Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 31 (2023)

The Round Complexity of Statistical MPC with Optimal Resiliency*

Benny Applebaum† Eliran Kachlon† Arpita Patra‡

Abstract

In STOC 1989, Rabin and Ben-Or (RB) established an important milestone in the fields of cryptography and distributed computing by showing that every functionality can be computed with statistical (information-theoretic) security in the presence of an active (aka Byzantine) rushing adversary that controls up to half of the parties. We study the round complexity of general secure multiparty computation and several related tasks in the RB model.

Our main result shows that every functionality can be realized in only four rounds of interaction which is known to be optimal. This completely settles the round complexity of statistical actively-secure optimally-resilient MPC, resolving a long line of research.

Along the way, we construct the first round-optimal statistically-secure verifiable secret sharing protocol (Chor, Goldwasser, Micali, and Awerbuch; STOC 1985), show that every single-input functionality (e.g., multi-verifier zero-knowledge) can be realized in 3 rounds, and prove that the latter bound is optimal. The complexity of all our protocols is exponential in the number of parties, and the question of deriving polynomially-efficient protocols is left for future research.

Our main technical contribution is a construction of a new type of statistically-secure signature scheme whose existence was open even for smaller resiliency thresholds. We also describe a new statistical compiler that lifts up passively-secure protocols to actively-secure protocols in a round-efficient way via the aid of protocols for single-input functionalities. This compiler can be viewed as a statistical variant of the GMW compiler (Goldreich, Micali, Wigderson; STOC, 1987) that originally employed zero-knowledge proofs and public-key encryption.

*This is a full version of a paper published in STOC'23.

†Tel-Aviv University, Israel bennyap@post.tau.ac.il, elirn.chalon@gmail.com. Supported by the Israel Science Foundation grant no. 2805/21 and by the European Union (ERC, NFITSC, 101097959). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

‡Indian Institute of Science, Bangalore, India arpita@iisc.ac.in. Financial support from C3iHub, IIT Kanpur, J. P. Morgan Chase Faculty Award and SONY Faculty Innovation Award are acknowledged.

ISSN 1433-8092

Contents

1 Introduction

The round complexity of interactive protocols is one of their most important efficiency measures. Consequently, a huge amount of research has been devoted towards characterizing the round complexity of various distributed tasks (e.g., Byzantine agreement [LF82, DR85, FM85], coin flipping [Cle86, MNS16], zero-knowledge proofs [GK96, CKPR01], verifiable secret sharing [GIKR01, KPR10] and general secure multiparty computation [Yao86, BMR90, GS18, BL18]) under different security models.

In this work, we focus on the round complexity of protocols that achieve *full information-theoretic* security, including *guaranteed output delivery* in the presence of an active (aka Byzantine or malicious), static, computationally-unbounded, rushing adversary. We assume that there are n parties that communicate over *secure point-to-point channels*, and also that they have an access to a *broadcast* channel. Feasibility results in this model were first proved in the classic works of Ben-Or, Goldwasser, and Wigderson [BGW88] and Chaum, Crépeau and Damgård [CCD88]. Specifically, it is known that *perfect security* is achievable if and only if the adversary corrupts less than a third of the parties, i.e., the best-achievable *resiliency threshold* is $t = \lfloor (n-1)/3 \rfloor$. Quite remarkably, Rabin and Ben-Or [RB89] later showed that, by compromising on *statistical security*, the resiliency can be improved to $t = \lfloor (n-1)/2 \rfloor$. Put differently, a standard "honest majority" is sufficient if one is willing to tolerate a negligible statistical error in privacy and correctness. This is the best that one can hope for since an honest majority is known to be necessary even for weaker notions like *passive* statistical security [CK89] or active *computational* security with guaranteed output delivery [Cle86].

Our goal in this paper is to determine the round complexity of general multiparty computation (MPC) in the statistical setting (aka the Rabin-Ben-Or setting). Indeed, following recent results that settled the round complexity of MPC for the perfect (aka BGW) setting [ABT18, GIS18, ABT19, ACGJ19, AKP20b] and for different variants of the computational setting [GLS15, GS17, GS18, BL18, ACJ17, BGJ+18, BHP17, ACGJ18, RCCG+20, BJMS20, HHPV21, AKP21], the statistical setting is arguably the last main challenge in this domain. We therefore ask:

> What is the optimal round complexity of general MPC with full statistical security, including guaranteed output delivery, and optimal resiliency?

Indeed, the round complexity of statistically-secure protocols has remained wide open for any resiliency t larger than $n/3$, let alone for the central case of *optimal resiliency* of $t = \lfloor (n-1)/2 \rfloor$. In fact, in this setting, we do not even know what is the exact round complexity of much more basic primitives, such as statistically-secure *verifiable secret sharing*.

1.1 Our Results

In this work, we settle the round complexity of general MPC, verifiable secret sharing, and several related primitives. Details follow.

1.1.1 Verifiable Secret Sharing

Verifiable secret sharing (VSS) [CGMA85] is arguably the most basic primitive in information-theoretic multiparty computation, and it is known to be necessary for the construction of general MPC protocols both in the perfect setting (see [AKP20b]), and in the statistical setting

(see [AKP20a]).[1] At a high level, a VSS scheme consists of two phases: a sharing phase, which allows a dealer to share a secret s among the parties, and a reconstruction phase, which allows the parties to recover the secret s. For an honest dealer, VSS has the same guarantees as robust secret sharing, that is, the adversary learns no information about s in the sharing phase (privacy), and the secret s will always be reconstructed properly in the reconstruction phase despite the misbehavior of the adversary (correctness). In addition, for a corrupt dealer, we require commitment, which means that at the end of the sharing phase there is some value s' that will always be reconstructed in the reconstruction phase.

While the round complexity of perfect VSS is well-understood (see [FGG+06, KKK09, GIKR01]), this problem is still open in the statistical settings. The best-known scheme [KPR10] achieves 3 rounds of sharing and 2 rounds of reconstruction. It is known that 3 rounds are necessary for the sharing phase [PCRR09, AKP20a], but it is unclear whether 2 rounds of reconstruction are necessary. In fact, as shown by [CDF01], one can derive a VSS in which the reconstruction phase consists of a single round by first constructing such a protocol under the assumption that the dealer is honest (aka robust secret sharing) [RB89], and then using statistical MPC to emulate the honest dealer. This approach suffers, however, from a large number of rounds in the sharing phase, which is inherently sub-optimal since it employs a VSS with multiple rounds of reconstruction as a building block (as part of the MPC protocol). Apart from the MPC-based approach, it is unknown how to achieve a single round of reconstruction even when more than 3 rounds of sharing are being employed [RB89, CDD+99, KPR10]. Let us note that single-round reconstruction has a qualitative advantage due to its non-interactive nature (parties can just "speak" once without waiting for others).

Overall, the existence of a statistical VSS that *simultaneously* achieves 3 rounds of sharing and a single round of reconstruction is open. In fact, the question is not fully resolved even if one adds another round of reconstruction (as in [KPR10]) since the known construction [KPR10] lacks some important properties that are typically employed as part of MPC protocols. Most notably, it is not *linearly homomorphic*, i.e., parties cannot locally combine shares of different secrets into a new share of a linear combination of the underlying secrets.

In this work, we provide the first construction of a round-optimal VSS scheme, that requires three rounds of sharing, and only *one round of reconstruction*. Our construction is also linearly-homomorphic (and has other "MPC-friendly" features, See Section 6).

Theorem 1.1. *For a security parameter κ, number of parties n, and number of corrupt parties $t < n/2$, there exists a protocol for verifiable secret sharing with three rounds of sharing and one round of reconstruction, that provides statistical security against an active, static, rushing unbounded adversary that corrupts up to t parties, with error $2^{-\kappa}$. The running time of the protocol is $\mathrm{poly}(\kappa, 2^n)$.*

Remark 1.2 (On the exponential dependency on n)**.** *Our VSS protocol, as well as the rest of our protocols, have exponential dependency on the number of parties n, so they are only efficient when $n = O(\log \kappa)$. We emphasize that in the settings of statistical security, even inefficient protocols are meaningful, since the protocols are secure even against a* computationally-unbounded *adversary. We also mention that the question of an efficient VSS scheme with three rounds of sharing is open even if we allow more than one round of reconstruction. Indeed, the construction of [KPR10] also has exponential dependency on the number of parties, even though it allows two rounds of reconstruction.*

[1] Unless stated otherwise, whenever we refer to the *perfect* setting and *statistical* setting, we assume that the resiliency threshold is taken to be optimal, i.e., $t_{\text{perfect}} = \lfloor (n-1)/3 \rfloor$ and $t_{\text{stat}} = \lfloor (n-1)/2 \rfloor$, respectively.

1.1.2 Single Input Functionalities

Before moving to the case of MPC for general functionalities, it is useful to consider the special case of *single input functionalities* [GIKR02], whose output depends on the input of a single party, called the dealer. Single input functionalities capture a large class of non-trivial tasks, including secure multicast and multi-verifier zero-knowledge. In the perfect setting, the sharing phase of VSS can be also captured by SIF. However, in the statistical setting, where there is merely an honest majority, it can be shown that SIF *cannot* capture VSS. (See Appendix A for proof.)

Augmented SIF. We present a stronger notion of SIF, called *augmented SIF*, that captures VSS as well as other related tasks (that will be useful later as building blocks for general MPC). Intuitively, it allows the computation of a single input functionality together with some verification information that will allow every party to publicly open its output, and convince the rest of the parties of its validity. Formally, for a single input functionality $\mathcal{F}$, the corresponding augmented single input functionality is a two-phase functionality $\mathcal{F}'$. The first phase, the computation phase, consists merely of the computation of $\mathcal{F}$. That is, the dealer inputs $\mathbf{x}$ to the functionality $\mathcal{F}'$, and the i-th party receives the value y_i as an output, where $\mathcal{F}(\mathbf{x}) = (y_1, \ldots, y_n)$. In the second phase, the opening phase, every P_i can input a command "open" to $\mathcal{F}'$, and the functionality will return y_i to the rest of the parties. It is not hard to verify that an augmented SIF of any $(t+1)$-out-of-n secret sharing scheme is indeed a VSS scheme.

The round complexity of SIF. Not much is known about the round complexity of SIF and augmented SIF. The three-round lower bound for the sharing phase of VSS [PCRR09, AKP20a] implies that the computation phase of an augmented SIF requires at least three rounds. First, we extend this lower bound to hold for standard SIF as well.

Theorem 1.3 (Lower Bound for SIF)**.** *Let $n \geq 3$ and $t \geq n/3$ be positive integers. Then there exists an n-party single input functionality that cannot be computed in two rounds with resiliency t and error $1/12$.*

The exact round complexity of both, SIF and augmented SIF, is open, and the best upper bound is some large constant [IK00, IK02, GIKR02].[2] We fully resolve this question and provide tight upper bounds. We show that every augmented SIF can be realized in three rounds for the computation phase, and one round for the opening phase. This implies that every SIF can be realized in three rounds (ignoring the opening phase).

Theorem 1.4 (Upper Bound for augmented SIF and SIF)**.** *For a security parameter κ, number of parties n, and number of corrupt parties $t < n/2$, for every single input functionality $\mathcal{F}$, the* augmented single input functionality *$\mathcal{F}'$ can be realized in three rounds for the computation phase, and one round for the opening phase, with statistical security against an active, static, rushing unbounded adversary that corrupts up to t parties, with error $2^{-\kappa}$. The running time of the protocol is $\mathrm{poly}(\kappa, 2^n, s)$, where s is the size of the boolean circuit computing $\mathcal{F}$. Consequently, the single-input functionality $\mathcal{F}$ can be realized in three rounds with similar complexity in the same setting.*

[2] The obvious approach is to use some MPC-friendly VSS, e.g., the VSS of [KPR10] whose sharing takes 4 rounds and reconstruction takes 2 rounds. Then we can use the protocol of [CDD+99] to compute a degree-2 SIF (which is complete for general SIF [GIKR02]). Computing one multiplication takes more than 10 rounds in [CDD+99], and so the protocol requires more than 14 rounds. By using standard tricks (sharing random multiplication triples [Bea91]) this can probably be improved to 7 rounds (4 rounds for sharing, 1 round for generating random challenges that are needed for the generation of random multiplication triples, and 2 rounds for opening).

Previously, the best 3-round SIF protocol with active information-theoretic security achieved a threshold of $t \leq \lfloor (n-1)/4 \rfloor$ [ABT19]. We also mention that if one is willing to relax the security to computational then the protocols of [AKP22] provide a 2-round SIF based on cryptographic assumptions (essentially non-interactive commitments).

Application: Multi-verifier zero-knowledge. In multi-verifier zero-knowledge [BD91] for an NP relation R, there is a single prover and k verifiers, all holding the same statement x. The prover wants to prove that she is holding a secret witness w so that $R(x,w) = 1$, without revealing any information about w. Observe that this task is captured by a single input functionality, that takes (x,w) from the prover, and returns $(x, \text{"true"})$ to all the parties if $R(x,w) = 1$, and $(x, \text{"false"})$ otherwise. Therefore, when there is an honest majority among the $k+1$ parties, we can use our SIF protocol to derive the first multi-verifier zero-knowledge proof system that runs in three rounds and achieves statistical security.We highlight the special case of a single prover and two verifiers (i.e., $k = 2$), allowing a single corruption. That is, by adding just a single verifier to the standard zero-knowledge settings, we obtain, for the first time, an efficient zero-knowledge proof with *statistical security* under no cryptographic assumptions. As a bonus, we even provide UC-security [Can01], which implies *straight-line black-box simulation*, as well as *knowledge extraction*. In comparison, in the standard settings of (single-verifier) zero-knowledge proofs three rounds protocols require non-black box simulation [GK96].

1.1.3 General Multiparty Computation

The round complexity of statistical-MPC for general functionalities is a long-standing open problem. For a long time, it is known that constant-round protocols exist [IK00, IK02], where the exact number of rounds is some large constant. More recently, [AKP20a] proved that at least four rounds are required for general statistical MPC. In this work, we close the gap and prove that four rounds are also sufficient.

Theorem 1.5 (General MPC in four rounds)**.** *For a security parameter κ, number of parties n, and number of corrupt parties $t < n/2$, every functionality $\mathcal{F}$ can be realized in four rounds with statistical security against an active, static, rushing unbounded adversary that corrupts up to t parties, with error $2^{-\kappa}$. The running time of the protocol is $\mathrm{poly}(\kappa, 2^n, s, 2^d)$, where s is the size of the boolean circuit computing $\mathcal{F}$, and d is the depth of the circuit.*

As in all known constructions of constant-round information-theoretic MPC, there is an exponential dependency on the depth of the circuit. Getting rid of this dependency, even in weaker adversarial models (e.g., passive adversary and resiliency of $t = 1$), is a famous open problem that goes back to [BMR90]. A potentially more accessible goal is to get-rid of the exponential dependency in the number of parties n (see Remark 1.2). Based on current techniques, $\mathrm{poly}(n)$-time protocols seem to require at least 7 rounds, and so the gap between efficient and inefficient solutions is quite large in this case.

Overall, our protocol is only efficient for $n = O(\log \kappa)$ and for NC^1 functionalities.[3] Nevertheless, even for general functions, for which our construction is inefficient, the result remains meaningful since the protocol resists computationally unbounded adversaries. More generally,

[3] As in similar cases, this can be pushed up to log-space functionalities since they securely reduce to NC^1 functionalities via non-interactive reductions [IK02].

ignoring efficiency aspects, one can view our theorems as *computability results* that characterize the minimal computational model (in terms of rounds of interactions) in which universal computation can be carried out with statistical security and optimal resiliency.

2 Technical Overview

In this section, we provide a detailed technical overview of our construction. We denote the parties by $P_1, \ldots, P_n$, and we assume that at most $t < n/2$ of the parties are corrupt. We denote the security parameter by κ, and throughout, we think of $\mathbb{F}$ as a finite field of size $\exp(n, \kappa)$, and of $1, \ldots, n$ as n distinct non-zero field elements.

At a high level, our construction consists of two main parts: the construction of a round-optimal VSS scheme, and a transformation from VSS to general MPC. In Section 2.1 we provide a detailed overview of our VSS scheme, that constitutes our main technical contribution. In Section 2.2 we provide a short overview of the transformation from VSS to general MPC via augmented SIF.

2.1 Verifiable Secret Sharing

Background. We begin with some background on the qualitative difference between VSS in the perfect setting and VSS in the statistical setting. It will be instructive to start with the simpler task of *robust secret sharing* where an *honest* dealer $\mathcal{D}$ shares a secret s among n players. We require t-privacy, i.e., t shares reveal no information about s, and also perfect t-robustness, which means that if all the parties send their shares to a receiver $\mathcal{R}$, then $\mathcal{R}$ can recover s even if t shares are maliciously chosen and might depend on the honest shares. When $n = 3t + 1$, it is known that Shamir's secret sharing satisfies those requirements since Reed-Solomon codes allow to recover the secret from t errors. On the other hand, when $n = 2t + 1$, it is not hard to see that this task is impossible. Indeed, if $(s_1, \ldots, s_n)$ are shares of 0, and $(s'_1, \ldots, s'_n)$ are shares of 1, then the Hamming distance between the two vectors has to be at least $2t+1 = n$, or otherwise t-robustness is violated. But this means that we can distinguish a secret sharing of 0 from a secret sharing of 1 based on a single share, so t-privacy is violated. In fact, this argument shows that perfect robustness is possible only when $n \geq 3t+1$. We will see that this qualitative difference propagates up to the more challenging task of VSS.

In the perfect setting, when $n \geq 3t + 1$, the canonical approach [BGW88, GIKR01, FGG+06, KKK09, AL17, AKP20b] is to design an MPC protocol for the single input functionality that takes an input s and randomness r from the dealer, generates the shares $s_1, \ldots, s_n$ according to Shamir's scheme, and delivers s_i to P_i. Privacy follows from the privacy of the secret sharing scheme, while commitment (and correctness) follows from the perfect robustness property: Even if the dealer is dishonest, and therefore knows all the shares of the honest parties and has full control of the shares of t corrupt parties, perfect robustness guarantees that there *exists* exactly one valid opening in the reconstruction phase. Indeed, this approach leads to VSS protocols with an optimal round complexity [GIKR01, FGG+06, KKK09, AKP20b] (3 rounds of sharing and a single round of reconstruction) and so the problem in the perfect setting is well understood.

The situation in the statistical setting is more subtle. As already observed, we cannot hope for perfect robustness whenever $t \geq n/3$, and so the commitment property cannot be based on the nonexistence of ambiguous openings. Instead, one has to argue that it is infeasible to find such

ambiguous openings given the adversary's view. That is, we have to inject some *private* randomness into the shares of the honest parties.[4] Indeed, following [RB89], the canonical approach here is to augment each share s_i with a private "proof-of-validity", that allows P_i to convince the rest of the parties in the validity of its share s_i. If P_i tries to open an invalid share $s'_i \neq s_i$, then with high probability P_i will fail to generate a proof-of-validity for s'_i, and the rest of the parties will set the share of P_i to an *erasure*. Since t erasures can be handled when $n \geq 2t + 1$, such proofs-of-validity suffices. Of course, some of the randomness used to generate the proofs-of-validity has to come from the honest parties and should remain hidden from the adversary. Furthermore, the use of interactive reconstruction (which allows for interactive verification of validity) seems to be of significant help. (See, e.g., the discussions in [CDF01, CFOR12, FY20] in the context of robust secret sharing.) In contrast, in the perfect setting, it can be shown that interaction is useless in the reconstruction phase [GIKR02].)

Interactive signatures. The main tool for constructing the proofs-of-validity is some form of *information-theoretic interactive signatures* (aka information-checking protocols [RB89]). This is essentially a weak version of VSS in which the opening is conducted by some designated party $\mathcal{I}$. Following the definition of [PCR08, PCPR09], an interactive signature is a protocol involving n parties, where two of them are distinguished: the dealer $\mathcal{D}$ and the intermediary $\mathcal{I}$. (Say that P_1 is $\mathcal{D}$ and that P_2 is $\mathcal{I}$.) The protocol consists of 3 phases as follows:

1. *Distribution phase*: $\mathcal{D}$ sends to $\mathcal{I}$ a secret s together with some authentication information, and some verification information to the rest of the parties.

2. *Verification phase*: The parties verify that the information that $\mathcal{D}$ sent is "valid" and "consistent" with the secret s that $\mathcal{I}$ holds, and terminate the phase with a public decision on success or failure that is taken based on public information (i.e., broadcasts).

3. *Selective opening phase*: Assuming that the verification phase succeeds, $\mathcal{I}$ can publicly open the value s to all the parties. The parties decide whether to accept or reject this opening. Crucially, the decision of whether to open the value is in the hands of $\mathcal{I}$ and may depend on external reasons. (Hence the term "selective".)

Correctness and *privacy* are defined in a natural way: When $\mathcal{D}$ and $\mathcal{I}$ are honest, the verification succeeds, the opening of $\mathcal{I}$ is accepted by all honest parties (correctness), and the adversary learns no information about s in the distribution phase and verification phase (privacy). The commitment property from the VSS is replaced with the following three fine-grained requirements that are all conditioned on the success of the verification phase: (a) *unforgeability*: If $\mathcal{D}$ is honest and $\mathcal{I}$ is corrupt, the honest parties will reject an opening of $s' \neq s$ by $\mathcal{I}$; (b) *nonrepudiation*: If $\mathcal{D}$ is corrupt and $\mathcal{I}$ is honest, then the honest parties will accept the opening of s by $\mathcal{I}$; and (c) *agreement*: All honest parties agree on whether to accept or reject the opening of $\mathcal{I}$, even if both $\mathcal{I}$ and $\mathcal{D}$ are corrupt. If verification fails, unforgeability, nonrepudiation, and agreement are vacuously satisfied.

[4] Consequently, in the honest majority statistical setting, VSS cannot be realized by a single input functionality. This is true even if multiple rounds of reconstruction are allowed as we prove in Section A.

VSS from signatures. The work of [KPR10] implicitly shows a VSS scheme with three rounds of sharing and one round of reconstruction (which is also MPC friendly) can be based on any signature scheme with one round of distribution, two rounds of verification, and a single round of selective opening that can be executed in *parallel* to the second round of the verification phase. We call such a signature scheme a $(1,2,1)$-signature. Unfortunately, all known constructions of interactive signatures [RB89, CDD+99, PCR08, KPR10], regardless of the round-complexity of the distribution phase and the verification phase, require an interactive (two-round) sub-protocol for selective opening. To bypass this barrier, let us first take a fresh look at existing constructions [PCR08, PCPR09] which originally rely on polynomials, and abstract them by using general linear secret-sharing schemes.

Abstraction of previous constructions. Consider a linear secret-sharing scheme over $\mathbb{F}$ for N secret-sharing players $Q_1, \ldots, Q_N$. Linearity means that in order to share a secret $s \in \mathbb{F}$, we sample a random vector $\boldsymbol{\rho}_s$ whose first entry is s, and set the i-th share to $L_i(\boldsymbol{\rho}_s)$ where L_i is some public non-degenerate linear operator that is associated with the i-th player Q_i. The scheme is parameterized by a threshold $T > n$ and we assume that any coalition of size n learns nothing about the secret and coalitions of size T can recover the secret and *the randomness vector* $\boldsymbol{\rho}_s$. The latter property implies that the mapping $L : \boldsymbol{\rho}_s \mapsto (L_i(\boldsymbol{\rho}_s))_{i\in[N]}$ forms a linear code of distance $\Delta = N - T + 1$. Jumping ahead, we will have exponentially many "virtual" secret-sharing players, i.e., $N = \exp(n,\kappa)$, but the threshold T is polynomial in n and κ. We will employ only n (randomly chosen) sharing players, so the overall complexity can be, in principle, $\mathrm{poly}(n,\kappa)$.[5] An interactive signature (with 2 rounds of selective opening) can be constructed as follows.

1. *Single-round Distribution phase*: Given a secret $s \in \mathbb{F}$, the dealer samples a random mask $r \in \mathbb{F}$, and random vectors $\boldsymbol{\rho}_s$ and $\boldsymbol{\rho}_r$ whose first entry is s and r, respectively. All these random values are sent to $\mathcal{I}$. In addition, for every P_i, the dealer picks a random index $\alpha_i \in [N]$ that represents some (virtual) secret-sharing player, and sends to party P_i the index α_i together with the corresponding shares $s_i := L_{\alpha_i}(\boldsymbol{\rho}_s)$ and $r_i := L_{\alpha_i}(\boldsymbol{\rho}_r)$, which will be used as "authenticators". This step reveals no information about s and r since the adversary can see at most n shares.

2. *2-round Verification phase*: $\mathcal{I}$ broadcasts a random linear combination of the randomizers $\boldsymbol{\rho}_s$ and $\boldsymbol{\rho}_r$, i.e., $\mathcal{I}$ samples a non-zero scalar $c \in \mathbb{F}$ and broadcasts $(c, \quad \boldsymbol{\rho} := \boldsymbol{\rho}_s + c \cdot \boldsymbol{\rho}_r)$. This equation is being checked by the dealer who broadcasts a public complaint if $\boldsymbol{\rho}_s + c \cdot \boldsymbol{\rho}_r \neq \boldsymbol{\rho}$. If such a complaint is issued verification fails, otherwise verification succeeds. These messages do not violate privacy since $\boldsymbol{\rho}_r$ masks the value of $\boldsymbol{\rho}_s$.

3. *2-round Selective opening phase*: In order to open the secret $\mathcal{I}$ broadcasts $\boldsymbol{\rho}_s$. We say that P_i *votes* for the opening if either (a) $L_{\alpha_i}(\boldsymbol{\rho}_s) = s_i$ or (b) $L_{\alpha_i}(\boldsymbol{\rho}) \neq s_i + c \cdot r_i$ and $\mathcal{D}$ did not broadcast a complaint.[6] In the second round, every party broadcasts its vote, and the parties accept if a majority of the parties vote for the opening. (If the verification phase failed, then the parties simply ignore the selective opening phase.)

[5] To achieve such a complexity, the secret sharing should be strongly explicit, i.e., the ith share should be computable in time $\mathrm{poly}(T, \log(N)) = \mathrm{poly}(n,\kappa)$. For example, one can use Shamir's $(n+1)$-out-of-N secret sharing scheme with a field of size $|\mathbb{F}| = N + 1$ and $T = n + 1$.

[6] In the latter case, P_i thinks that $\mathcal{D}$ is corrupt and so he shouldn't worry about unforgeability and there is no harm in accepting the opening.

