^\eta_j(0), o^\gamma_{0,j}, o^\eta_{0,j}, o^\rho_{0,j}\}_{j\in\mathsf{C}}$, as well as the shares and openings from $\langle 1\rangle$, where all the shares are 1, and the openings are all computed according to $\mathsf{commit}_{\mathsf{crs}}(1;\vec{0})$, (4) the values and openings $L^{\mathsf{out}}_j = \{(i,j), b^G_{ij}, o^G_{ij}\}_{i\in I_j\setminus\{\alpha,\beta\}}$ for $j \in \mathsf{H}$, and (5) $\{g(j)\}_{j\in\{1,\ldots,n\}}$. The simulator also sends $(\mathbf{a}^{\mathsf{out}}, \delta^{G,\mathsf{out}}, g(0))$ as the output of $\mathcal{F}^{\mathsf{out}}_{\mathsf{glinear}}$.

For the leakage of $\mathcal{F}^u_{\mathsf{glinear}}$, the simulator sends to $\mathcal{A}$, (1) the commitments $\{C_{\alpha j}\}_{j\in\{0,\ldots,n\}}$ and $\{C^\gamma_{0j}\}_{j\in\{0,\ldots,n\}}$, (2) the honest parties indicator vectors $\{\delta^{i,u} := (\delta^i_\alpha, 1)\}_{i\in\mathsf{H}}$ and $\delta^{G,u} := (\delta^G_\alpha, 1)$, (3) the vector $\mathbf{a}^u := (1,-1)$ and the values $\{b^G_{\alpha j}, o^G_{\alpha j}\}_{j\in\mathsf{C}}$ and $\{\bar{f}^\gamma(j), o^\gamma_{0j}\}_{j\in\mathsf{C}}$, (4) the values and openings $L^u_j = \{(\alpha,j), b^G_{\alpha j}, o^G_{\alpha j}\}$ for every $j \in \mathsf{H}$ such that $\alpha \in I_j$, and (5) the values $\{\bar{f}^u(j)\}_{j\in\{1,\ldots,n\}}$. The simulator also sends $(\mathbf{a}^u, \delta^{G,u}, \bar{f}^u(0))$ as the output of $\mathcal{F}^u_{\mathsf{glinear}}$.

For the leakage of $\mathcal{F}^v_{\mathsf{glinear}}$, the simulator sends to $\mathcal{A}$, (1) the commitments $\{C_{\beta j}\}_{j\in\{0,\ldots,n\}}$ and $\{C^\rho_{0j}\}_{j\in\{0,\ldots,n\}}$, (2) the honest parties indicator vectors $\{\delta^{i,v} := (\delta^i_\beta, 1)\}_{i\in\mathsf{H}}$ and $\delta^{G,v} := (\delta^G_\beta, 1)$, (3) the vector $\mathbf{a}^v := (1,-1)$ and the values $\{b^G_{\beta j}, o^G_{\beta j}\}_{j\in\mathsf{C}}$ and $\{\bar{f}^\rho(j), o^\rho_{0j}\}_{j\in\mathsf{C}}$, (4) The values and openings $L^v_j = \{(\beta,j), b^G_{\beta j}, o^G_{\beta j}\}$ for every $j \in \mathsf{H}$ such that $\beta \in I_j$, and (5) $\{\bar{f}^v(j)\}_{j\in\{1,\ldots,n\}}$. The simulator also sends $(\mathbf{a}^v, \delta^{G,v}, \bar{f}^v(0))$ as the output of $\mathcal{F}^v_{\mathsf{glinear}}$.

At this stage, the simulator receives the $\mathcal{F}_{\mathsf{tss}}$ inputs flag_i of the corrupt parties. The simulator sets W to be the set of all corrupt parties P_i with $\mathsf{flag}_i = 1$, and sends $(\mathsf{W}, \{o^{\gamma}_{ij}, o^{\rho}_{ij}, o^{\eta}_{ij}\}_{i \in \mathsf{W}, j \in \{0,\ldots,n\}})$ to $\mathcal{A}$ as the output of the verification phase of $\mathcal{F}_{\mathsf{tss}}$.

We assume without loss of generality that $\mathcal{A}$ is the dummy adversary (see Section A.2), and we fix a polynomial-time environment $\mathcal{Z}$ with input ζ. We may assume without loss of generality that $\mathcal{Z}$ is deterministic. We begin by showing that the view of $\mathcal{Z}$ in the real-world is close to the view of $\mathcal{Z}$ in the ideal world.

$\mathcal{Z}$'s view. Consider the real-world random variables

$$\Big(\mathsf{crs}, \{\mathbf{C}^w, \mathbf{O}^w_i\}_{w \in \{\gamma,\rho,\eta\}, i \in \mathsf{C}}, f^{\gamma}(x), f^{\rho}(x), f^{\eta}(x)\Big),$$

where $f^w(x) := F^w(x, 0)$, for $w \in \{\gamma, \rho, \eta\}$. We also consider the ideal-world random variables

$$\Big(\mathsf{crs}, \{\mathbf{C}^w, \mathbf{O}^w_i\}_{w \in \{\gamma,\rho,\eta\}, i \in \mathsf{C}}, \bar{f}^{\gamma}(x), \bar{f}^{\rho}(x), \bar{f}^{\eta}(x)\Big),$$

where $\{\mathbf{C}^w, \mathbf{O}^w_i\}_{w \in \{\gamma,\rho,\eta\}, i \in \mathsf{C}}$ are generated by the simulator, and $\bar{f}^{\gamma}(x)$, $\bar{f}^{\rho}(x)$ and $\bar{f}^{\eta}(x)$ are random degree-t polynomials conditioned on $\bar{f}^{\gamma}(0) \cdot \bar{f}^{\rho}(0) = \bar{f}^{\eta}(0)$ and $\bar{f}^{w}(i) = \mathsf{open}_{\mathsf{crs}}(C^w_{i0}, o^w_{i0})$ for all $i \in \mathsf{C}$ and $w \in \{\gamma, \rho, \eta\}$. Observe that those random variables are $O(n^2\epsilon)$-indistinguishable.

In both worlds the view of $\mathcal{Z}$ can be obtained by the following efficient process.

- *(Process' inputs)* The process receives $(\mathsf{crs}, \{\mathbf{C}^w, \mathbf{O}^w_i\}_{w \in \{\gamma,\rho,\eta\}, i \in \mathsf{C}}, f^{\gamma}(x), f^{\rho}(x), f^{\eta}(x))$ as inputs.
- *(Offline-round simulation)* The process simulates the offline-round view by giving $\mathcal{Z}$ the CRS, and the values $(\mathbf{C}^w, \mathbf{O}^w_i)$, for $w \in \{\gamma, \rho, \eta\}$ and $\mathsf{happy}_i = 1$, as the output of a corrupt P_i in the sharing phase of the tss call.
- *(Honest parties' inputs.)* At this stage, the process obtains from $\mathcal{Z}$ the inputs of the honest parties.
- *(Online-round simulation)* If $\delta^G_{\alpha} = 1$, let $f^{w^{\alpha}}(x)$ be the degree-t polynomial corresponding to $\{b^G_{\alpha j}\}_{j \in \{0,\ldots,n\}}$, let $f^u(x) := f^{w^{\alpha}}(x) - f^{\gamma}(x)$ and $u := f^u(0)$. Otherwise let $f^u(x) := 0 - f^{\gamma}(x)$ and $u := f^u(0)$. Similarly, if $\delta^G_{\beta} = 1$, let $f^{w^{\beta}}(x)$ be the degree-t polynomial corresponding to $\{b^G_{\beta j}\}_{j \in \{0,\ldots,n\}}$ and let $f^v(x) := f^{w^{\beta}}(x) - f^{\rho}(x)$ and $v := f^v(0)$. Otherwise, let $f^v(x) := 0 - f^{\rho}(x)$, and $v := f^v(0)$. Let $g(x) := \delta^G_{\alpha} \cdot \delta^G_{\beta} \cdot a_0(v f^{\gamma}(x) + u f^{\rho}(x) + f^{\eta}(x) + uv) + \sum_{i \in \delta^G \setminus \{\alpha,\beta\}} a_i f^{w^i}(x)$, where $f^{w^i}(x)$ is the degree-t polynomial corresponding to $\{b^G_{ij}\}_{j \in \{0,\ldots,n\}}$. (Observe that for any $i \in \delta^G \setminus \{\alpha, \beta\}$, $f^{w^i}(x)$ is indeed a degree-t polynomial.)

  Continue the simulation of the online-round just like the simulator, using the honest parties' inputs obtained from $\mathcal{Z}$, the polynomials $f^u(x)$, $f^v(x)$ and $g(x)$, and the values u and v.

Indeed, it is not hard to see that when the inputs $(\mathsf{crs}, \{\mathbf{C}^v, \mathbf{O}^v_i\}_{v \in \{\gamma,\rho,\eta\}, i \in \mathsf{C}}, f^{\gamma}(x), f^{\rho}(x), f^{\eta}(x))$ are sampled from the real-world, the above process perfectly simulates the view of $\mathcal{Z}$. Similarly, when the inputs are take from the ideal-world, if $\delta^G_\alpha \wedge \delta^G_\beta = 0$ then it is not hard to see that the above process perfectly simulates the view of $\mathcal{Z}$. Hence we focus on the case where $\delta^G_\alpha \wedge \delta^G_\beta = 1$.

Fix any $(\mathsf{crs}, \{\mathbf{C}^w, \mathbf{O}^w_i\}_{w\in\{\gamma,\rho,\eta\}, i\in\mathsf{C}})$, which also fixes $\{f^\gamma(i), f^\rho(i), f^\eta(i)\}_{i\in\mathsf{C}}$. Observe that the random variables $(f^u(x), f^v(x), uf^\rho(x) + vf^\gamma(x) + f^\eta(x) + uv)$ which are generated by the process are distributed just like in Experiment 1 in Fact A.8, and so they have the same distribution as the random variables $(\bar{f}^u(x), \bar{f}^v(x), f(x))$, where $\bar{f}^u(x)$ is a uniform degree-t polynomial conditioned on $\bar{f}^u(i) = f^{w^\alpha}(i) - f^\gamma(i)$ for $i \in \mathsf{C}$, $\bar{f}^v(x)$ is a uniform degree-t polynomial conditioned on $\bar{f}^v(i) = f^{w^\beta}(i) - f^\rho(i)$ for $i \in \mathsf{C}$, and $f(x)$ is a uniform degree-t polynomial conditioned on $f(0) = f^{w^\alpha}(0) \cdot f^{w^\beta}(0)$ and $f(i) = \bar{u}f^\rho(i) + \bar{v}f^\gamma(i) + f^\eta(i) + \bar{u}\bar{v}$ for $i \in \mathsf{C}$, where $\bar{u} = \bar{f}^u(0)$ and $\bar{v} = \bar{f}^v(0)$. Observe that $(\bar{f}^u(x), \bar{f}^v(x))$ are distributed just like in the ideal-world, and fix those values as well. Since $a_0 \neq 0$, the random variable $g(x)$ which is generated by the simulator has the same distribution as $a_0 f(x) + \sum_{i\in\delta^G\setminus\{\alpha,\beta\}} a_i f^{w^i}(x)$, that has the same distribution as the random variable $g(x)$ generated by the process. This concludes the analysis of $\mathcal{Z}$'s view.

Honest parties' output. We claim that both in the real-world and the ideal-world, the outputs of the honest parties in the output phase can be extracted from a View by the following efficient deterministic process: For each $i \in \{\alpha, \beta, 1, \ldots, m\}$ such that $\delta^G_i = 1$, let $b^G_i := b^G_{i,0}$, and otherwise let $b^G_i = 0$, where δ^G and $b^G_{i,0}$ are the inputs of the honest guide according to View. Then each honest party outputs $(\mathbf{a}, \delta^G, a_0 \cdot b^G_\alpha b^G_\beta + \sum_{i\in\delta^G\setminus\{\alpha,\beta\}} a_i \cdot b^G_i)$.

This is clearly the output of the honest parties when View is taken from the ideal-world, so it remains to analyse the real-world. Fix any real-world view of the adversary. Observe that, since the guide is honest, the guide is never discarded. Indeed, the verification phase of $\mathcal{F}_{\mathsf{tss}}$ does not result in "D is corrupt", and none of the honest parties are in W. In addition, non of the calls to $\mathcal{F}_{\mathsf{glinear}}$ result in "G is corrupt", and it always holds that (1) $\mathbf{a}^{\mathsf{out}} = (a_1, \ldots, a_m, a_0\delta^G_\alpha\delta^G_\beta v, a_0\delta^G_\alpha\delta^G_\beta u, a_0\delta^G_\alpha\delta^G_\beta, a_0\delta^G_\alpha\delta^G_\beta uv)$, (2) $\mathbf{a}^u = (1, -1)$, (3) $\mathbf{a}^v = (1, -1)$, (4) $\delta^{\mathsf{out}} = (\delta^G_1, \ldots, \delta^G_n, 1, 1, 1, 1)$, (5) $\delta^u = (\delta^G_\alpha, 1)$, and (6) $\delta^v = (\delta^G_\beta, 1)$.

When $\delta^G_\alpha \wedge \delta^G_\beta = 1$ then, Fact A.8, the sum in the output of $\mathcal{F}^{\mathsf{out}}_{\mathsf{glinear}}$ is $a_0 \cdot b^G_{\alpha,0} \cdot b^G_{\beta,0} + \sum_{i\in\delta^G\setminus\{\alpha,\beta\}} a_i b^G_{i,0}$, so the output is $(\mathbf{a}, \delta^G, a_0 \cdot b^G_{\alpha 0} b^G_{\beta 0} + \sum_{i\in\delta^G\setminus\{\alpha,\beta\}} a_i \cdot b^G_{i0})$. When $\delta^G_\alpha \wedge \delta^G_\beta = 0$ then the sum in the output of $\mathcal{F}^{\mathsf{out}}_{\mathsf{glinear}}$ is $\sum_{i\in\delta^G\setminus\{\alpha,\beta\}} a_i \cdot b^G_{i0}$, so the output is $(\mathbf{a}, \delta^G, \sum_{i\in\delta^G\setminus\{\alpha,\beta\}} a_i \cdot b^G_{i0})$. This concludes the analysis of the honest parties' output.

B.6.2 Corrupt Guide

The simulator, that holds all the inputs of the honest parties, takes the role of the honest parties and simulates an execution of gdtc in the following way.

Offline round. The guide receives from $\mathcal{A}$ three pairs $(\mathbf{C}^\gamma, \mathbf{O}^\gamma)$, $(\mathbf{C}^\rho, \mathbf{O}^\rho)$, $(\mathbf{C}^\eta, \mathbf{O}^\eta)$ and $\{\mathsf{happy}_i\}_{i\in\{1,\ldots,n\}}$, as an input to $\mathcal{F}_{\mathsf{tss}}$. The guide sends $(\mathbf{C}^\gamma, \mathbf{O}^\gamma_i)$, $(\mathbf{C}^\rho, \mathbf{O}^\rho_i)$, $(\mathbf{C}^\eta, \mathbf{O}^\eta_i)$ and happy_i to P_i as the output of the sharing phase of $\mathcal{F}_{\mathsf{tss}}$.

Online round. The simulator receives the following leakage from $\mathcal{F}_{\mathsf{gdtc}}$: (1) the commitments $(C_{ij})_{i\in\{\alpha,\beta,1,\ldots,m\}, j\in\{0,\ldots,n\}}$, (2) the honest parties' indicator vectors $\{\delta^i\}_{i\in\mathsf{H}}$, and (3) the honest parties' values $\{\mathbf{b}^i\}_{i\in\mathsf{H}}$. The simulator, that holds all honest parties' inputs, continues the execution of gdtc by taking the role of the honest parties, for which the simulator holds all the inputs. At the end of the execution the simulator computes the outputs of the honest parties.

If the honest parties output "G is corrupt" then the simulator sends $\mathsf{flag} := 1$ to $\mathcal{F}_{\mathsf{gdtc}}$ (the rest of the inputs do not matter). Otherwise, let the broadcast of the corrupt guide be $(\mathbf{a}, \delta^G)$. Denote the adversary's input to $\mathcal{F}^u_{\mathsf{glinear}}$ by $(\mathbf{a}^{G,u}, \mathbf{b}^{G,u}, \{o^{G,u}_{\alpha j}, o^{\gamma}_{0j}\}_{j\in\{0,\ldots,n\}}, \delta^{G,u})$, and the output by $(\mathbf{a}^{G,u}, \delta^{G,u}, u)$. Similarly, denote the adversary's input to $\mathcal{F}^v_{\mathsf{glinear}}$ by $(\mathbf{a}^{G,v}, \mathbf{b}^{G,v}, \{o^{G,v}_{\alpha j}, o^{\gamma}_{0j}\}_{j\in\{0,\ldots,n\}}, \delta^{G,v})$, and the output by $(\mathbf{a}^{G,v}, \delta^{G,v}, v)$. Denote the adversary's input to $\mathcal{F}^{\mathsf{out}}_{\mathsf{glinear}}$ by $(\mathbf{a}^{G,\mathsf{out}}, \mathbf{b}^{G,\mathsf{out}}, \{o^{G,\mathsf{out}}_{ij}\}_{i\in\{1,\ldots,m\},j\in\{0,\ldots,n\}}, \mathbf{O}^{\gamma}_0, \mathbf{O}^{\rho}_0, \mathbf{O}^{\eta}_0, \delta^{G,u})$, and the output by $(\mathbf{a}^{G,\mathsf{out}}, \delta^{G,\mathsf{out}}, \mathsf{out})$. The simulator inputs (1) the list of coefficients $\mathbf{a}$, (2) the list of value $\mathbf{b}^G := (\mathbf{b}^{G,u}_{\alpha}, \mathbf{b}^{G,v}_{\beta}, \mathbf{b}^{G,\mathsf{out}}_1, \ldots, \mathbf{b}^{G,\mathsf{out}}_m)$, (3) the lists of openings $\{o^{G,u}_{\alpha,j}\}_{j\in\{0,\ldots,n\}}$, $\{o^{G,v}_{\beta,j}\}_{j\in\{0,\ldots,n\}}$, and $\{o^{G,\mathsf{out}}_{ij}\}_{i\in\{1,\ldots,m\},j\in\{0,\ldots,n\}}$, (4) the indicator vector δ^G, (5) the bit $\mathsf{flag} = 0$.

Fix any polynomial-time environment $\mathcal{Z}$ and input ζ to the environment. We show that the view of $\mathcal{Z}$ in the real world is statistically close to the view of $\mathcal{Z}$ in the ideal world, even when the underlying commitment scheme are merely computationally-hiding.

Adversary's view. Since the simulator, that takes the role of the honest parties, holds all the honest parties' inputs, the adversary's view has the same distribution in both worlds.

Honest parties' outputs. We say that a view View is "good" if for any commitment C that appears in the view, and for any two openings o, and o' that appear in the view, it holds that either $\mathsf{open}_{\mathsf{crs}}(C, o) = \perp$ or $\mathsf{open}_{\mathsf{crs}}(C, o') = \perp$ or $\mathsf{open}_{\mathsf{crs}}(C, o) = \mathsf{open}_{\mathsf{crs}}(C, o)$. Observe that a view is good with probability at least $1 - \epsilon$, and fix any good view.

If the output of the honest parties according to View is "G is corrupt" then the simulator sets $\mathsf{flag}_i = 1$ and so the output of the honest parties in the ideal-world is also "G is corrupt". Hence, we focus on the case where the output of the honest parties according to View is not "G is corrupt". In this case, in the real-world the output of the honest parties is $(\mathbf{a}, \delta^G, \mathsf{out})$. We show that this holds with probability 1 in the ideal world as well.

Let $F^{\gamma}(x,y)$, $F^{\rho}(x,y)$ and $F^{\eta}(x,y)$ be the sharing polynomials of the weak double t-sharing defined by the $\mathcal{F}_{\mathsf{tss}}$ calls, and let $f^{\gamma}(x) := F^{\gamma}(x,0)$, $f^{\rho}(x) := F^{\rho}(x,0)$ and $f^{\eta}(x) := F^{\eta}(x,0)$, and note that, by the correctness of the $\mathcal{F}_{\mathsf{tss}}$ functionality, $f^{\gamma}(0) \cdot f^{\rho}(0) = f^{\eta}(0)$. First, observe that in the real-world the adversary inputs $f^{\gamma}(j)$ (resp., $f^{\rho}(j)$) to $\mathcal{F}^u_{\mathsf{glinear}}$ (resp., $\mathcal{F}^v_{\mathsf{glinear}}$) as the committed value of C^{γ}_{0j} (resp., C^{ρ}_{0j}), for all $j \in \mathsf{H}$. Indeed, if this is not the case then, since View is good, it is not hard to see that the guide is disqualified. Similarly, the adversary inputs $f^{\gamma}(j)$, $f^{\rho}(j)$ and $f^{\eta}(j)$ to $\mathcal{F}^{\mathsf{out}}_{\mathsf{glinear}}$ as the committed values of C^{γ}_{0j}, C^{ρ}_{0j}, and C^{η}_{0j}, respectively, for every $j \in \mathsf{H}$.

In addition, in the ideal-world, for any $j \in \mathsf{H}$ and $i \in I_j$ it holds that $\mathsf{open}_{\mathsf{crs}}(C_{ij}, o^G_{ij}) = b^G_{ij}$. Indeed, if this equation does not hold, then, in the real-world, for $i = \alpha$ the guide will be disqualified in $\mathcal{F}^u_{\mathsf{glinear}}$, for $i = \beta$ the guide will be disqualified in $\mathcal{F}^v_{\mathsf{glinear}}$, and for $i \in \{1, \ldots, m\}$ the guide will be disqualified in $\mathcal{F}^{\mathsf{out}}_{\mathsf{glinear}}$, in contradiction. Also, observe that $a_0 \neq 0$, or otherwise, the guide will be disqualified.

We continue by splitting into two cases.

- Assume that $\delta^G_{\alpha} \wedge \delta^G_{\beta} = 1$. Let $f^{\alpha}(x)$ be the polynomial obtained by interpolating $\{b^G_{\alpha,j}\}_{j\in\mathsf{H}}$, let $f^{\beta}(x)$ be the polynomial obtained by interpolating $\{b^G_{\beta,j}\}_{j\in\mathsf{H}}$, and let $g(x)$ be the polynomial obtained by interpolating $\{\sum_{i\in\delta^G\setminus\{\alpha,\beta\}} a_i b^G_{i,j}\}_{j\in\mathsf{H}}$. We continue by showing that all of them are of degree-t.

Consider the polynomial $f^u(x)$ obtained by interpolating $\{b^G_{\alpha,j} - f^\gamma(j)\}_{j\in\mathsf{H}}$. Since the output of $\mathcal{F}^u_{\mathsf{glinear}}$ is not "G is corrupt", the degree of $f^u(x)$ is at most t. Since $f^\gamma(x)$ is of degree-t, we conclude that the degree of $f^\alpha(x)$ is at most t. Similarly, let $f^v(x)$ be the degree-t polynomial obtained by interpolating $\{b^G_{\beta,j} - f^\rho(j)\}_{j\in\mathsf{H}}$, and observe that, by a similar analysis, the degree of $f^\beta(x)$ is at most t. Let $h(x)$ be the polynomial obtained by interpolating $\{a_0(uf^\rho(j) + vf^\gamma(j) + f^\eta(j) + uv) + \sum_{i\in\delta^G\setminus\{\alpha,\beta\}} a_i b^G_{i,j}\}_{j\in\mathsf{H}}$. Since the output of $\mathcal{F}^{\mathsf{out}}_{\mathsf{glinear}}$ is not "G is corrupt", the degree of $h(x)$ is at most t. Since $f^\rho(x)$, $f^\gamma(x)$ and $f^\eta(x)$ are of degree at most t, we conclude that $g(x)$ is of degree at most t.

Finally, observe that according to View, $u = f^u(0)$, $v = f^v(0)$ and $\mathsf{out} = h(0)$. By the above analysis, and by Fact A.8, it is not hard to see that $\mathsf{out} = a_0 f^\alpha(0) f^\beta(0) + g(0)$, and that the output in the ideal-world is also $(\mathbf{a}, \delta^G, \mathsf{out})$.

- Assume that $\delta^G_\alpha \wedge \delta^G_\beta = 0$. Let $g(x)$ be the polynomial obtained by interpolating $\{\sum_{i\in\delta^G\setminus\{\alpha,\beta\}} a_i b^G_{i,j}\}_{j\in\mathsf{H}}$. Since the output of $\mathcal{F}^{\mathsf{out}}_{\mathsf{glinear}}$ is not "G is corrupt", the degree of $g(x)$ is at most t, and $\mathsf{out} = g(0)$. By the above analysis, it is not hard to see that the output in the ideal-world is also $(\mathbf{a}, \delta^G, g(0))$. This concludes the proof of security.

□

B.7 Degree-2 Computation

Proof of Lemma 3.17. In this section we prove that protocol dtc UC-emulates $\mathcal{F}_{\mathsf{dtc}}$ (with everlasting security when the underlying commitment scheme are statistically-hiding). From the composition properties of UC-security, it is enough to prove security in the $(\mathcal{F}_{\mathsf{vss}}, \mathcal{F}_{\mathsf{tss}}, \mathcal{F}_{\mathsf{sif}}, \mathcal{F}_{\mathsf{vao}}, \mathcal{F}_{\mathsf{dtc}}, \mathcal{F}_{\mathsf{glinear}},)$-hybrid model. Let $\mathcal{A}$ be an efficient adversary against gdtc. We define the simulator $\mathcal{S}$ as follows. The simulator $\mathcal{S}$ uses $\mathcal{A}$ in a black-box manner, and forwards all messages between $\mathcal{Z}$ and $\mathcal{A}$. The simulator first receives the set of corrupt C parties from $\mathcal{Z}$.

B.7.1 Round 1

$\mathcal{F}_{\mathsf{vss}}$ simulation. For every honest P_i, the simulator simulates the calls of P_i to $\mathcal{F}_{\mathsf{vss}}$ in the following way.

- For every $j \in \{L_{i-1}+1, \ldots, L_{i-1}+\ell_i\}$, the simulator picks a random strong double t-sharing $\langle\langle 0\rangle\rangle$ denoted $(\mathbf{C}^{w_j}, \mathbf{O}^{w_j})$, and uses it as the input to $\mathcal{F}^{w_j}_{\mathsf{vss}}$. That is, the simulator gives to the adversary $(\mathbf{C}^{w_j}, \mathbf{O}^{w_j}_r)$ for every $r \in \mathsf{C}$ as the output of the sharing phase of $\mathcal{F}^{w_j}_{\mathsf{vss}}$. For each $r \in \mathsf{C}$, denote by $\bar{f}^{w_j}_r(x)$ the degree-t polynomial defined by $(\mathbf{C}^{w_j}_r, \mathbf{O}^{w_j}_r)$.
- For every $k \in \{1, \ldots, m\}$ the simulator picks a random value p^k_i (resp., z^k_i), picks a random strong double t-sharing $\langle\langle p^k_i\rangle\rangle$ (resp., $\langle\langle z^k_i\rangle\rangle$), denoted $(\mathbf{C}^{p^k_i}, \mathbf{O}^{p^k_i})$ (resp.,$(\mathbf{C}^{z^k_i}, \mathbf{O}^{z^k_i})$) , and uses it as the input to $\mathcal{F}^{p^k_i}_{\mathsf{vss}}$ (resp., $\mathcal{F}^{z^k_i}_{\mathsf{vss}}$). That is, the simulator gives to the adversary $(\mathbf{C}^{p^k_i}, \mathbf{O}^{p^k_i}_j)$ (resp., $(\mathbf{C}^{z^k_i}, \mathbf{O}^{z^k_i}_j)$) for every $j \in \mathsf{C}$ as the output of $\mathcal{F}^{p^k_i}_{\mathsf{vss}}$ (resp., $\mathcal{F}^{z^k_i}_{\mathsf{vss}}$). For each $r \in \mathsf{C}$, denote by $\bar{f}^{p^k_i}_r(x)$ and $\bar{f}^{z^k_i}_r(x)$ the degree-t polynomial defined by $(\mathbf{C}^{p^k_i}_r, \mathbf{O}^{p^k_i}_r)$ and $(\mathbf{C}^{z^k_i}_r, \mathbf{O}^{z^k_i}_r)$, respectively.

- For every $k \in \{1, \ldots, m\}$ the simulator picks a random strong double t-sharing of 0, denoted $(\mathbf{C}^{0_i^k}, \mathbf{O}^{0_i^k})$ and uses it as the input to $\mathcal{F}_{\mathsf{vss}}^{0_i^k}$. That is, the simulator gives to the adversary $(\mathbf{C}^{0_i^k}, \mathbf{O}_k^{0_i^k})$ for every $k \in \mathsf{C}$ as the output of the sharing phase of $\mathcal{F}_{\mathsf{vss}}^{0_i^k}$. For each $r \in \mathsf{C}$, denote by $\bar{f}_r^{0_i^k}(x)$ the degree-t polynomial defined by $(\mathbf{C}_r^{0_i^k}, \mathbf{O}_r^{0_i^k})$.