Analysis (sketch). Correctness, privacy, and agreement are straightforward. For unforgeability, we assume that $\mathcal{D}$ is honest, $\mathcal{I}$ is corrupt, verification passes and $\mathcal{I}$ opens $\boldsymbol{\rho}'_s \neq \boldsymbol{\rho}_s$. Since the code L has distance Δ, the probability that $L_{\alpha_i}(\boldsymbol{\rho}_s) \neq L_{\alpha_i}(\boldsymbol{\rho}'_s)$, for a random $\alpha_i \in [N]$, is at least $\Delta/N > 1 - T/N$, and so every honest party is likely to vote against the opening. Here we crucially relied on the fact that the adversary does not know α_i. For nonrepudiation, assume that $\mathcal{D}$ is corrupt, $\mathcal{I}$ is honest, verification passes but $\mathcal{I}$ is rejected, i.e., at least one honest party P_i voted against the opening. Thus, $L_{\alpha_i}(\boldsymbol{\rho}_s) \neq s_i$ but $L_{\alpha_i}(\boldsymbol{\rho}) = s_i + c \cdot r_i$. By linearity, this happens iff $c \cdot (r_i - L_{\alpha_i}(\boldsymbol{\rho}_r)) = (L_{\alpha_i}(\boldsymbol{\rho}_s) - s_i)$, and since the RHS is non-zero, this happens with a probability of at most $1/(|\mathbb{F}| - 1)$ over the choice of the random non-zero scalar c.

Reducing a round? We can try to reduce one round of the selective opening by letting each party decide locally based on his own vote. However, in this case, an adversary that corrupts both $\mathcal{D}$ and $\mathcal{I}$ can violate the agreement property by generating vectors $\boldsymbol{\rho}_s, \boldsymbol{\rho}, (\alpha_i, s_i, r_i)$ and (α_j, s_j, r_j), so that an honest P_i accepts the opening while an honest P_j rejects the opening. One could also try to let every P_i broadcast the authentication values (α_i, s_i, r_i) in the first round of the selective opening phase, so the rest of the parties will be able to compute the vote of P_i based on $\boldsymbol{\rho}_s$ and (α_i, s_i, r_i). However, given this information, a corrupt rushing $\mathcal{I}$ can efficiently find an invalid opening that will be accepted by the honest parties, violating the unforgeability requirement. This problem can be fixed by increasing the distance of the code and setting the privacy threshold below n. But in this case, the authenticators (α_i, s_i, r_i) prematurely reveal the secret before we even know whether the intermediate wishes to open the secret, thus violating privacy. Overall, the challenge is to reveal enough information that allows the parties to reach an agreement (in case the secret is opened), while keeping enough uncertainty about the secret and its "authenticators" (for privacy and unforgeability).[7]

At a high level, we solve the problem by letting each party spread some partial, randomized, pieces of information about his local authenticators. In particular, each party P_i will receive *many* secret shares from the dealer and will spread random linear combinations of these shares to the other parties. Crucially, P_i will use a local private source of randomness. (This deviates from all previous approaches in which only $\mathcal{D}$ and $\mathcal{I}$ were randomized). The actual implementation of this approach requires some care. We will start with a simplified model that includes additional (virtual) verifiers (Section 2.1.1), and then explain how to emulate the verifiers in a round-preserving way (Section 2.1.2), in order to obtain a $(1, 2, 1)$-signature in the standard model.

2.1.1 Step I: Signature Scheme with Virtual Verifiers

A simplified model. In previous constructions, every P_i had a dual role: P_i acted both as a verifier, that had to vote for/against the opening of $\mathcal{I}$, and also as a receiver, that had to accept/reject the opening of $\mathcal{I}$. We consider a *simplified model* where this role is divided between two entities: a verifier $\mathcal{V}_i$ and a receiver P_i. Formally, the model consists of m verifiers $\mathcal{V}_1, \ldots, \mathcal{V}_m$, and n receivers $P_1, \ldots, P_n$ and we assume that the dealer $\mathcal{D}$ is P_1 and the intermediary $\mathcal{I}$ is P_2. The adversary can corrupt any number of the receivers, and can *weakly corrupt* any subset of the verifiers with one

[7] In contrast, in the reconstruction phase in VSS we do not care about the privacy of the secret, since the opening is not selective and all the parties know that the secret should be revealed. In this sense, signatures (with a single round of opening) are more challenging than VSS. Indeed, to the best of our knowledge, the question is open even when the resiliency threshold t is smaller than $n/3$ and perfect-VSS is available.

limitation: When $\mathcal{D}$ is honest and $\mathcal{I}$ is corrupt there must be at least one honest verifier. The notion of weak corruption is non-standard: A weakly corrupted verifier passes all her incoming messages to the adversary but keeps her internal state (i.e., her random tape) and the messages that she sends to the other honest parties hidden. In addition, the adversary is allowed to abort a weakly corrupted verifier at any time. If a verifier is not aborted, it plays its role honestly.

Hadamard-based secret sharing. Recall that we employ a linear secret-sharing scheme over $\mathbb{F}$ that is defined by N distinct linear mappings $\{L_i\}_{i\in[N]}$, one for each virtual party. We will need the (highly non-standard) property that this set of functions forms a linear space $\mathcal{L}$. For this, we will take $\mathcal{L}$ to be the space of *all* linear functions from $\mathbb{F}^v$ to $\mathbb{F}$ and think of each function as a vector in $\mathbb{F}^v$ (so $N = |\mathbb{F}|^v$). This is not a valid threshold secret sharing since some small coalitions (that span the vector $\mathbf{e}_1 = (1, 0, \ldots, 0)$) can recover the secret, and some huge coalitions (that do not span $\mathbf{e}_1$) may not be able to recover the secret. However, for a randomly chosen coalition of size $\gg v$ (resp., $\ll v$) correctness (resp., privacy) holds with high probability. These relaxed properties suffice (since privacy and correctness will only be needed when the dealer is honest and in this case the virtual parties will be selected at random). From a coding perspective, this secret sharing corresponds to the Hadamard code over a large field. Let u be polynomially larger than $\kappa \cdot m \cdot n$ and let v be polynomially larger than um. We modify the previous construction as follows:

1. *Single-round distribution phase:* As before, the dealer samples the randomizers $\boldsymbol{\rho}_s$ and $\boldsymbol{\rho}_r$ and sends them to $\mathcal{I}$. In addition, $\mathcal{D}$ allocates to each verifier u random virtual secret-sharing parties by sampling a random $u \times v$ matrix A_i, and sends to $\mathcal{V}_i$ the "names" of the virtual parties and their shares, $(A_i, \quad \mathbf{s}_i := A_i \cdot \boldsymbol{\rho}_s, \quad \mathbf{r}_i := A_i \cdot \boldsymbol{\rho}_r)$. Since $v \gg um$, with a very high probability the row-span $(A_i)_{i\in[m]}$ does not include the unit vector $\mathbf{e}_1 = (1, 0, \ldots, 0)$, which means that all the messages that the verifiers receive from $\mathcal{D}$ reveal no information about s and r.

2. *2-round verification phase:* As in the previous protocol, the intermediate publishes a random non-zero scalar c and a linear combination of the secret sharing randomizers $\boldsymbol{\rho} := \boldsymbol{\rho}_s + c \cdot \boldsymbol{\rho}_r$, and the dealer $\mathcal{D}$ announces whether verification succeeds by verifying the above equality. In addition, in the second round, every verifier $\mathcal{V}_i$ does the following: (a) broadcasts a public complaint if its local authenticators are inconsistent with the published information, i.e., if $A_i \cdot \boldsymbol{\rho} \neq \mathbf{s}_i + c \cdot \mathbf{r}_i$; and (b) privately sends to each receiver P_j, $j \in [n]$, a random linear combination of his s-shares $(\mathbf{t}_{i,j}, \quad \mathbf{a}_{i,j} := \mathbf{t}_{i,j} \cdot A_i, \quad s_{i,j} := \mathbf{t}_{i,j} \cdot \mathbf{s}_i)$, where $\mathbf{t}_{i,j} \leftarrow \mathbb{F}^u$ is a random row vector.

3. *Single-round selective opening phase*: To open the secret s, the intermediate $\mathcal{I}$ broadcasts the vector $\boldsymbol{\rho}_s$ to all the parties. After this, each receiver P_i locally computes a vote for each verifier $\mathcal{V}_j$ and rejects the opening if at least one verifier votes against the opening. The receiver P_i thinks that the verifier $\mathcal{V}_j$ votes against the opening if the following conditions hold: (1) $\mathcal{V}_j$ did not broadcast a complaint[8], (2) $\mathcal{V}_j$ did not abort, and (3) there is an inconsistency $\mathbf{a}_{i,j} \cdot \boldsymbol{\rho}_s \neq s_{i,j}$. Observe that this phase can be executed in parallel to the second round of the verification phase, so that if the verification phase failed the opening is simply ignored.

[8] Again, in case of a complaint the verifier (who always operates honestly) claims that the dealer is cheating, and so it's safe to accept the opening without worrying about forgery.

Analysis (sketch). It is not hard to see that correctness and privacy hold even if none of the verifiers are honest. For unforgeability, assume that $\mathcal{D}$ is honest, $\mathcal{I}$ is corrupt, some verifier, say $\mathcal{V}_1$ is honest, and the verification phase succeeds. By following the previous argument, unforgeability boils down to showing that the vector $\mathbf{a}_{1,j}$ that an honest receiver P_j gets from $\mathcal{V}_1$ is (almost) uniformly distributed. Moreover, this should hold even when conditioning on the adversary's view that consists of all the vectors $\mathbf{a}_{1,\ell}$ that $\mathcal{V}_1$ sent to the corrupted receivers. To see this, observe that the vectors $\mathbf{a} = (\mathbf{a}_{1,1}, \ldots, \mathbf{a}_{1,n})$ were generated by taking n random linear combinations $T = (\mathbf{t}_{1,1}, \ldots, \mathbf{t}_{1,n})$ of the rows of a random matrix A_1. Since T is likely to be linearly independent (as each $\mathbf{t}_{1,j}$ is of dimension $u \gg n$) and since A_1 is uniform, the outcome $\mathbf{a}$ is also uniform.

For nonrepudiation, assume that $\mathcal{D}$ is corrupt and $\mathcal{I}$ is honest, and let $\mathcal{V}_i$ be any verifier that did not abort. If $A_i \cdot \boldsymbol{\rho}_s \neq \mathbf{s}_i$ then even weakly-corrupt $\mathcal{V}_i$ is likely to broadcast a public complaint (the argument is similar to the one used in the previous scheme); Otherwise, $A_i \cdot \boldsymbol{\rho}_s = \mathbf{s}_i$ and all the local authenticators that were sent by $\mathcal{V}_i$ will be consistent. In any case, no honest party rejects due to $\mathcal{V}_i$.

Finally, for agreement, we show that even if $\mathcal{D}$ and $\mathcal{I}$ are corrupt, if the verification phase succeeds, then all the honest receivers are likely to see, for every (possibly weakly-corrupt) verifier $\mathcal{V}_i$, the *same* vote. This is trivially true if $\mathcal{V}_i$ aborts or broadcast a complaint. If this is not the case, then, except with probability of $1/|\mathbb{F}| = \mathsf{negl}(n, \kappa)$ over the choice of the linear combination $\mathbf{t}_{i,j}$, the local equation tested by an honest receiver P_j, which can be written as $\mathbf{t}_{i,j} \cdot (A_i \cdot \boldsymbol{\rho}_s) = \mathbf{t}_{i,j} \cdot (\mathbf{s}_i)$ holds if and only if $\mathcal{V}_i$'s equation $A_i \cdot \boldsymbol{\rho}_s = \mathbf{s}_i$ holds.

Multiple-authenticators variant. It will be useful to consider a variant of the protocol in which every $\mathcal{V}_i$ sends multiple authenticators $(\mathbf{t}^k_{i,j}, \quad \mathbf{a}^k_{i,j} := \mathbf{t}^k_{i,j} \cdot A_i, \quad s^k_{i,j} := \mathbf{t}^k_{i,j} \cdot \mathbf{s}_i)_{k \in [\ell]}$ to every receiver P_j. Accordingly, in the opening phase, P_j will treat any inequality of the form $\mathbf{a}^k_{i,j} \cdot \boldsymbol{\rho}_s \neq s^k_{i,j}$ for some $k \in [\ell]$, as an inconsistency, and will reject accordingly (unless $\mathcal{V}_i$ aborted or issued a public complaint). We think of the random tape of $\mathcal{V}_i$ as composed of ℓ blocks where the kth block consists of all the vectors $(\mathbf{t}^k_{i,j})_{j \in [n]}$. By slightly modifying the parameters u and v, we can securely support this extension even if the adversary partially controls the choice of the linear combinations $\mathbf{t}^k_{i,j}$ of non-honest verifiers $\mathcal{V}_i$. Specifically, if either $\mathcal{D}$ or $\mathcal{I}$ is corrupt, the adversary will be allowed to choose, for every non-honest $\mathcal{V}_i$, all the random tape except for one block that is sampled uniformly at random and remains unknown to the adversary. (If $\mathcal{D}$ and $\mathcal{I}$ are honest then we allow the adversary to pick all the vectors $\mathbf{t}^k_{i,j}$ that are generated by $\mathcal{V}_i$.)

2.1.2 Step II: Emulating the verifiers

We return to the standard model with n parties $P_1, \ldots, P_n$ where at most $t < n/2$ of them are corrupt. We consider the multiple-authenticators variant of the protocol in the simplified model as the *outer protocol*, and we use a virtualization technique to emulate the verifiers in a round-preserving way. For ease of presentation, we assume that the parties have an access to an idealized single-round signature scheme which is also *linearly homomorphic*. That is, if a signer A *signs* to B over secrets $(s_1, \ldots, s_q)$ then B can pick any vector of coefficients $\boldsymbol{\beta}_j = (\boldsymbol{\beta}_j[1], \ldots, \boldsymbol{\beta}_j[q])$ and *privately open* the vector of coefficients and linear combination $(\boldsymbol{\beta}, \quad \sum_{i \in [q]} \boldsymbol{\beta}_j[i] \cdot s_i)$ to some party P_j while certifying that these values were "signed" by A. The opening is *designated* to P_j, and so we use the terms *A signs to B* and *B opens to P_j*. While it may seem paradoxical to make this assumption at this point, we will later see that it can be replaced with a weak form of interactive

signatures, and thus can be easily realized.

The emulation. We identify the verifiers with all subsets of t parties that do not include $\mathcal{D}$ and $\mathcal{I}$, so $m = \binom{n-2}{t}$.[9] In a nutshell, messages that will be sent to the virtual verifier $\mathcal{V}_i$ will be delivered to all the parties in the corresponding committee, and whenever $\mathcal{V}_i$ sends a message, we let each party in the committee send the message as well where the message will be computed with respect to his own independent randomness. We think of $\mathcal{V}_i$ as weakly corrupt if it contains a corrupt party P_j. If all the parties in $\mathcal{V}_i$ are honest, the virtual party will be viewed as an honest party.

In more detail, we think of the distribution phase and verification phase as a three-round protocol. In the first round of the outer protocol, i.e., in the distribution phase, $\mathcal{D}$ sends every $\mathcal{V}_i$ a vector $(A_i, \mathbf{s}_i, \mathbf{r}_i)$. We emulate this step by letting $\mathcal{D}$ pass these values to every P_j in $\mathcal{V}_i$ together with linear private-opening (LPO) signatures. To make sure that $\mathcal{D}$ sent the same vector to all parties in $\mathcal{V}_i$, we let the parties perform a public secure pairwise comparison of those values in the following way. For every $\mathcal{V}_i$ and every P_j and P_k in $\mathcal{V}_i$, we let P_j sign a random pad $\mathbf{r}_{i,j,k}$ to P_k in the first round using the LPO signature, and both parties broadcast $(A_i, \mathbf{s}_i, \mathbf{r}_i) + \mathbf{r}_{i,j,k}$ in the second round. Any inconsistency in the broadcasts implies that the comparison failed, in which case we think of $\mathcal{V}_i$ as an aborting verifier.

In the third round of the outer protocol, i.e., in the second round of the verification phase, every $\mathcal{V}_i$ verifies that $A_i \cdot \boldsymbol{\rho} = \mathbf{s}_i + c \cdot \mathbf{r}_i$, and also generates ℓ authenticators and sends them to the receivers. We emulate this step by letting every P_j in $\mathcal{V}_i$, verify that $A_i \cdot \boldsymbol{\rho} = \mathbf{s}_i + c \cdot \mathbf{r}_i$, using the vector $(A_i, \mathbf{s}_i, \mathbf{r}_i)$ that P_j received from $\mathcal{D}$, and broadcast a public complaint if equality does not hold. If any P_j in $\mathcal{V}_i$ broadcasts a complaint, we think of $\mathcal{V}_i$ as a complaining verifier. We set the number of authenticators generated by every verifier $\mathcal{V}_i$ to be $\ell = t$, and we let every P_k in $\mathcal{V}_i$ generate a random vector $\mathbf{t}^k_{i,j}$ for every P_j, and *privately open* to P_j the values $(\mathbf{t}^k_{i,j}, \quad \mathbf{a}^k_{i,j} := \mathbf{t}^k_{i,j} \cdot A_i, \quad s^k_{i,j} := \mathbf{t}^k_{i,j} \cdot \mathbf{s}_i)$ which are linear combinations of the values on which $\mathcal{D}$ signed to P_k. That is, every party in $\mathcal{V}_i$ generates a single authenticator for P_j. (If P_j rejects the opening of P_k, then P_j simply ignores the authenticator of P_k.)

Analysis. For honest $\mathcal{D}$ and $\mathcal{I}$, we allowed all the verifiers to be non-honest in the outer protocol. Observe that every verifier $\mathcal{V}_i$ contains a corrupt party, that can broadcast a false complaint. However, since a complaining $\mathcal{V}_i$ is equivalent to an aborting $\mathcal{V}_i$ in the outer protocol, this behavior is allowed. In addition, when $\mathcal{D}$ is honest, the unforgeability of the LPO signature implies that for every non-aborting $\mathcal{V}_i$, all authenticators generated by the parties in $\mathcal{V}_i$ are of the correct form $(\mathbf{t}^k_{i,j}, \mathbf{t}^k_{i,j} \cdot A_i, \mathbf{t}^k_{i,j} \cdot \mathbf{s}_i)$. If, in addition, $\mathcal{I}$ is corrupt, then there exists a verifier that contains only honest parties that acts exactly like an honest verifier in the outer protocol, and the adversary has no information about the internal state of this verifier. Moreover, every verifier $\mathcal{V}_i$ contains at least one honest party P_k, for which the authenticators $(\mathbf{t}^k_{i,j})_{j\in[n]}$ are uniformly distributed, and the adversary has no information about the vectors corresponding to the honest parties.

When $\mathcal{D}$ is corrupt, every $\mathcal{V}_i$ contains at least one honest party, and if $\mathcal{V}_i$ did not abort then the honest parties in $\mathcal{V}_i$ agree on the values $(A_i, \mathbf{s}_i, \mathbf{r}_i)$. This means that for every $\mathcal{V}_i$ at least one authenticator is honestly generated. However, the authenticators that the corrupt parties generate are not necessarily consistent with the values $(A_i, \mathbf{s}_i, \mathbf{r}_i)$ that the honest parties hold. To solve this problem, we observe that each pair of parties P_ℓ and P_k in a non-aborting $\mathcal{V}_i$ already publicly

[9] This is the point where complexity becomes exponential in n.

agreed (via broadcast) on the masked values $\mathbf{b}_{i,\ell,k} = (A_i, \mathbf{s}_i, \mathbf{r}_i) + \mathbf{r}_{i,\ell,k}$. Also P_ℓ signed the random pad $\mathbf{r}_{i,\ell,k}$ to P_k, and since $\mathbf{b}_{i,\ell,k}$ is public, this signature is effectively a signature on the "plaintext" $(A_i, \mathbf{s}_i, \mathbf{r}_i)$. In particular, P_k can prove to any party P_j that her authenticator, $(\mathbf{t}^k_{i,j}, \quad \mathbf{a}^k_{i,j} := \mathbf{t}^k_{i,j} \cdot A_i, \quad s^k_{i,j} := \mathbf{t}^k_{i,j} \cdot \mathbf{s}_i)$, is consistent with P_ℓ, by opening to P_j the linear combinations of the random pad $\mathbf{r}_{i,\ell,k}$ that correspond to $\mathbf{t}^k_{i,j}$. (See Section 5 for details.)

Realizing the linear private-opening signatures. So far we assumed that we have an idealized version of the linear private-opening signatures. To replace these signatures we construct information-theoretic linear private-opening signatures. Since the openings are private, we do not require agreement and show that such a scheme can be realized with a single round of distribution, two rounds of verification, and a single round of opening that can be executed in parallel to the second round of verification. Our construction follows the blueprints presented in previous works [PCR08, PCPR09] and reviewed in Section 2.1. (Full details appear in Section 4.) Despite their interactive nature, our signatures can be employed in the above protocol without increasing the round complexity. The distribution phase is executed in the first round, and the verification phase is executed in the second and third round. A failures of an LPO verification, which is a public event, is translated to an "abort" of the corresponding virtual verifier. The final construction of the $(1, 2, 1)$-signature scheme satisfies several additional properties (e.g., linearity and refined versions of openings) that are needed later for the other constructions. See Section 5 for details.

2.2 From VSS to General MPC

At a high level, our (long and winding) road to round-optimal general MPC has few additional steps. First, we note that our VSS scheme satisfies some useful properties for the construction of round-optimal general MPC protocol. We then use those properties to construct (standard) single-input functionalities, and show how to enhance SIF to augmented SIF. Finally, we use augmented single input functionalities for the construction of round-optimal general MPC. We continue with a short explanation about each step.

2.5-rounds VSS. In order to obtain round-optimal protocols, we need to perform operations on the shares *before* the execution of VSS terminated. This idea can be traced back to [ABT19], and was used in several papers on round-optimal MPC [AKP20b, AKP22]. We call the shares that the parties received in the first round of the VSS protocol *tentative shares*, and we observe that in some special cases we can perform linear operations over those shares. In the first special case, a single dealer shared many secrets $s_1, \ldots, s_m$, and the parties securely compute a linear function $\sum_{i\in[m]} \alpha_i \cdot s_i$ of the secrets already in the third round of VSS (see Section 7.1.1). In the second special case there are two dealers $\mathcal{D}_1$ and $\mathcal{D}_2$ that share the secrets s_1 and s_2, respectively, and the parties securely compute the value $s_1 - s_2$ already in the third round (see Section 8.2.1).

Single input functionalities. Recently, [AKP22] implicitly showed a round-preserving transformation from a VSS scheme that allows performing linear operations over the tentative shares into a protocol for single input functionalities. We follow this blueprint and show that it can be adopted to the statistical setting as well. Roughly, the transformation has two steps: (1) Based on VSS, we construct (Section 7.1.2) a three-round protocol for *triple secret sharing* (TSS) that allows a dealer $\mathcal{D}$ to share a triple (a, b, c) among the parties via VSS, and also prove in zero-knowledge that the triple

satisfies $c = ab$; and (2) We use the TSS protocol in order to construct a three-round SIF protocol for a degree-2 functionality by letting the dealer shares its inputs and all the degree-2 monomials (via TSS) and then let the parties compute linear operations over the tentative shares. Since general SIF non-interactively reduces to degree-2 SIF [GIKR02], we get a 3-round SIF protocol.

While this approach is sufficient for the construction of SIF, it is insufficient for the construction of an augmented SIF, since a party P_i cannot convince the other parties of the validity of its outputs. In order to obtain an augmented SIF, we first present a new primitive, called *verifiable sharing and transferring* (VST), that allows a dealer $\mathcal{D}$ to share a secret s among the parties via VSS, while delegating the ability to perform the verifiable opening of s to a designated receiver R (who also gets to learn the secret). The protocol follows similar ideas to VSS, and is presented in Section 7.2. Now given a SIF functionality f, we realize an augmented SIF as follows. Instead of delivering the ith output, $f_i(\mathbf{x})$, privately to P_i, we use SIF to send to all the parties a masked version of the output $f_i(\mathbf{x}) + r_i$, and share the mask r_i via VST while delegating the opening to P_i as the receiver. As a result, P_i learns the output $f_i(\mathbf{x})$ and gets the ability to verifiably open r_i, and let all the parties learn $f_i(\mathbf{x})$.[10]

From Single Input Functionality to General Multiparty Computation To construct a 4-round MPC protocol for general functionalities, we begin with 2-round perfectly secure protocol Π^{sm} against passive adversaries (e.g., [ABT18]), and use a 3-round augmented SIF to force an honest behavior while increasing the total round complexity by only one round. This can be viewed as a new round-efficient statistical realization of the GMW paradigm [GMW87]. Since the underlying protocol Π^{sm} uses private channels, we face a consistency problem: How should Alice convince Bob and Charlie that she behaves well when they have different views on her behavior? To solve this problem GMW eliminate all private communication and pass it, encrypted under public-key encryption, over a broadcast channel, thus providing a common "point of reference" for all the parties. This public-key assumption was carried to round-efficient realizations of the GMW compiler [GLS15, ACGJ18], and was recently relaxed to a symmetric assumption (the existence of commitments) in [AKP21]. We get rid of computational assumptions and present a statistical variant of this round-efficient compiler. For this, we exploit the full power of the augmented SIF protocol and some of its special properties such as the ability to compute the difference between the outputs of two single input functionalities with different dealers. See Section 10.

Organization. In Section 4 we present the construction of linear private-opening interactive signatures, and we use it in Section 5 to construct a (1,2,1)-signatures. Section 6 is devoted to the new VSS scheme, Section 7 contains the related notions of TSS and VST, and Section 8 combines these notions into a single $\mathcal{F}_{\mathsf{sh\text{-}comp}}$ functionality. The augmented SIF protocol appears in Section 9, and the MPC protocol appears in Section 10. A flow-chart of our construction is presented in Figure 1.

[10] In order to construct augmented single input functionalities, we need to execute multiple instances of VSS, TSS and VST, and perform linear operations over the shares. All these calls should be correlated, i.e., for every pair of parties (P_i, P_j), all instances of VSS, TSS, and VST should use the same underlying instance of linear (1,2,1)-signature with P_i in the role of $\mathcal{D}$ and P_j in the role of $\mathcal{I}$. In order to handle this correlation, it will be convenient to capture the execution of *all* VSS, TSS and VST instances by a *single* ideal functionality $\mathcal{F}_{\mathsf{sh\text{-}comp}}$, that will formalize both the task of sharing values by the parties and of computing linear operations over the shares.

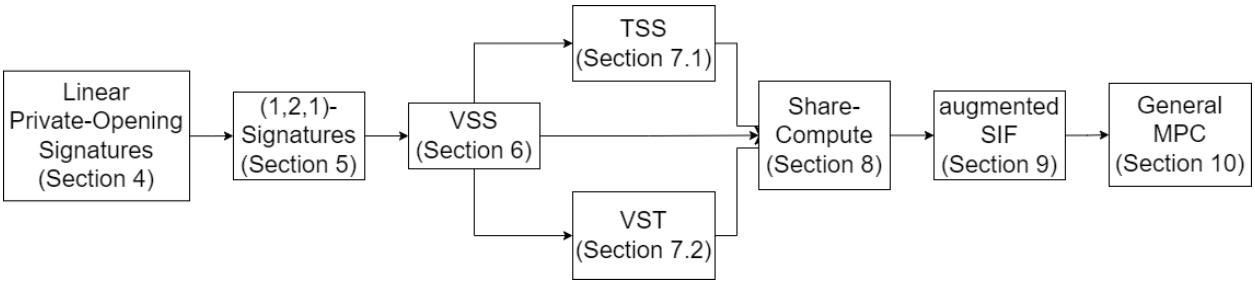

Figure 1: A block diagram of our constructions.