$\mathcal{F}_{\mathsf{tss}}$ **simulation.** The simulator simulates the calls of P_i to $\mathcal{F}_{\mathsf{tss}}$ in the following way. For every $k \in \{1, \ldots, m\}$, the simulator picks three random strong double t-sharings $\langle\!\langle 0 \rangle\!\rangle$ denoted $(\mathbf{C}^{\gamma_i^k}, \mathbf{O}^{\gamma_i^k})$, $(\mathbf{C}^{\rho_i^k}, \mathbf{O}^{\rho_i^k})$ and $(\mathbf{C}^{\eta_i^k}, \mathbf{O}^{\eta_i^k})$, and uses it as the input to $\mathcal{F}_{\mathsf{tss}}^{k,i}$. That is, the simulator gives to the adversary $(\mathbf{C}^{\gamma_i^k}, \mathbf{O}_j^{\gamma_i^k})$, $(\mathbf{C}^{\rho_i^k}, \mathbf{O}_j^{\rho_i^k})$ and $(\mathbf{C}^{\eta_i^k}, \mathbf{O}_j^{\eta_i^k})$ and $\mathsf{happy}_j = 1$ for every $j \in \mathsf{C}$ as the output of $\mathcal{F}_{\mathsf{tss}}^{k,i}$. For each $r \in \mathsf{C}$, denote by $\bar{f}_r^{\gamma_i^k}(x)$, $\bar{f}_r^{\rho_i^k}(x)$ and $\bar{f}_r^{\eta_i^k}(x)$ the degree-t polynomial defined by $(\mathbf{C}_r^{\gamma_i^k}, \mathbf{O}_r^{\gamma_i^k})$, $(\mathbf{C}_r^{\rho_i^k}, \mathbf{O}_r^{\rho_i^k})$ and $(\mathbf{C}_r^{\eta_i^k}, \mathbf{O}_r^{\eta_i^k})$, respectively.

This completes the communication from honest parties to corrupt parties in the first round.

Adversary's communication. At this stage the simulator receives the inputs of the corrupt parties to the $\mathcal{F}_{\mathsf{vss}}$ and $\mathcal{F}_{\mathsf{tss}}$ calls. For each $i \in \mathsf{C}$ and $j \in \{L_{i-1}+1, \ldots, L_{i-1}+\ell_i\}$ the simulator receives $(\mathbf{C}^{w_j}, \mathbf{O}^{w_j})$ as the input to $\mathcal{F}_{\mathsf{vss}}^{w_j}$, and gives the adversary $(\mathbf{C}^{w_j}, \mathbf{O}_r^{w_j})$ for each $r \in \mathsf{C}$ as the output. Similarly, for each $i \in \mathsf{C}$ and $k \in \{1, \ldots, m\}$ the simulator receives $(\mathbf{C}^{p_i^k}, \mathbf{O}^{p_i^k})$ (resp.,$(\mathbf{C}^{z_i^k}, \mathbf{O}^{z_i^k})$), as the input to $\mathcal{F}_{\mathsf{vss}}^{p_i^k}$ (resp., $\mathcal{F}_{\mathsf{vss}}^{z_i^k}$) and gives the adversary $(\mathbf{C}^{p_i^k}, \mathbf{O}_j^{p_i^k})$ (resp., $(\mathbf{C}^{z_i^k}, \mathbf{O}_j^{z_i^k})$) for every $j \in \mathsf{C}$ as the output. In addition, the simulator receives $(\mathbf{C}^{0_i^k}, \mathbf{O}^{0_i^k})$, as the input to $\mathcal{F}_{\mathsf{vss}}^{0_i^k}$ and gives the adversary $(\mathbf{C}^{0_i^k}, \mathbf{O}_j^{0_i^k})$ for every $j \in \mathsf{C}$ as the output. Finally, for every $i \in \mathsf{C}$ and $k \in \{1, \ldots, m\}$ the simulator receives $(\mathbf{C}^{\gamma_i^k}, \mathbf{O}^{\gamma_i^k})$, $(\mathbf{C}^{\rho_i^k}, \mathbf{O}^{\rho_i^k})$, $(\mathbf{C}^{\eta_i^k}, \mathbf{O}^{\eta_i^k})$, and $\{\mathsf{happy}_{ij}^k\}_{j \in \{1,\ldots,n\}}$ as the input to $\mathcal{F}_{\mathsf{tss}}^{k,i}$, and gives the adversary $(\mathbf{C}^{\gamma_i^k}, \mathbf{O}_j^{\gamma_i^k})$, $(\mathbf{C}^{\rho_i^k}, \mathbf{O}_j^{\rho_i^k})$ $(\mathbf{C}^{\eta_i^k}, \mathbf{O}_j^{\eta_i^k})$, and happy_{ij}^k for each $j \in \mathsf{C}$ as the output. This completes the simulation of the first round.

B.7.2 Round 2

Broadcast simulation. For every $i, j \in \mathsf{H}$ the simulator sets $\mathsf{flag}_{ij} = 0$, and for every $i \in \mathsf{H}$ and $j \in \mathsf{C}$ the simulator computes flag_{ij} like in the protocol, using the values received from the adversary as inputs to $\mathcal{F}_{\mathsf{vss}}$ and $\mathcal{F}_{\mathsf{tss}}$. In addition, on behalf of each honest P_i, if a corrupt P_j sent an invalid pair to P_i in some $\mathcal{F}_{\mathsf{vss}}$ or $\mathcal{F}_{\mathsf{tss}}$ call, then we turn all openings sent from P_j to P_i, in all $\mathcal{F}_{\mathsf{vss}}$ and $\mathcal{F}_{\mathsf{tss}}$ calls, to $\perp$.

For each $i \in \mathsf{H}$, the simulator gives $\{\mathsf{flag}_{ij}\}_{j \in \{1,\ldots,n\}}$ as the broadcasts of the honest parties.

$\mathcal{F}_{\mathsf{vss}}$ **and** $\mathcal{F}_{\mathsf{tss}}$ **leakage.** For every $\mathcal{F}_{\mathsf{vss}}$ and $\mathcal{F}_{\mathsf{tss}}$ call with P_j as the dealer, the simulator gives the adversary $\{\mathsf{flag}_{ij}\}_{i \in \mathsf{H}}$ as the leakage.

$\mathcal{F}_{\mathsf{sif}}$ **simulation.** For any honest P_i and any value s that belongs to P_i, the output phase of $\mathcal{F}_{\mathsf{sif}}^s$ is simulated by giving the adversary the output $\mathbf{C}^s$, and $\mathsf{out} = 1$. Similarly, for any honest P_i and every $k \in \{1, \ldots, m\}$, the output phase of $\mathcal{F}_{\mathsf{sif}}^{0_i^k}$ is simulated by giving the adversary the output $\mathbf{C}^{0_i^k}$, and $\mathsf{out} = 1$.

$\mathcal{F}_{\mathsf{vao}}$ **simulation.** For any honest P_i, any value s that belongs to P_i, and any honest P_j the simulator gives the adversary the leakage $\mathbf{C}_j^s$, $\{o_{kj}^s\}_{j\in\mathsf{C}}$, $\mathsf{verify}_D = 0$, and the output "verification succeeded" of $\mathcal{F}_{\mathsf{vao}}^{s,j}$. For any corrupt P_i, any value s shared by P_i, and any honest P_j, the simulator computes the leakage and output of $\mathcal{F}_{\mathsf{vao}}^{s,j}$ using (1) $\mathbf{C}_j^s$ as the input of the honest parties, (2) $\mathbf{O}_j^s$ and $\mathsf{flag}_D = \mathsf{flag}_{ji}$ as the input of the dealer P_j, and (3) o_{kj}^s as the input of each honest P_k. Similarly, for every value s and any corrupt P_j, the simulator computes the leakage of $\mathcal{F}_{\mathsf{vao}}^{s,j}$ using (1) $\mathbf{C}_j^s$ as the input of the honest parties, and (2) o_{kj}^s as the input of each honest P_k.

$\mathcal{F}_{\mathsf{gdtc}}^{k,i}$ **simulation for** $i \in \mathsf{H}$. For every $k \in \{1,\ldots,m\}$ and $i \in \mathsf{H}$, the simulator simulates the leakage and output of $\mathcal{F}_{\mathsf{gdtc}}^{k,i}$ in the following way. Let $y^k = x^\alpha x^\beta + x^1 + \ldots + x^n$. The simulator sends to $\mathcal{A}$ the leakage (1) the commitments $\mathbf{C}_i^{x^\alpha}, \mathbf{C}_i^{x^\beta}, \mathbf{C}_i^{x^1}, \ldots, \mathbf{C}_i^{x^n}$, $\mathbf{C}_i^{z_1^k}, \ldots, \mathbf{C}_i^{z_n^k}$ and $\mathbf{C}_0^{p_i^k}$, (2) the indicator vectors $\{\delta^j\}_{j\in\mathsf{H}}$, where for every $j \in \mathsf{H}$, (a) $\delta^j_{p_i^k} = 1$, (b) for every $q \in \{\alpha,\beta,1,\ldots,n\}$, $\delta^j_{x^q} = \overline{\mathsf{flag}}_{j\mathsf{Dof}(x^q)}$,[13], and (c) for every $r \in \{1,\ldots,n\}$, $\delta^j_{z_r^k} = \overline{\mathsf{flag}}_{jr}$, (3) the vector $\mathbf{a} = (1,\ldots,1,\lambda_i^1,\ldots,\lambda_i^n,1)$, (4) the openings and values $\{\mathsf{open}_{\mathsf{crs}}(C_{ij}^{x^r}, o_{ij}^{x^r}), o_{ij}^{x^r}\}_{j\in\mathsf{C},r\in\{\alpha,\beta,1,\ldots,n\}}$, $\{\mathsf{open}_{\mathsf{crs}}(C_{ij}^{z_r^k}, o_{ij}^{z_r^k}), o_{ij}^{z_r^k}\}_{j\in\mathsf{C},r\in\{1,\ldots,n\}}$, and $\{\mathsf{open}_{\mathsf{crs}}(C_{0j}^{p_i^k}, o_{0j}^{p_i^k}), o_{0j}^{p_i^k}\}_{j\in\mathsf{C}}$, (5) the indicator vector $\delta^G := \delta^i$, and (6) the sets L_j for every $j \in \mathsf{H}$, where L_j contains the following elements: (a) $(\mathsf{open}_{\mathsf{crs}}(C_{ij}^{x^q}, o_{ij}^{x^q}), o_{ij}^{x^q})$ for every $q \in \{\alpha,\beta,1,\ldots,n\}$, such that $\mathsf{Dof}(x^q) \in \mathsf{C}$, $\delta^j_{x^q} = 0$ and $\delta^G_{x^q} = 1$, (b) $(\mathsf{open}_{\mathsf{crs}}(C_{ij}^{z_r^k}, o_{ij}^{z_r^k}), o_{ij}^{z_r^k})$ for every $r \in \mathsf{C}$, such that $\delta^j_{z_r^k} = 0$ and $\delta^G_{z_r^k} = 1$.

In addition, the simulator picks a degree-t polynomial $\bar{f}_{\mathsf{gdtc}}^{k,i}(x)$ in the following way. If $\delta^G_\alpha \wedge \delta^G_\beta = 1$ then $\bar{f}_{\mathsf{gdtc}}^{k,i}(x)$ is a random degree-t polynomial, and if $\delta^G_\alpha \wedge \delta^G_\beta = 0$ then $\bar{f}_{\mathsf{gdtc}}^{k,i}(x)$ is a random degree-t polynomial conditioned on

$$\bar{f}_{\mathsf{gdtc}}^{k,i}(j) = \sum_{r\in\{1,\ldots,n\}:\delta^G_{x^r}=1} \mathsf{open}_{\mathsf{crs}}(C_{ij}^{x^r}, o_{ij}^{x^r}) + \sum_{r\in\{1,\ldots,n\}:\delta^G_{z_r^k}=1} \lambda_i^r \cdot \mathsf{open}_{\mathsf{crs}}(C_{ij}^{z_r^k}, o_{ij}^{z_r^k}) + \mathsf{open}_{\mathsf{crs}}(C_{0j}^{p_i^k}, o_{0j}^{p_i^k}),$$

for every $j \in \mathsf{C}$. If $\delta^G_\alpha \wedge \delta^G_\beta = 0$ then the simulator gives $\{\bar{f}_{\mathsf{gdtc}}^{k,i}(j)\}_{j\in\{1,\ldots,n\}}$ to the adversary as additional leakage. Finally, the simulator gives the adversary $(\mathbf{a}, \delta^G, \bar{f}_{\mathsf{gdtc}}^{k,i}(0))$ as the output of $\mathcal{F}_{\mathsf{gdtc}}^{k,i}$.

$\mathcal{F}_{\mathsf{gdtc}}^{k,i}$ **leakage for** $i \in \mathsf{C}$. For every $k \in \{1,\ldots,m\}$ and $i \in \mathsf{C}$ the simulator simulates the leakage of $\mathcal{F}_{\mathsf{gdtc}}^{k,i}$ in the following way. The simulator sends to $\mathcal{A}$ the leakage (1) $\mathbf{C}_i^{x^\alpha}, \mathbf{C}_i^{x^\beta}, \mathbf{C}_i^{x^1}, \ldots, \mathbf{C}_i^{x^n}$, $\mathbf{C}_i^{z_1^k}, \ldots, \mathbf{C}_i^{z_n^k}$ and $\mathbf{C}_0^{p_i^k}$, (2) the indicator vectors $\{\delta^j\}_{j\in\mathsf{H}}$ where for every $j \in \mathsf{H}$: (a) $\delta^j_{p_i^k} = \overline{\mathsf{flag}}_{ji}$, (b) for every $q \in \{\alpha,\beta,1,\ldots,n\}$ $\delta^j_{x^q} = \overline{\mathsf{flag}}_{j\mathsf{Dof}(x^q)}$, and (c) for every $r \in \{1,\ldots,n\}$, $\delta^j_{z_r^k} = \overline{\mathsf{flag}}_{jr}$, (3) for every $j \in \mathsf{H}$, the values $\{\mathsf{open}_{\mathsf{crs}}(C_{ji}^{x^r}, o_{ji}^{x^r})\}_{r\in\{\alpha,\beta,1,\ldots,n\}}$, $\{\mathsf{open}_{\mathsf{crs}}(C_{ji}^{z_r^k}, o_{ji}^{z_r^k})\}_{r\in\{1,\ldots,n\}}$ and, $\mathsf{open}_{\mathsf{crs}}(C_{j0}^{p_i^k}, o_{j0}^{p_i^k})$.