3 Preliminaries

Notation. Throughout, we let n be the number of parties, $t < n/2$ be the number of corrupt parties, κ be the security parameter, and we let $\mathbb{F}$ be a sufficiently large finite field, where we usually think of the size of $\mathbb{F}$ as exponential in n and κ. For a vector $\boldsymbol{\beta}$ of length m, we denote $\boldsymbol{\beta} = (\boldsymbol{\beta}[1], \ldots, \boldsymbol{\beta}[m])$. For two vectors $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{F}^m$, we denote $\boldsymbol{\alpha} \cdot \boldsymbol{\beta} := \sum_{i=1}^{m} \boldsymbol{\alpha}[i] \cdot \boldsymbol{\beta}[i]$.

UC-security. All our results are proved in the framework of universal composability [Can01]. For more information about UC-security, the reader is referred to [Can01]. See also the overview in [AKP21, Appendix A], with computational-indistinguishability replaced with statistical-indistinguishability.

Randomized Encoding The following is taken with minor changes from [App17]. Let X, Y, Z and R be finite sets.

Definition 3.1 (Perfect randomized encoding [IK00, AIK06])**.** *Let $f : X \to Y$ be a function. We say that a function $\hat{f} : X \times R \to Z$ is a* perfect randomized encoding *of f if there exists a pair of randomized algorithms, decoder* dec *and simulator* Sim*, for which the following hold:*

- (Correctness) *For any input $x \in X$, $\Pr_{r \leftarrow R}[\mathsf{dec}(\hat{f}(x; r)) = f(x)] = 1$.*
- (Privacy) *For any $x \in X$ and any computationally-unbounded distinguisher $\mathcal{A}$, $|\Pr[\mathcal{A}(\mathsf{Sim}(f(x))) = 1] - \Pr_{r \leftarrow R}[\mathcal{A}(\hat{f}(x; r)) = 1]| = 0$.*

We refer to the second input of $\hat{f}$ as its random input*.*

Definition 3.2 (Perfect decomposable randomized encoding)**.** *Assume that the function f is an arithmetic function whose input $x = (x_1, \ldots, x_n)$ is a vector of elements of some ring X. We say that a randomized encoding $\hat{f}$ is a* decomposable *randomized encoding if each output of $\hat{f}$ depends on at most a single input x_i. Namely, $\hat{f}$ decomposes to $(\hat{f}_1(x_1; r), \ldots, \hat{f}_n(x_n; r))$, where $\hat{f}_i$ might output several ring elements.*

We will also be interested in the special case of 2-decomposable randomized encoding.

Definition 3.3 (2-decomposable randomized encoding)**.** *Let $f : X \to Y$ be a function where $X = X_1 \times X_2$. A randomized encoding $\hat{f}$ of f is* 2-decomposable *(also known as* 2-party private simultaneous messages*) if $\hat{f}(x_1, x_2; r)$ decomposes to $(\hat{f}_1(x_1; r), \hat{f}_2(x_2; r))$.*

We will always be interested in *efficiently constructible* randomized encoding whose corresponding algorithms $\hat{f}$, dec and Sim are computable by polynomial-size circuits that can be constructed efficiently given a description of f. The following theorem, due to [IK02, CFIK03] shows that such a construction can be obtained for the class of arithmetic circuits with logarithmic depth.

Theorem 3.4. *There exists an efficiently constructible perfect decomposable randomized encoding for the class of polynomial-size arithmetic circuits with logarithmic depth over an arbitrary ring. In particular, there is a compiler that takes a size-S depth-D arithmetic circuit for f and in time $\mathrm{poly}(S, 2^D)$ outputs arithmetic circuits for $\hat{f}$, dec and Sim.*

4 Linear Private-Opening Interactive Signature scheme

In this section we present the linear private-opening interactive signature scheme, that was discussed in Section 2.1.2. We formalize the requirements from the signature scheme by an ideal functionality $\mathcal{F}_{\mathsf{poSig}}$, parameterized by integers m and ℓ, that correspond to the number of inputs that $\mathcal{D}$ signs and the number of linear combinations that every party receives, respectively. We slightly deviate from the structure of the interactive signature as presented in Section 2.1.2 and include the second round of the verification phase within the opening phase, where both run in parallel in the same round. The functionality (Figure 2) as well as the protocol (Figure 3) take care of this change. Recall that we will only require private opening for this signature scheme and this is accounted for in our functionality and protocol.

Functionality $\mathcal{F}_{\mathsf{poSig}}^{m,\ell}$

The functionality receives the set of corrupt parties C.

Distribution phase. $\mathcal{D}$ inputs $\mathbf{s} = (s_1, \ldots, s_m) \in \mathbb{F}^m$. The functionality returns $\mathbf{s}$ to $\mathcal{I}$.

Verification phase.

- *(Inputs)* A corrupt $\mathcal{I}$ inputs a bit $\mathsf{reveal}_{\mathcal{I}} \in \{0,1\}$. For an honest $\mathcal{I}$ set $\mathsf{reveal}_{\mathcal{I}} = 0$.
- *(Outputs)* The functionality returns $\mathsf{reveal}_{\mathcal{I}}$ to $\mathcal{D}$.

Opening phase.

- *($\mathcal{I}$'s inputs)* $\mathcal{I}$ inputs lists of coefficients $(\boldsymbol{\beta}_{i,j})_{i\in[n],j\in[\ell]}$, where $\boldsymbol{\beta}_{i,j} \in \mathbb{F}^m$ for every $i \in [n], j \in [\ell]$. A corrupt $\mathcal{I}$ also inputs bits $\mathsf{abort}_1, \ldots, \mathsf{abort}_n \in \{0,1\}$ and field elements $(z_{i,j})_{i\in[n],j\in[\ell]}$, where $z_{i,j} \in \mathbb{F}$.
- *(Leakage)* The functionality returns $(\boldsymbol{\beta}_{i,j})_{i\in\mathsf{C},j\in[\ell]}$ to the adversary.
- *(Corrupt $\mathcal{D}$'s inputs)* A corrupt $\mathcal{D}$ inputs a bit $\mathsf{reveal}_{\mathcal{D}}$ and values $s'_1, \ldots, s'_m \in \mathbb{F}$. For an honest $\mathcal{D}$ set $\mathsf{reveal}_{\mathcal{D}} = 0$ and $s'_i := s_i$ for all $i \in [m]$.
- *(Outputs)* The functionality returns $b := (\mathsf{reveal}_{\mathcal{I}} \vee \mathsf{reveal}_{\mathcal{D}})$ to all the players. If $b = 1$ then the functionality returns $s'_1, \ldots, s'_m$ to all parties and terminates. Otherwise, the functionality does as follows.
  - *(Honest $\mathcal{D}, \mathcal{I}$)* For every $i \in [n]$, the functionality returns $(\boldsymbol{\beta}_{i,j}, \boldsymbol{\beta}_{i,j} \cdot \mathbf{s})_{j\in[\ell]}$ to P_i.
  - *(Honest $\mathcal{D}$, corrupt $\mathcal{I}$)* For every $i \in [n]$, if $\mathsf{abort}_i = 1$ then the functionality returns $\perp$ to P_i, and otherwise it returns $(\boldsymbol{\beta}_{i,j}, \boldsymbol{\beta}_{i,j} \cdot \mathbf{s})_{j\in[\ell]}$ to P_i.

- *(Corrupt $\mathcal{D}$, honest $\mathcal{I}$)* For every $i \in [n]$, the functionality returns $(\boldsymbol{\beta}_{i,j}, \boldsymbol{\beta}_{i,j} \cdot \mathbf{s})_{j\in[\ell]}$ to P_i.
- *(Corrupt $\mathcal{D}, \mathcal{I}$)* For every $i \in [n]$, if $\mathsf{abort}_i = 1$ then the functionality returns $\perp$ to P_i, and otherwise it returns $(\boldsymbol{\beta}_{i,j}, z_{i,j})_{j\in[\ell]}$ to P_i.

Figure 2: Functionality $\mathcal{F}_{\mathsf{poSig}}^{m,\ell}$

One can realize $\mathcal{F}_{\mathsf{poSig}}^{m,\ell}$ based on a general linear secret-sharing scheme with strong recovery properties (authorized sets can recover the randomness of the dealer). For concreteness, we describe an instantiation of this approach based on Shamir's secret sharing scheme.

Protocol $\mathsf{poSig}^{m,\ell}$

Public parameters: All parties share a statistical security parameter 1^κ.

(Distribution phase or poSig.dis**):** D holds input $\mathbf{s} = (s_1, \ldots, s_m)$ at the beginning of distribution phase, and does the following.

- *(Authentication information)* $\mathcal{D}$ sets $d := n$ and picks random degree-d polynomials $f_1(x), \ldots, f_m(x) \in \mathbb{F}[x]$ conditioned on $f_k(0) = s_k$ for every $k \in [m]$. It further picks random degree-d polynomials $r_1(x), \ldots, r_m(x) \in \mathbb{F}[x]$. $\mathcal{D}$ sends $(f_k(x), r_k(x))_{k\in[m]}$ to $\mathcal{I}$.
- *(Verification information)* For every P_i, $\mathcal{D}$ samples a random non-zero point $\alpha_i \in \mathbb{F} \setminus \{0\}$, computes $f_{k,i} := f_k(\alpha_i)$ and $r_{k,i} := r_k(\alpha_i)$ for every $k \in [m]$, and sends $(\alpha_i, f_{1,i}, \ldots, f_{m,i}, r_{1,i}, \ldots, r_{m,i})$ to P_i.

At the end of the round $\mathcal{I}$ outputs $(f_1(0), \ldots, f_m(0))$.

(Verification phase or poSig.ver**):** $\mathcal{I}$ picks a random non-zero field element $c \in \mathbb{F} \setminus \{0\}$, and broadcasts $(c, f_k(x) + c \cdot r_k(x))_{k\in[m]}$.

Let $(c, d_k(x))_{k\in[m]}$ be the broadcast of $\mathcal{I}$. At the end of the round, if $d_k(x) = f_k(x) + c \cdot r_k(x)$ for every $k \in [m]$ then $\mathcal{D}$ outputs $\mathsf{reveal}_\mathcal{I} = 0$. Otherwise, $\mathcal{D}$ outputs $\mathsf{reveal}_\mathcal{I} = 1$.

(Opening phase or poSig.open**):**

- *(Checking verification test)* $\mathcal{D}$ broadcasts "accept" if $\mathsf{reveal}_\mathcal{I} = 0$, and broadcasts $\mathbf{s}$ otherwise.
- *(Private opening)* The intermediary $\mathcal{I}$ receives the coefficient vectors $(\boldsymbol{\beta}_{i,j})_{i\in[n],j\in[\ell]}$ as inputs, and for every $i \in [n]$, she sends to P_i the values $(\boldsymbol{\beta}_{i,j}, \mathsf{Out}_{i,j}(x))_{j\in[\ell]}$ where $\mathsf{Out}_{i,j}(x)$ is the univariate polynomial $\boldsymbol{\beta}_{i,j}[1] \cdot f_1(x) + \ldots + \boldsymbol{\beta}_{i,j}[m] \cdot f_m(x)$.

(Local computation): Every P_i does the following local computation.

- If $\mathcal{D}$ broadcasted $\mathbf{s}$ in the verification phase, then P_i sets $b = 1$, outputs $\mathbf{s}$ and terminates.
- Otherwise, P_i sets $b = 0$. If either (1) $d_k(\alpha_i) \neq f_{k,i} + cr_{k,i}$ for some $k \in [m]$, or (2) it holds that $\mathsf{Out}_{i,j}(\alpha_i) = \boldsymbol{\beta}_{i,j}[1]f_{1,i} + \ldots + \boldsymbol{\beta}_{i,j}[m]f_{m,i}$ for every $j \in [\ell]$, then P_i outputs $(\boldsymbol{\beta}_{i,j}, \mathsf{Out}_{i,j}(0))_{j\in[\ell]}$.
- Otherwise, P_i outputs a failure symbol $\perp$.

Figure 3: Protocol $\mathsf{poSig}^{m,\ell}$

Notation 1 (Conflicts in private-openings signatures)**.** *We say that $\mathcal{D}$ is* poSign-conflicted *with $\mathcal{I}$ if $\mathcal{D}$ does not broadcasts "accept" in the verification phase (i.e., if $b = 1$). If $\mathcal{D}$ and $\mathcal{I}$ are not conflicted, we say that the opening of $\mathcal{I}$ is* accepted *by P_i if P_i does not output $\perp$. Otherwise, if the opening of $\mathcal{I}$ is not accepted by P_i, we say that the opening was* rejected *or that it* failed*.*

Notation 2 (Specifying inputs to poSig)**.** *When we use* poSig *as a subprotocol of a protocol* Π*, we always execute the distribution phase in the first round, the verification phase in the second round, and the opening phase in the third round. It will be convenient to use the following notational conventions.*

In the distribution phase, the signed values $s_1, \ldots, s_m$ *will have many different contexts. Therefore, it will be convenient to specify each input in its own context, instead of specifying all of them together. Hence, whenever we introduce a new input* s_i *of* $\mathcal{D}$ *to* poSig*, we will say that* $\mathcal{D}$ *executes* $\mathsf{poSig.dis}(\mathcal{D}, \mathcal{I}, s_i)$*, which means that* s_i *is the* i*-th input of* $\mathcal{D}$ *in the execution of* poSig *with* $\mathcal{I}$ *as the intermediary. In fact, we always assume that every input of* poSig *has some predetermined role in the protocol* Π*, so no confusion can arise regarding the order of the inputs, that we usually ignore. For example, for three field elements,* a, b, c*, when we say that* $\mathcal{D}$ *executes* $\mathsf{poSig.dis}(\mathcal{D}, \mathcal{I}, a)$*,* $\mathsf{poSig.dis}(\mathcal{D}, \mathcal{I}, b)$ *and* $\mathsf{poSig.dis}(\mathcal{D}, \mathcal{I}, c)$ *we mean that* $\mathcal{D}$ *executes the distribution phase with* $\mathcal{I}$ *as the intermediary, and with inputs* (a, b, c)*.*

Similarly, instead of specifying all the coefficients that $\mathcal{I}$ *inputs in the opening phase of* poSig *at the same time, it will be convenient to specify the coefficients in their own context. Therefore, whenever we introduce a new vector of coefficients* $\boldsymbol{\beta}$*, we will say that* $\mathcal{I}$ *executes* $\mathsf{poSig.open}(\mathcal{D}, \mathcal{I}, P_i, \boldsymbol{\beta} \cdot \mathbf{s})$ *if* $\mathcal{I}$ *inputs the coefficients* $\boldsymbol{\beta}$ *to* $\mathsf{poSig}^{m,\ell}$ *in* poSig.open *at the role of some* $\boldsymbol{\beta}_{i,j}$*. Again, we assume that every coefficient in the opening phase of* poSig *has some predetermined role in the protocol* Π*, so no confusion can arise regarding the order of the coefficients, that we usually ignore.*

We will sometimes think of $\mathbf{s}$ *as a* matrix M *(say of dimensions* $a \times b$*) rather than a vector, and use the matrix notation* $\mathsf{poSig.open}(\mathcal{D}, \mathcal{I}, P_i, \boldsymbol{\beta} \cdot M)$ *to specify the opening of the* b *linear combinations* $\boldsymbol{\beta} \cdot M_1, \ldots, \boldsymbol{\beta} \cdot M_b$*, where* $\boldsymbol{\beta}$ *is a row vector and* M_j *is the* j*-th column of* M*.*

In some cases we will only be interested in opening a linear combination of a subset of inputs $s_{i_1}, \ldots, s_{i_k}$ *where* $k < m$*. In this case we simplify notation and denote the opening by* $\mathsf{poSig.open}(\mathcal{D}, \mathcal{I}, \boldsymbol{\beta} \cdot (s_{i_1}, \ldots, s_{i_k}))$ *where* $\boldsymbol{\beta} \in \mathbb{F}^k$*. Observe that* $\boldsymbol{\beta}$ *can be naturally translated into a length-*m *vector* $\bar{\boldsymbol{\beta}} \in \mathbb{F}^m$ *so that* $\bar{\boldsymbol{\beta}} \cdot \mathbf{s} = \boldsymbol{\beta}[1]s_{i_1} + \ldots + \boldsymbol{\beta}[k]s_{i_k}$*, by setting* $\bar{\boldsymbol{\beta}}[i_j] = \boldsymbol{\beta}[j]$ *for all* $j \in [k]$*, and* $\bar{\boldsymbol{\beta}}[j] = 0$ *in the remaining entries. In this case it will be convenient to say that the output is* $(\boldsymbol{\beta}, \boldsymbol{\beta} \cdot (s_{i_1}, \ldots, s_{i_k}))$ *instead of saying that the output is* $(\bar{\boldsymbol{\beta}}, \bar{\boldsymbol{\beta}} \cdot \mathbf{s})$*. This extends naturally to the matrix notation as well.*

Notation 3 (On m and ℓ)**.** *When using* poSig *as a subprotocol in a protocol* Π*, the number of inputs and outputs that every instance of* poSig *requires can be deduced from the definition of* Π*. Therefore, in order to simplify notation, we usually ignore input parameter* m *and the output parameter* ℓ *when using* poSig *as a subprotocol.*

In Section C we prove the following theorem.

Theorem 4.1. *Let* κ *be a security parameter, and let* n *be the number of parties and* $t < n/2$ *the number of corrupt parties. Let* m, ℓ *be the parameters of* poSig*, and let* $\mathbb{F}$ *be a field of size greater than* $2^\kappa \cdot 2n^2 m\ell$*. Then protocol* $\mathsf{poSig}^{m,\ell}$ *is a UC-secure implementation of* $\mathcal{F}_{\mathsf{poSig}}^{m,\ell}$ *against a static, active, rushing adversary corrupting up to* t *parties. The complexity of* $\mathsf{poSig}^{m,\ell}$ *is* $\mathrm{poly}(\kappa, n, m, \ell, \log|\mathbb{F}|)$*.*

5 Linear Interactive Signature Scheme

In this section we present our protocol for (1,2,1)-signatures, that we call *linear interactive signature*, discussed in Section 2.1. We formalize the requirements by an ideal functionality $\mathcal{F}_{\mathsf{iSig}}$, parameterized by an input-parameter m and and output-parameter ℓ. As in $\mathcal{F}_{\mathsf{poSig}}$, the functionality consists of a single distribution phase, and a single verification phase. However, for $\mathcal{F}_{\mathsf{iSig}}$ it will be useful to have two opening phases, where in the first opening phase $\mathcal{I}$ can perform both private

and public opening of linear combination of the secrets, and in the second opening phase $\mathcal{I}$ can perform only public opening of a subset of secrets.[11] The functionality is described in Figure 4.

Functionality $\mathcal{F}_{\mathsf{iSig}}^{m,\ell}$

The functionality receives the set of corrupt parties C.

Distribution phase. $\mathcal{D}$ inputs $\mathbf{s} = (s_1, \ldots, s_m) \in \mathbb{F}^m$. The functionality returns $\mathbf{s}$ to $\mathcal{I}$.

Verification phase.

- *(Inputs)* A corrupt $\mathcal{I}$ inputs a bit $\mathsf{reveal}_{\mathcal{I}} \in \{0,1\}$. For an honest $\mathcal{I}$ set $\mathsf{reveal}_{\mathcal{I}} = 0$.
- *(Outputs)* The functionality returns $\mathsf{reveal}_{\mathcal{I}}$ to $\mathcal{D}$.

Opening phase 1 (for private and public opening of linear combination of the secrets).

- *($\mathcal{I}$'s inputs)* $\mathcal{I}$ inputs lists of coefficients, $(\boldsymbol{\beta}_j^{\mathsf{pub}})_{j\in[\ell]}$ and $(\boldsymbol{\beta}_{i,j}^{\mathsf{pri}})_{i\in[n],j\in[\ell]}$, where $\boldsymbol{\beta}_j^{\mathsf{pub}}, \boldsymbol{\beta}_{i,j}^{\mathsf{pri}} \in \mathbb{F}^m$ for every $i \in [n], j \in [\ell]$. A corrupt $\mathcal{I}$ also inputs bits $\mathsf{abort}^{\mathsf{pub}}, \mathsf{abort}_1^{\mathsf{pri}}, \ldots, \mathsf{abort}_n^{\mathsf{pri}} \in \{0,1\}$ and field elements $(z_j^{\mathsf{pub}})_{j\in[\ell]}$ and $(z_{i,j}^{\mathsf{pri}})_{i\in[n],j\in[\ell]}$, where $z_j^{\mathsf{pub}}, z_{i,j}^{\mathsf{pri}} \in \mathbb{F}$ for every $i \in [n], j \in [\ell]$.
- *(Leakage)* The values $(\boldsymbol{\beta}_j^{\mathsf{pub}})_{j\in[\ell]}$ and $(\boldsymbol{\beta}_{i,j}^{\mathsf{pri}})_{i\in\mathsf{C},j\in[\ell]}$ are leaked to the adversary.
- *($\mathcal{D}$'s inputs)* A corrupt $\mathcal{D}$ inputs a bit $\mathsf{reveal}_{\mathcal{D}}$ and field elements $s'_1, \ldots, s'_m$. For an honest $\mathcal{D}$ set $\mathsf{reveal}_{\mathcal{D}} = 0$ and $s'_i := s_i$ for all $i \in [m]$.
- *(Outputs)* The functionality returns $b := (\mathsf{reveal}_{\mathcal{I}} \vee \mathsf{reveal}_{\mathcal{D}})$ to all the players. If $b = 1$ then the functionality returns $s'_1, \ldots, s'_m$ to all parties and terminates. Otherwise, the functionality does as follows.
  - *(Public opening)*
    - *(Honest $\mathcal{D}, \mathcal{I}$)* The functionality returns $(\boldsymbol{\beta}_i^{\mathsf{pub}}, \boldsymbol{\beta}_i^{\mathsf{pub}} \cdot \mathbf{s})_{i\in[\ell]}$ to all parties.
    - *(Honest $\mathcal{D}$, corrupt $\mathcal{I}$)* If $\mathsf{abort}^{\mathsf{pub}} = 1$ then the functionality returns $\perp$ to all parties. Otherwise, the functionality returns $(\boldsymbol{\beta}_i^{\mathsf{pub}}, \boldsymbol{\beta}_i^{\mathsf{pub}} \cdot \mathbf{s})_{i\in[\ell]}$ to all parties.
    - *(Corrupt $\mathcal{D}$, honest $\mathcal{I}$)* The functionality returns $(\boldsymbol{\beta}_i^{\mathsf{pub}}, \boldsymbol{\beta}_i^{\mathsf{pub}} \cdot \mathbf{s})_{i\in[\ell]}$ to all parties.
    - *(Corrupt $\mathcal{D}, \mathcal{I}$)* If $\mathsf{abort}^{\mathsf{pub}} = 1$ then the functionality returns $\perp$ to all parties. Otherwise, the functionality returns $(\boldsymbol{\beta}_i^{\mathsf{pub}}, z_i^{\mathsf{pub}})_{i\in[\ell]}$ to all parties.
  - *(Private opening)* If $\mathcal{I}$ is corrupt and $\mathsf{abort}^{\mathsf{pub}} = 1$ then the functionality returns $\perp$ to all the parties. Otherwise, the functionality does as follows.
    - *(Honest $\mathcal{D}, \mathcal{I}$)* For every $i \in [n]$, the functionality returns $(\boldsymbol{\beta}_{i,j}^{\mathsf{pri}}, \boldsymbol{\beta}_{i,j}^{\mathsf{pri}} \cdot \mathbf{s})_{j\in[\ell]}$ to P_i.
    - *(Honest $\mathcal{D}$, corrupt $\mathcal{I}$)* For every $i \in [n]$, if $\mathsf{abort}_i^{\mathsf{pri}} = 1$ then the functionality returns $\perp$ to P_i, and otherwise it returns $(\boldsymbol{\beta}_{i,j}^{\mathsf{pri}}, \boldsymbol{\beta}_{i,j}^{\mathsf{pri}} \cdot \mathbf{s})_{j\in[\ell]}$ to P_i.
    - *(Corrupt $\mathcal{D}$, honest $\mathcal{I}$)* For every $i \in [n]$, the functionality returns $(\boldsymbol{\beta}_{i,j}^{\mathsf{pri}}, \boldsymbol{\beta}_{i,j}^{\mathsf{pri}} \cdot \mathbf{s})_{j\in[\ell]}$ to P_i.
    - *(Corrupt $\mathcal{D}, \mathcal{I}$)* For every $i \in [n]$, if $\mathsf{abort}_i^{\mathsf{pri}} = 1$ then the functionality returns $\perp$ to P_i, and otherwise it returns $(\boldsymbol{\beta}_{i,j}^{\mathsf{pri}}, z_{i,j}^{\mathsf{pri}})_{j\in[\ell]}$ to P_i.

[11] We could allow $\mathcal{I}$ perform private and public opening of linear combination in the second opening phase as well, however it will not be necessary for the applications.

Opening phase 2 (for public opening of a subset of the secrets). If (1) $b = 1$, or (2) $\mathcal{I}$ is corrupt and $\mathsf{abort}^{\mathsf{pub}} = 1$, then return $\perp$ and terminate.[a]

Otherwise, the adversary inputs a bit abort and values $s''_1, \ldots, s''_m$. We split into cases.

- *(Honest $\mathcal{D}, \mathcal{I}$)* On inputs $0 \leq \mu \leq m$ and $k_1, \ldots, k_\mu \in [m]$ from $\mathcal{I}$, return $(k_i, s_{k_i})_{i\in[\mu]}$ to all parties.
- *(Honest $\mathcal{D}$, corrupt $\mathcal{I}$)* If abort $= 1$ return $\perp$ to all the parties and terminate. Otherwise, on inputs $0 \leq \mu \leq m$ and $k_1, \ldots, k_\mu \in [m]$ from $\mathcal{I}$, return $(k_i, s_{k_i})_{i\in[\mu]}$ to all parties.
- *(Corrupt $\mathcal{D}$, honest $\mathcal{I}$)* On inputs $0 \leq \mu \leq m$ and $k_1, \ldots, k_\mu \in [m]$ from $\mathcal{I}$, return $(k_i, s_{k_i})_{i\in[\mu]}$ to all parties.
- *(Corrupt $\mathcal{D}, \mathcal{I}$)* On inputs $0 \leq \mu \leq m$ and $k_1, \ldots, k_\mu \in [m]$ from $\mathcal{I}$, return $(k_i, s''_{k_i})_{i\in[\mu]}$ to all parties.