[13]Here, $\overline{\mathsf{flag}}$ is $1 - \mathsf{flag}$.

$\mathcal{F}_{\mathsf{glinear}}^{k,i,u}$ **and** $\mathcal{F}_{\mathsf{glinear}}^{k,i,v}$ **simulation for** $i \in \mathsf{H}$. For every $k \in \{1,\ldots,m\}$ and $i \in \mathsf{H}$ the simulator simulates the leakage of $\mathcal{F}_{\mathsf{glinear}}^{k,i,u}$ by sending $\mathcal{A}$ the leakage (1) $\mathbf{C}_i^{x^\alpha}$ and $\mathbf{C}_0^{\gamma_i^k}$, (2) the indicator vectors $\{\delta^j\}_{j\in\mathsf{H}}$ where for every $j \in \mathsf{H}$, $\delta_1^j = \overline{\mathsf{flag}}_{j\mathsf{Dof}(x^\alpha)}$, and $\delta_2^j = 1$, (3) the vector $\mathbf{a} = (1,-1)$, the openings and values $(\mathsf{open}_{\mathsf{crs}}(C_{ij}^{x^\alpha}, o_{ij}^{x^\alpha}), o_{ij}^{x^\alpha})$ and $(\mathsf{open}_{\mathsf{crs}}(C_{0j}^{\gamma_i^k}, o_{0j}^{\gamma_i^k}), o_{0j}^{\gamma_i^k})$ for every $j \in \mathsf{C}$, and the vector $\delta^G := \delta^i$, (4) the sets L_j for every $j \in \mathsf{H}$, that contain $(\mathsf{open}_{\mathsf{crs}}(C_{ij}^{x^\alpha}, o_{ij}^{x^\alpha}), o_{ij}^{x^\alpha})$ if $\delta_1^G = 1$ and $\delta_1^j = 0$.

In addition, if $\delta_1^G = 0$ the simulator sets $\bar{f}_{\mathsf{glinear}}^{k,i,u}(x)$ to be a random degree-t polynomial conditioned on $\bar{f}_{\mathsf{glinear}}^{k,i,u}(j) = 0 - \mathsf{open}_{\mathsf{crs}}(C_{j0}^{\gamma_i^k}, o_{j0}^{\gamma_i^k})$ for all $j \in \mathsf{C}$. Otherwise, if $\delta_1^G = 1$, simulator picks a random degree-t polynomial $\bar{f}_{\mathsf{glinear}}^{k,i,u}(x)$ conditioned on

$$\bar{f}_{\mathsf{glinear}}^{k,i,u}(j) = \mathsf{open}_{\mathsf{crs}}(C_{ij}^{x^\alpha}, o_{ij}^{x^\alpha}) - \mathsf{open}_{\mathsf{crs}}(C_{j0}^{\gamma_i^k}, o_{j0}^{\gamma_i^k}),$$

for all $j \in \mathsf{C}$. The simulator returns $\{\bar{f}_{\mathsf{glinear}}^{k,i,u}(j)\}_{j\in\{1,\ldots,n\}}$ to the adversary as additional leakage. Finally, the simulator gives the adversary $(\mathbf{a}, \delta^G, \bar{f}_{\mathsf{glinear}}^{k,i,u}(0))$ as the output of $\mathcal{F}_{\mathsf{glinear}}^{k,i,u}$.

The simulation of $\mathcal{F}_{\mathsf{glinear}}^{k,i,v}$ follows the same line - simply replace α with β, γ with ρ, and u with v.

$\mathcal{F}_{\mathsf{glinear}}^{k,i,u}$ **and** $\mathcal{F}_{\mathsf{glinear}}^{k,i,v}$ **leakage for** $i \in \mathsf{C}$. For every $k \in \{1,\ldots,m\}$ and $i \in \mathsf{C}$, the simulator simulates the leakage of $\mathcal{F}_{\mathsf{glinear}}^{k,i,u}$ in the following way. The simulator sends to $\mathcal{A}$ the leakage (1) $\mathbf{C}_i^{x^\alpha}$ and $\mathbf{C}_0^{\gamma_i^k}$, (2) the indicator vectors $\{\delta^j\}_{j\in\mathsf{H}}$ where for every $j \in \mathsf{H}$, $\delta_1^j = \overline{\mathsf{flag}}_{j\mathsf{Dof}(x^\alpha)}$, and $\delta_2^j = \overline{\mathsf{flag}}_{ji}$, and (3) for every $j \in \mathsf{H}$ the values $\mathsf{open}_{\mathsf{crs}}(C_{ji}^{x^\alpha}, o_{ji}^{x^\alpha})$ and $\mathsf{open}_{\mathsf{crs}}(C_{j0}^{\gamma_i^k}, o_{j0}^{\gamma_i^k})$.

The simulation of $\mathcal{F}_{\mathsf{glinear}}^{k,i,v}$ follows the same line - simply replace α with β, γ with ρ, and u with v.

Adversary's communication. At this stage the simulator receives from $\mathcal{A}$ the messages from the corrupt parties to the honest parties, which include the broadcasts $\{\mathsf{flag}_{ij}\}_{j\in\{1,\ldots,n\}}$ for every corrupt P_i, and also the inputs of the corrupt parties to the functionalities. Upon receiving such inputs for $\mathcal{F}_{\mathsf{gdtc}}^{k,i}$ or $\mathcal{F}_{\mathsf{glinear}}^{k,i,u}$ or $\mathcal{F}_{\mathsf{glinear}}^{k,i,v}$, for $i \in \mathsf{C}$, the simulator holds all inputs to the functionality, and so can compute the output of the functionality and give it to $\mathcal{A}$. Similarly, upon receiving such inputs for $\mathcal{F}_{\mathsf{vss}}$ calls or $\mathcal{F}_{\mathsf{tss}}$ calls in which the dealer is corrupt, the simulator holds all inputs to the functionality, and so can compute the output of the functionality and give it to $\mathcal{A}$. Finally, for $\mathcal{F}_{\mathsf{vss}}$ calls or $\mathcal{F}_{\mathsf{tss}}$ calls in which the dealer is honest, only the corrupt parties may raise a flag, and since the dealer holds the rows that correspond to corrupt parties, the simulator can compute the output of the functionality. This completes the simulation .

B.7.3 Communication with $\mathcal{F}_{\mathsf{dtc}}$

The simulator computes the set V by following the protocol and using the view of the corrupt parties. For any corrupt P_i which is not in V, the simulator sets all of the inputs $w_{L_{i-1}+1}, \ldots w_{L_{i-1}+\ell_i}$ to be 0, and sends them to $\mathcal{F}_{\mathsf{dtc}}$. For every corrupt P_i which is in V, and every value s that belongs to P_i, the corresponding $\mathcal{F}_{\mathsf{vss}}$ or $\mathcal{F}_{\mathsf{tss}}$ call did not end with "D is corrupt", and the simulator computes the sharing polynomial of $[\![s]\!]$, denoted $F^s(x,y)$. For any $j \in \{L_{i-1}+1,\ldots,L_{i-1}+\ell_i\}$, the simulator sets $w^j := F^{w^j}(0,0)$, and sends w^j to $\mathcal{F}_{\mathsf{dtc}}$. Finally, the simulator receives the output of $\mathcal{F}_{\mathsf{dtc}}$, denoted $(y^1,\ldots,y^m)$.

B.7.4 Round 3

For every value s and any $i \in \{1,\ldots,n\}$, we define $\bar{\mathbf{O}}_i^s$ in the following way: (1) if s belongs to an honest party, and $i \in \mathsf{C}$, let $\bar{\mathbf{O}}_i^s := \mathbf{O}_i^s$, (2) if s belongs to a corrupt party in V, and $i \in \mathsf{H}$, let $\bar{\mathbf{O}}_i^s$ be the openings of honest party P_i according to $[\![s]\!]$ (Observe that $\bar{\mathbf{O}}_i^s$ was sent by the corrupt party to the corresponding $\mathcal{F}_{\mathsf{vss}}$ or $\mathcal{F}_{\mathsf{tss}}$ functionality, either in the sharing phase or in the verification phase, and so it is known to the simulator), (3) if s belongs to a corrupt party outside V, let $\bar{o}_{ij}^s = \bot$ for all $i,j \in \{0,\ldots,n\}$, and (4) if s belongs to an honest party, $i \in \mathsf{H}$ and $j \in \mathsf{C}$, then set $\bar{o}_{ij}^s := \bar{o}_{ji}^s$. (We don't care about the other cases.) In addition, for every value s that belongs to a corrupt party in V, let $\bar{f}_i^s(x) := F^s(x,i)$, for $i \in \{1,\ldots,n\}$.

For each $k \in \{1,\ldots,m\}$ the simulator simulates the computation of $y^k = x^\alpha x^\beta + x^1 + \ldots + x^n$ in the following way. We split into cases.

Exiting through linear functions. If P_α or P_β (or both) are not in V, then the simulator picks a symmetric bivariate polynomial $\bar{F}^k(x,y)$ of degree at most t in each variable, conditioned on $F^k(0,0) = y^k$ and

$$\bar{F}^k(x,i) = \sum_{j:\mathsf{Dof}(x^j)\in\mathsf{V}} \bar{f}_i^{x^j}(x) + \sum_{j\in\mathsf{V}} \bar{f}_i^{0_j^k}(x),$$

for every $i \in \mathsf{C}$.

For every $i \in \mathsf{V}\cap\mathsf{H}$ the simulator simulates the call to $(\mathcal{F}_{\mathsf{glinear}})_{\mathsf{lin}}^i$ in the following way. The simulator sends to $\mathcal{A}$ (1) the commitments $\{\mathbf{C}^{x^j}\}_{j\in\{1,\ldots,n\}}$ and $\{\mathbf{C}^{0_j^k}\}_{j\in\{1,\ldots,n\}}$, (2) the vectors $\{\delta^j\}_{j\in\mathsf{H}}$ where for every $j \in \mathsf{H}$ and $r \in \{1,\ldots,n\}$, $\delta_{x^r}^j = 1$ if $\mathsf{Dof}(x^r) \in \mathsf{V}$, (resp., $\delta_{0_r^k}^j = 1$ if $r \in \mathsf{V}$) and $\delta_{x^r}^j = 0$ (resp., $\delta_{0_r^k}^j = 0$) otherwise, (3) the vector $\mathbf{a} = (1,\ldots,1)$, (4) the openings and values $\{\bar{o}_{ir}^{x^j}, \mathsf{open}_{\mathsf{crs}}(C_{ir}^{x^j}, \bar{o}_{ir}^{x^j})\}_{j\in\{1,\ldots,n\},r\in\mathsf{C}}$, and $\{\bar{o}_{ir}^{0_j^k}, \mathsf{open}_{\mathsf{crs}}(C_{ir}^{0_j^k}, \bar{o}_{ir}^{0_j^k})\}_{j\in\{1,\ldots,n\},r\in\mathsf{C}}$ (5) the vector $\delta^G := \delta^i$ (6) empty sets $L_j = \varnothing$ for every $j \in \mathsf{H}$, and (7) the values $\bar{F}^k(j,i)$ for all $j \in \{1,\ldots,n\}$. The simulator also sends to $\mathcal{A}$ the values $(\mathbf{a}, \delta^G, \bar{F}^k(0,i))$ as the output of $(\mathcal{F}_{\mathsf{glinear}})_{\mathsf{lin}}^i$.

For every $i \in \mathsf{V}\setminus\mathsf{H}$ the simulator simulates the leakage of $(\mathcal{F}_{\mathsf{glinear}})_{\mathsf{lin}}^i$ in the following way. The simulator sends to $\mathcal{A}$ (1) the commitments $\{\mathbf{C}^{x^j}\}_{j\in\{1,\ldots,n\}}$ and $\{\mathbf{C}^{0_j^k}\}_{j\in\{1,\ldots,n\}}$, (2) the vectors $\{\delta^j\}_{j\in\mathsf{H}}$ where for every $j \in \mathsf{H}$ and $r \in \{1,\ldots,n\}$, $\delta_{x^r}^j = 1$ if $\mathsf{Dof}(x^r) \in \mathsf{V}$, (resp., $\delta_{0_r^k}^j = 1$ if $r \in \mathsf{V}$) and $\delta_{x^r}^j = 0$ (resp., $\delta_{0_r^k}^j = 0$) otherwise, (3) for each $j \in \mathsf{H}$, the values $\{\mathsf{open}_{\mathsf{crs}}(C_{ji}^{x^r}, \bar{o}_{ji}^{x^r})\}_{r\in\{1,\ldots,n\}}$ and $\{\mathsf{open}_{\mathsf{crs}}(C_{ji}^{0_r^k}, \bar{o}_{ji}^{0_r^k})\}_{r\in\{1,\ldots,n\}}$.

Later, the simulator receives from $\mathcal{A}$ the inputs of the corrupt parties to the functionalities $(\mathcal{F}_{\mathsf{glinear}})_{\mathsf{lin}}^i$ for $i \in \mathsf{V}\setminus\mathsf{H}$. Upon receiving such inputs, the simulator holds all inputs to the functionality, and so can compute the output of the functionality and give it to $\mathcal{A}$.

Correcting shares. If P_α and P_β are in V then the simulator picks a random degree-$2t$ polynomial $\bar{f}^k(x)$ conditioned on $\bar{f}^k(0) = y^k$, and

$$\bar{f}^k(i) = \bar{f}_i^{x^\alpha}(0)\bar{f}_i^{x^\beta}(0) + \sum_{j:\mathsf{Dof}(x^j)\in\mathsf{V}} \bar{f}_i^{x^j}(0) + \sum_{j\in\mathsf{V}} \lambda_i^j \bar{f}_i^{z_j^k}(0),$$

for every $i \in \mathsf{C}$. For every P_i the parties do as follows.

- If P_i is not in V, the simulator sends to $\mathcal{A}$ the following broadcast on behalf of any honest P_j. For every P_r in V, such that $\mathsf{flag}_{jr} = 0$, and every value s that belongs to P_r, P_j broadcasts $\bar{o}_{jr}^s$.
- If P_i is in V, denote the (simulated) output of $\mathcal{F}_{\mathsf{gdtc}}^{k,i}$ by $(\mathbf{a}, \delta^G, y_i^k)$. For every corrupt P_j not in V, and any value s that belongs to P_j such that the verification phase of $\mathcal{F}_{\mathsf{vao}}^{s,i}$ ended with "verification succeeded", the simulator simulates the leakage and output of the opening phase of $\mathcal{F}_{\mathsf{vao}}^{s,i}$ in the following way. If P_i is honest then the simulator uses (1) $\mathbf{C}_i^s$ as the input of the honest parties, (2) $\mathbf{O}_i^s$ and $\mathsf{flag}_D = \mathsf{flag}_{ij}$ as the input of the dealer P_i, and (3) o_{ki}^s as the input of each honest P_k to simulate the leakage and output. Otherwise, if P_i is corrupt, then the simulator holds the inputs of all parties to the verification phase of the functionality, and can compute the output of the opening phase as well.