[a] This phase takes place after the former opening phase. This is why we have taken into account whether condition (2) (i.e. $\mathcal{I}$ is corrupt and $\mathsf{abort}^{\mathsf{pub}} = 1$) is already the case.

Figure 4: Functionality $\mathcal{F}_{\mathsf{iSig}}^{m,\ell}$

We construct a protocol that realizes every phase in exactly one round. We let $\mathcal{S}$ be the family of all sets that contain t players and do not contain $\mathcal{D}$ and $\mathcal{I}$. The protocol is presented in Figure 5.

Protocol $\mathsf{iSig}^{m,\ell}$

All parties share a statistical security parameter 1^κ. Let $u > tn$ and $v > u \cdot 2^n$.

(Distribution phase or iSig.dis**):** $\mathcal{D}$ holds inputs $\mathbf{s} = (s_1, \ldots, s_m)$ and does as follows.

- *(Authentication information)* For every $k \in [m]$, $\mathcal{D}$ picks a random vector $\mathbf{f}_k \in \mathbb{F}^v$ whose first entry is the secret s_k. It further picks a random vector $\mathbf{r}_k \in \mathbb{F}^v$. $\mathcal{D}$ sends $(\mathbf{f}_k, \mathbf{r}_k)_{k\in[m]}$ to $\mathcal{I}$.
- *(Verification information)* For every set $S \in \mathcal{S}$, $\mathcal{D}$ picks a random $u \times v$ matrix $A^S \in \mathbb{F}^{u\times v}$, computes $\mathbf{f}_k^S := A^S \cdot \mathbf{f}_k \in \mathbb{F}^u$ and $\mathbf{r}_k^S := A^S \cdot \mathbf{r}_k \in \mathbb{F}^u$, and executes $\mathsf{poSig.dis}(\mathcal{D}, P_i, (A^S, (\mathbf{f}_k^S, \mathbf{r}_k^S)_{k\in[m]}))$ with every P_i in S.[a]

The players do as follows.

- *(Exchanging random pads)* For every set $S \in \mathcal{S}$, and every pair (P_i, P_j) in S, P_i picks a random pad $\mathbf{r}_{i,j}^S \in \mathbb{F}^{uv+2mu}$ and executes $\mathsf{poSig.dis}(P_i, P_j, \mathbf{r}_{i,j}^S)$.
- *($\mathcal{I}$'s output)* $\mathcal{I}$ outputs $(\mathbf{f}_k[1])_{k\in[m]}$, where $\mathbf{f}_k[1]$ is the first entry of $\mathbf{f}_k$.

(Verification phase or iSig.ver**):** $\mathcal{I}$ does as follows.

- *(Verification test)* $\mathcal{I}$ picks a random non-zero field element $c \in \mathbb{F}\setminus\{0\}$ and broadcasts $(c, \mathbf{f}_k + c\mathbf{r}_k)_{k\in[m]}$.

The players do as follows.

- *(*poSig *execution)* The players continue with the verification phase of all poSig executions.
- *(Comparing verification information)* For every set $S \in \mathcal{S}$, and every pair (P_i, P_j) in S, P_i broadcast $\mathbf{a}_{i,j}^S = (A^S, (\mathbf{f}_k^S, \mathbf{r}_k^S)_{k\in[m]}) + \mathbf{r}_{i,j}^S$ and $\mathbf{b}_{i,j}^S = (A^S, (\mathbf{f}_k^S, \mathbf{r}_k^S)_{k\in[m]}) + \mathbf{r}_{j,i}^S$.

$\mathcal{D}$ does as follows.

- *(Checking verification test)* Let $(c, \mathbf{d}_k)_{k\in[m]}$ be the broadcast of $\mathcal{I}$. $\mathcal{D}$ sets $\mathsf{reveal}_\mathcal{I} = 0$ if $\mathbf{d}_k = \mathbf{f}_k + c\mathbf{r}_k$ for every $k \in [m]$, and $\mathsf{reveal}_\mathcal{I} = 1$ otherwise.

(Opening phase 1 or $\mathsf{iSig.open}_1$): The dealer does as follows.

- $\mathcal{D}$ broadcasts "accept" if $\mathsf{reveal}_{\mathcal{I}} = 0$, and broadcasts $\mathbf{s}$ otherwise.

In addition, for every $S \in \mathcal{S}$ every P_i in S does as follows.

- *(Checking verification test)* If $A^S \cdot \mathbf{d}_k \neq \mathbf{f}_k^S + c\mathbf{r}_k^S$ for some $k \in [m]$ then P_i broadcasts "not equal".
- *(Sending verification information)* P_i does as follows for every player P_j (not necessarily in S).
  - P_i samples a row vector $\mathbf{t}_{i,j}^S \in \mathbb{F}^u$.
  - P_i executes $\mathsf{poSig.open}(\mathcal{D}, P_i, P_j, \mathbf{t}_{i,j}^S \cdot A^S)$. (Here we think of P_i opening v linear combinations to P_j, denoted $\mathbf{t}_{i,j}^S \cdot A^S[1], \ldots, \mathbf{t}_{i,j}^S \cdot A^S[v]$, where $A^S[k]$ is the k-th column of A^S.)
  - For every $P_{i'}$ in S, P_i executes $\mathsf{poSig.open}(P_{i'}, P_i, P_j, \mathbf{t}_{i,j}^S \cdot \mathbf{r}_{i',i}^S[A^S])$. (Here, $\mathbf{r}_{i',i}^S[A^S]$ is the part of $\mathbf{r}_{i',i}^S$ that pads A^S, where we think of $\mathbf{r}_{i',i}^S[A^S]$ as a $u \times v$ matrix.)
  - P_i executes $\mathsf{poSig.open}(\mathcal{D}, P_i, P_j, \mathbf{t}_{i,j}^S \cdot \mathbf{f}_k^S)$ for every $k \in [m]$.
  - For every P_i' in S and every $k \in [m]$, P_i executes $\mathsf{poSig.open}(P_{i'}, P_i, P_j, \mathbf{t}_{i,j}^S \cdot \mathbf{r}_{i',i}^S[\mathbf{f}_k^S])$. (Here, $\mathbf{r}_{i',i}^S[\mathbf{f}_k^S]$ is the part of $\mathbf{r}_{i',i}^S$ that pads $\mathbf{f}_k^S$, where we think of $\mathbf{r}_{i',i}^S[\mathbf{f}_k^S]$ as a vector of length u.)

$\mathcal{I}$, that holds inputs $(\boldsymbol{\beta}_j^{\mathsf{pub}})_{j\in[\ell]}$ and $(\boldsymbol{\beta}_{i,j}^{\mathsf{pri}})_{i\in[n],j\in[\ell]}$, does as follows.

- *(Public opening)* $\mathcal{I}$ broadcasts $(\boldsymbol{\beta}_j^{\mathsf{pub}}, \sum_{k\in[m]} \boldsymbol{\beta}_j^{\mathsf{pub}}[k] \cdot \mathbf{f}_k)_{j\in[\ell]}$.
- *(Private opening)* $\mathcal{I}$ sends $(\boldsymbol{\beta}_{i,j}^{\mathsf{pri}}, \sum_{k\in[m]} \boldsymbol{\beta}_{i,j}^{\mathsf{pri}}[k] \cdot \mathbf{f}_k)_{j\in[\ell]}$ to P_i via private message.

Each party P_j does the following local computation.

- If $\mathcal{D}$ broadcasted $\mathbf{s}$ in the verification phase, then all parties output $\mathbf{s}$ and terminate.
- Let $\mathcal{G} \subseteq \mathcal{S}$ be the set containing all sets $S \in \mathcal{S}$ so that (1) for every P_i in S, $\mathcal{D}$ is not *poSign-conflicted* with P_i in $\mathsf{poSig}(\mathcal{D}, P_i, (A^S, (\mathbf{f}_k^S, \mathbf{r}_k^S)_{k\in[m]}))$, (2) for every P_i and $P_{i'}$ in S it holds that $\mathbf{a}_{i,i'}^S = \mathbf{b}_{i',i}^S$ (3) for every P_i and $P_{i'}$ in S it holds that P_i is not *poSign-conflicted* with $P_{i'}$ in $\mathsf{poSig}(P_i, P_{i'}, \mathbf{r}_{i,i'}^S)$, and (4) for every P_i in S it holds that P_i did not broadcast "not equal" in Open Phase 1.[b]
- For every $S \in \mathcal{G}$, P_j defines a set $G^{S,j}$ that contains all player P_i in S so that:
  1. P_j accepted the openings of P_i in the below instances, and P_i uses the same vector $\mathbf{t}_{i,j}^S$ in all of them:
     - $\mathsf{poSig.open}(\mathcal{D}, P_i, P_j, \mathbf{t}_{i,j}^S \cdot A^S)$,
     - $(\mathsf{poSig.open}(P_{i'}, P_i, P_j, \mathbf{t}_{i,j}^S \cdot \mathbf{r}_{i',i}^S[A^S]))_{P_{i'}\in S}$,
     - $(\mathsf{poSig.open}(\mathcal{D}, P_i, P_j, \mathbf{t}_{i,j}^S \cdot \mathbf{f}_k^S))_{k\in[m]}$ and
     - $(\mathsf{poSig.open}(P_{i'}, P_i, P_j, \mathbf{t}_{i,j}^S \cdot \mathbf{r}_{i',i}^S[\mathbf{f}_k^S]))_{P_{i'}\in S, k\in[m]}$.

     Denote the outputs by $\boldsymbol{\alpha}_i^{S,j}$, $(\boldsymbol{\rho}_{i',i}^{S,j})_{P_{i'}\in S}$, $(\phi_{i,k}^{S,j})_{k\in[m]}$ and $(\gamma_{i',i,k}^{S,j})_{P_{i'}\in S, k\in[m]}$, respectively.
  2. For every $P_{i'}$ in S it holds that $\boldsymbol{\alpha}_i^{S,j} + \boldsymbol{\rho}_{i',i}^{S,j} = \mathbf{t}_{i,j}^S \cdot \mathbf{b}_{i,i'}^S[A^S]$ and $\phi_{i,k}^{S,j} + \gamma_{i',i,k}^{S,j} = \mathbf{t}_{i,j}^S \cdot \mathbf{b}_{i,i'}^S[\mathbf{f}_k^S]$ for every $k \in [m]$.[c]
- *(Acceptance of public opening)* If there exists a set $S \in \mathcal{G}$, a player $P_i \in G^{S,j}$, and an index $j' \in [\ell]$ so that

$$\boldsymbol{\alpha}_i^{S,j} \cdot \sum_{k\in[m]} \boldsymbol{\beta}_{j'}^{\mathsf{pub}}[k] \cdot \mathbf{f}_k \neq \sum_{k\in[m]} \boldsymbol{\beta}_{j'}^{\mathsf{pub}}[k] \cdot \phi_{i,k}^{S,j}$$

  then P_j outputs $\perp$ as the public output. Otherwise, P_j outputs the first entry of $\sum_{k\in[m]} \boldsymbol{\beta}_{j'}^{\mathsf{pub}}[k] \cdot \mathbf{f}_k$ for every $j' \in [\ell]$.

- *(Acceptance of private opening)* If the public output is $\perp$ then so is the private ouptut. Otherwise, if there exists a set $S \in \mathcal{G}$, a player $P_i \in G^{S,j}$, and an index $j' \in [\ell]$ so that

$$\boldsymbol{\alpha}_i^{S,j} \cdot \sum_{k\in[m]} \boldsymbol{\beta}^{\mathsf{pri}}_{j,j'}[k] \cdot \mathbf{f}_k \neq \sum_{k\in[m]} \boldsymbol{\beta}^{\mathsf{pri}}_{j,j'}[k] \cdot \phi^{S,j}_{i,k}$$

  then P_j outputs $\perp$ as the private output. Otherwise, P_j outputs the first entry of $\sum_{k\in[m]} \boldsymbol{\beta}^{\mathsf{pri}}_{j,j'}[k] \cdot \mathbf{f}_k$ for every $j' \in [\ell]$.

(Opening phase 2 or $\mathsf{iSig.open}_2$): On input $\mu, k_1, \ldots, k_\mu \in [m]$, $\mathcal{I}$ broadcasts $(k_i, \mathbf{f}_{k_i})_{i\in[\mu]}$.

Every P_j does the following local computation. If $\mathcal{D}$ broadcasted $\mathbf{s}$ in the verify phase, or if the public output in $\mathsf{iSig.open}_1$ was $\perp$, then P_j outputs $\perp$ and terminate. Otherwise, If there exists a set $S \in \mathcal{G}$ and a player $P_{i'} \in G^{S,j}$ so that $\boldsymbol{\alpha}^{S,j}_{i'} \cdot \mathbf{f}_{k_i} \neq \phi^{S,j}_{i',k_i}$ for some $i \in [\mu]$ then P_j outputs $\perp$. Otherwise, P_j outputs $(k_i, \mathbf{f}_{k_i}[1])_{i\in[\mu]}$.

[a] Here we think of $(A^S, (\mathbf{f}^S_k, \mathbf{r}^S_k)_{k\in[m]})$ as a vector in $\mathbb{F}^{uv+2mu}$.
[b] Note that all honest parties agree on the set $\mathcal{G}$.
[c] Note that the set $G^{S,j}$ depends on the private view of P_j, and may differ for two honest parties.

Figure 5: Protocol $\mathsf{iSig}^{m,\ell}$

Notation 4. *We say that $\mathcal{D}$ is* iSign-conflicted *with $\mathcal{I}$ if $\mathcal{D}$ does not broadcasts "accept" in the verification phase (i.e., if $b = 1$). If $\mathcal{D}$ and $\mathcal{I}$ are not conflicted, we say that the opening of $\mathcal{I}$ is* accepted *by P_i if P_i does not output $\perp$. Otherwise, if the opening of $\mathcal{I}$ is not accepted by P_i, we say that the opening was* rejected *or that it* failed.

When we use iSig *as a subprotocol of a protocol Π, we always execute the distribution phase in the first round, the verification phase in the second round, the first opening phase in the third round, and the second opening phase in the fourth round (if exists). We therefore follow the same conventions in Notation 2 and Notation 3.*

Notation 5 (Private versus Public Opening)**.** *When we invoke the private opening of* $\mathsf{iSig.open}_1$*, we invoke the protocol with the receiver's identity along with the identities of $\mathcal{D}$ and $\mathcal{I}$ i.e.* $\mathsf{iSig.open}_1(\mathcal{D}, \mathcal{I}, R, v)$*. When no receiver is specified, it means we invoke the public opening of* $\mathsf{iSig.open}_1$*.*

Notation 6. *It will be convenient to consider the case where $\mathcal{D}$ and $\mathcal{I}$ are the same party. In this case, in the distribution and verification phase there is no communication. In opening phase 1 we let $\mathcal{I}$ broadcast $(\boldsymbol{\beta}^{\mathsf{pub}}_j, \boldsymbol{\beta}^{\mathsf{pub}}_j \cdot \mathbf{s})_{j\in[\ell]}$ and send $(\boldsymbol{\beta}^{\mathsf{pri}}_{i,j}, \boldsymbol{\beta}^{\mathsf{pri}}_{i,j} \cdot \mathbf{s})_{j\in[\ell]}$ to P_i, and we assume that P_i always accepts the opening. Similarly, in open phase 2 we let $\mathcal{I}$ broadcast $(k_i, s_i)_{i\in[\mu]}$, and we assume that all the players accept the opening.*

Throughout, we assume that $\mathbb{F}$ is a sufficiently large field of size $\exp(n, \kappa, \ell, m)$, that allows the execution of poSig as required by iSig. In Section D we prove the following theorem.

Theorem 5.1. *Let κ be a security parameter, and let n be the number of parties and $t < n/2$ the number of corrupt parties. Let m, ℓ be the parameters of* poSig*, and let $\mathbb{F}$ be a field of sufficiently large size as a function of (n, κ, ℓ, m). Then protocol* $\mathsf{iSig}^{m,\ell}$ *is a UC-secure implementation of $\mathcal{F}^{m,\ell}_{\mathsf{iSig}}$ against a static, active, rushing adversary corrupting up to t parties. The complexity of* $\mathsf{iSig}^{m,\ell}$ *is* $\mathrm{poly}(\kappa, 2^n, m, \ell, \log|\mathbb{F}|)$*.*

6 Verifiable Secret Sharing

In this section we present a protocol for verifiable secret sharing with three rounds of sharing and a single round of opening. We start by discussing the underlying secret sharing scheme.

The secret-sharing scheme. We consider an extension of the classical Shamir's secret sharing scheme, based on *symmetric bivariate polynomials $F(x, y)$ of degree at most t in each variable*. It will be convenient to think of the shares as an $n \times n$ symmetric matrix where the (i, j)-th share is $F(j, i)$, for $i, j \in [n]$. It is well known (see Fact B.3) that if $F(x, y)$ is chosen uniformly at random conditioned on $F(0, 0) = s$, then every t rows of the matrix reveal no information about s, but any $t + 1$ rows fully determine the matrix (see Fact B.2).

From secret sharing to VSS. We note that the task of constructing a VSS protocol can be reduced to the construction of a two-phase protocol vss with the following guarantees: (1) in the first phase, the sharing phase, the dealer P_d shares a polynomial $F(x, y)$ among the parties, so that every P_i learns the univariate polynomial $f_i(x) := F(x, i)$ and nothing else, and (2) in the second phase, the reconstruction phase, every P_i can publicly open $f_i(x)$, and nothing else. Indeed, if P_d picks $F(x, y)$ at random conditioned on $F(0, 0) = s$, then in sharing phase the adversary can see at most t shares, so privacy is guaranteed. In addition, in the reconstruction phase, since there are $n-t \geq t+1$ honest parties, $F(x, y)$ will be revealed, so all parties can recover $F(0, 0)$, which means that correctness and commitment hold.

The rest of the section is organised as follows. In Section 6.1 we present the construction of our vss protocol, that requires three round of sharing and only a single round of reconstruction. In Section 6.2 we analyse protocol vss and prove that it indeed implies a VSS protocol. In Section 6.3 we show that our VSS protocol is linearly-homomorphic, and in Section 6.4 we present the notion of *tentative shares* and provide some properties.

6.1 The Construction

In the first round of the protocol, the dealer P_d signs the values $(F(j, i))_{j\in[n]}$ for every P_i. In addition, every P_i sends to every P_j a random pad $r_{i,j}$, that will be used for a pairwise-consistency test in the second round, together with a signature. P_i also sends $r_{i,j}$ to P_d with a signature. In the second round, the parties execute the pairwise consistency test, where every P_i broadcasts $a_{i,j} = F(j, i)+r_{i,j}$ and $b_{i,j} = F(j, i)+r_{j,i}$ for every $j \in [n]$, and P_d broadcasts $a^d_{i,j} = F(j, i)+r_{i,j}$ for every $i, j \in [n]$. Observe that, because of the random pads, this step reveals no information about the shares of the honest parties. In the third round, if P_d is conflicted with P_i (i.e., if $a^d_{i,j} \neq a_{i,j}$ for some $j \in [n]$) then P_d and P_i open all the signatures that correspond to the shares of P_i. Similarly, if P_i and P_j are conflicted (i.e., if $a_{i,j} \neq b_{j,i}$ or $a_{j,i} \neq b_{i,j}$), then P_i and P_j open all the signatures that correspond to the (i, j)-th share. We mention that parties also become conflicted if they are *iSign-conflicted* . If the dealer is honest then every two honest parties are consistent, so no information about their shares is revealed to the adversary. Since we use vss as a subprotocol in our general MPC protocol, we also allow the parties to inputs flags, that indicate whether they are conflicted due to some *external reason*.

At the end of the execution the parties first verify that, given the opened signatures, no player has a clear malicious behaviour. Every party found to be corrupt is added to the set of bad parties

B. If P_d is found to be corrupt then P_d is discarded. Otherwise, for every P_i conflicted with P_d, the shares of P_i will be computed based on the public openings of P_d and P_i, and will be known to all the parties. Similarly, if P_i and P_j are conflicted, then the (i,j)-th share will be computed based on the public openings of P_i and P_j, and will be known to all the parties. However, if P_i and P_j are not conflicted, then the (i,j)-th share is not known to all the parties, and it is set to be $b_{i,j} - r_{j,i}$, where $b_{i,j}$ is a public value, and P_i holds a signature of $r_{j,i}$ from P_j. In the analysis we show that if P_d is not discarded, then the shares of the honest parties are consistent with some polynomial $F'(x,y)$, so that $F'(x,y) = F(x,y)$ if the dealer is honest. In addition, the shares of every corrupt P_i are either consistent with $F'(x,i)$, or correspond to a polynomial of degree more than t, so that in the opening phase the parties will turn the shares into an erasure.

We continue with a formal description of the protocol in Figure 6.

Protocol vss

Inputs: The dealer P_d has a symmetric bivariate polynomial of degree t in each variable $F(x,y)$. All parties share a statistical security parameter 1^κ.

Round 1 (share and signature generation):

- P_d lets $f_i(x) := F(x,i)$. P_d executes $\mathsf{iSig.dis}(P_d, P_i, f_i(j))$ for every $i,j \in [n]$. Let every party P_i receive $f'_i(x)$ via the signatures.
- Every P_i picks a random pad $r_{i,j} \in \mathbb{F}$ for each P_j and executes $\mathsf{iSig.dis}(P_i, P_j, r_{i,j})$ and $\mathsf{iSig.dis}(P_i, P_d, r_{i,j})$. Let the value received by P_j and P_d be denoted as $r'_{i,j}$ and $r^d_{i,j}$.

Round 2 (Pairwise consistency): The parties do the following.

- P_i broadcasts $a_{i,j} = f'_i(j) + r_{i,j}$ and $b_{i,j} = f'_i(j) + r'_{j,i}$ for every $j \in [n]$.
- P_d broadcasts $a^d_{i,j} = f_i(j) + r^d_{i,j}$ for every $i,j \in [n]$.
- Run $\mathsf{iSig.ver}(P_d, P_i, f_i(j))$, $\mathsf{iSig.ver}(P_i, P_j, r_{i,j})$ and $\mathsf{iSig.ver}(P_i, P_d, r_{i,j})$ for every $i,j \in [n]$.
- If P_i received $f'_i(x)$ which is not a polynomial of degree t, then P_i broadcasts a public complaint.

At the end of the round, the parties do the following local computation.

- If P_d *iSign-conflicted* with P_i OR P_i complained against P_d in Round 2 OR $a_{i,j} \neq a^d_{i,j}$ for some $j \in [n]$, then we say that P_d is *internally-vss-conflicted* with P_i. (This state is known to P_d at the end of the round.)
- If P_i *iSign-conflicted* with P_j OR $a_{i,j} \neq b_{j,i}$ OR $a_{j,i} \neq b_{i,j}$ then we say that P_i is *internally-vss-conflicted* with P_j. (This state is known to P_i at the end of the round.)
- If P_i *iSign-conflicted* with P_d then we say that P_i is *internally-vss-conflicted* with P_d. (This state is known to P_i at the end of the round.)

Round 3 (Resolution by signature opening): Every P_i inputs a flag-bit flag_i together with flag-bits $(\mathsf{flag}_{i,j})_{j\in[n]}$. In addition, P_d inputs flags $\mathsf{flag}^d_1, \ldots, \mathsf{flag}^d_n$.[a] The parties do the following.

- Every party broadcasts all of its flags.

- If P_d is *internally-vss-conflicted* with P_i, or $\mathsf{flag}_i^d = 1$, then we say that P_d is *vss-conflicted* with P_i. P_d executes $\mathsf{iSig.open}_1(P_i, P_d, r_{i,k}^d)$ and $\mathsf{iSig.open}_1(P_k, P_d, r_{k,i}^d)$ for all $k \in [n]$. Let W be the set of parties that P_d are *vss-conflicted* with. (Observe that all parties agree on W at the end of the round since *iSign-conflicted* become public at the end of the round.)
- If P_i *internally-vss-conflicted* with P_j OR $\mathsf{flag}_{i,j} = 1$, then we say that P_i is *vss-conflicted* with P_j. P_i executes $\mathsf{iSig.open}_1(P_d, P_i, f_i'(j))$ and $\mathsf{iSig.open}_1(P_j, P_i, r_{j,i}')$. (Observe that all parties agree on the conflicts at the end of the round since *iSign-conflicted* become public at the end of the round.)
- If P_i *internally-vss-conflicted* with P_d OR $\mathsf{flag}_i = 1$ then we say that P_i is *vss-conflicted* with P_d. P_i executes $\mathsf{iSig.open}_1(P_d, P_i, f_i'(k))$ and $\mathsf{iSig.open}_1(P_k, P_i, r_{k,i}')$ for all $k \in [n]$. Let W' be the set of all P_i that are *vss-conflicted* with P_d and are not in W. (Observe that all parties agree on W' at the end of the round since *iSign-conflicted* become public at the end of the round.)
- Complete $\mathsf{iSig.open}_1(P_d, P_i, f_i(j))$, $\mathsf{iSig.open}_1(P_i, P_j, r_{i,j})$ and $\mathsf{iSig.open}_1(P_i, P_d, r_{i,j})$ for every $i, j \in [n]$.

(Local computation):

Initialize a set $B := \varnothing$ of bad players. Add a party P_i to B, if one of the following is true:

- There exists an instance of public opening $\mathsf{iSig.open}_1(*, P_i, *)$ that failed (but did not have *iSign-confliction*), or P_i did not open the correct value (i.e. it used the wrong linear coefficients βs).
- There is some P_j so that (1) P_d and P_j are not *iSign-conflicted* with P_i, (2) P_i executed $\mathsf{iSig.open}_1(P_d, P_i, f_i'(j))$ and $\mathsf{iSig.open}_1(P_j, P_i, r_{j,i}')$ and opened the values $f_i'(j)$ and $r_{j,i}'$, and (3) it holds that $b_{i,j} \neq f_i'(j) + r_{j,i}'$.

Remove from W and W' all parties P_i that are in B.

P_d is **discarded** if one of the following holds.

- *(Failed opening)* There exists an instance of public opening $\mathsf{iSig.open}_1(*, P_d, *)$ that failed (but did not have *iSign-conflict*), or P_d did not open the correct value (i.e. it used the wrong linear coefficients βs).
- *(W verification I)* There exists $P_i \in W$ that is not *iSign-conflicted* with P_d, and the polynomial defined by $(a_{i,j}^d - r_{i,j}^d)_{j \in [n]}$, denoted $f_i^d(x)$, is not of degree-t.
- *(W verification II)* There exists $P_i \in W$ for which the following holds. Consider the polynomial defined by $(a_{k,i}^d - r_{k,i}^d)$ for every P_k not *iSign-conflicted* with P_d, and denote it by $g_i^d(x)$. Then $g_i^d(x)$ is not of degree-t. If P_i is *iSign-conflicted* with P_d, define $f_i^d(x) := g_i^d(x)$.
- *(W verification III)* There exists $P_i \in W$ that is not *iSign-conflicted* with P_d, for which $f_i^d(x) \neq g_i^d(x)$.
- *(W' verification)* There exists $P_i \in W'$ such that the opening $(\mathsf{iSig.open}_1(P_d, P_i, f_i'(k)))_{k \in [n]}$ was accepted, but $(f_i'(k))_{k \in [n]}$ does not lie on a degree-t polynomial. Otherwise, denote the polynomial by $f_i^d(x)$.
- *(Pairwise consistency of polynomials in $W \cup W'$)* There exist $P_i, P_j \in W \cup W'$ such that $f_i^d(j) \neq f_j^d(i)$.
- *(Pairwise Consistency of polynomials)* There exist P_i, P_j that P_d is not *iSign-conflicted* with, so that $\mathsf{iSig.open}_1(P_d, P_i, f_i'(j))$ and $\mathsf{iSig.open}_1(P_d, P_j, f_j'(i))$ resulted in successful opening, and $f_i'(j) \neq f_j'(i)$.
- *(Pairwise Consistency between $W \cup W'$ and its complement I)* There exists $P_i \in W \cup W'$ and some P_j with a successful opening in $\mathsf{iSig.open}_1(P_d, P_j, f_j'(i))$ so that $f_j'(i) \neq f_i^d(j)$.
- *(Pairwise Consistency between $W \cup W'$ and its complement II)* There exists $P_i \in W \cup W'$ and some P_j where P_i is not *iSign-conflicted* with P_d in $\mathsf{iSig}(P_i, P_d, r_{i,j}^d)$, and $\mathsf{iSig.open}_1(P_i, P_d, r_{i,j}^d)$ was successfully opened, so that $a_{i,j}^d \neq f_i^d(j) + r_{i,j}^d$.

- *(Pairwise Consistency between $W \cup W'$ and its complement III)* There exists $P_i \in W \cup W'$ so that $\mathsf{iSig.open}_1(P_j, P_d, r^d_{j,i})$ was successfully opened for some P_j and $a^d_{j,i} \neq f^d_i(j) + r^d_{j,i}$.

If P_d is discarded then the parties output $\perp$. Otherwise, the shares are computed as follows.

For every $P_i, P_j \notin B$, the parties compute the (i,j)-th share as follows:

- *($P_i \in W \cup W'$)* The parties set the (i,j)-th share to be the public value $f^d_i(j)$.
- *($P_i \notin W \cup W'$)* We split into cases.
  - For $P_j \in W \cup W'$ the parties set the (i,j)-th share to be the public value $f^d_j(i)$.
  - Consider $P_j \notin W \cup W'$ that did not *vss-conflict* with P_i.
    If P_i is not *vss-conflicted* with P_j the parties set the (i,j)-th share to be $b_{i,j} - r_{j,i}$, where $b_{i,j}$ is public, and P_i can open $\mathsf{iSig.open}_2(P_j, P_i, r_{j,i})$ since P_j is not *iSign-conflicted* with P_i.
    If P_i is *vss-conflicted* with P_j then P_i opened $f'_i(j)$ and $r'_{j,i}$ and it holds that $b_{i,j} = f'_i(j) + r'_{j,i}$ (or otherwise $P_i \in B$). The parties set the (i,j)-the share to be $f'_i(j)$.
  - Consider $P_j \notin W \cup W'$ that *vss-conflicted* with P_i.
    If P_i is not *vss-conflicted* with P_j, then P_j opened $f'_j(i)$ and $r'_{i,j}$ and it holds that $b_{j,i} = f'_j(i) + r'_{i,j}$ (or otherwise $P_j \in B$). The parties set the (i,j)-th share to be $f'_j(i)$.
    If P_i is *vss-conflicted* with P_j then P_i opened $f'_i(j)$ and P_j opened $f'_j(i)$ and $f'_i(j) = f'_j(i)$ (or otherwise P_d is discarded). The parties set the (i,j)-th share to be $f'_j(i)$.

[a] We allow the parties to input flags, that can make them conflicted with other parties in the vss protocol due to an *external reason*. This will be useful when vss is used as a sub-protocol, and we would like to reflect conflicts in the outer protocol in the vss protocol as well.

Figure 6: Protocol vss

For completeness, we also provide the reconstruction protocol, that requires one round.

Protocol vrec

Round 1 (Reconstruction): Every $P_i \notin B \cup W \cup W'$ executes $\mathsf{iSig.open}_2(P_j, P_i, r'_{j,i})$ for every $P_j \notin B \cup W \cup W'$ so that P_j is not *vss-conflicted* with P_i and P_i is not *vss-conflicted* with P_j.

Local Computation Party P_i initializes the set of accepted parties $A := W \cup W'$. P_i adds $P_j \notin B \cup W \cup W'$ to A if all the following are true.

1. For every $P_k \notin B \cup W \cup W'$ so that P_k is not *vss-conflicted* with P_j and P_j is not *vss-conflicted* with P_k, the opening $\mathsf{iSig.open}_2(P_k, P_j, r'_{k,j})$ is accepted. Let the (j,k)-th share be $b_{j,k} - r'_{k,j}$.
2. The interpolation on all (j,k)-th shares, $P_k \notin B$, results in a degree-t polynomial, denoted $f_j(x)$.

For every $P_j \in W \cup W'$ let $f_j(x) := f^d_j(x)$. Use $f_j(x)$ of every P_j in A to reconstruct the unique symmetric bivariate polynomial $F(x,y)$. Output $F(0,0)$.

Figure 7: Protocol vrec

6.2 Analysis

Protocol vss can be used to implement a verifiable secret sharing scheme with three rounds for the sharing and one round for the opening, as per Definition A.1, in the following way.[12] Upon holding a secret $s \in \mathbb{F}$, the dealer P_d picks a random symmetric bivariate polynomial $F(x,y)$ of degree at most t in each variable conditioned on $F(0,0) = s$. The sharing phase consists of an execution of vss with input $F(x,y)$ to P_d, and where all the flags are set to 0. For the opening phase, the parties execute vrec. (If P_d is discarded in the sharing phase, simply set the secret to be 0.)

For completeness, we continue with an analysis of the properties of vss in the $\mathcal{F}_{\mathsf{iSig}}$-hybrid model.

Privacy. A simulator is provided in Appendix E.1, and its security is implicitly proved as part of the security-proof of protocol sh-comp. Since the privacy of protocol vss follows from known facts about bivariate polynomials (see Fact B.3) and analysis similar to previous works (see [KPR10, Section 5]), we skip a full analysis here.

Correctness. Assume that P_d is honest, and let $F(x,y)$ be the input of P_d. It will be useful to prove correctness even for the case where the honest parties might raise flags in the third round. Observe that P_d is not discarded in the execution of vss, and that none of the honest parties are in B. We continue with the following claim.

Claim 6.1. *For every corrupt $P_i \notin B$, and every honest P_j the (i,j)-th share, as well as the (j,i)-th share, are $F(i,j) = F(j,i)$.*

Proof. We split into cases.

- Assume that P_d is *vss-conflicted* with P_i. Then P_d opens $r^d_{k,i}$ for every P_k which is not *iSign-conflicted* with P_d. Since there are at least $n - t \geq t + 1$ honest parties that are not *iSign-conflicted* with P_d, and since $a^d_{k,i} = F(i,k) + r^d_{k,i}$ for every P_k not *iSign-conflicted* with P_d, then $g^d_i(x) = F(x,i)$.

  If P_i is *iSign-conflicted* with P_d then $f^d_i(x) := g^d_i(x)$, so the (i,j)-th share and the (j,i)-th share are set to be $f^d_i(j) = F(j,i) = F(i,j)$, as required.

  If P_i is not *iSign-conflicted* with P_d then P_d also opens $(r^d_{i,k})_{k\in[n]}$. Since $a^d_{i,k} = F(k,i) + r^d_{i,k}$ for every $k \in [n]$, then $f^d_i(x) = F(x,i)$ so the (i,j)-th share and the (j,i)-th share are set to be $f^d_i(j) = F(j,i) = F(i,j)$, as required.

- Assume that P_d is not *vss-conflicted* with P_i, but P_i is *vss-conflicted* with P_d. In this case P_i opens $f'_i(k) = F(k,i)$ for every $k \in [n]$ (as well as $r'_{k,i}$ for every P_k that is not *iSign-conflicted* with P_i). Therefore the (i,j)-th share and the (j,i)-th share are set to be $f'_i(j) = F(j,i) = F(i,j)$, as required.

- Assume that P_d is not *vss-conflicted* with P_i, and that P_i is not *vss-conflicted* with P_d. We split into cases.

[12] A functionality-based VSS protocol is captured by the sharing functionality $\mathcal{F}_{\mathsf{sh\text{-}comp}}$ in Section 8. The reason that we do not use such a definition here, is that it is inconvenient to employ VSS later as an ideal functionality (see the discussion in Footnote 10).

– Assume that P_i is *vss-conflicted* with P_j or that P_j is *vss-conflicted* with P_i. In this case at least one of the values $f'_i(j) = f'_j(i)$ is opened, and set to be the share. Since $f'_i(j) = f'_j(i) = F(j,i) = F(i,j)$, the (i,j)-th share and the (j,i)-th share are $F(j,i) = F(i,j)$, as required.

– Assume that P_i is not *vss-conflicted* with P_j and that P_j is not *vss-conflicted* with P_i. In this case it holds that $a_{j,i} = b_{i,j}$ so $b_{i,j} - r'_{j,i} = a_{j,i} - r_{j,i} = F(j,i)$, as required. Since P_j is not *vss-conflicted* with P_i then P_j is not *iSign-conflicted* with P_i. Therefore, upon opening $r'_{j,i}$ the parties recover the correct (i,j)-th share $F(j,i)$. Similarly, the (j,i)-th share is set to be $b_{j,i} - r'_{i,j} = F(j,i)$, as required.

This concludes the proof of the claim. □

We conclude that a corrupt P_i has to open at least $n - t \geq t + 1$ points that are consistent with $F(x,i)$. Therefore, the polynomial $f_i(x)$ that P_i reveals in vrec is either $F(x,i)$, or it has degree more than t in which case P_i is not an accepted party. We continue with the following claim.

Claim 6.2. *For every honest P_i, and every $P_j \notin B$. it holds that the (i,j)-th share is $F(i,j) = F(j,i)$.*

Proof. If P_j is corrupt then this follows from Claim 6.1. Otherwise, if P_j is honest, then P_i, P_j and P_d are not *internally-vss-conflicted*. If $\mathsf{flag}_i^d = 1$ or $\mathsf{flag}_j^d = 1$ or $\mathsf{flag}_i = 1$ or $\mathsf{flag}_j = 1$ or $\mathsf{flag}_{i,j} = 1$ or $\mathsf{flag}_{j,i} = 1$ then, by the same analysis as in the proof of Claim 6.1, the (i,j)-th share is set to be the public value $F(j,i)$. Otherwise, $P_i, P_j \notin W \cup W'$, and the (i,j)-th share is $b_{i,j} - r_{j,i} = F(j,i)$, as required. This concludes the proof of the claim. □

We conclude that in vrec all honest parties are in A, and for every corrupt P_i in A it holds that $f_i(x) = F(x,i)$. Since there are at least $n - t \geq t + 1$ honest parties, the polynomial $F(x,y)$ will be recovered, and the output will be $F(0,0) = s$. This concludes the analysis of correctness.

Commitment. We continue with the analysis of the commitment property. We show that even if P_d is corrupt, in every execution in which P_d is not discarded, at the end of vss there exists a symmetric bivariate polynomial $F(x,y)$ of degree at most t in each variable that will be opened in vrec with probability 1, and we take the committed value to be $F(0,0)$. In fact, we prove that this is true even if the parties are allowed to raise flags in the third round. As before, we observe that none of the honest parties are in B. We continue with the following claim.

Claim 6.3. *For every honest $P_i \notin W \cup W'$ and every honest P_j it holds that (1) the (i,j)-th share is consistent with the degree-t polynomial $f'_i(x)$ that P_i received from P_d in the first round, and (2) the (i,j)-th share is equal to the (j,i)-th share.*

Proof. We split into cases.

- Assume that $P_j \notin W \cup W'$. Since P_i and P_j are honest then they are not *iSign-conflicted* , and it is not hard to see that if P_i is *internally-vss-conflicted* with P_j or P_j is *internally-vss-conflicted* with P_i then P_d is discarded since $f'_i(j) \neq f'_j(i)$.

  Otherwise, they are not *internally-vss-conflicted*, which means that $a_{i,j} = b_{j,i}$ so $f'_i(j) = f'_j(i)$. If $\mathsf{flag}_{i,j} = 1$ or $\mathsf{flag}_{j,i} = 1$, then since $a_{i,j} = b_{j,i}$ it must hold that the correct share $f'_i(j)$ is made public.

Otherwise, P_i and P_j are not *vss-conflicted*. In this case the (i,j)-th share is $b_{i,j} - r'_{j,i} = f'_i(j)$, and the (j,i)-th share is $b_{j,i} - r'_{i,j} = f'_j(i)$, and it holds that $f'_i(j) = b_{i,j} - r'_{j,i} = a_{j,i} - r_{j,i} = f'_j(i)$, as required.

- Assume that $P_j \in W$. In this case the (i,j)-th share and the (j,i)-th share are both set to be $f^d_j(i)$. Since $P_i \notin W \cup W'$ then P_i is not *iSign-conflicted* with P_d and $a^d_{i,j} = a_{i,j}$. Therefore P_d has to open $r^d_{i,j}$, and P_d is discarded unless $f^d_j(i) = a^d_{i,j} - r^d_{i,j} = a_{i,j} - r_{i,j} = f'_i(j)$, as required.
- Assume that $P_j \in W'$. As before, if P_i is *internally-vss-conflicted* with P_j, or P_j is *internally-vss-conflicted* with P_i then P_d is discarded. Otherwise, it must hold that $a_{i,j} = b_{j,i}$ so $f'_i(j) = f'_j(i)$. Since $P_j \in W'$, the (i,j)-th share and the (j,i)-th share are both set to be $f^d_j(i) = f'_j(i)$, as required.

This concludes the proof of the claim. □

Consider now any honest $P_i \in W \cup W'$. We've already argued that $f^d_i(x)$ is consistent with $f'_j(x)$ for any honest $P_j \notin W \cup W'$. Observe that it also has to be consistent with $f^d_j(x)$ for any honest $P_j \in W \cup W'$ or otherwise P_d is discarded. We conclude that the polynomials

$$(f'_j(x))_{P_j \in \mathsf{H}, P_j \notin W \cup W'} \cup (f^d_j(x))_{P_j \in \mathsf{H}, P_j \in W \cup W'}$$

are pairwise-consistent. Since there are $n - t \geq t + 1$ honest parties, those polynomials define a unique symmetric bivariate polynomial of degree at most t in each variable $F(x,y)$ (see Fact B.2).

We continue by showing that all the shares of the honest parties are consistent with $F(x,y)$. This is clearly true for any $P_i \in W \cup W'$. We continue with the following claim.

Claim 6.4. *For every honest $P_i \notin W \cup W'$, and every corrupt $P_j \notin B$ it holds that the (i,j)-th and the (j,i)-th share are both $f'_i(j) = F(j,i)$.*

Proof. We split into cases.

- Assume that $P_j \notin W \cup W'$. We split into cases.
  - Assume that P_i is *vss-conflicted* with P_j and that P_j is *vss-conflicted* with P_i. In this case P_i opens $f'_i(j)$ and P_j opens $f'_j(i)$. If they are not the same then P_d is discarded, and otherwise, the (i,j)-th share, as well as the (j,i)-th share, are set to be $f'_i(j)$.
  - Assume that P_i is *vss-conflicted* with P_j and that P_j is not *vss-conflicted* with P_i. In this case P_i opens $f'_i(j)$ and the (i,j)-th share, as well as the (j,i)-th share, are set to be $f'_i(j)$.
  - Assume that P_i is not *vss-conflicted* with P_j and that P_j is *vss-conflicted* with P_i. In this case P_j opens $f'_j(i)$ and $r'_{i,j}$, and the (i,j)-th share, as well as the (j,i)-th share, are set to be $f'_j(i)$. It must hold that $b_{j,i} = f'_j(i) + r'_{i,j}$ or otherwise P_j is in B. But then $a_{i,j} = b_{j,i}$ or otherwise P_i is *vss-conflicted* with P_j, so necessarily $f'_j(i) = f'_i(j)$, as required.
  - Assume that P_i is not *vss-conflicted* with P_j and that P_j is not *vss-conflicted* with P_i. In this case the (i,j)-th share is $b_{i,j} - r'_{j,i} = f'_i(j)$, and the (j,i)-th share is $b_{j,i} - r'_{i,j} = a_{i,j} - r_{i,j} = f'_i(j)$, as required.
- Assume that $P_j \in W$. In this case the (i,j)-th share and the (j,i)-th share are both set to be $f^d_j(i)$. Observe that P_d opened $r^d_{i,j}$, and that P_d is discarded unless $f^d_j(i) = a^d_{i,j} - r^d_{i,j} = a_{i,j} - r_{i,j} = f'_i(j)$, as required.

- Assume that $P_j \in W'$. In this case the (i,j)-th share, as well as the (j,i)-th share, are set to be $f'_j(i)$. We split into cases.
  - Assume that P_i is *vss-conflicted* with P_j. In this case P_i opens $f'_i(j)$. If $f'_i(j) \neq f'_j(i)$ then P_d is discarded, and otherwise, the (i,j)-th share, as well as the (j,i)-th share, are set to be $f'_i(j)$.
  - Assume that P_i is not *vss-conflicted* with P_j. Since $P_j \in W'$ then P_j also opens $r'_{i,j}$. Since P_i is not *vss-conflicted* with P_j, it must hold that $a_{i,j} = b_{j,i}$ and since $P_j \notin B$ then $b_{j,i} = f'_j(i) + r_{i,j}$. But since $a_{i,j} = f'_i(j) + r_{i,j}$ it must hold that $f'_j(i) = f'_i(j)$, as required.

This concludes the proof of the claim. □

We conclude that in the vrec execution all honest parties are in A, and that for every honest party it holds that $f_i(x) = F(x,i)$. In addition, by the above analysis, for every corrupt $P_j \notin B$, and every honest P_i it holds that the (j,i)-th share is $F(j,i) = F(i,j)$. This means that $f_j(x)$ agrees with $F(x,j)$ on $n - t \geq t + 1$ points, so if $P_j \in A$ then $f_j(x) = F(x,j)$. Finally, since there are at least $n - t \geq t + 1$ honest parties that will provide their shares in the opening phase, the polynomial $F(x,y)$ will be recovered, and the output will be $F(0,0)$. This concludes the analysis of commitment.

For future reference, let us record the following lemma whose proof follows from the above analysis.

Lemma 6.5 (Properties of the shares.)**.** *In any execution of* vss *in the $\mathcal{F}_{\mathsf{iSig}}$-hybrid model, if P_d is honest then P_d is not discarded. In addition, the following is true for any execution of* vss *in which P_d (that might be corrupt) was not discarded.*

1. *The set B does not contain honest parties.*
2. *At the end of the execution, the shares of the honest parties (i.e., the (i,j)-th shares for $P_i \in \mathsf{H}$ and $P_j \notin B$) are consistent with a unique symmetric bivariate polynomial of degree at most t in each variable, denoted by $F'(x,y)$. If P_d is honest then $F'(x,y) = F(x,y)$.*
3. *If P_d is honest then for every $P_i, P_j \notin B$, if the (i,j)-th share is public then the (i,j)-th share is equal to $F(j,i)$.*
4. *For every $P_i \in W \cup W'$ the shares of P_i are public and they are consistent with $F'(x,i)$.*
5. *For $P_i, P_j \notin B$, if P_i is honest or P_j is honest then the (i,j)-th share is $F'(j,i) = F'(i,j)$.*
6. *For an honest $P_i \notin W \cup W'$ and every $P_j \notin B$, the (i,j)-th and (j,i)-th shares are equal to $f'_i(j)$ and it holds that $F'(j,i) = f'_i(j)$.*

6.3 Linearly-Homomorphic VSS

In this section we show that our VSS scheme is linearly-homomorphic, i.e., it allows parties to combine shares of different secrets into a new share of a linear combination of the underlying secrets. Assume that the parties shared m secrets $s_1, \ldots, s_m$ via m instances of vss, denoted by $\mathsf{vss}^1, \ldots, \mathsf{vss}^m$. To obtain linearity, we require that for every pair of parties (P_i, P_j), all instances of

vss use *the same* underlying signature scheme iSig, where P_i is $\mathcal{D}$ and P_j is $\mathcal{I}$, denoted $\mathsf{iSig}^{i,j}$. The m pads used in these instances act as the m messages in $\mathsf{iSig}^{i,j}$.

The underlying idea to reveal a linear combination of the secrets $s_1, \ldots, s_m$ is similar to vrec. For every $i, j \in [n]$ we let P_i open a linear combination of the (i,j)-th shares over all instances of $\mathsf{vss}^1, \ldots, \mathsf{vss}^m$, by using the linearity of the signatures scheme (by setting the βs for all cases to the required linear combination). Then, we let the parties verify that the shares that P_i revealed correspond to a degree t polynomial, and otherwise we set P_i's share to an erasure. Finally, we interpolate over all valid shares to obtain the output.

Formally, assume that the parties want to compute the value $s := \sum_{k=1}^{m} \beta_k \cdot s_k$. Let $S \subset [m]$ be the set of all indices $k \in [m]$ so that the dealer in vss^k was not discarded, and for every $k \in S$ let $F^k(x,y)$ be the polynomial defined by the shares of the honest parties in vss^k, so that $s_k = F^k(0,0)$, let $r^k_{i,j}$ be the random pad that P_i signed to P_j in vss^k, let $b^k_{i,j}$ be the broadcast of P_i, and let B^k be the set of bad parties in vss^k. Denote $B = \cup_{k \in S} B^k$. Since every discarded dealer is necessarily corrupt, we simply reset $s_k = 0$ for every $k \notin S$, so it is enough to compute $\sum_{k \in S} \beta_k \cdot F^k(x,y)$. For every $k \in S$ denote by $f^k_{i,j}$ the (i,j)-th share in vss^k, and for every $P_i, P_j \notin B$, let $S_{i,j}$ be the set of all indices $k \in S$, so that the (i,j)-th share in vss^k is not public.

For the computation of the linear function, every $P_i \notin B$ reveals his shares of s as follows. For every $P_j \notin B$ we let P_i open $\sigma_{i,j} := \sum_{k \in S_{i,j}} \beta_k \cdot r^k_{j,i}$ by using the linearity of the signature scheme.[13] Observe that this allows the secure computation of $(\sum_{k \in S_{i,j}} \beta_k \cdot b^k_{i,j}) - \sigma_{i,j} = \sum_{k \in S_{i,j}} \beta_k \cdot (b^k_{i,j} - r^k_{i,j}) = \sum_{k \in S_{i,j}} \beta_k \cdot f^k_{i,j}$ by the parties. The parties set $f_{i,j} := (\sum_{k \in S_{i,j}} b^k_{i,j}) - \sigma_{i,j} + \sum_{k \in S \setminus S_{i,j}} f^k_{i,j} = \sum_{k \in S} f^k_{i,j}$ to be the (i,j)-th share of s. If the shares $(f_{i,j})_{P_j \notin B}$ do not correspond to a degree-t polynomial $f_i(x)$ then the parties ignore the shares of P_i. Finally, the parties interpolate all valid polynomials $f_i(x)$ to obtain $F(x,y)$, and output $F(0,0)$. Using Lemma 6.5 one can verify that the output is indeed s.

6.4 On Tentative Shares

In order to obtain round-optimal MPC protocol, it will be crucial to perform operations on the shares *before* the termination of the vss protocol. For any honest P_i that received a degree-t polynomial $f'_i(x)$ in the first round, think of the shares defined by $f'_i(x)$ as the *tentative* shares. We note that the tentative shares satisfy the following property.

Observation 6.6 (Tentative shares.)**.** *Tentative shares of an honest P_i might change in Round 3 only if (1) P_i received a polynomial of degree more than t in the first round, or (2) P_i is* iSign-conflicted *with P_d, or (3) there exists some $j \in [n]$ so that $a^d_{i,j} \neq a_{i,j}$. This means that P_i knows already at the end of the second round whether its shares might change in the third round.*

Indeed, assume that P_i received a degree-t polynomial, that P_i is not *iSign-conflicted* with P_d, and that for every $j \in [n]$ it holds that $a^d_{i,j} = a_{i,j}$. We split into cases.

- If P_d is *vss-conflicted* with P_i, then P_d must open the random pads $(r_{i,j})_{j \in [n]}$. Since P_i is not *iSign-conflicted* with P_d, and since $a^d_{i,j} = a_{i,j}$ for every $j \in [n]$, it must hold that $a^d_{i,j} - r_{i,j} = a_{i,j} - r_{i,j} = f'_i(j)$. Therefore, the polynomial $f^d_i(x)$ is equal to $f'_i(x)$.

[13] The linear computation is executed in the fourth round of iSig, i.e., in $\mathsf{iSig.open}_2$. Technically, iSig does not support linear operations in $\mathsf{iSig.open}_2$, since those will not be required for the construction of general MPC. However, it is straightforward to modify $\mathsf{iSig.open}_2$ to support linear operations by following the same protocol as in $\mathsf{iSig.open}_1$.

- If P_d is not *vss-conflicted* with P_i, but P_i is *vss-conflicted* with P_d, then the (i,j)-share is set to be $f'_i(j)$ for all $j \in [n]$.
- Otherwise, P_d is not *vss-conflicted* with P_i, and P_i is not *vss-conflicted* with P_j. The claim now follows from Lemma 6.5.

7 The VSS Suite

In this section we present two VSS-related primitives, called *triple secret sharing* (TSS) and verifiable sharing and transferring (VST). In principle, we could abstract these protocols by ideal functionalities. However, for technical reasons, it is inconvenient to employ them later as ideal functionalities (as discussed in Footnote 10). We will therefore prove directly some useful properties of these protocols, that suffices for our applications, namely, for the construction of the Share-Compute functionality (Section 8).

7.1 Triple Secret Sharing

In triple secret sharing the dealer P_d shares three polynomials $F^a(x,y), F^b(x,y)$ and $F^c(x,y)$ and also proves that $F^c(0,0) = F^a(0,0) \cdot F^b(0,0)$. In order to do so, we first develop a sub-protocol for performing linear operations over tentative shares from a single dealer.