  The next step of the simulation depends on whether P_i is honest or corrupt.

  Honest P_i and $\delta_\alpha^G \wedge \delta_\beta^G = 1$. The simulator samples a degree-t polynomial $g_i^k(x)$ conditioned on $g_i^k(j) = f_j^{p_i^k}(0)$ for all $j \in \mathsf{C}$, and

$$g_i^k(0) = y_i^k - f^k(i) + \sum_{j:\mathsf{Dof}(x^j)\in\mathsf{V}\wedge\delta_{x^j}^G=0} \bar{f}_i^{x^j}(0) + \sum_{j\in\mathsf{V}:\delta_{z_j^k}^G=0} \bar{f}_i^{z_j^k}(0)$$
$$- \sum_{j:\mathsf{Dof}(x^j)\notin\mathsf{V}\wedge\delta_{x^j}^G=1} \mathsf{open}_{\mathsf{crs}}(C_{i0}^{x^j}, o_{i0}^{x^j}) - \sum_{j\in\mathsf{V}:\delta_{z_j^k}^G=1} \mathsf{open}_{\mathsf{crs}}(C_{i0}^{z_j^k}, o_{i0}^{z_j^k}),$$

  where for each j such that $\mathsf{Dof}(x) \notin \mathsf{V}$ (resp., $j \notin \mathsf{V}$) the value x^j (resp., z_j^k) belongs to a corrupt party, so $C_{i0}^{x^j}$ and $o_{i0}^{x^j}$ (resp., $C_{i0}^{z_j^k}$ and $o_{i0}^{z_j^k}$) are fixed, and for every j such that $\mathsf{Dof}(x^j) \in \mathsf{V}$ and $\delta_{x^j}^G = 0$ (resp., $j \in \mathsf{V}$ and $\delta_{z_j^k}^G = 0$) the value x^j (resp., z_j^k) belongs to a corrupt party, so $\bar{f}_i^{x^j}(x)$ (resp., $\bar{f}_i^{z_j^k}(x)$) is fixed. For each honest P_j, it holds that $\mathsf{flag}_{ji} = 0$, and the simulator sends $(C_{j0}^{p_i^k}, g_i^k(j))$ as the output of $\mathcal{F}_{\mathsf{sif}}^{kji}$.

  Later, the simulator receives from $\mathcal{A}$ the inputs for $\mathcal{F}_{\mathsf{sif}}^{kji}$ for corrupt P_j's, computes the output of the functionality and gives it back to $\mathcal{A}$.

  Honest P_i and $\delta_\alpha^G \wedge \delta_\beta^G = 0$. The simulator computes u and v like in the protocol using the corrupt parties view. The simulator picks a random symmetric bivariate polynomial

$\bar{G}_i^k(x,y)$ of degree at most t in each variable, conditioned on $\bar{G}_i^k(x,j) = v\bar{f}_j^{\gamma_i^k}(x) + u\bar{f}_j^{\rho_i^k}(x) + \bar{f}_j^{\eta_i^k}(x) + uv - \bar{f}_j^{p_i^k}(x)$, for all $j \in \mathsf{C}$, and

$$\bar{G}_i^k(0,0) = \bar{f}^k(i) - y_i^k - \sum_{j:\mathsf{Dof}(x^j)\in\mathsf{V}\wedge\delta^G_{x^j}=0} \bar{f}_i^{x^j}(0) - \sum_{j\in\mathsf{V}:\delta^G_{z_j^k}=0} \bar{f}_i^{z_j^k}(0)$$
$$+ \sum_{j:\mathsf{Dof}(x^j)\notin\mathsf{V}\wedge\delta^G_{x^j}=1} \mathsf{open}_{\mathsf{crs}}(C_{i0}^{x^j}, o_{i0}^{x^j}) + \sum_{j\notin\mathsf{V}:\delta^G_{z_j^k}=1} \mathsf{open}_{\mathsf{crs}}(C_{i0}^{z_j^k}, o_{i0}^{z_j^k}),$$

where for each j such that $\mathsf{Dof}(x) \notin \mathsf{V}$ (resp., $j \notin \mathsf{V}$) the value x^j (resp., z_j^k) belongs to a corrupt party, so $C_{i0}^{x^j}$ and $o_{i0}^{x^j}$ (resp., $C_{i0}^{z_j^k}$ and $o_{i0}^{z_j^k}$) are fixed, and for every j such that $\mathsf{Dof}(x^j) \in \mathsf{V}$ and $\delta^G_{x^j} = 0$ (resp., $j \in \mathsf{V}$ and $\delta^G_{z_j^k} = 0$) the value x^j (resp., z_j^k) belongs to a corrupt party, so $\bar{f}_i^{x^j}(x)$ (resp., $\bar{f}_i^{z_j^k}(x)$) is fixed.
For every P_j in V, the simulator do as follows.

– For every honest P_j in V, the simulator simulates the leakage and output of $(\mathcal{F}_{\mathsf{glinear}})^{k,i,j}_{\mathsf{correct}}$ in the following way. The simulator simulates the leakage by sending to $\mathcal{A}$ (1) the j-th rows $\mathbf{C}_j^{\gamma_i^k}$, $\mathbf{C}_j^{\rho_i^k}$, $\mathbf{C}_j^{\eta_i^k}$, $\mathbf{C}^1$, and $\mathbf{C}_j^{p_i^k}$, where $\mathbf{C}^1$ are the commitments that correspond to default sharing $\langle 1 \rangle$, (2) the indicator vectors $\{\delta^r\}_{r\in\mathsf{H}}$ such that $\delta^r = (1,1,1,1,1)$ for all $r \in \mathsf{H}$, (3) the vector $\mathbf{a} = (1,1,1,1,-1)$, values and openings $\{\bar{f}_r^{\gamma_i^k}(j), \bar{o}_{jr}^{\gamma_i^k}\}_{r\in\mathsf{C}}$, $\{\bar{f}_r^{\rho_i^k}(j), \bar{o}_{jr}^{\rho_i^k}\}_{r\in\mathsf{C}}$, $\{\bar{f}_r^{\eta_i^k}(j), \bar{o}_{jr}^{\eta_i^k}\}_{r\in\mathsf{C}}$, $\{\bar{f}_r^{p_i^k}(j), \bar{o}_{jr}^{p_i^k}\}_{r\in\mathsf{C}}$ and $\{(1, o^1)\}_{r\in\mathsf{C}}$ where o^1 is the opening in the default sharing of $\langle 1 \rangle$, (4) $\delta^G = (1,1,1,1,1)$, (5) empty sets L_r for every $r \in \mathsf{H}$, and (6) the values $\bar{G}_i^k(r,j)$ for every $r \in \{1,\ldots,n\}$. Finally, the simulator gives $(\mathbf{a}, \delta^G, \bar{G}_i^k(0,j))$ to $\mathcal{A}$ as the output of the functionality.
– For every corrupt P_j in V, the simulator simulates the leakage of $(\mathcal{F}_{\mathsf{glinear}})^{k,i,j}_{\mathsf{correct}}$ in the following way. The adversary receives (1) the j-th rows $\mathbf{C}_j^{\gamma_i^k}$, $\mathbf{C}_j^{\rho_i^k}$, $\mathbf{C}_j^{\eta_i^k}$, $\mathbf{C}^1$, and $\mathbf{C}_j^{p_i^k}$, where $\mathbf{C}^1$ are the commitments that correspond to default sharing $\langle 1 \rangle$, (2) the indicator vectors $\{\delta^r\}_{r\in\mathsf{H}}$ such that $\delta^r = (1,1,1,1,1)$ for all $r \in \mathsf{H}$, and (3) the values $(\bar{f}_j^{\gamma_i^k}(r), \bar{f}_j^{\rho_i^k}(r), \bar{f}_j^{\eta_i^k}(r), 1, \bar{f}_j^{p_i^k}(r))$ for every $r \in \mathsf{H}$.
  Later, the simulator receives from $\mathcal{A}$ the inputs of the corrupt parties to the functionality. Upon receiving such inputs, the simulator holds all inputs to the functionality, and so can compute the output of the functionality and give it to $\mathcal{A}$.

Corrupt P_i and $\delta^G_\alpha \wedge \delta^G_\beta = 1$. For each honest P_j in V with $\mathsf{flag}_{ji} = 0$, the simulator sends $(C_{j0}^{p_i^k}, \mathsf{open}_{\mathsf{crs}}(C_{j0}^{p_i^k}, o_{j0}^{p_i^k}))$ as the output of $\mathcal{F}^{kji}_{\mathsf{sif}}$. Later, the simulator receives from $\mathcal{A}$ the inputs for $\mathcal{F}^{kji}_{\mathsf{sif}}$ for corrupt P_j's, computes the output of the functionality and gives it back to $\mathcal{A}$.

Corrupt P_i and $\delta^G_\alpha \wedge \delta^G_\beta = 0$. The simulator computes u and v like in the protocol using the corrupt parties view. The simulator sets

$$\bar{G}_i^k(x,y) := vF^{\gamma_i^k}(x,y) + uF^{\rho_i^k}(x,y) + F^{\eta_i^k}(x,y) + uv - F^{p_i^k}(x,y).$$

For every P_j in V, the simulator do as follows.

- For every honest P_j in V, the simulator simulates the leakage and output of $(\mathcal{F}_{\mathsf{glinear}})^{k,i,j}_{\mathsf{correct}}$ in the following way. The simulator simulates the leakage by sending to $\mathcal{A}$ (1) the j-th rows $\mathbf{C}^{\gamma_i^k}_j$, $\mathbf{C}^{\rho_i^k}_j$, $\mathbf{C}^{\eta_i^k}_j$, $\mathbf{C}^1$, and $\mathbf{C}^{p_i^k}_j$, where $\mathbf{C}^1$ are the commitments that correspond to default sharing $\langle 1 \rangle$, (2) the indicator vectors $\{\delta^r\}_{r\in\mathsf{H}}$ such that $\delta^r = (1,1,1,1,1)$ for all $r \in \mathsf{H}$, (3) the vector $\mathbf{a} = (1,1,1,1,-1)$, values and openings $\{F^{\gamma_i^k}(r,j), \bar{o}^{\gamma_i^k}_{jr}\}_{r\in\mathsf{C}}$, $\{F^{\rho_i^k}(r,j), \bar{o}^{\rho_i^k}_{jr}\}_{r\in\mathsf{C}}$, $\{F^{\eta_i^k}(r,j), \bar{o}^{\eta_i^k}_{jr}\}_{r\in\mathsf{C}}$, $\{F^{p_i^k}(r,j), \bar{o}^{p_i^k}_{jr}\}_{r\in\mathsf{C}}$ and $\{(1, o^1)\}_{r\in\mathsf{C}}$ where o^1 is the opening in the default sharing of $\langle 1 \rangle$, (4) $\delta^G = (1,1,1,1,1)$, (5) empty sets L_r for every $r \in \mathsf{H}$, and (6) the values $\bar{G}^k_i(r,j)$ for every $r \in \{1,\ldots,n\}$. Finally, the simulator gives $(\mathbf{a}, \delta^G, \bar{G}^k_i(0,j))$ to $\mathcal{A}$ as the output of the functionality.
- For every $j \in \mathsf{C}$ the simulator simulates the leakage of $(\mathcal{F}_{\mathsf{glinear}})^{k,i,j}_{\mathsf{correct}}$ in the following way. The adversary receives (1) the j-th rows $\mathbf{C}^{\gamma_i^k}_j$, $\mathbf{C}^{\rho_i^k}_j$, $\mathbf{C}^{\eta_i^k}_j$, $\mathbf{C}^1$, and $\mathbf{C}^{p_i^k}_j$, where $\mathbf{C}^1$ are the commitments that correspond to default sharing $\langle 1 \rangle$, (2) the indicator vectors $\{\delta^r\}_{r\in\mathsf{H}}$ such that $\delta^r = (1,1,1,1,1)$ for all $r \in \mathsf{H}$, and (3) the values $(F^{\gamma_i^k}(r,j), F^{\rho_i^k}(r,j), F^{\eta_i^k}(r,j), 1, F^{p_i^k}(r,j))$ for every $r \in \mathsf{H}$.
  Later, the simulator receives from $\mathcal{A}$ the inputs of the corrupt parties to the functionality. Upon receiving such inputs, the simulator holds all inputs to the functionality, and so can compute the output of the functionality and give it to $\mathcal{A}$.

This concludes the simulation.

B.7.5 Hybrid-Worlds

Fix a polynomial time environment $\mathcal{Z}$ and input ζ to the environment, and assume without loss of generality that $\mathcal{Z}$ is deterministic. Consider the following hybrid worlds, where we assume that the honest parties know the set H.

Hybrid 1. In Hybrid 1, the honest parties act like in the real-world, except that (1) in the first round, every honest P_i sends $F^s(x,j)$ to any honest P_j, for any value s that belongs to P_i, where $F^s(x,y)$ is the sharing polynomial of s, (2) at the end of the first round an honest P_i does not verify the validity of rows received from an honest P_j, but is always happy with P_j and sets $\mathsf{flag}_{ij} = 0$, (3) for every honest P_i and every s that belongs to P_i, in any call to $\mathcal{F}_{\mathsf{glinear}}$ or $\mathcal{F}_{\mathsf{gdtc}}$ that involves s, an honest P_j uses the values $F^s(x,j)$ received from P_i, instead of computing $\{\mathsf{open}_{\mathsf{crs}}(C^s_{jk}, o^s_{jk})\}_{k\in\{0,\ldots,n\}}$, (4) for any value s that belongs to an honest party, and any $i \in \mathsf{H}$, we change $\mathcal{F}^{s,i}_{\mathsf{vao}}$ so that (a) it only leaks $(C^s_{i1},\ldots,C^s_{in})$, the openings $\{o^s_{ij}\}_{j\in\mathsf{C}}$, and $\mathsf{verify}_D = 0$, and (b) the verification phase always ends with "verification succeeded", (the opening phase is never executed) (5) for any value s that belongs to an honest party, the functionality $\mathcal{F}^s_{\mathsf{sif}}$ that computes $\mathcal{F}^{\mathsf{sif}}_{\langle\langle\cdot\rangle\rangle}$ always returns $\mathbf{C}^s$ and $\mathsf{out} = 1$; In addition, for an honest P_i and $k \in \{1,\ldots,m\}$, the functionality $\mathcal{F}^{0_i^k}_{\mathsf{sif}}$ that computes $\mathcal{F}^{\mathsf{sif}}_{\langle\langle 0\rangle\rangle}$ returns $\mathbf{C}^{0_i^k}$ and $\mathsf{out} = 1$, (6) for each $k \in \{1,\ldots,m\}$ and $i,j \in \mathsf{H}$, the output phase of $\mathcal{F}^{kji}_{\mathsf{sif}}$ that computes $\mathcal{F}^{\mathsf{sif}}_{\mathsf{co}}$ always returns $(C^{p_i^k}_{j0}, F^{p_i^k}(0,j))$, were $F^{p_i^k}(x,j)$ is the polynomial that P_j received from P_i in the first round.

Hybrid 2. In Hybrid 2, the honest parties act like in the real-world, with the following changes.

- For every honest P_i and every input w_j that belongs to P_i, P_i sets $(\mathbf{C}^{w_j}, \mathbf{O}^{w_j})$ to be a random strong double t-sharing of 0 (instead of w_j), and we denote by $F^{w_j}(x,y)$ the corresponding sharing polynomial. Then, P_i samples a random polynomial $\bar{F}^{w_j}(x,y)$ conditioned on $\bar{F}^{w_j}(x,k) = F^{w_j}(x,k)$ for $k \in \mathsf{C}$, and $\bar{F}^{w_j}(0,0) = w_j$.

- For every honest P_i and any $k \in \{1, \dots, m\}$, P_i sets $(\mathbf{C}^{\gamma_i^k}, \mathbf{O}^{\gamma_i^k})$, $(\mathbf{C}^{\rho_i^k}, \mathbf{O}^{\rho_i^k})$ and $(\mathbf{C}^{\eta_i^k}, \mathbf{O}^{\eta_i^k})$ to be a random strong double t-sharings of 0, and we denote by $F^{\gamma_i^k}(x,y)$, $F^{\rho_i^k}(x,y)$ and $F^{\eta_i^k}(x,y)$ the corresponding sharing polynomials. Then, P_i samples random polynomials $\bar{F}^{\gamma_i^k}(x,y)$, $\bar{F}^{\rho_i^k}(x,y)$ and $\bar{F}^{\eta_i^k}(x,y)$ conditioned on $\bar{F}^{\gamma_i^k}(x,j) = F^{\gamma_i^k}(x,j)$, $\bar{F}^{\rho_i^k}(x,j) = F^{\rho_i^k}(x,j)$ and $\bar{F}^{\eta_i^k}(x,j) = F^{\eta_i^k}(x,j)$ for all $j \in \mathsf{C}$, and $\bar{F}^{\gamma_i^k}(0,0) \cdot \bar{F}^{\rho_i^k}(0,0) = \bar{F}^{\eta_i^k}(0,0)$.

- For every non-input and non-triple value s that belongs to an honest P_i, P_i samples $(\mathbf{C}^s, \mathbf{O}^s)$ like in the Hybrid 1, and we denote the sharing polynomial by $F^s(x,y)$. If s is p_i^k or z_i^k then P_i samples $\bar{F}^s(x,y)$ conditioned on $\bar{F}^s(x,k) = F^s(x,k)$ for $k \in \mathsf{C}$. Otherwise, if s is 0_i^k, then P_i samples samples $\bar{F}^s(x,y)$ conditioned on $\bar{F}^s(x,k) = F^s(x,k)$ for $k \in \mathsf{C}$ and $\bar{F}^s(0,0) = 0$.

- For every $i \in \mathsf{H}$, every value s that belongs to P_i, and every $j \in \mathsf{H}$, P_i sends $\bar{F}^s(x,j)$ to P_j (instead of $F^s(x,j)$ as in Hybrid 1).

- At the end of the first round an honest P_i does not verify the validity of rows received from an honest P_j, but is always happy with P_j and sets $\mathsf{flag}_{ij} = 0$.

- For every honest P_i and every s that belongs to P_i, in any call to $\mathcal{F}_{\mathsf{glinear}}$ or $\mathcal{F}_{\mathsf{gdtc}}$ that involves s an honest P_j uses the values $\bar{F}^s(x,j)$ received from P_i, instead of computing $\{\mathsf{open}_{\mathsf{crs}}(C^s_{jk}, o^s_{jk})\}_{k \in \{0,\dots,n\}}$.

  Since the values $\bar{F}^s(x,j)$ are not necessarily consistent with $\{\mathsf{open}_{\mathsf{crs}}(C^s_{jk}, o^s_{jk})\}_{k \in \{0,\dots,n\}}$, we change $\mathcal{F}_{\mathsf{glinear}}$ and $\mathcal{F}_{\mathsf{gdtc}}$ for an honest guide, so that it would have the exact same leakage and output, even if some of the guide's values $(b^G_{i,0}, \dots, b^G_{i,n})$ are not consistent with the guide's openings $(o^G_{i,0}, \dots, o^G_{i,n})$ (but they are consistent with some degree-t polynomial).

- For any value s that belongs to an honest party, and any $i \in \mathsf{H}$, We change $\mathcal{F}^{s,i}_{\mathsf{vao}}$ so that (a) it only leaks $(C^s_{i1}, \dots, C^s_{in})$, the openings $\{o^s_{ij}\}_{j \in \mathsf{C}}$, and verify_D, and (b) the verification phase always ends with "verification succeeded" (the opening phase is never executed).

- For any value s that belongs to an honest party, the functionality $\mathcal{F}^s_{\mathsf{sif}}$ that computes $\mathcal{F}^{\mathsf{sif}}_{\langle\langle \cdot \rangle\rangle}$ always returns $\mathbf{C}^s$ and $\mathsf{out} = 1$. In addition, for an honest P_i and $k \in \{1, \dots, m\}$, the functionality $\mathcal{F}^{0_i^k}_{\mathsf{sif}}$ that computes $\mathcal{F}^{\mathsf{sif}}_{\langle\langle 0 \rangle\rangle}$ returns $\mathbf{C}^{0_i^k}$ and $\mathsf{out} = 1$.

- For each $k \in \{1, \dots, m\}$ and $i, j \in \mathsf{H}$, the output phase of $\mathcal{F}^{kji}_{\mathsf{sif}}$ that computes $\mathcal{F}^{\mathsf{sif}}_{\mathsf{co}}$ always returns $(C^{p_i^k}_{j0}, F^{p_i^k}(0,j))$, were $F^{p_i^k}(x,j)$ is the polynomial that P_j received from P_i in the first round.

We continue by showing that the view in the real-world has the same distribution as the view in Hybrid 1, that the view in Hybrid 1 is close to the view in Hybrid 2, and that the view in Hybrid 2 is close to the view in the ideal-world.

B.7.6 Real-World vs. Hybrid 1

We claim that the real-world view has the same distribution as the view in Hybrid 1. This follows by noting that in the real-world (1) for any $i, j \in \mathsf{H}$ it holds that $\mathsf{flag}_{ij} = 0$, (2) the inputs to $\mathcal{F}_{\mathsf{glinear}}$ and $\mathcal{F}_{\mathsf{gdtc}}$ are exactly as specified in Hybrid 1, (3) for any s that belongs to an honest party, and any $i \in \mathsf{H}$, the leakage and output of the verification phase of $\mathcal{F}_{\mathsf{vao}}^{s,i}$ are exactly as specified in Hybrid 1, and the opening phase is never executed since all honest parties are in V, (4) the output of $\mathcal{F}_{\mathsf{sif}}^{s}$, for any s that belongs to an honest party, is as specified in Hybrid 1, (5) similarly, the output of $\mathcal{F}_{\mathsf{sif}}^{0_i^k}$, for any $k \in \{1, \ldots, m\}$ and $i \in \mathsf{H}$, is as specified in Hybrid 1, (6) for each $k \in \{1, \ldots, m\}$ and $i, j \in \mathsf{H}$, the output phase of $\mathcal{F}_{\mathsf{sif}}^{kji}$ returns the same as in Hybrid 1.

B.7.7 Hybrid 1 vs. Hybrid 2

We claim that Hybrid 1 is $O(m \cdot L_{n+1} \cdot n^2\epsilon)$-close to Hybrid 2. Consider the Hybrid 1 random variables

$$(\mathsf{crs}, \{w^{L_{i-1}+1}, \ldots, w^{L_{i-1}+\ell_i}\}_{i \in \mathsf{H}}, ((\mathbf{C}^s, \{\mathbf{O}_i^s\}_{i \in \mathsf{C}}), \{F^s(x, y)\})_{s:\mathsf{Dof}(s) \in \mathsf{H}})$$

where $\{w^{L_{i-1}+1}, \ldots, w^{L_{i-1}+\ell_i}\}_{i \in \mathsf{H}}$ are the inputs of the honest parties, and s ranges over all values that belong to honest parties, and the Hybrid 2 random variables

$$(\mathsf{crs}, \{w^{L_{i-1}+1}, \ldots, w^{L_{i-1}+\ell_i}\}_{i \in \mathsf{H}}, ((\mathbf{C}^s, \{\mathbf{O}_i^s\}_{i \in \mathsf{C}}), \{\bar{F}^s(x, y)\})_{s:\mathsf{Dof}(s) \in \mathsf{H}}),$$

observe that they are $O(k \cdot L_{n+1} \cdot n^2\epsilon)$-close, and that in both worlds the view can be obtained from them by the same random process, that continues the execution as in Hybrid 2 using the above values, and setting $o_{ij}^s = \perp$ for any s such that $\mathsf{Dof}(s) \in \mathsf{H}$ and $i, j \in \mathsf{H}$ (note that such openings are never used in Hybrid 1 and in Hybrid 2).

B.7.8 Hybrid 2 vs. Ideal-World

We claim that the view in Hybrid 2 is $O(\epsilon)$-statistically-close to the view in the ideal-world, even when the commitment scheme is only computationally-hiding.

Round 1. Consider the random variables

$$(\mathsf{crs}, \{w^{L_{i-1}+1}, \ldots, w^{L_{i-1}+\ell_i}\}_{i \in \mathsf{H}}, (\mathbf{C}^s, \{\mathbf{O}_i^s\}_{i \in \mathsf{C}}), \{\bar{F}^s(x, i)\}_{i \in \mathsf{C}})_{s:\mathsf{Dof}(s) \in \mathsf{H}}$$

and observe that they have the same distribution in both worlds. Fix those random variables, and observe that this fixes the messages from the honest parties to the corrupt parties in both worlds. At this stage the corrupt parties send messages to the honest parties, and so they have the same distribution as well. This concludes the analysis of the first round's view.

Round 2. We say that the first-round view is "good" if for any commitment C that appears in the view, and for any two openings o, and o' that appear in the view, it holds that either $\mathsf{open}_{\mathsf{crs}}(C, o) = \perp$ or $\mathsf{open}_{\mathsf{crs}}(C, o') = \perp$ or $\mathsf{open}_{\mathsf{crs}}(C, o) = \mathsf{open}_{\mathsf{crs}}(C, o)$. Observe that a first-round view is good with probability at least $1 - \epsilon$, and fix any such view.

In the Hybrid 2, since an honest party P_i always receives a valid sharing from any other honest P_j, we conclude that $\mathsf{flag}_{ij} = 0$ for all $i, j \in \mathsf{H}$, just like in the ideal-world. In addition, for $i \in$

H and $j \in \mathsf{C}$, the value flag_{ij} is computed in the same way in both worlds. We conclude that the broadcasts $\{\mathsf{flag}_{ij}\}_{j\in\{1,\dots,n\}}$ of the honest parties, as well as the leakage of the $\mathcal{F}_{\mathsf{vss}}$ and $\mathcal{F}_{\mathsf{tss}}$ functionalities, are the same in both worlds.

One can verify that (1) the leakage and outputs of the $\mathcal{F}_{\mathsf{sif}}$ and $\mathcal{F}_{\mathsf{vao}}$ calls in which the dealer is honest are the same in both worlds, (2) the leakage of an $\mathcal{F}_{\mathsf{vao}}$ call in which the dealer is corrupt is the same in both worlds, and (3) the leakage of $\mathcal{F}^{k,i}_{\mathsf{gdtc}}$, $\mathcal{F}^{k,i,u}_{\mathsf{glinear}}$ and $\mathcal{F}^{k,i,v}_{\mathsf{glinear}}$, for $i \in \mathsf{C}$, is the same in both worlds. We continue with the analysis of $\mathcal{F}^{k,i}_{\mathsf{gdtc}}$, $\mathcal{F}^{k,i,u}_{\mathsf{glinear}}$ and $\mathcal{F}^{k,i,v}_{\mathsf{glinear}}$ for $i \in \mathsf{H}$.

$\mathcal{F}^{k,i}_{\mathsf{gdtc}}$ for $i \in \mathsf{H}$. We continue with the analysis of an $\mathcal{F}^{k,i}_{\mathsf{gdtc}}$ call, for $k \in \{1,\dots,m\}$ and $i \in \mathsf{H}$. Observe that in the ideal-world, parts (1)–(5) of the leakage sent by the simulator to the adversary, is fixed and and equal to the leakage in the hybrid-world. Regarding part (6) of the leakage, the sets L_j for $j \in \mathsf{H}$, observe that since the view is good then in the hybrid-world L_j contains only elements that correspond to x^q (resp., z^k_r) such that $\delta^j_{x^q} = 0$ and $\delta^G_{x^q} = 1$, (resp., $\delta^j_{z^k_r} = 0$ and $\delta^G_{z^k_r} = 1$). Since this may occur only when x^q and z^k_r belong to a corrupt party, we conclude that those sets are fixed and equal in both worlds.

In the hybrid-world, observe that whenever $\delta^G_\alpha \wedge \delta^G_\beta = 1$ then, even conditioned on the adversary's view, the value $\bar{F}^{p^k_i}(0,0)$ is uniformly distributed (by Fact A.5), and so the output of $\mathcal{F}^{k,i}_{\mathsf{gdtc}}$ has the same distribution in both worlds.