7.1.1 Linear Operations over Tentative Shares from a Single Dealer

Consider the case where a single dealer P_d performs m executions of vss in parallel, using the polynomials $F^1(x,y), \ldots, F^m(x,y)$. In the third round (the last round of the vss execution), some P_i wishes to privately reveal a linear combination of its shares to some receiver R, using the coefficients $\boldsymbol{\beta} = (\boldsymbol{\beta}[1], \ldots, \boldsymbol{\beta}[m])$. As in Section 6.3, we assume that all instances of vss use *the same* underlying signature scheme iSig where P_i is $\mathcal{D}$ and P_j is $\mathcal{I}$. In the k-th instance of vss, we denote the random pad that P_j sent to P_i by $r^k_{j,i}$, the second-round broadcast messages of P_i by $(a^k_{i,j}, b^k_{i,j})_{j\in[n]}$, and the set of bad players by B^k.

The main idea is to extend a conflict between two parties in some vss execution into a conflict between the same parties in *all* vss executions. For example, if an honest P_d is conflicted with some P_j, then necessarily P_j is corrupt, so P_d can become conflicted with P_j in *all* vss executions, and makes all the shares of P_j public. In particular, the (i,j)-th share becomes public in every vss execution, so R can compute the (i,j)-th share of the output based on the public values. Similarly, if an honest P_j is conflicted with P_d, then P_j becomes conflicted with P_d in *all* vss executions. Note that if an honest P_j is conflicted with some P_k, then either P_k or P_d is corrupt. In both cases there is no need to keep the (j,k)-th share secure, so P_j becomes conflicted with P_k in *all* vss executions. Finally, if P_i is not conflicted with P_j, then we let P_i privately open the linear combination $\sum_{k\in[m]} \boldsymbol{\beta}[k] \cdot r^k_{j,i}$ to R, so R could compute $(\sum_{k\in[m]} \boldsymbol{\beta}[k] \cdot b^k_{i,j}) - (\sum_{k\in[m]} \boldsymbol{\beta}[k] \cdot r^k_{j,i}) = \sum_{k\in[m]} F(j,i)$. Given all the shares, R can recover the polynomial $\sum_{k\in[m]} \boldsymbol{\beta}[k] \cdot F(x,i)$. We continue with a sub-protocol for linear operations over the tentative shares that is executed in parallel to the vss executions.

Protocol linop_1

Inputs. P_i inputs a vector of coefficients $\boldsymbol{\beta} = (\boldsymbol{\beta}[1], \ldots, \boldsymbol{\beta}[m])$. All parties share a statistical security parameter 1^κ.

Assumption. We assume that the first and the second round of all vss instances were executed. In addition, we assume that in the third round of the outer protocol, the parties are instructed to execute the vss instances as follows.

- If P_d is *internally-vss-conflicted* with some P_j in some vss execution, or if $\mathsf{flag}_j^d = 1$ in some vss execution, then P_d raises a flag flag_j^d in all vss executions.
- If some P_j is *internally-vss-conflicted* with P_d in some vss execution, or if $\mathsf{flag}_j = 1$ in some vss execution, then P_j raises a flag flag_j in all vss executions.
- If some P_j is *internally-vss-conflicted* with some $P_{j'}$ in some vss execution, or if $\mathsf{flag}_{j,j'} = 1$ in some vss execution, then P_j raises a flag $\mathsf{flag}_{j,j'}$ in all vss executions.

Round 3. P_i inputs the flags $(\mathsf{flag}_i, \mathsf{flag}_{i,j})_{j\in[n]}$. (Those are the flags of P_i in the vss executions.)

P_i sends $\boldsymbol{\beta}$ to R. In addition, P_i does as follows.

- If $\mathsf{flag}_i = 1$ then P_i does nothing.
- Otherwise, for every P_j, P_i does as follows.
  - If $\mathsf{flag}_{i,j} = 1$ then P_i does nothing.
  - Otherwise, P_i executes $\mathsf{iSig.open}_1(P_j, P_i, R, \sum_{k\in[m]} \boldsymbol{\beta}[k] r_{j,i}^k)$.

Local computation. If some vss execution ended with P_d discarded, or if P_d is *vss-conflicted* with some P_j in some vss execution but not in others then R outputs $\perp$ and terminates.

Otherwise, let $B := \cup_{k\in[m]} B^k$. Add to B any P_j so that either (1) P_j is *vss-conflicted* with P_d in some vss executions but not in others, or (2) P_j is *vss-conflicted* with some $P_{j'}$ in some vss executions but not in others. If P_i is in B, then R outputs $\perp$ and terminates.

Otherwise P_i is not in B. If P_d is *vss-conflicted* with P_i in all vss executions, or P_i is *vss-conflicted* with P_d in all vss executions, then the shares $f_i^1(x), \ldots, f_i^m(x)$ of P_i are public. In this case R sets $g_i(x) := \sum_{k\in[m]} \boldsymbol{\beta}[k] \cdot f_i^k(x)$, outputs $(\boldsymbol{\beta}, g_i(x))$ and terminates.

Otherwise, P_d and P_i are not *vss-conflicted* in any execution. For every $P_j \notin B$, R does as follows.

- If P_i is *vss-conflicted* with P_j, or P_j is *vss-conflicted* with P_i, or P_d is *vss-conflicted* with P_j, or P_j is *vss-conflicted* with P_d, then the shares $f_i^1(j), \ldots, f_i^m(j)$ are public, and R computes $g_{i,j} := \sum_{k\in[m]} \boldsymbol{\beta}[k] \cdot f_i^k(j)$.
- Otherwise, R computes $g_{i,j} := (\sum_{k\in[m]} \boldsymbol{\beta}[k] \cdot b_{i,j}^k) - (\sum_{k\in[m]} \boldsymbol{\beta}[k] \cdot r_{j,i}^k)$, where the first sum can be computed based on the public values $b_{i,j}^1, \ldots, b_{i,j}^m$, and the second sum was obtained from $\mathsf{iSig.open}(P_j, P_i, R, \sum_{k\in[m]} \boldsymbol{\beta}[k] r_{j,i}^k)$.

If P_i did not use the same vector $\boldsymbol{\beta}$ in all $\mathsf{iSig.open}(P_j, P_i, R, \sum_{k\in[m]} \boldsymbol{\beta}[k] r_{j,i}^k)$ instances, or the opening terminated with $\perp$, or if the shares $(g_{i,j})_{P_j\notin B}$ do not correspond to a degree-t polynomial, then R outputs $\perp$. Otherwise let $g_i(x)$ be the degree-t polynomial corresponding to the shares $(g_{i,j})_{P_j\notin B}$, and R outputs $(\boldsymbol{\beta}, g_i(x))$.

Figure 8: Protocol linop_1

We continue with a short analysis in the $\mathcal{F}_{\mathsf{iSig}}$-hybrid model. In the analysis of the vss protocol (see Lemma 6.5) we have seen that any vss execution defines a polynomial $F(x, y)$ so that (1) for every $P_k \in W \cup W'$ it holds that the public polynomial $f_k^d(x)$ is equal to $F(x, k)$, (2) for every

honest $P_k \notin W \cup W'$ and $P_j \notin \mathcal{B}$ the (k,j)-th share is $F(j,k)$, and (3) for every corrupt P_k and honest P_j, the (k,j)-th share is equal to $F(j,k)$.

Assuming that the outer-protocol satisfies the assumptions of linop_1, it is not hard to see that the set B does not contain honest parties. Therefore, for an honest P_i, the correct value $g_i(x) = \sum_{k\in[m]} \boldsymbol{\beta}[k] \cdot F^k(x,i)$ will be recovered by R. For a corrupt P_i, the polynomial $g_i(x)$ has to be consistent with the shares of at least $n - t \geq t + 1$ honest parties, so either $g_i(x) = \sum_{k\in[m]} \boldsymbol{\beta}[k] \cdot F^k(x,i)$, or $g_i(x)$ has degree more than t in which case R outputs $\perp$.

We proved the following lemma.

Lemma 7.1. *Consider any execution of* $\mathsf{vss}^1, \ldots, \mathsf{vss}^m$ *and* linop_1 *in the* $\mathcal{F}_{\mathsf{iSig}}$*-hybrid model, where the parties are instructed to act according to the assumptions of* linop_1*. Assume that* P_d *was not discarded, and let* $F^1(x,y), \ldots, F^m(x,y)$ *be the polynomials defined by the shares of the honest parties. Then the set* B *does not contain honest parties, and the following holds.*

- *If* P_i *is honest then* R *outputs* $(\boldsymbol{\beta}, \sum_{k\in[m]} \boldsymbol{\beta}[k] \cdot F^k(x,i))$.
- *If* P_i *is corrupt then* R *outputs either* $(\boldsymbol{\beta}, \sum_{k\in[m]} \boldsymbol{\beta}[k] \cdot F^k(x,i))$ *or* $\perp$.

7.1.2 The Triple Secret Sharing Protocol

The dealer, on inputs $F^a(x,y)$, $F^b(x,y)$ and $F^c(x,y)$ so that $F^c(0,0) = F^a(0,0) \cdot F^b(0,0)$, picks two random degree-d polynomials $A(x), B(x)$ conditioned on $A(0) = F^a(0,0)$, $B(0) = F^b(0,0)$, and sets the degree-$2d$ polynomial $C(x) := A(x) \cdot B(x)$. The dealer shares the coefficients of the polynomials $A(x), B(x)$ and $C(x)$ via vss, where the free coefficients are shared by using the polynomials $F^a(x,y)$, $F^b(x,y)$ and $F^c(x,y)$. As in Section 6.3, we assume that all instances of vss use *the same* underlying signature scheme iSig, where P_i is $\mathcal{D}$ and P_j is $\mathcal{I}$.

The goal now is to convince the parties that $F^c(0,0) = F^a(0,0) \cdot F^b(0,0)$, and to do so it is enough to convince the parties that $C(x) = A(x) \cdot B(x)$. At a high level, the idea is to generate a random challenge $\beta \in \mathbb{F} \setminus \{0\}$, let all the parties compute $A(\beta), B(\beta)$ and $C(\beta)$ by performing linear operations over the tentative shares, and then verify that $C(\beta) = A(\beta) \cdot B(\beta)$ and reject the dealer if equality does not hold. Since $A(x)$ and $B(x)$ are random polynomials, this will reveal no information about the free coefficients of $A(x), B(x)$ and $C(x)$. In addition, if $C(x) \neq A(x) \cdot B(x)$, then with high probability $C(\beta) \neq A(\beta) \cdot B(\beta)$. The main challenge in implementing this idea is that we cannot let the adversary know the challenge β before the termination of the vss protocol. Indeed, since the tentative share might change even in the last round of vss, a corrupt dealer who knows β before the termination of the vss protocol could pick polynomials $A(x), B(x)$ and $C(x)$ so that $C(\beta) = A(\beta) \cdot B(\beta)$ but $C(x) \neq A(x) \cdot B(x)$.

To solve this problem we use the honest majority, and let every subset Q of $t+1$ parties to generate random challenges $\beta_{Q,1}, \ldots, \beta_{Q,n}$, and privately open the shares of $A(\beta_{Q,i}), B(\beta_{Q,i}), C(\beta_{Q,i})$ to P_i by using linop_1. First we note that there is at least one set Q that contains only honest parties, that for this set the challenges that correspond to honest parties are unknown to the adversary, and that every honest P_i receives $t+1$ shares of $A(\beta_{Q,i}), B(\beta_{Q,i}), C(\beta_{Q,i})$, so P_i can recover those values and verify that $C(\beta_{Q,i}) = A(\beta_{Q,i}) \cdot B(\beta_{Q,i})$. In addition, for every set Q that contains a corrupt party, by the correctness of linop_1, the corrupt party can either open the correct linear combination or $\perp$, so an honest dealer will not be discarded by sets that contain corrupt parties. Finally, if the

degree d is large enough, then even though the adversary sees many evaluations of the polynomials $A(x), B(x)$, and $C(x)$, the adversary still holds no information about the free coefficients of $A(x), B(x)$, and $C(x)$.

We continue with a description of the protocol in Figure 9.

Protocol tss

Inputs: The dealer P_d has three polynomials $F^a(x,y), F^b(x,y), F^c(a,b)$ such that $F^c(0,0) = F^a(0,0)F^b(0,0)$. All parties share a statistical security parameter 1^κ.

Round 1. The dealer does as follows.

- P_d picks three random polynomials $A(x), B(x)$, and $C(x)$ of degree d, d and $2d$, respectively, such that $C(x) = A(x) \cdot B(x)$, and $A(0) = a$, $B(0) = b$ and $C(0) = c$, where $d := \binom{n}{t+1} \cdot (t+1)^2$. Let A^k, B^k, and C^k denote the k-th coefficient of $A(x), B(x)$ and $C(x)$, respectively, where $A^k = B^k = 0$ for $k > d$.
- For each $k \in \{1, \ldots, d\}$, D shares A^k and B^k via two vss instances, $\mathsf{vss}^{a,k}$, and $\mathsf{vss}^{b,k}$ respectively, and we denote the corresponding sharing polynomials by $F^{a,k}(x,y)$ and $F^{b,k}(x,y)$. For each $k \in \{1, \ldots, 2d\}$, D shares C^k via a vss instance, $\mathsf{vss}^{c,k}$, and we denote the corresponding sharing polynomial by $F^{c,k}(x,y)$.
- 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 $F^a(x,y), F^b(x,y)$ and $F^c(x,y)$, respectively. We denote the corresponding sharing polynomials by $F^{a,0}(x,y), F^{b,0}(x,y)$ and $F^{c,0}(x,y)$.

Round 2. The parties continue with all the vss executions. For every subset Q of exactly $t+1$ parties, the parties do as follows. Let P_Q be the party with the minimum index in Q. Then P_Q picks non-zero random field elements $\beta_{Q,i} \leftarrow \mathbb{F} \setminus \{0\}$ for every $i \in [n]$, and sends them to all parties in Q via private channels.

Round 3. Every P_i inputs a flag-bit flag_i together with flag-bits $(\mathsf{flag}_{i,j})_{j \in [n]}$. In addition, P_d inputs flags $\mathsf{flag}_1^d, \ldots, \mathsf{flag}_n^d$.

The parties continue with all the vss executions with the following flags. If P_i is *internally-vss-conflicted* with P_d in some instance of vss, or $\mathsf{flag}_i = 1$, then P_i inputs $\mathsf{flag}_i = 1$ in all instances of vss; Otherwise, P_i inputs $\mathsf{flag}_i = 0$ in all instances of vss. In addition, if in some vss instance P_d is *internally-vss-conflicts* with P_i, or $\mathsf{flag}_i^d = 1$, then P_d inputs $\mathsf{flag}_i^d = 1$ in all vss executions; Otherwise, P_d inputs $\mathsf{flag}_i^d = 0$ in all instances of vss. Also, if P_i is *internally-vss-conflicted* with P_j in some vss execution, or $\mathsf{flag}_{i,j} = 1$, then P_i inputs $\mathsf{flag}_{i,j} = 1$ in all vss executions; Otherwise, P_i inputs $\mathsf{flag}_{i,j} = 0$ in all instances of vss.

For every set Q of $t+1$ parties, every $j \in [n]$, and every $P_i \in Q$, P_i sends $\beta_{Q,j}$ to P_j and opens

$$\beta_{Q,j}^0 \cdot F^{a,0}(x,i) + \beta_{Q,j}^1 \cdot F^{a,1}(x,i) + \ldots + \beta_{Q,j}^d \cdot F^{c,d}(x,i)$$
$$\beta_{Q,j}^0 \cdot F^{b,0}(x,i) + \beta_{Q,j}^1 \cdot F^{b,1}(x,i) + \ldots + \beta_{Q,j}^d \cdot F^{b,d}(x,i)$$
$$\beta_{Q,j}^0 \cdot F^{c,0}(x,i) + \beta_{Q,j}^1 \cdot F^{c,1}(x,i) + \ldots + \beta_{Q,j}^{2d} \cdot F^{c,2d}(x,i)$$

to P_j by executing three instances of linop_1: the first instance, $\mathsf{linop}_1^{Q,a,i,j}$, is executed with respect to $\mathsf{vss}^{a,0}, \ldots, \mathsf{vss}^{a,d}$ and with vector of coefficients $\boldsymbol{\beta} = (\beta_{Q,j}^0, \ldots, \beta_{Q,j}^d)$; the second instance, $\mathsf{linop}_1^{Q,b,i,j}$, is executed with respect to $\mathsf{vss}^{b,0}, \ldots, \mathsf{vss}^{b,d}$ and with vector of coefficients $\boldsymbol{\beta} = (\beta_{Q,j}^0, \ldots, \beta_{Q,j}^d)$; and the third instance, $\mathsf{linop}_1^{Q,c,i,j}$, is executed with respect to $\mathsf{vss}^{c,0}, \ldots, \mathsf{vss}^{c,2d}$ and with vector of coefficients

$\boldsymbol{\beta} = (\beta^0_{Q,j}, \ldots, \beta^{2d}_{Q,j})$. In all of the instances P_i is using the flags $(\mathsf{flag}_i, \mathsf{flag}_{i,j})_{j\in[n]}$ that are used as inputs in the vss instances, as we defined above.

(Local computation): Every P_i does as follows.

- Run the local computation of every vss execution. If some vss execution ended with P_d discarded, or if P_d is vss-conflicted with some P_j in some vss execution but not in others then P_d is discarded.
- Otherwise, run the local computation of every linop_1 instance. Denote the output of $\mathsf{linop}_1^{Q,k,j,i}$ that did not terminate with $\perp$ by $(\boldsymbol{\beta}^{Q,k,j,i}, g^{Q,k,j,i}(x))$.
- Let $B := (\cup_{j\in\{a,b\},k\in\{0,\ldots,d\}} B^{j,k}) \cup (\cup_{k\in\{0,\ldots,2d\}} B^{c,k})$, where $B^{j,k}$ is the set of bad players defined in $\mathsf{vss}^{j,k}$. Add to B any P_j so that either (1) P_j is vss-conflicted with P_d in some vss executions but not in others, or (2) P_j is vss-conflicted with some $P_{j'}$ in some vss executions but not in others.
- A set Q is bad if (1) $Q_k \cap B \neq \varnothing$, or (2) not all parties in Q send the same $\beta_{Q,i}$, or (3) there is some party $P_j \in Q$ and $k \in \{a,b,c\}$ for which the output of $\mathsf{linop}_1^{Q,k,j,i}$ is either $\perp$, or $\boldsymbol{\beta}^{Q,k,j,i} \neq (\beta^0_{Q,i}, \ldots, \beta^d_{Q,i})$ if $k \in \{a,b\}$, or $\boldsymbol{\beta}^{Q,k,j,i} \neq (\beta^0_{Q,i}, \ldots, \beta^{2d}_{Q,i})$ if $k = c$.
- For every good set Q reconstruct $A(\beta_{Q,i}), B(\beta_{Q,i})$ and $C(\beta_{Q,i})$ from $(g^{Q,a,j,i}(x), g^{Q,b,j,i}(x), g^{Q,c,j,i}(x))_{P_j\in Q}$ and verify that $C(\beta_{Q,i}) = A(\beta_{Q,i}) \cdot B(\beta_{Q,i})$. If equality does not hold then P_d is discarded. Otherwise, set the shares as defined in $\mathsf{vss}^{a,0}$, $\mathsf{vss}^{b,0}$ and $\mathsf{vss}^{c,0}$.

Figure 9: Protocol tss

Analysis. We continue with a short analysis of the correctness of tss.[14] We first note that by the properties of vss (see Lemma 6.5), the set B does not contain an honest party. Consider an honest P_d and observe that P_d is not discarded in the first step of the local computation phase. In addition, for every honest P_i and for every set Q that is good for P_i, the correctness of linop_1 (see Lemma 7.1) implies that the shares that P_i receives from the parties in Q define the correct values $A(\beta_{Q,i}), B(\beta_{Q,i})$ and $C(\beta_{Q,i})$, so necessarily $C(\beta_{Q,i}) = A(\beta_{Q,i}) \cdot B(\beta_{Q,i})$. Therefore, an honest P_d is not discarded by the honest parties.

Consider now a corrupt P_d that was not discarded in the first step of the local computation phase. By Lemma 6.5, the shares of the honest parties in the internal vss execution define polynomials $F^{v,k}(x,y)$ for $v \in \{a,b,c\}$ and $k \in \{0,\ldots,2d\}$, where we set $F^{a,k}(x,y) = F^{b,k}(x,y) = 0$ for $k > d$. Those polynomials fully defined the univariate polynomials $A(x), B(x)$ and $C(x)$ in tss. Like in the case of an honest P_d, if $C(x) = A(x) \cdot B(x)$, then for every honest P_i and every set Q that is good for P_i, the dealer is not discarded due to Q. We continue by proving that the probability that $C(x) \neq A(x) \cdot B(x)$ and that P_d is not discarded by *all* honest parties is at most $2nd/(|\mathbb{F}|-1)$. We let Q^* be a set of $t+1$ honest parties, and we say that the tss execution is "bad" if (1) $C(x) \neq A(x) \cdot B(x)$ and (2) there exists an honest P_i so that $C(\beta_{Q^*,i}) = A(\beta_{Q^*,i}) \cdot B(\beta_{Q^*,i})$. Observe that, since the values $(\beta_{Q^*,i})_{i\in\mathsf{H}}$ are uniformly distributed even conditioned on the view of the adversary in the tss execution, the probability that the execution is bad is at most $2nd/(|\mathbb{F}|-1)$, as required. Finally, it is not hard to see that in every good execution correctness holds.

7.2 Verifiable Sharing and Transferring

In verifiable sharing and transferring the dealer P_d *shares* a polynomial $F(x,y)$ among the parties, and, in addition, the same polynomial $F(x,y)$ is also *transferred* to a special player R, so that R can

[14] A formal proof of security will be provided as part of the share-compute protocol (Section 8).

reconstruct $F(x,y)$ for the other parties, and convince them that the reconstruction is correct.

In order to share $F(x,y)$ among the parties, we simply let the dealer execute the vss protocol with input $F(x,y)$. In order to transfer $F(x,y)$ to R, we follow similar ideas to those of vss. In the first round, we let the dealer sign the values $(F(j,i))_{i,j\in[n]}$ for R. In the second round, every P_i publicly compares the values $(F(j,i))_{j\in[n]}$ with both R and P_d. In addition, we let P_d and R publicly compare the values $(F(j,i))_{i\in[n],j\in[n]}$. Finally, in the third round, there is a conflict resolution process. Observe that if an honest P_i is conflicted with R, then either P_d or R are corrupt, which means that the adversary holds the polynomial $F(x,y)$. Therefore, in this case we allow P_i to raise a flag in the vss instance so that the share of P_i will become public, even though it is possible that P_d is honest. Similarly, if P_d is conflicted with R, then P_d knows that R is corrupt and that the adversary knows the polynomial $F(x,y)$. Therefore, we allow P_d to raise a flag flag_i for every $i\in[n]$, including for honest parties, in order to make the shares of all the parties public in the vss execution. The protocol is described in Figure 10.

Protocol vst

Inputs: The dealer P_d has a polynomial $F(x,y)$. All parties share a statistical security parameter 1^κ.

Round 1 (share and signature generation):

- The players execute the first round of vss, where P_d inputs $F(x,y)$. For every P_i we use the same notation as in vss, and denote the polynomial of P_i by $f'_i(x)$, the random pad that P_j received from P_i by $r'_{i,j}$ and the random pad that P_d received from P_i by $r^d_{i,j}$.
- P_d executes $\mathsf{iSig.dis}(P_d,R,F(i,j))$ for every $i,j\in[n]$. Assume that P_d received $F'(x,y)$ via the signatures.
- R picks random pads $(\rho_{R,i,j})_{i,j\in[n]}$ and executes $\mathsf{iSig.dis}(R,P_d,\rho_{R,i,j})$ for every $i,j\in[n]$. In addition, R executes $\mathsf{iSig.dis}(R,P_i,\rho_{R,i,j})$ for every P_i and $j\in[n]$. Denote the random pads that P_d received by $\rho^d_{R,i,j}$, and the random pads that P_i received by $\rho'_{R,i,j}$.
- Every P_i picks random pads $(\rho_{i,j,R})_{j\in[n]}$ and executes $\mathsf{iSig.dis}(P_i,P_d,\rho_{i,j,R})$ and $\mathsf{iSig.dis}(P_i,R,\rho_{i,j,R})$ for every $j\in[n]$. Denote the random pads that P_d received by $\rho^d_{i,j,R}$ and the random pads that R received by $\rho'_{i,j,R}$.

Round 2 (Pairwise consistency): The parties execute the second round of vss. For every P_i we use the same notation as in vss, and denote the broadcasts of P_i by $(a_{i,j},b_{i,j})_{j\in[n]}$. Similarly, we denoted the broadcasts of P_d by $(a^d_{i,j})_{i,j\in[n]}$.

In addition, the parties do the following.

- R broadcasts $\alpha_{R,i,j}=F'(j,i)+\rho_{R,i,j}$ and $\beta_{R,i,j}=F'(j,i)+\rho'_{i,j,R}$ for every $i,j\in[n]$.
- P_i broadcasts $\alpha_{i,j,R}=f'_i(j)+\rho_{i,j,R}$ and $\beta_{i,j,R}=f'_i(j)+\rho'_{R,i,j}$ for every $j\in[n]$.
- P_d broadcasts $\alpha^d_{R,i,j}=f_i(j)+\rho^d_{R,i,j}$ and $\alpha^d_{i,j,R}=f_i(j)+\rho^d_{i,j,R}$ for every $i,j\in[n]$.
- Run $\mathsf{iSig.ver}(P_d,R,F(i,j))$, $\mathsf{iSig.ver}(R,P_d,\rho_{R,i,j})$, $\mathsf{iSig.ver}(R,P_i,\rho_{R,i,j})$, $\mathsf{iSig.ver}(P_i,P_d,\rho_{i,j,R})$ and $\mathsf{iSig.ver}(P_i,R,\rho_{i,j,R})$ for every $i,j\in[n]$.
- If the polynomial $F'(x,y)$ that R received is not a symmetric bivariate polynomial of degree at most t in each variable, or if $F'(x,y)$ is not consistent with the shares that R received in the vss execution, then R broadcasts a public complaint.

At the end of the round the parties do the following local computation.

- If P_d is *iSign-conflicted* with R OR P_d is internally-vss-conflicted with R OR R complained against P_d OR $\alpha_{R,i,j} \neq \alpha^d_{R,i,j}$ for some $i, j \in [n]$, then we say that P_d is *internally-vst-conflicted* with R. (This state is known to P_d at the end of the round.)
- If P_d is *iSign-conflicted* with P_i OR P_d is internally-vss-conflicted with P_i OR $\alpha_{i,j,R} \neq \alpha^d_{i,j,R}$ for some $j \in [n]$, then we say that P_d is *internally-vst-conflicted* with P_i. (This state is known to P_d at the end of the round.)
- If R is *iSign-conflicted* with P_d then we say that R is *internally-vst-conflicted* with P_d. (This state is known to R at the end of the round.)
- If R is *iSign-conflicted* with P_i OR $\alpha_{R,i,j} \neq \beta_{i,j,R}$ for some $j \in [n]$ OR $\alpha_{i,j,R} \neq \beta_{R,i,j}$ for some $j \in [n]$, then we say that R is *internally-vst-conflicted* with P_i. (This state is known to R at the end of the round.)
- If P_i is *iSign-conflicted* with P_d OR P_i is internally-vss-conflicted with P_d then we say that P_i is *internally-vst-conflicted* with P_d. (This state is known to P_i at the end of the round.)
- If P_i is *iSign-conflicted* with R OR P_i is internally-vss-conflicted with R OR $\alpha_{i,j,R} \neq \beta_{R,i,j}$ for some $j \in [n]$ OR $\alpha_{R,i,j} \neq \beta_{i,j,R}$ for some $j \in [n]$ then we say that P_i is *internally-vst-conflicted* with R. (This state is known to P_i at the end of the round.)

Round 3 (Resolution by signature opening): Every P_i inputs flag-bits $\mathsf{flag}_{i,d}, \mathsf{flag}_{i,R}$ together with flag-bits $(\mathsf{flag}_{i,j})_{j\in[n]}$. In addition, P_d inputs flags $\mathsf{flag}^d_1, \ldots, \mathsf{flag}^d_n, \mathsf{flag}^d_R$ and R inputs flags $\mathsf{flag}^R_1, \ldots, \mathsf{flag}^R_n, \mathsf{flag}^R_d$. The parties reset the flags as follows.

- If P_d is internally-vst-conflicted with P_i then P_d locally resets $\mathsf{flag}^d_i = 1$. If P_d is internally-vst-conflicted with R, or if $\mathsf{flag}^d_R = 1$, then P_d locally resets $\mathsf{flag}^d_R = 1$ and $\mathsf{flag}^d_i = 1$ for every $i \in [n]$.
- If R is internally-vst-conflicted with P_i then R locally resets $\mathsf{flag}^R_i = 1$. Similarly, if R is internally-vst-conflicted with P_d, or if $\mathsf{flag}^R_d = 1$, then R locally resets $\mathsf{flag}^R_d = 1$ and $\mathsf{flag}^R_i = 1$ for every $i \in [n]$.
- If P_i is internally-vst-conflicted with P_d, or P_i is internally-vst-conflicted with R, or $\mathsf{flag}_{i,d} = 1$, or $\mathsf{flag}_{i,R} = 1$, then P_i locally resets $\mathsf{flag}_{i,d} = 1$ and $\mathsf{flag}_{i,R} = 1$.

In addition, the parties do as follows.

- Every party broadcasts its flags.
- If P_d is internally-vst-conflicted with R OR $\mathsf{flag}^d_R = 1$ then we say that P_d is *vst-conflicted* with R. In this case P_d executes $\mathsf{iSig.open}_1(P_i, P_d, r^d_{i,j})$, $\mathsf{iSig.open}_1(P_i, P_d, \rho^d_{i,j,R})$ and $\mathsf{iSig.open}_1(R, P_d, \rho^d_{R,i,j})$ for all $i, j \in [n]$.
- If P_d internally-vst-conflicted with P_i OR $\mathsf{flag}^d_i = 1$, then we say that P_d is *vst-conflicted* with P_i. In this case P_d executes $\mathsf{iSig.open}_1(P_i, P_d, \rho^d_{i,j,R})$ and $\mathsf{iSig.open}_1(R, P_d, \rho^d_{R,j,i})$ for all $j \in [n]$. Let V be the set of parties that P_d is vst-conflicted with. (Observe that all parties agree on V at the end of the round since iSign-conflicts and vss-conflicts become public at the end of the round.)
- If R internally-vst-conflicted with P_d OR $\mathsf{flag}^R_d = 1$, then we say that R is *vst-conflicted* with P_d. In this case R executes $\mathsf{iSig.open}_1(P_i, R, \rho'_{i,j,R})$ and $\mathsf{iSig.open}_1(P_d, R, F'(i,j))$ for all $i, j \in [n]$.
- If R internally-vst-conflicted with P_i OR $\mathsf{flag}^R_i = 1$ then we say that R is *vst-conflicted* with P_i. In this case R executes $\mathsf{iSig.open}_1(P_i, R, \rho'_{i,j,R})$, $\mathsf{iSig.open}_1(P_d, R, F'(i,j))$, $\mathsf{iSig.open}_1(P_j, R, \rho'_{j,i,R})$, $\mathsf{iSig.open}_1(P_d, R, F'(j,i))$, for all $j \in [n]$. Let U be the set of parties that R is vst-conflicted with.

(Observe that all parties agree on U at the end of the round since iSign-conflicts and vss-conflicts become public at the end of the round.)

- If P_i internally-vst-conflicted with P_d OR $\mathsf{flag}_{i,d} = 1$, then we say that P_i *vst-conflictes* with P_d. P_i and executes $\mathsf{iSig.open}_1(P_d, P_i, f'_i(j))$ and $\mathsf{iSig.open}_1(R, P_i, \rho'_{R,i,j})$, for all $j \in [n]$. Let V' be the set of all P_i that are vst-conflicted with P_d and are not in V. (Observe that all parties agree on V' at the end of the round since iSign-conflicts and vss-conflicts become public at the end of the round.)
- If P_i internally-vst-conflicted with R OR $\mathsf{flag}_{i,R} = 1$ then we say that P_i is *vst-conflicted* with R. In this case P_i executes $\mathsf{iSig.open}_1(P_d, P_i, f'_i(j))$ and $\mathsf{iSig.open}_1(R, P_i, \rho'_{R,i,j})$ for all $j \in [n]$. Let U' be the set of all P_i that are vst-conflicted with R and are not in U. (Observe that all parties agree on U' at the end of the round since iSign-conflicts and vss-conflicts become public at the end of the round.)
- Complete the execution of $\mathsf{iSig.open}_1(P_d, R, F'(i,j))$, $\mathsf{iSig.open}_1(R, P_d, \rho^d_{R,i,j})$, $\mathsf{iSig.open}_1(R, P_i, \rho'_{R,i,j})$, $\mathsf{iSig.open}_1(P_i, P_d, \rho^d_{i,j,R})$ and $\mathsf{iSig.open}_1(P_i, R, \rho'_{i,j,R})$ for every $i, j \in [n]$.
- The parties execute the last round of vss where P_i holds flag_i and $(\mathsf{flag}_{i,j})_{j\in[n]}$, and P_d holds $\mathsf{flag}^d_1, \ldots, \mathsf{flag}^d_n$.

(Local computation):

Every party executes the local computation of vss. Let B be the set of bad parties, and let W, W' be the sets computed in vss.

Add a party P_i to B, if one of the following is true:

- There exists an instance of public opening $\mathsf{iSig.open}_1(*, P_i, *)$ that failed (but did not have *iSign-conflict*), or P_i did not open the correct value (i.e. it used the wrong linear coefficients βs).
- There is some $j \in [n]$ so that (1) P_d and R are not *iSign-conflicted* with P_i, (2) P_i executed $\mathsf{iSig.open}_1(P_d, P_i, f'_i(j))$ and $\mathsf{iSig.open}_1(R, P_i, \rho'_{R,i,j})$ and opened the values $f'_i(j)$ and $\rho'_{R,i,j}$, and (3) it holds that $\beta_{i,j,R} \neq f'_i(j) + \rho'_{R,i,j}$.
- P_i is vst-conflicted with P_d or R, but $\mathsf{flag}_{i,d} = 0$ in vst or vss.

Remove from W, W', V, V', U, U' the parties in B.

R is **discarded** if one of the following holds.

- There exists an instance of public opening $\mathsf{iSig.open}_1(*, R, *)$ that failed (but did not have *iSign-conflict*), or R did not open the correct value (i.e. it used the wrong linear coefficients βs).
- There exists some P_i, and some $j \in [n]$ so that (1) P_d and P_i are not *iSign-conflicted* with R, (2) R executed $\mathsf{iSig.open}_1(P_d, R, F'(j,i))$ and $\mathsf{iSig.open}_1(P_i, R, \rho_{i,j,R})$ and opened the values $F'(j,i)$ and $\rho'_{i,j,R}$, and (3) it holds that $\beta_{R,i,j} \neq F'(j,i) + \rho'_{i,j,R}$.
- R did not broadcast a complaint against P_d, and there exists $P_i \in U$ that is not *iSign-conflicted* with R, so that the polynomial $h^R_i(x)$, defined by $(\beta_{R,i,j} - \rho'_{i,j,R})_{j\in[n]}$, has degree more than t.
- R did not broadcast a complaint against P_d, and there exists $P_i \in U'$ so that the values $(\alpha_{R,i,j} - \rho'_{R,i,j})_{j\in[n]}$ do not correspond to a degree-t polynomial.
- R did not broadcast a complaint against P_d, and there exist some P_i, P_j that are not *iSign-conflicted* with R, R opened $\rho'_{i,j,R}$ and $\rho'_{j,i,R}$ and it holds that $\beta_{R,i,j} - \rho'_{i,j,R} \neq \beta_{R,j,i} - \rho'_{j,i,R}$.

P_d is **discarded** if one of the following holds.

- P_d is discarded in vss.
- There exists P_i so that P_d is vst-conflicted with P_i, but $\mathsf{flag}^d_i = 0$ in vst or vss. If P_d is not discarded here then $W = V$ and $W' = V'$, and we continue with the notation of W, W'.
- P_d is vst-conflicted with R but $\mathsf{flag}^d_i = 0$ in vst or vss for some $i \in [n]$.

- *(Failed opening)* There exists an instance of public opening $\mathsf{iSig.open}_1(*, P_d, *)$ that failed (but did not have *iSign-conflict*), or P_d did not open the correct value (i.e. it used the wrong linear coefficients βs).
- *(W verification I)* There exists $P_i \in W$ which is not *iSign-conflicted* with P_d so that the polynomial defined by $(\alpha^d_{i,j,R} - \rho^d_{i,j,R})_{j\in[n]}$ is not $f^d_i(x)$, where $f^d_i(x)$ was defined in the local computation of vss.
- *(W verification II)* R is not *iSign-conflicted* with P_d, and there exists $P_i \in W$ for which the polynomial defined by $(\alpha^d_{R,i,j} - \rho^d_{R,i,j})_{j\in[n]}$ is not $f^d_i(x)$.
- *(U verification)* P_d is not *iSign-conflicted* with R, and there exists $P_i \in U$ for which the polynomial defined by the values $(F'(j,i))_{j\in[n]}$ is of degree more than t.
- P_d is not *iSign-conflicted* with R, and there exist $i, j \in [n]$ such that R executed $\mathsf{iSig.open}_1(P_d, R, F'(j,i))$ and $\mathsf{iSig.open}_1(P_d, R, F'(i,j))$ and it holds that $F'(i,j) \neq F'(j,i)$.
- *(Pairwise Consistency between $W \cup W'$ and its complement I)* There exists $P_i \in W \cup W'$ and some P_j, where P_i is not *iSign-conflicted* with P_d in $\mathsf{iSig}(P_i, P_d, \rho^d_{i,j,R})$, and $\mathsf{iSig.open}_1(P_i, P_d, \rho^d_{i,j,R})$ was successfully opened, so that $\alpha^d_{i,j,R} \neq f^d_i(j) + \rho^d_{i,j,R}$.
- *(Pairwise Consistency between $W \cup W'$ and its complement II)* There exists $P_i \in W \cup W'$ and some $j \in [n]$, so that P_d is not *iSign-conflicted* with R, and R opened $F'(j,i)$ so that $F'(j,i) \neq f^d_i(j)$.
- *(Pairwise Consistency between $W \cup W'$ and its complement III)* There exists $P_i \in W \cup W'$ and some $j \in [n]$, so that P_d is not *iSign-conflicted* with R, and R opened $F'(i,j)$ so that $F'(i,j) \neq f^d_i(j)$.
- *(Pairwise Consistency between U and its complement I)* There exists $P_i \in U$ and some P_j, so that P_d is not *iSign-conflicted* with P_i and R, P_i opened $\mathsf{iSig.open}_1(P_d, P_i, f'_i(j))$, R opened $\mathsf{iSig.open}_1(P_d, R, F'(j,i))$, and it holds that $F'(j,i) \neq f'_i(j)$.
- *(Pairwise Consistency between U and its complement II)* There exists $P_i \in U$ and some P_j, so that P_d is not *iSign-conflicted* with P_j and R, P_j opened $\mathsf{iSig.open}_1(P_d, P_i, f'_j(i))$, R opened $\mathsf{iSig.open}_1(P_d, R, F'(j,i))$, and it holds that $F'(j,i) \neq f'_j(i)$.

If P_d is discarded then the parties output $\perp$. Otherwise, they output their shares as defined in vss.

If R is not discarded, then for every $P_i, P_j \notin B$, the (i,j)-th share that R holds is defined as follows.

- If $P_i \in W \cup W'$ then the (i,j)-th share is $f^d_i(j)$.
- Otherwise, if $P_j \in W \cup W'$ then the (i,j)-th share is $f^d_j(i)$.
- Otherwise, if $P_i \in U$ then the (i,j)-th share is set to be the public value $h^R_i(j)$.[a]
- Otherwise, if $P_j \in U$ then the (i,j)-th share is set to be the public value $h^R_j(i)$.
- Otherwise, the (i,j)-th share is defined to be $\beta_{R,i,j} - \rho'_{i,j,R}$, where $\beta_{R,i,j}$ is public, and R can open the signature $\mathsf{iSig}(P_i, R, \rho'_{i,j,R})$.

[a] Observe that P_i is not *iSign-conflicted* with R. Indeed, if P_i is *iSign-conflicted* with R then either $P_i \in W'$ since P_i raises a flag in vss as instructed, or $P_i \in B$. Therefore, $h^R_i(x)$ is well-defined.

Figure 10: Protocol vst

Remark 7.2 (On conflicts between honest parties.)**.** *Consider the case of an honest P_d and a corrupt R, which means that the adversary holds all the information regarding $F(x,y)$. In this case we allow an honest P_i to be conflicted with an honest P_d in the* vss *instance, since this conflict reveals no information that the adversary doesn't already know.*

For completeness, we also provide the reconstruction protocol for R.

Protocol vrec_R

Round 1 (Reconstruction): For every $P_i, P_j \notin B$ so that $P_i, P_j \notin W \cup W' \cup U$, R executes $\mathsf{iSig.open}_2(P_i, R, \rho'_{i,j,R})$.

Local Computation For every party $P_i \notin B$, the parties do as follows.

1. If $P_i \in W \cup W'$, set $f_i(x) := f_i^d(x)$.
2. Otherwise, if $P_i \in U$, set $f_i(x) := h_i^R(x)$.[a]
3. Otherwise, for every $P_j \notin B$ do as follows.
   - If $P_j \in W \cup W'$ set $f_{i,j} := f_j^d(i)$.
   - Otherwise, if $P_j \in U$ set $f_{i,j} := h_j^R(i)$.
   - Otherwise, set $f_{i,j} := \beta_{R,i,j} - \rho'_{i,j,R}$.

   Interpolate over all $(f_{i,j})_{P_j \notin B}$ to obtain a polynomial $f_i(x)$.

If there exists an instance of public opening $\mathsf{iSig.open}_2(*, R, *)$ that failed (but did not have *iSign-conflict*), or R did not open the correct value (i.e. it used the wrong linear coefficients βs), or if for some $P_i \notin B$ the degree of $f_i(x)$ is more than t, or there exist $P_i, P_j \notin B$ so that $f_i(j) \neq f_j(i)$ then output $\perp$. Otherwise, interpolate over $(f_i(x))_{P_i \notin B}$ to obtain $F(x, y)$. Output $F(x, y)$.

[a] Observe that P_i is not *iSign-conflicted* with R. Indeed, if P_i is *iSign-conflicted* with R then either $P_i \in W'$ since P_i raises a flag in vss as instructed, or $P_i \in B$. Therefore, $h_i^R(x)$ is well-defined.

Figure 11: Protocol vrec_R

7.2.1 Analysis

In this section we analyse the properties of protocol vst in the $\mathcal{F}_{\mathsf{iSig}}$-hybrid model. In Section 6 we argued that if P_d is not discarded then the shares of the honest parties are consistent with a unique polynomial $F'(x, y)$ and that $F'(x, y) = F(x, y)$ for an honest P_d (see Lemma 6.5 for a formal statement). We continue by proving that an honest R can open the polynomial $F'(x, y)$ by executing vrec_R, and that a corrupt R can either open $F'(x, y)$ or $\perp$. We observe that in every execution of vst the set B contains only corrupt parties, and we split into cases.

Honest P_d, R. Assume that P_d and R are honest, and that $F(x, y)$ is the input of P_d. Observe that P_d and R are not discarded. We show that for every $P_i, P_j \notin B$ the (i, j)-th share of R is $F(j, i)$. Observe that if $P_i \in W \cup W'$ or $P_j \in W \cup W'$ then the correct shares are made public in the vss execution (see Lemma 6.5). Otherwise, if $P_i \in U$ or $P_j \in U$ then the correct shares are made public in the vst execution, since $h_i^R(x) = F(x, i)$ or $h_j^R(x) = F(x, j)$. Otherwise, the (i, j)-th share is set to be $\beta_{R,i,j} - \rho'_{i,j,R} = F(j, i)$, where $\beta_{R,i,j}$ is public and R can open the value $\rho'_{i,j,R}$. Since there are at least $n - t \geq t + 1$ honest parties, it is not hard to see that by executing vrec_R the parties will recover the polynomial $F(x, y)$.

Honest P_d, corrupt R. Assume that P_d is honest with input $F(x, y)$, and that R is corrupt. Again, we observe that P_d is not discarded. Our goal is to show that if R is not discarded, then for every honest P_i, the receiver R can either open $f_i(x) = F(x, i)$ in vrec_R, or a polynomial $f_i(x)$ of degree more than t. Since there are are $n - t \geq t + 1$ honest parties, this would imply that the only bivariate polynomial that R can open in vrec_R is $F(x, y)$. In order to do so, it is enough to prove that for

every honest P_j, the (i,j)-th share of R is $F(j,i)$. Indeed, since there are $n - t \geq t + 1$ honest parties, this would imply that $f_i(x)$ is either $F(x,i)$ or has degree more than t.

Observe that if $P_i \in W \cup W'$, or $P_j \in W \cup W'$ then the correct shares are made public in the vss execution (see Lemma 6.5) . Otherwise, assume that $P_i \in U$. Since P_i is not in $W \cup W'$, it must hold that $\beta_{R,i,j} = \alpha_{i,j,R} = F(j,i) + \rho_{i,j,R}$ for all $j \in [n]$, and P_i is not *iSign-conflicted* with R. Therefore it must hold that $h_i^R(j) = F(j,i)$, as required. Otherwise, if $P_i \notin U$ but $P_j \in U$, then a similar argument shows that the (i,j)-th share is indeed $F(j,i) = F(i,j)$. Otherwise, if $P_i, P_j \notin U$, it must hold that $\beta_{R,i,j} = \alpha_{i,j,R} = F(j,i) + \rho'_{i,j,R}$, and the (i,j)-th share is set to be $\beta_{R,i,j} - \rho'_{i,j,R} = F(j,i)$, where $\beta_{R,i,j}$ is public and R can open only the value $\rho'_{i,j,R}$, as required.

Corrupt P_d, R. Assume that P_d and R are corrupt, and let $F(x,y)$ be the polynomial defined by the shares of the honest parties in vss (see Lemma 6.5). Our goal is to show that in every execution in which P_d and R are not discarded, for every honest P_i and any $P_j \notin B$, the (i,j)-th share of R is $F(j,i)$. Since there are $n - t \geq t + 1$ honest parties, this would imply that the only polynomial that R can open in vrec_R is $F(x,y)$.

Observe that if $P_i \in W \cup W'$ or $P_j \in W \cup W'$ then the correct shares are made public in the vss execution. Otherwise $P_i, P_j \notin W \cup W'$, which means that P_i and P_j are not *iSign-conflicted* with R, and that $\beta_{R,i,j} = \alpha_{i,j,R}$. We split into cases.

- Assume that $P_i \in U$. Then the (i,j)-th share is set to be $h_i^R(j) = \beta_{R,i,j} - \rho'_{i,j,R} = \alpha_{i,j,R} - \rho'_{i,j,R} = f'_i(j) = F(j,i)$, where the last equality follows from Lemma 6.5.
- Otherwise $P_i \notin U$. Assume that $P_j \in U$. Then R opens $\mathsf{iSig.open}(P_i, R, \rho'_{i,j,R})$, $\mathsf{iSig.open}(P_d, R, F'(i,j))$, $\mathsf{iSig.open}(P_j, R, \rho'_{j,i,R})$, $\mathsf{iSig.open}(P_d, R, F'(j,i))$, and the (i,j)-th share is set to be $h_j^R(i) = \beta_{R,j,i} - \rho'_{j,i,R}$. In addition it must hold that $\beta_{R,j,i} - \rho'_{j,i,R} = \beta_{R,i,j} - \rho'_{i,j,R}$ or otherwise R is discarded. Therefore, the (i,j)-th share is $h_j^R(i) = \beta_{R,j,i} - \rho'_{j,i,R} = \beta_{R,i,j} - \rho'_{i,j,R} = f'_i(j) = F(j,i)$, where the last equality follows from Lemma 6.5.
- Otherwise $P_j \notin U$. Then the (i,j)-th share is $\beta_{R,i,j} - \rho'_{i,j,R}$, where $\beta_{R,i,j}$ is public and R can open $\rho'_{i,j,R}$. Therefore $\beta_{R,i,j} - \rho'_{i,j,R} = \alpha_{i,j,R} - \rho'_{i,j,R} = f'_i(j) = F(j,i)$, where the last equality follows from Lemma 6.5.

Corrupt P_d, honest R. Assume that P_d is corrupt and that R is honest. In Lemma 6.5 we've seen that (1) the shares of the honest parties are consistent with a unique symmetric bivariate polynomial of degree at most t in each variable, denoted $\bar{F}(x,y)$, (2) for every $P_i \in W \cup W'$ the public shares of P_i are consistent with $\bar{F}(x,y)$, and (3) for every $P_i, P_j \notin B$, if P_i is honest or P_j is honest then the (i,j)-th share is $\bar{F}(j,i)$.

Observe that R is never discarded, and consider any execution in which P_d is not discarded. If P_d is vst-conflicted with R, then all the shares of $\bar{F}(x,y)$ are made public in vss, as required. Otherwise, P_d is not vst-conflicted with R, which means that the polynomial $F'(x,y)$ that R received from P_d is a symmetric bivariate polynomial of degree at most t in each variable. Assume that R is vst-conflicted with P_d, and note that R publicly opens the polynomial $F'(x,y)$, and that all parties are in U. We show that for every $P_i, P_j \notin B$ the (i,j)-th share of R is $F'(j,i)$. In addition, if P_i is honest then $F'(i,j) = \bar{F}(j,i)$.

- Assume that $P_i \in W \cup W'$. If $f_i^d(x) \neq F'(x,i)$ then P_d is discarded. Otherwise, the (i,j)-th share is set to be $f_i^d(j) = F'(j,i)$, and since $f_i^d(j) = \bar{F}(j,i)$ then $F'(j,i) = \bar{F}(j,i)$, as required.

- Otherwise $P_i \notin W \cup W'$. Assume that $P_j \in W \cup W'$. Then the same argument shows that the (i,j)-th share is $\bar{F}(j,i) = \bar{F}(i,j) = f_j^d(i) = F'(i,j) = F'(j,i)$, as required.

- Otherwise $P_j \notin W \cup W'$, and $P_i \in U$. This means that P_i is not *iSign-conflicted* with R, and the (i,j)-th share is set to be $h_i^R(j) = \beta_{R,i,j} - \rho'_{i,j,R} = F'(j,i)$, as required.

  Assume that P_i is honest. Then $\beta_{R,i,j} = \alpha_{R,i,j} = f'_i(j) + \rho_{i,j,R}$ or otherwise $P_i \in W \cup W'$. Therefore, $\bar{F}(j,i) = f'_i(j) = F'(j,i)$, as required.

We conclude that all the shares are consistent with the polynomial $F'(x,y)$ so $F'(x,y)$ will be recovered in vrec_R with probability 1. Since there are $n - t \geq t + 1$ honest parties, and since $F'(x,i) = \bar{F}(x,i)$ for every honest P_i, we conclude that $F'(x,y) = \bar{F}(x,y)$, as required.

Otherwise R is not vst-conflicted with P_d. We show that for every honest P_i and every $P_j \notin B$ it holds that (1) the (i,j)-th and the (j,i)-th shares of R are consistent with $\bar{F}(x,y)$, and (2) $F'(j,i) = \bar{F}(j,i)$.

- Assume that $P_i \in W$. Then the polynomial $f_i^d(x) = \bar{F}(x,i)$ is public, and the (i,j)-th and (j,i)-th shares are set to be $f_i^d(j) = \bar{F}(j,i)$ as required. Since $P_i \in W$ it must hold that P_d opened $\rho^d_{R,i,j}$, and since R is not vst-conflicted with P_d it must hold that $\alpha^d_{R,i,j} - \rho^d_{R,i,j} = \alpha_{R,i,j} - \rho^d_{R,i,j} = F'(j,i)$. Since P_d is not discarded it must hold that $f_i^d(j) = \alpha^d_{R,i,j} - \rho^d_{R,i,j}$ so $F'(j,i) = \bar{F}(j,i)$, as required.

- Otherwise $P_i \notin W$. Assume that $P_j \in W$. Then, by using the symmetry of $\bar{F}(x,y)$ and $F'(x,y)$, the same argument shows that the (i,j)-th and (j,i)-th shares are $\bar{F}(j,i) = F'(j,i)$.

- Otherwise $P_i, P_j \notin W$. Assume that $P_i \in W'$. Then the (i,j)-th and (j,i)-th shares are set to be $f_i^d(j) = \bar{F}(j,i)$ as required. We split into cases.

  - Assume that R is vst-conflicted with P_i. Then $P_i \in U$ and R opens $F'(x,i)$, and it must hold that $f_i^d(j) = F'(j,i)$ or otherwise P_d is discarded.
  - Assume that R is not vst-conflicted with P_i, so $\alpha_{R,i,j} = \beta_{i,j,R}$. Since $P_i \in W'$ it must hold that P_i opened $f_i^d(j)$ and $\rho'_{R,i,j}$ (note that P_d and R are not *iSign-conflicted* with P_i). But then $f_i^d(j) + \rho'_{R,i,j} = \beta_{i,j,R} = \alpha_{R,i,j} = F'(j,i) + \rho'_{R,i,j}$, or otherwise $P_i \in B$. Therefore $f_i^d(j) = F'(j,i)$, as required.

- Assume that $P_j \in W'$. Then, by using the symmetry of $\bar{F}(x,y)$ and $F'(x,y)$, the same argument shows that the (i,j)-th and (j,i)-th shares are $\bar{F}(j,i) = F'(j,i)$.