Otherwise, $\delta^G_\alpha \wedge \delta^G_\beta = 0$, and in the hybrid-world, for each x^r (resp., z^k_r) such that $\delta^G_{x^r} = 1$ (resp., $\delta^G_{z^k_r} = 1$), we let $f^{x^r}_i(x)$ (resp., $f^{z^k_r}_i(x)$) be the corresponding degree-t polynomial that P_i holds. Let $f^{p^k_i}(x) := \bar{F}^{p^k_i}(x,0)$, and set

$$f^{k,i}_{\mathsf{gdtc}}(x) := \sum_{r\in\{1,\dots,n\}:\delta^G_{x^r}=1} f^{x^r}_i(x) + \sum_{r\in\{1,\dots,n\}:\delta^G_{z^k_r}=1} \lambda^r_i f^{z^k_r}_i(x) + f^{p^k_i}(x).$$

Observe that, in the hybrid-world, the functionality additionally leaks $\{f^{k,i}_{\mathsf{gdtc}}(j)\}_{j\in\{1,\dots,n\}}$ and the output is $(\mathbf{a}, \delta^G, f^{k,i}_{\mathsf{gdtc}}(0))$. Since $f^{p^k_i}(x)$ is uniformly distributed conditioned on the values $\{f^{p^k_i}(j)\}_{j\in\mathsf{C}}$ which are fixed by the view, we conclude that $f^{k,i}_{\mathsf{gdtc}}(x)$ and $\bar{f}^{k,i}_{\mathsf{gdtc}}(x)$ have the same distribution in both worlds. This concludes the analysis of $\mathcal{F}^{k,i}_{\mathsf{gdtc}}$ for $i \in \mathsf{H}$.

$\mathcal{F}^{k,i,u}_{\mathsf{glinear}}$ and $\mathcal{F}^{k,i,v}_{\mathsf{glinear}}$ for $i \in \mathsf{H}$. We continue with the analysis of an $\mathcal{F}^{k,i,u}_{\mathsf{glinear}}$ call, for $k \in \{1,\dots,m\}$ and $i \in \mathsf{H}$. The analysis of $\mathcal{F}^{k,i,v}_{\mathsf{glinear}}$ is similar, and so it is omitted. Similarly to the $\mathcal{F}_{\mathsf{gdtc}}$ functionality, it is not hard to see that parts (1)–(4) of the leakage are fixed, and the same in both worlds.

In the hybrid-world, if $\delta^G_1 = 0$, then the values $\{0 - \bar{F}^{\gamma^k_i}(0,j)\}_{j\in\{1,\dots,n\}}$ are also leaked to the adversary, and the output is $(\mathbf{a}, \delta^G, 0 - \bar{F}^{\gamma^k_i}(0,j))$. Since $0 - \bar{F}^{\gamma^k_i}(0,x)$ is uniformly distributed conditioned on the values $\{0 - \bar{F}^{\gamma^k_i}(0,j)\}_{j\in\mathsf{C}}$ that are known to the adversary, it is not hard to see that those values have the same distribution in both worlds, as required.

Otherwise, if $\delta^G = 1$, in the hybrid-world let $f^{x^\alpha}_i(x)$ be the degree-t polynomial corresponding to $(\mathbf{C}^{x^\alpha}_i, \mathbf{O}^{x^\alpha}_i)$, and let $f^{\gamma^k_i}(x) := \bar{F}^{\gamma^k_i}(x,0)$. Let

$$f^{k,i,u}_{\mathsf{glinear}}(x) = f^{x^\alpha}_i(x) - f^{\gamma^k_i}(x),$$

and observe that the leakage of the functionality is $\{f^{k,i,u}_{\mathsf{glinear}}(j)\}_{j\in\{1,\dots,n\}}$ and that the output is $(\mathbf{a}, \delta^G, f^{k,i,u}_{\mathsf{glinear}}(0))$. Since $f^{\gamma^k_i}(x)$ is uniformly distributed conditioned on the values $\{f^{\gamma^k_i}(j)\}_{j\in\mathsf{C}}$ which are fixed by the view, we conclude that $f^{k,i,u}_{\mathsf{glinear}}(x)$ and $\bar{f}^{k,i,u}_{\mathsf{glinear}}(x)$ have the same distribution. This concludes the analysis of $\mathcal{F}^{k,i,u}_{\mathsf{glinear}}$ for $i \in \mathsf{H}$.

Adversary's communication. Conditioned on the above values, the adversary's inputs to the functionalities are picked in the same way, and the outputs are computed exactly like in the hybrid-world. We conclude that conditioned on any good first-round view, the second-round view has the same distribution in both worlds.

Round 3. We say that the second-round view is "good" if for any commitment C that appears in the view, and for any two openings o, and o' that appear in the view, it holds that either $\mathsf{open}_{\mathsf{crs}}(C,o) = \perp$ or $\mathsf{open}_{\mathsf{crs}}(C,o') = \perp$ or $\mathsf{open}_{\mathsf{crs}}(C,o) = \mathsf{open}_{\mathsf{crs}}(C,o)$. Observe that a second-round view is good with probability at least $1-\epsilon$, and fix any such view.

In the hybrid-world, conditioned on a good view by the correctness of the $\mathcal{F}_{\mathsf{sif}}$ functionality, each value s that belongs to a corrupt party in V is strongly shared, and denote the sharing polynomial by $F^s(x,y)$. We also denote by $\bar{o}^s_{i0}, \dots, \bar{o}^s_{in}$ the openings that a party P_i holds at this stage according to the protocol.

We continue with the analysis of $y^k = x^\alpha x^\beta + x^1 + \ldots + x^n$, for $k \in \{1,\dots,m\}$.

Exiting through linear functions. Assume that P_α or P_β (or both) are not in V. In the hybrid-world, let

$$F^k(x,y) = \sum_{j:\mathsf{Dof}(x^j)\in\mathsf{V}} \bar{F}^{x^j}(x,y) + \sum_{j\in\mathsf{V}} \bar{F}^{0^k_j}(x,y).$$

By the correctness of $\mathcal{F}_{\mathsf{sif}}$, and since View is good, it follows that $\bar{F}^{0^k_j}(0,0) = 0$ for every $j \in \mathsf{V}$. Since V contains an honest party, there exists $j \in \mathsf{V}$ such that $\bar{F}^{0^k_j}(x,y)$ is uniformly distributed conditioned on $\{f^{0^k_j}_i(x)\}_{i\in\mathsf{C}}$ and $\bar{F}^{0^k_j}(0,0) = 0$. Therefore, the distribution of $\bar{F}^k(x,y)$ is the same as the distribution of the ideal-world random variable $\bar{F}^k(x,y)$, and it is independent of the reconstruction of $y^{k'}$ for $k' \neq k$.

Conditioned on those values, it is not hard to see that the leakage and output of $(\mathcal{F}_{\mathsf{glinear}})^i_{\mathsf{lin}}$ for $i \in \mathsf{H}$ is fixed, and has the same distribution in both worlds. Similarly, the leakage of $(\mathcal{F}_{\mathsf{glinear}})^i_{\mathsf{lin}}$ for $i \in \mathsf{C}$ is fixed, and has the same distribution in both worlds. This concludes the case of exiting through linear functions.

Correcting shares. Assume that P_α and P_β are in V. In the hybrid-world, let $f^k(x)$ be the degree-$2t$ polynomial, obtained by interpolating

$$\{F^{x^\alpha}(0,i)F^{x^\beta}(0,i) + \sum_{j:\mathsf{Dof}(x^j)\in\mathsf{V}} F^{x^j}(0,i) + \sum_{j\in\mathsf{V}} \lambda^j_i F^{z^k_j}(0,i)\}_{i\in\{1,\dots,n\}}.$$

Since for every x^j that belongs to an honest party the value $F^{x^j}(0,0)$ is equal to x^j, and by definition of the simulator for every x^j that belongs to a corrupt party in V the value x^j is equal to $F^{x^j}(0,0)$, we conclude that the value $F^{x^\alpha}(0,0)F^{x^\beta}(0,0) + \sum_{j:\in\mathsf{Dof}(x^j)\mathsf{V}} F^{x^j}(0,0)$ is equal to the

y^k that the simulator obtains from the ideal functionality $\mathcal{F}_{\mathsf{dtc}}$. By Fact A.10 we conclude that the polynomial $f^k(x)$ has the same distribution as $\bar{f}^k(x)$. Fix those polynomials.

We continue by analysing the reconstruction of the i-th share of y^k. We split into cases.

- Assume that P_i is not in V, so P_i is corrupt. In both worlds, every honest party P_j in V broadcasts $\bar{o}^s_{ji}$ for every value s that belongs to a party P_r in V such that $\mathsf{flag}_{jr} = 0$.
- Assume that P_i is in V, and denote the output of $\mathcal{F}^{k,i}_{\mathsf{gdtc}}$ by $(\mathbf{a}, \delta^G, y^k_i)$. Observe that the opening phase of $\mathcal{F}^{s,i}_{\mathsf{vao}}$ has the same distribution in both worlds, for any s that belongs to a party not in V (recall that every party not in V is corrupt). We continue by splitting into cases.

Honest P_i and $\delta^G_\alpha \wedge \delta^G_\beta = 1$. In this case, in the hybrid-world, observe that $F^{p^k_i}(x,y)$ is uniformly distributed, conditioned on $F^{p^k_i}(x,j) = f^{p^k_i}_j(x)$ for all $j \in \mathsf{C}$, and

$$\begin{aligned}
F^{p^k_i}(0,0) &= \mathsf{open}_{\mathsf{crs}}(C^{p^k_i}_{00}, o^{p^k_i}_{00}) \\
&= y^k_i - \mathsf{open}_{\mathsf{crs}}(C^{x^\alpha}_{i0}, o^{x^\alpha}_{i0})\mathsf{open}_{\mathsf{crs}}(C^{x^\beta}_{i0}, o^{x^\beta}_{i0}) \\
&\quad - \sum_{j\in\{1,\dots,n\}:\delta^G_{x^j}=1} \mathsf{open}_{\mathsf{crs}}(C^{x^j}_{i0}, o^{x^j}_{i0}) - \sum_{j\in\{1,\dots,n\}:\delta^G_{z^k_j}=1} \lambda^j_i \mathsf{open}_{\mathsf{crs}}(C^{z^k_j}_{i0}, o^{z^k_j}_{i0}) \\
&= y^k_i - F^{x^\alpha}(0,i)F^{x^\beta}(0,i) - \sum_{j:\delta^G_{w^j}=1\wedge \mathsf{Dof}(x^j)\in\mathsf{V}} F^{x^j}(0,i) - \sum_{j:\delta^G_{z^k_j}=1\wedge j\in\mathsf{V}} F^{z^k_j}(0,i) \\
&\quad - \sum_{j:\delta^G_{x^j}=1\wedge \mathsf{Dof}(j)\notin\mathsf{V}} \mathsf{open}_{\mathsf{crs}}(C^{x^j}_{i0}, o^{x^j}_{i0}) - \sum_{j:\delta^G_{z^k_j}=1\wedge j\notin\mathsf{V}} \mathsf{open}_{\mathsf{crs}}(C^{z^k_j}_{i0}, o^{z^k_j}_{i0}) \\
&= y^k_i - f^k(i) + \sum_{j:\mathsf{Dof}(x^j)\in\mathsf{V}\wedge\delta^G_{x^j}=0} F^{x^j}(0,i) + \sum_{j:\mathsf{Dof}(z^k_j)\in\mathsf{V}\wedge\delta^G_{z^k_j}=0} F^{z^k_j}(0,i) \\
&\quad - \sum_{j:\delta^G_{w^j}=1\wedge \mathsf{Dof}(x^j)\notin\mathsf{V}} \mathsf{open}_{\mathsf{crs}}(C^{x^j}_{i0}, o^{x^j}_{i0}) - \sum_{j:\delta^G_{z^k_j}=1\wedge j\notin\mathsf{V}} \mathsf{open}_{\mathsf{crs}}(C^{z^k_j}_{i0}, o^{z^k_j}_{i0}),
\end{aligned}$$

where we used the fact that View is good, so (1) $F^{p^k_i}(0,0) = \mathsf{open}_{\mathsf{crs}}(C^{p^k_i}_{00}, o^{p^k_i}_{00})$, (2) for every $j \in \{\alpha, \beta, 1, \dots, n\}$ such that $\mathsf{Dof}(x^j) \in \mathsf{V}$ and $\delta^G_{x^j} = 1$, it holds that $\mathsf{open}_{\mathsf{crs}}(C^{x^j}_{i0}, o^{x^j}_{i0}) = F^{x^j}(0,i)$, and (3) for every $j \in \mathsf{V}$ such that $\delta^G_{z^k_j} = 1$, it holds that $\mathsf{open}_{\mathsf{crs}}(C^{z^k_j}_{i0}, o^{z^k_j}_{i0}) = F^{z^k_j}(0,i)$.

Observe that for each j such that $\mathsf{Dof}(x^j) \notin \mathsf{V}$ (resp., $j \notin \mathsf{V}$) the value x^j (resp., z^k_j) belongs to a corrupt party, so $C^{x^j}_{i0}$ and $o^{x^j}_{i0}$ (resp., $C^{z^k_j}_{i0}$ and $o^{z^k_j}_{i0}$) are fixed, and for every j such that $\mathsf{Dof}(x^j) \in \mathsf{V}$ and $\delta^G_{x^j} = 0$ (resp., $j \in \mathsf{V}$ and $\delta^G_{z^k_j} = 0$) the value x^j (resp., z^k_j) belongs to a corrupt party, so $F^{x^j}(x,y)$ (resp., $F^{z^k_j}(x,y)$) is fixed.

Observe that the polynomial $f^{p^k_i}(x) := F^{p^k_i}(x,0)$ has the same distribution as the ideal-world polynomial $g^k_i(x)$. Fix those polynomials, and note that for each honest P_j it holds that $\mathsf{flag}_{ji} = 0$, and the output of $\mathcal{F}_{\mathsf{sif}}^{ji}$ in the hybrid-world is $(C_{j0}^{p_i^k}, f^{p_i^k}(j))$, just like in the ideal-world.

Honest P_i and $\delta_\alpha^G \wedge \delta_\beta^G = 0$. In the hybrid-world, a similar analysis to the one above shows that

$$\begin{aligned}
F^{p_i^k}(0,0) &= \mathsf{open}_{\mathsf{crs}}(C_{00}^{p_i^k}, o_{00}^{p_i^k}) \\
&= y_i^k - \sum_{j\in\{1,\ldots,n\}:\delta_{x^j}^G=1} \mathsf{open}_{\mathsf{crs}}(C_{i0}^{x^j}, o_{i0}^{x^j}) - \sum_{j\in\{1,\ldots,n\}:\delta_{z_j^k}^G=1} \lambda_i^j \mathsf{open}_{\mathsf{crs}}(C_{i0}^{z_j^k}, o_{i0}^{z_j^k}) \\
&= y_i^k - \sum_{j:\delta_{x^j}^G=1\wedge \mathsf{Dof}(x^j)\in\mathsf{V}} F^{x^j}(0,i) - \sum_{j:\delta_{z_j^k}^G=1\wedge j\in\mathsf{V}} F^{z_j^k}(0,i) \\
&\quad - \sum_{j:\delta_{x^j}^G=1\wedge \mathsf{Dof}(x^j)\notin\mathsf{V}^k} \mathsf{open}_{\mathsf{crs}}(C_{i0}^{x^j}, o_{i0}^{x^j}) - \sum_{j:\delta_{z_j^k}^G=1\wedge j\notin\mathsf{V}} \mathsf{open}_{\mathsf{crs}}(C_{i0}^{z_j^k}, o_{i0}^{z_j^k}) \\
&= y_i^k - f^k(i) + F^{x^\alpha}(0,i)F^{x^\beta}(0,i) + \sum_{j:\mathsf{Dof}(x^j)\in\mathsf{V}\wedge\delta_{x^j}^G=0} F^{x^j}(0,i) + \sum_{j:j\in\mathsf{V}\wedge\delta_{z_j^k}^G=0} F^{z_j^k}(0,i) \\
&\quad - \sum_{j:\delta_{x^j}^G=1\wedge \mathsf{Dof}(x^j)\notin\mathsf{V}} \mathsf{open}_{\mathsf{crs}}(C_{i0}^{x^j}, o_{i0}^{x^j}) - \sum_{j\in\{1,\ldots,n\}:\delta_{z_j^k}^G=1\wedge j\notin\mathsf{V}} \mathsf{open}_{\mathsf{crs}}(C_{i0}^{z_j^k}, o_{i0}^{z_j^k}),
\end{aligned}$$

where, again, we used the fact that View is good. Let u and v be defined as in the protocol, and consider the polynomial $G_i^k(x,y) := vF^{\gamma_i^k}(x,y) + uF^{\rho_i^k}(x,y) + F^{\eta_i^k}(x,y) + uv - F^{p_i^k}(x,y)$. By Fact A.9 we conclude that $G_i^k(x,y)$ is uniformly distributed, conditioned on $G_i^k(x,j) = vf_j^{\gamma_i^k}(x) + uf_j^{\rho_i^k}(x) + f_j^{\eta_i^k}(x) + uv - f_j^{p_i^k}(x)$ and

$$\begin{aligned}
G_i^k(0,0) = f^k(i) - y_i^k &- \sum_{j:\mathsf{Dof}(x^j)\in\mathsf{V}\wedge\delta_{x^j}^G=0} F^{x^j}(0,i) - \sum_{j\in\mathsf{V}:\delta_{z_j^k}^G=0} F^{z_j^k}(0,i) \\
&+ \sum_{j:\delta_{w^j}^G=1\wedge\mathsf{Dof}(x^j)\notin\mathsf{V}} \mathsf{open}_{\mathsf{crs}}(C_{i0}^{x^j}, o_{i0}^{x^j}) + \sum_{j:\delta_{z_j^k}^G=1\wedge j\notin\mathsf{V}} \mathsf{open}_{\mathsf{crs}}(C_{i0}^{z_j^k}, o_{i0}^{z_j^k}),
\end{aligned}$$

where for each j such that $\mathsf{Dof}(x^j) \notin \mathsf{V}$ (resp., $j \notin \mathsf{V}$) the value x^j (resp., z_j^k) belongs to a corrupt party, so $C_{i0}^{x^j}$ and $o_{i0}^{x^j}$ (resp., $C_{i0}^{z_j^k}$ and $o_{i0}^{z_j^k}$) are fixed, and for every j such that $\mathsf{Dof}(x^j) \in \mathsf{V}$ and $\delta_{x^j}^G = 0$ (resp., $j \in \mathsf{V}$ such that $\delta_{z_j^k}^G = 0$) the value x^j (resp., z_j^k) belongs to a corrupt party, so $F^{x^j}(x,y)$ (resp., $F^{z_j^k}(x,y)$) is fixed.

Observe that the polynomial $G_i^k(x,y)$ and the ideal-world polynomial $\bar{G}_i^k(x,y)$ have the same distribution, and that conditioned on those values, for any $j \in \mathsf{H}$ the leakage and output of the $(\mathcal{F}_{\mathsf{glinear}})_{\mathsf{correct}}^{k,i,j}$ call have the same distribution in both worlds, and that for any $j \in \mathsf{C}$, the leakage of the $(\mathcal{F}_{\mathsf{glinear}})_{\mathsf{correct}}^{k,i,j}$ call has the same distribution in both worlds.

Corrupt P_i and $\delta^G_\alpha \wedge \delta^G_\beta = 1$. In this case, it is not hard to see that for each honest P_j in V with $\mathsf{flag}_{ji} = 0$, the output of $\mathcal{F}^{ji}_{\mathsf{sif}}$ has the same distribution in both worlds.

Corrupt P_i and $\delta^G_\alpha \wedge \delta^G_\beta = 0$. In this case, in the hybrid-world let u and v be defined as in the protocol, and consider the polynomial $G^k_i(x,y) := vF^{\gamma^k_i}(x,y) + uF^{\rho^k_i}(x,y) + F^{\eta^k_i}(x,y) + uv - F^{p^k_i}(x,y)$, and observe that it is fixed, and equal to the ideal-world polynomial $\bar{G}^k_i(x,y)$. It is not hard to see that for any $j \in \mathsf{H}$ the leakage and output of the $(\mathcal{F}_{\mathsf{glinear}})^{k,i,j}_{\mathsf{correct}}$ calls have the same distribution in both worlds, and that for any $j \in \mathsf{C}$, the leakage of the $(\mathcal{F}_{\mathsf{glinear}})^{k,i,j}_{\mathsf{correct}}$ calls have the same distribution in both worlds.

Adversary's communication. Conditioned on the above values, the adversary's inputs to the functionalities are picked in the same way, and the outputs are computed exactly like in the hybrid-world. This concludes the analysis of the adversary's view.

B.7.9 Honest Parties' Output

We say that a view is "good" if for any commitment C that appears in the view, and for any two openings o, and o' that appear in the view, it holds that either $\mathsf{open}_{\mathsf{crs}}(C,o) = \perp$ or $\mathsf{open}_{\mathsf{crs}}(C,o') = \perp$ or $\mathsf{open}_{\mathsf{crs}}(C,o) = \mathsf{open}_{\mathsf{crs}}(C,o)$. Observe that a view is good with probability at least $1-\epsilon$.

We claim that both in the real-world and the ideal-world, the outputs of the honest parties can be extracted from a good View by the following efficient deterministic process: (1) denote the inputs of an honest P_i by $w^{L_{i-1}+1}, \ldots w^{L_{i-1}+\ell_i}$, (2) for any corrupt P_i which is not in V, set $w^{L_{i-1}+1}, \ldots w^{L_{i-1}+\ell_i}$ to be 0, (3) for every corrupt P_i which is in V, and every value s that belongs to P_i, and for any $j \in \{L_{i-1}+1, \ldots, L_{i-1}+\ell_i\}$, the $\mathcal{F}^{w^j}_{\mathsf{vss}}$ call did not end with "D is corrupt", so let $F^{w^j}(x,y)$ be the corresponding sharing polynomial, and set $w^j := F^{w^j}(0,0)$, (4) the output is $y(\mathbf{w})$.

By the definition of the simulator it follows that this is indeed the output in the ideal-world, so it remains to analyse the real-world. For every good view View, it holds that (1) for every value s that belongs to an honest party, and for every $i \in \mathsf{C}$ and $j \in \{0, \ldots, n\}$, the adversary did not provide any opening o such that $\mathsf{open}_{\mathsf{crs}}(C^s_{ij}, o)$ is not $\perp$ and not equal to $F^s(j,i)$, where $F^s(x,y)$ is the sharing polynomial picked by the honest party, and (2) for every value s that belongs to a corrupt party in V, and for every $i,j \in \{0,\ldots,n\}$, the adversary did not provide any opening o such that $\mathsf{open}_{\mathsf{crs}}(C^s_{ij}, o)$ is not $\perp$ and not equal to $F^s(j,i)$, where $F^s(x,y)$ is the sharing polynomial defined at the end of the corresponding $\mathcal{F}_{\mathsf{vss}}$ or $\mathcal{F}_{\mathsf{tss}}$ call.

Fix some $k \in \{1,\ldots,m\}$, and we show that $y^k = x^\alpha x^\beta + x^1 + \ldots + x^m$ has the same value as in the process. We split into cases.

- Assume that P_α or P_β (or both) are not in V. This means that P_α or P_β are corrupt, so in the ideal-world $y^k = x^1 + \ldots + x^m$.

  In the real-world, define $\mathbf{a}$ and δ^G such $\mathbf{a}$ is the all-one vector, and for every $j \in \{1,\ldots,n\}$ it holds that $\delta^G_j = 1$ if $\mathsf{Dof}(x^j) \in \mathsf{V}$ (resp., $\delta^G_{0^k_j} = 1$ if $j \in \mathsf{V}$), and $\delta^G_j = 0$ (resp., $\delta^G_{0^k_j} = 0$) otherwise. For each $i \in \mathsf{V} \cap \mathsf{H}$ the output of $(\mathcal{F}_{\mathsf{glinear}})^{k,i}_{\mathsf{lin}}$ is

$(\mathbf{a}, \delta^G, \sum_{j:\mathsf{Dof}(x^j)\in\mathsf{V}} F^{w^j}(0,i) + \sum_{j\in\mathsf{V}} F^{0^k_j}(0,i))$. In addition, since the view is good, for every corrupt P_i in V such that the output of $(\mathcal{F}_{\mathsf{glinear}})^{k,i}_{\mathsf{lin}}$ is of the form $(\mathbf{a}, \delta^G, s_i)$, it holds that $s_i = \sum_{j:\mathsf{Dof}(x^j)\in\mathsf{V}} F^{x^j}(0,i) + \sum_{j\in\mathsf{V}} F^{0^k_j}(0,i)$. We conclude that the parties recover the polynomial $f^k(x) = \sum_{j:\mathsf{Dof}(x^j)\in\mathsf{V}} F^{x^j}(0,x) + \sum_{j\in\mathsf{V}} F^{0^k_j}(0,x)$, whose free-coefficient is $\sum_{j:\mathsf{Dof}(x^j)\in\mathsf{V}} F^{x^j}(0,0) + \sum_{j\in\mathsf{V}} F^{0^k_j}(0,0)$. Finally, since View is good and for each $j \in \mathsf{V}$ the output of $\mathcal{F}^{0^k_j}_{\mathsf{sif}}$ is $\mathbf{C}^{0^k_j}$, and out $= 1$, it holds that $F^{0^k_j}(0,0) = 0$ for every $j \in \mathsf{V}$. We conclude that in the real-world $y^k = \sum_{j:\mathsf{Dof}(x^j)\in\mathsf{V}} F^{x^j}(0,0)$, which is equal to the output of the process, as required.

- Otherwise, both P_α and P_β are in V. In the real-world, for every $i \in \{1,\ldots,n\}$ define $f^k_i(x) := F^\alpha(x,i)F^\beta(x,i) + \sum_{j:\mathsf{Dof}(x^j)\in\mathsf{V}} F^{x^j}(x,i) + \sum_{j\in\mathsf{V}} \lambda^j_i F^{z^k_j}(x,i)$. Let $f^k(x)$ be the degree-$2t$ polynomial obtained by interpolating $\{f^k_i(0)\}_{i\in\{1,\ldots,n\}}$, and observe that, by Fact A.10, it holds that $f^k(0) = F^\alpha(0,0)F^\beta(0,0) + \sum_{j:\mathsf{Dof}(x^j)\in\mathsf{V}} F^{x^j}(0,0)$. We continue by showing that for every $i \in \{1,\ldots,n\}$ the value $\bar{y}^k_i$ recovered by the corrupt parties is equal to $f^k(i)$.
  - If P_i is not in V, then, since View is good, it holds that the parties recover the value $\bar{y}^k_i = F^\alpha(0,i)F^\beta(0,i) + \sum_{j:\mathsf{Dof}(x^j)\in\mathsf{V}} F^{x^j}(0,i) + \sum_{j\in\mathsf{V}} \lambda^j_i F^{z^k_j}(0,i) = f^k(i)$, as required.
  - If P_i is in V, let $(\mathbf{a}, \delta^G, y^k_i)$ be the output of $\mathcal{F}^{k,i}_{\mathsf{gdtc}}$. We split into cases.

    Honest P_i**.** In this case it is not hard to see that $\bar{y}^k_i = F^\alpha(0,i)F^\beta(0,i) + \sum_{j:\mathsf{Dof}(x^j)\in\mathsf{V}} F^{x^j}(0,i) + \sum_{j\in\mathsf{V}} \lambda^j_i F^{z^k_j}(0,i) = f^k(i)$.

    Corrupt P_i**.** Since P_i is in V then, for every s such that $\delta^G_s = 1$ in $\mathcal{F}^{k,i}_{\mathsf{gdtc}}$, it holds that the verification phase of $\mathcal{F}^{s,i}_{\mathsf{vao}}$ ended with "verification succeeded", and so the corrupt P_i holds openings to $\mathbf{C}^s_i$ such that the shares of the honest parties correspond to a degree-t polynomial $h^s(x)$. Since View is good, if $\delta^G_\alpha \wedge \delta^G_\beta = 1$ then $y^k_i = h^{x^\alpha}(0)h^{x^\beta}(0) + \sum_{j\in\{1,\ldots,n\}:\delta^G_{w^j}=1} h^{x^j}(0) + \sum_{j\in\{1,\ldots,n\}:\delta^G_{z^k_j}=1} \lambda^j_i h^{z^k_j}(0) + h^{p^k_i}(0)$. On the other hand, if $\delta^G_\alpha \wedge \delta^G_\beta = 0$, then $y^k_i = \sum_{j\in\{1,\ldots,n\}:\delta^G_{x^j}=1} h^{x^j}(0) + \sum_{j\in\{1,\ldots,n\}:\delta^G_{z^k_j}=1} \lambda^j_i h^{z^k_j}(0) + h^{p^k_i}(0)$. In addition, for every $\mathcal{F}^{s,i}_{\mathsf{vao}}$, for which the parties execute the opening phase it holds that the output is $h^s(0)$. We conclude that the parties recover $\bar{y}^k_i = F^\alpha(0,i)F^\beta(0,i) + \sum_{j:\mathsf{Dof}(x^j)\in\mathsf{V}} F^{x^j}(0,i) + \sum_{j\in\mathsf{V}} \lambda^j_i F^{z^k_j}(0,i) = f^k(i)$.

  We conclude that in the real-world it holds that $y^k = f^k(0) = F^\alpha(0,0)F^\beta(0,0) + \sum_{j:\mathsf{Dof}(x^j)\in\mathsf{V}} F^{x^j}(0,0)$ which is equal to the output of the process, as required.

This concludes the analysis of the honest parties' outputs, and the proof of security of $\mathcal{F}_{\mathsf{dtc}}$. $\square$