- Otherwise $P_i, P_j \notin W \cup W'$. Assume that $P_i \in U$. Observe that P_i is not *iSign-conflicted* with R, so R opens $h_i^R(x)$ and $h_i^R(x) = F'(x,i)$, so the (i,j)-th and (j,i)-th shares are set to be $F'(j,i)$, as required. In addition, it holds that $\alpha_{R,i,j} = \beta_{i,j,R}$ or otherwise $P_i \in W \cup W'$. Therefore $F'(j,i) = f'_i(j) = \bar{F}(j,i)$, as required.

- Assume that $P_j \in U$. In this case R opens $h_j^R(x) = F'(x,j)$, so the (i,j)-th and (j,i)-th shares are set to be $F'(i,j) = F'(j,i)$, as required. Since $P_i \notin W \cup W'$ then $\alpha_{R,i,j} = \beta_{i,j,R}$, so $F'(j,i) = f'_i(j) = \bar{F}(j,i)$, as required.

- Otherwise $P_i, P_j \notin U$. But then the (i,j)-th share is $F'(j,i) = \beta_{R,i,j} - \rho'_{i,j,R} = \alpha_{i,j,R} - \rho'_{i,j,R} = f'_i(j) = \bar{F}(j,i)$, as required. Similarly, the (j,i)-th share is $\beta_{R,j,i} - \rho'_{j,i,R} = \alpha_{j,i,R} - \rho'_{j,i,R} = F'(i,j) = F'(j,i)$, and we've seen that $F'(j,i) = \bar{F}(j,i)$, as required.

Since $\bar{F}(x,i) = F'(x,i)$ for every honest P_i, and since there are at least $n - t \geq t + 1$ honest parties, we conclude that $\bar{F}(x,y) = F'(x,y)$. It remains to show that for corrupt $P_i, P_j \notin B$ the (i,j)-th share is $\bar{F}(j,i)$. By the above analysis, if P_i or P_j are in $W \cup W' \cup U$ then the above holds. Consider any $P_i, P_j \notin B$ so that $P_i, P_j \notin W \cup W' \cup U$. Then the (i,j)-th share is $\beta_{R,i,j} - \rho'_{i,j,R} = F'(j,i) = \bar{F}(j,i)$, as required. This concludes the analysis.

We proved the following lemma.

Lemma 7.3. *Consider any execution of* vst *in the $\mathcal{F}_{\mathsf{iSig}}$-hybrid model. Then an honest P_d is not discarded, an honest R is not discarded, and the set B contains no honest parties.*

For an honest P_d with input $F(x,y)$, if R is honest then for every $P_i, P_j \notin B$ the (i,j)-th share is $F(j,i)$, and by executing vrec_R, an honest R opens the polynomial $F(x,y)$ with probability 1. On the other hand, a corrupt R can either open $F(x,y)$ or $\perp$.

For a corrupt P_d that was not discarded, let $F'(x,y)$ be the polynomial defined by the shares of the honest parties in vss*. Then for an honest R, for every $P_i, P_j \notin B$ the (i,j)-th share of R is $F'(j,i)$, and by executing vrec_R, an honest R opens the polynomial $F'(x,y)$ with probability 1. On the other hand, a corrupt R can either open $F'(x,y)$ or $\perp$.*

8 Share and Compute

Here we present the share-compute functionality $\mathcal{F}_{\mathsf{sh\text{-}comp}}$ and realize it in a protocol sh-comp.

8.1 The Share-compute Functionality

The functionality $\mathcal{F}_{\mathsf{sh\text{-}comp}}$ captures and formalizes our requirements from VSS, TSS and VST, including the execution of linear operations over tentative shares. We continue with an explanation of each task that the functionality performs.

Sharing polynomials. Every party P_i inputs m polynomials $G^{i,1}(x,y), \ldots, G^{i,m}(x)$ that will be shared among the parties, i.e., every P_j receives $G^{i,1}(x,j), \ldots, G^{i,m}(x,j)$ from the functionality. The functionality also supports *triple sharing*, where every P_i also inputs additional polynomials $H^{i,j,k}(x,y)$ so that $H^{i,j,k}(0,0) = G^{i,j}(0,0) \cdot G^{i,k}(0,0)$ for every $j,k \in [m]$, and the functionality shares the polynomials among the parties, i.e., every $P_{i'}$ receives $H^{i,j,k}(x,i')$. Finally, the functionality supports the *sharing and transferring* operation, where P_i inputs ℓ polynomials $F^{i,1}(x,y), \ldots, F^{i,\ell}(x,y)$, the functionality shares the polynomials among the parties, and also transfers $F^{i,j}(x,y)$ to party $P_{\phi(i,j)}$ where $\phi : [n] \times [\ell] \to [n]$ is a mapping that, for every $i \in [n]$ and $v \in [\ell]$, specifies the receiver of $F^{i,v}(x,y)$. As part of the sharing and transferring, the functionality allows $P_{\phi(i,v)}$ to publicly open $F^{i,v}(x,y)$ at the opening phase.

Linear operations. The functionality also supports two kinds of linear operations over the shares.[15] First, for every P_i the functionality supports performing linear operations over the

[15] When we realize the functionality, we show that we can perform linear operations over the *tentative shares* to save rounds.

shares that P_i distributed. This means that the functionality is parameterized by coefficients $(\beta^v_{i,j}, \gamma^v_{i,j,k})_{i\in[n],j,k\in[m],v\in[\ell]}$ and returns

$$\sum_{j=1}^{m} \beta^v_{i,j} G^{i,j}(x,y) + \sum_{j,k\in[m]} \gamma^v_{i,j,k} H^{i,j,k}(x,y) + F^{i,v}(x,y)$$

to all the parties, for every $v \in [\ell]$. Since $H^{i,j,k}(0,0) = G^{i,j}(0,0) \cdot G^{i,k}(0,0)$ for every $j,k \in [m]$ this allows the computation of degree-2 functions where all the inputs belong to the same party. Looking forward, when using $\mathcal{F}_{\mathsf{sh\text{-}comp}}$ we will think of $F^{i,v}(x,y)$ as a random pad, so that only party $P_{\phi(i,v)}$ can learn $\sum_{j=1}^{m} \beta^v_{i,j} G^{i,j}(x,y) + \sum_{j,k\in[m]} \gamma^v_{i,j,k} H^{i,j,k}(x,y)$. Recall that $P_{\phi(i,v)}$ can publicly open $F^{i,v}(x,y)$, so $P_{\phi(i,v)}$ will be able to reveal $\sum_{j=1}^{m} \beta^v_{i,j} G^{i,j}(x,y) + \sum_{j,k\in[m]} \gamma^v_{i,j,k} H^{i,j,k}(x,y)$ to all the parties if needed.

The functionality also supports linear operations between shares of two different dealers. Formally, the functionality is parameterized by a set S of tuples $((i_1,v_1),(i_2,v_2)) \in (([n]\times[\ell])\times([n]\times[\ell]))$. The functionality returns $F^{i_1,v_1}(x,y) - F^{i_2,v_2}(x,y)$ to all the parties.

Communication with the adversary. The functionality allows the adversary to choose the inputs of the corrupt parties, after seeing the following leakage: (1) the shares of the corrupt parties from the polynomials that the honest parties input, (2) the outputs of the linear computation over the shares that the honest parties distributed, (3) $F^{i_1,v_1}(x,y) - F^{i_2,v_2}(x,y)$ for $((i_1,v_1),(i_2,v_2)) \in S$ with honest P_{i_1}, P_{i_2}, (4) $F^{i_1,v_1}(x,y)$ for $((i_1,v_1),(i_2,v_2)) \in S$ with corrupt P_{i_2}, and (5) $F^{i_2,v_2}(x,y)$ for $((i_1,v_1),(i_2,v_2)) \in S$ with corrupt P_{i_1}. Looking forward, we will need to make sure that our protocol is secure even if the adversary can choose its inputs based on the leakage.

We allow the adversary to cause every corrupt P_i to abort, by inputting $\mathsf{flag}_i = 1$ to the functionality. The functionality returns the set I, of all aborting corrupt parties, to all the parties.

Functionality $\mathcal{F}_{\mathsf{sh\text{-}comp}}$

The functionality is parametrized as follows. There is an input parameter $m \in \mathbb{N}$, an output parameter $\ell \in \mathbb{N}$, a function $\phi : [n]\times[\ell] \to [n]$ and coefficients $(\beta^v_{i,j}, \gamma^v_{i,j,k})_{i\in[n],j,k\in[m],v\in[\ell]}$. There is also a set $S \subseteq (([n]\times[\ell])\times([n]\times[\ell]))$ containing tuples $((i_1,v_1),(i_2,v_2))$ so that $\phi(i_1,v_1) = \phi(i_2,v_2)$ and every pair (i,v) appears in at most one tuple $((i_1,v_1),(i_2,v_2))$ in S.

The functionality receives the set of corrupt parties C.

Input phase.

- *(Triple sharing)* Every honest party P_i inputs m symmetric bivariate polynomials of degree at most t in each variable $G^{i,1}(x,y),\ldots,G^{i,m}(x,y)$ as well as m^2 symmetric bivariate polynomials of degree at most t in each variable $(H^{i,j,k}(x,y))_{j,k\in[m]}$ so that $H^{i,j,k}(0,0) = G^{i,j}(0,0)\cdot G^{i,k}(0,0)$ for every $j,k\in[m]$.
- *(Share and Transfer)* Every honest party P_i inputs ℓ symmetric bivariate polynomials of degree at most t in each variable $F^{i,1}(x,y),\ldots,F^{i,\ell}(x,y)$.
- *(Leakage)* The values $(F^{i,v}(x,i'), G^{i,j}(x,i'), H^{i,j,k}(x,i'))_{i\in\mathsf{H},i'\in\mathsf{C},j,k\in[m],v\in[\ell]}$ are leaked to the adversary. In addition, for every corrupt $P_{i'}$, and every $i\in\mathsf{H}$ and $v\in[\ell]$ so that $\phi(i,v) = i'$, the polynomial $F^{i,v}(x,y)$ is leaked to the adversary.

Linear computation phase.

- *(Leakage)* The functionality leaks the following to the adversary. For every honest P_i the functionality leaks $\left(\sum_{j=1}^{m} \beta_{i,j}^{v} G^{i,j}(x,y) + \sum_{j,k\in[m]} \gamma_{i,j,k}^{v} H^{i,j,k}(x,y) + F^{i,v}(x,y)\right)_{v\in[\ell]}$ to the adversary. In addition, for every $((i_1,v_1),(i_2,v_2)) \in S$ so that P_{i_1} and P_{i_2} are honest, the functionality returns $F^{i_1,v_1}(x,y) - F^{i_2,v_2}(x,y)$ to the adversary. Finally, for every $((i_1,v_1),(i_2,v_2)) \in S$ so that P_{i_1} is honest and P_{i_2} is corrupt, the functionality returns $F^{i_1,v_1}(x,y)$ to the adversary, and for every $((i_1,v_1),(i_2,v_2)) \in S$ so that P_{i_2} is honest and P_{i_1} is corrupt, the functionality returns $F^{i_2,v_2}(x,y)$ to the adversary.
- *(Corrupt parties' inputs)* Every corrupt P_i inputs m symmetric bivariate polynomials of degree at most t in each variable $G^{i,1}(x,y),\ldots,G^{i,m}(x,y)$ as well as m^2 symmetric bivariate polynomials of degree at most t in each variable $(H^{i,j,k}(x,y))_{j,k\in[m]}$ and a bit flag_i. Let I be the set of indices of all corrupt P_i with $\mathsf{flag}_i = 1$, or such that $H^{i,j,k}(0,0) \neq G^{i,j}(0,0)\cdot G^{i,k}(0,0)$ for some $j,k\in[m]$.
- *(Shares distribution)* The functionality returns

$$I, \quad (F^{i',v}(x,i), G^{i',j}(x,i), H^{i',j,k}(x,i))_{i'\notin I, j,k\in[m], v\in[\ell]}, \quad \text{and} \quad (F^{i',v}(x,y))_{i'\notin I, v\in[\ell]:\phi(i',v)=i)}$$

  to P_i.
- *(Public linear computation)* For every $i \notin I$, the functionality returns

$$\left(\sum_{j=1}^{m} \beta_{i,j}^{v} G^{i,j}(x,y) + \sum_{j,k\in[m]} \gamma_{i,j,k}^{v} H^{i,j,k}(x,y) + F^{i,v}(x,y)\right)_{v\in[\ell]}$$

  to all parties.
- In addition, for every $((i_1,v_1),(i_2,v_2)) \in S$ the functionality returns
  - *($i_1, i_2 \notin I$).* The tuple $((i_1,v_1),(i_2,v_2)), F^{i_1,v_1}(x,y) - F^{i_2,v_2}(x,y))$ to all parties.
  - *(Otherwise.)* The tuple $((i_1,v_1),(i_2,v_2)),\perp)$ to all parties.

Opening phase.

- *(Inputs of honest parties)* Every honest P_i inputs an integer $\mu_i \in \mathbb{N}$, and pairs $(j_{i,k},v_{i,k})_{k\in[\mu_i]}$ so that $j_{i,k} \in [n]\setminus I$, $v_{i,k}\in[\ell]$ and $\phi(j_{i,k},v_{i,k}) = i$. (The functionality just ignores pairs that are not of this form.)
- *(Leakage)* For every honest P_i the functionality returns $(\mu_i, (j_{i,k}, v_{i,k}, F^{j_{i,k},v_{i,k}}(x,y))_{k\in[\mu_i]}))$ to the adversary.
- *(Inputs of corrupt parties)* Every corrupt P_i inputs a bit abort_i, an integer $\mu_i \in \mathbb{N}$, and pairs $(j_{i,k},v_{i,k})_{k\in[\mu_i]}$ so that $j_{i,k} \in [n]\setminus I$, $v_{i,k}\in[\ell]$ and $\phi(j_{i,k},v_{i,k}) = i$. (The functionality just ignores pairs that are not of this form.)
- *(Output)* Let I' be the set I together with all the corrupt parties with $\mathsf{abort}_i = 1$. The functionality returns

$$(I', (\mu_i, j_{i,k}, v_{i,k}, F^{j_{i,k},v_{i,k}}(x,y))_{i\notin I', k\in[\mu_i]}).$$

  to all the parties.

Figure 12: Functionality $\mathcal{F}_{\mathsf{sh\text{-}comp}}$

In the next sections we realize $\mathcal{F}_{\mathsf{sh\text{-}comp}}$ by a protocol sh-comp. As always, we assume that $\mathbb{F}$ is a sufficiently large field of size $\exp(n,\kappa,\ell,m)$, that allows the execution of iSig as required

by sh-comp. We also assume that ϕ and S can be computed in time T. We prove the following statement in Section E.

Theorem 8.1. *Let κ be a security parameter, let n be the number of parties and let $t < n/2$ the number of corrupt parties. Let $\mathbb{F}$ be a sufficiently large finite field, as a function of (κ, n, ℓ, m). Then protocol* sh-comp *is a UC-secure implementation of $\mathcal{F}_{\text{sh-comp}}$ against a static, active, rushing adversary corrupting up to t parties. The complexity of* sh-comp *is* $\text{poly}(\kappa, 2^n, m, \ell, T, \log |\mathbb{F}|)$.

8.2 The Share-compute Protocol

The share-compute protocol requires multiple executions of vss, tss and vst. As in Section 6.3, we assume that all instances of vss, tss and vst use *the same* underlying signature scheme iSig where P_i is $\mathcal{D}$ and P_j is $\mathcal{I}$. First, in Section 8.2.1 and Section 8.2.2 we present two subprotocols for the computation of linear operations over tentative shares. Finally, in Section 8.2.3 we present the protocol sh-comp.

8.2.1 Linear Operations over Tentative Shares from Different Dealers

Consider the case where $\mathcal{D}_1$ and $\mathcal{D}_2$ share $F^1(x,y)$ and $F^2(x,y)$, respectively, via vss^1 and vss^2, and the parties want to reveal $F^1(x,y) - F^2(x,y)$ in the third round. For $k \in \{1,2\}$, in the execution of vss^k we denote the random pad that P_i sent to P_j by $r^k_{i,j}$, the broadcast messages of P_i by $(a^k_{i,j}, b^k_{i,j})_{j\in[n]}$, the set of bad players in vss^k by B^k, and the sets W, W' by W^k and $(W^k)'$. We remind the reader that, at the end of the second round of a vss execution, a party P_i thinks that its tentative shares might change if either (1) P_i received a polynomial of degree more than t in the first round, or (2) P_i is *iSign-conflicted* with P_d, or (3) there exists some $j \in [n]$ so that $a^d_{i,j} \neq a_{i,j}$ (see remark 6.6).

The idea underlying our protocol is similar to that of linop_1. We let every P_i who thinks that its shares might change in vss^1 to raise a flag in vss^1, and also publicly open its shares in vss^2, so that the parties will be able to recover $F^1(x,i) - F^2(x,i)$ from the public shares. We note that this does not violate privacy, because if P_i is honest and thinks that its shares might change in vss^1, then necessarily $\mathcal{D}_1$ is corrupt and there is no need for privacy, because the adversary can learn the polynomial $F^2(x,y)$ from $F^1(x,y) - F^2(x,y)$. P_i acts in a similar way if he thinks that its shares might change in vss^2. If P_i does not think that its share might change, then for every $P_j \notin B$, we let P_i open $r^1_{j,i} - r^2_{j,i}$ so the parties can compute $(b^1_{i,j} - b^2_{i,j}) - (r^1_{j,i} - r^2_{j,i}) = (b^1_{i,j} - r^1_{j,i}) - (b^2_{i,j} - r^2_{j,i})$. If P_i is honest, then, since P_i does not think that its share might change, this is indeed the correct (i,j)-th share. On the other hand, if P_i is corrupt, then P_i has to be consistent with at least $n - t \geq t + 1$ honest parties, which means that P_i's shares will either correspond to a degree-t polynomial, or set to an erasure. We continue with the description of the protocol.

Protocol linop_2

Assumption We assume that the first and the second round of all vss instances were executed. In addition, we assume that in the third round of the outer protocol, the parties are instructed to execute the vss instances as follows.

- For every $k \in \{1,2\}$, if P_i thinks that its share might change in vss^k then P_i is instructed to raise a flag $\mathsf{flag}_i^k = 1$ in vss^k.
- Let flag_i^1 and flag_i^2 be the flags of P_i in vss_1 and vss_2, respectively. If $\mathsf{flag}_i^1 = \mathsf{flag}_i^2 = 0$, then P_i is instructed to do as follows.
  - If P_i is internally-vss-conflicted with P_j in vss^1 and vss^2 then P_i sets $\mathsf{flag}_{i,j}^1 = \mathsf{flag}_{i,j}^2 = 1$ in vss^1 and vss^2.
  - If P_i is internally-vss-conflicted with P_j in vss^1 then P_i sets $\mathsf{flag}_{i,j}^1 = 1$ in vss^1.
  - If P_i is internally-vss-conflicted with P_j in vss^2 then P_i sets $\mathsf{flag}_{i,j}^2 = 1$ in vss^2.

Round 3 Every P_i inputs the flags $(\mathsf{flag}_i^k, \mathsf{flag}_{i,j}^k)_{j\in[n],k\in\{1,2\}}$. (Those are the flags that the players input into vss^1 and vss^2.)

Every P_i does as follows.

- If $\mathsf{flag}_i^1 = \mathsf{flag}_i^2 = 1$, then P_i broadcasts "complaint:1,2".
- Otherwise, if $\mathsf{flag}_i^1 = 1$ and $\mathsf{flag}_i^2 = 0$, then P_i broadcasts "complaint:1". In addition, P_i executes $\mathsf{iSig.open}_1(P_j, P_i, r_{j,i}^2)$ for all $j \in [n]$.
- Otherwise, if $\mathsf{flag}_i^2 = 1$ and $\mathsf{flag}_i^1 = 0$, then P_i broadcasts "complaint:2". In addition, P_i executes $\mathsf{iSig.open}_1(P_j, P_i, r_{j,i}^1)$ for all $j \in [n]$.
- Otherwise $\mathsf{flag}_i^1 = \mathsf{flag}_i^2 = 0$. P_i does as follows for every P_j.
  - If $\mathsf{flag}_{i,j}^1 = \mathsf{flag}_{i,j}^2 = 1$ the P_i does nothing.
  - If $\mathsf{flag}_{i,j}^1 = 1$ and $\mathsf{flag}_{i,j}^2 = 0$ then P_i executes $\mathsf{iSig.open}_1(P_j, P_i, r_{j,i}^2)$.
  - If $\mathsf{flag}_{i,j}^1 = 0$ and $\mathsf{flag}_{i,j}^2 = 1$ then P_i executes $\mathsf{iSig.open}_1(P_j, P_i, r_{j,i}^1)$.
  - If $\mathsf{flag}_{i,j}^1 = \mathsf{flag}_{i,j}^2 = 0$ the P_i executes $\mathsf{iSig.open}_1(P_j, P_i, r_{j,i}^1 - r_{j,i}^2)$.

Local computation The parties do as follows.

- Execute the local computation of vss^1 and vss^2, and observe that all the flags are public at the end of the vss executions. If $\mathcal{D}_1$ or $\mathcal{D}_2$ were discarded then the parties output $\perp$ and terminate.
- Compute $B = B^1 \cup B^2$.
- Add to B every P_i so that there exists an instance of public opening $\mathsf{iSig.open}_1(*, P_i, *)$ that failed (but did not have *iSign-conflict*), or P_i did not open the correct value (i.e. it used the wrong linear coefficients βs).
- Add to B every P_i that raised a flag in some vss execution but did not broadcast the corresponding complaint.
- Add to B every P_i that broadcasted "complaint:1,2" so that $P_i \notin W^1 \cup (W^1)'$ or $P_i \notin W^2 \cup (W^2)'$.
- Add to B every P_i that broadcasted "complaint:1" so that $P_i \notin W^1 \cup (W^1)'$.
- Add to B every P_i that broadcasted "complaint:2" so that $P_i \notin W^2 \cup (W^2)'$.
- For every $P_i \notin B$ that broadcasted a complaint, the parties do as follows.
  - If $P_i \in W^1 \cup (W^1)'$, then let $f_i^1(x)$ be the public polynomial that corresponds to the shares of P_i in vss^1. Otherwise, $P_i \notin W^1 \cup (W^1)'$, which means that P_i did not broadcast a complaint regarding vss^1, but did broadcast a complaint regarding vss^2. Since P_i executed $\mathsf{iSig.open}_1(P_j, P_i, r_{j,i}^1)$ for every P_j so that $\mathsf{flag}_{i,j}^1 = 0$, the (i,j)-th share of P_i in vss^1 is public for every $P_j \notin B$, and let $f_i^1(x)$ be the polynomial obtained by interpolating all the shares. If the degree of $f_i^1(x)$ is more than t then add P_i to B.

- If $P_i \in W^2 \cup (W^2)'$, then let $f_i^2(x)$ be the public polynomial that corresponds to the shares of P_i in vss^2. Otherwise, $P_i \notin W^2 \cup (W^2)'$, which means that P_i did not broadcast a complaint regarding vss^2, but did broadcast a complaint regarding vss^1. Since P_i executed $\mathsf{iSig.open}_1(P_j, P_i, r_{j,i}^2)$ for every P_j so that $\mathsf{flag}_{i,j}^2 = 0$, the (i,j)-th share of P_i in vss^2 is public for every $P_j \notin B$, and let $f_i^2(x)$ be the polynomial obtained by interpolating all the shares. If the degree of $f_i^2(x)$ is more than t then add P_i to B.
- Set $f_i(x) := f_i^1(x) - f_i^2(x)$.

- For every $P_i \notin B$ that did not broadcast a complaint, the parties do as follows.
  - If both $\mathcal{D}_1$ and $\mathcal{D}_2$ were vss-conflicted with P_i, then $P_i \in W^1$ and $P_i \in W^2$, and let $f_i^1(x)$ and $f_i^2(x)$ be the public polynomials corresponding to the shares of P_i. The parties compute $f_i(x) := f_i^1(x) - f_i^2(x)$.
  - Otherwise, for every $P_j \notin B$ the parties do as follows.
    * If P_j broadcasted a complaint, then set $f_{i,j} := f_j(i)$.
    * If P_i is vss-conflicted with P_j in some vss instance, then the (i,j)-th shares are public in both instances, and we denote them by $f_{i,j}^1, f_{i,j}^2$. The parties compute $f_{i,j} := f_{i,j}^1 - f_{i,j}^2$.
    * If P_j is vss-conflicted with P_i in some vss instance, then the (j,i)-th shares are public in both instances, and we denote them by $f_{i,j}^1, f_{i,j}^2$. The parties compute $f_{i,j} := f_{i,j}^1 - f_{i,j}^2$.
    * Otherwise P_i and P_j are not vss-conflicted in any vss execution. In this case we set $f_{i,j} := (b_{i,j}^1 - b_{i,j}^2) - (r_{j,i}^1 - r_{j,i}^2)$, where $b_{i,j}^1$ and $b_{i,j}^2$ are public values, and $(r_{j,i}^1 - r_{j,i}^2)$ was opened by P_i.

    The parties interpolate over $(f_{i,j})_{P_j \notin B}$ in order to obtain a polynomial $f_i(x)$. If $f_i(x)$ is of degree more than t then the parties add P_i to B. Otherwise, the i-th share is taken to be $f_i(x)$.
- Finally, the parties interpolate $(f_i(x))_{P_i \notin B}$ in order to obtain a symmetric bivariate polynomial of degree at most t in each variable $F(x,y)$, and output $F(x,y)$.

Figure 13: Protocol linop_2

We continue with an analysis of the correctness in the $\mathcal{F}_{\mathsf{iSig}}$-hybrid model in an execution in which the parties are instructed according to our assumptions. Recall that in any vss execution in which the dealer is not discarded, the shares of the honest parties fully define a symmetric bivariate polynomial of degree at most t in each variable (see Lemma 6.5). Consider any execution of vss^1 and vss^2 in which the dealers were not discarded, let $F^1(x,y)$ and $F^2(x,y)$ be the corresponding polynomials, and observe that the set B does not contain any honest party. We start by showing that for any $P_i \notin B$ that broadcasts a complaint it holds that $f_i(x) = F^1(x,i) - F^2(x,i)$. We split into cases.

- If $P_i \in W^1 \cup (W^1)'$ then $f_i^1(x) = F^1(x,i)$ by Lemma 6.5.
- Otherwise, if P_i is honest, then by Lemma 6.5, for every $P_j \notin B^1$ the (i,j)-th share of P_i is $F^1(j,i)$, and since there are at least $n - t \geq t + 1$ honest parties that are not in B, it holds that the (i,j)-th shares for all $P_j \notin B$ fully determine the polynomial $F^1(x,i)$, so $f_i^1(x) = F^1(x,i)$.
- Otherwise P_i is corrupt. By Lemma 6.5 for every honest P_j it holds that the (i,j)-th share of P_i is $F^1(j,i)$, and since there are at least $n - t \geq t + 1$ honest parties that are not in B either $f_i^1(x) = F^1(x,i)$, or $f_i^1(x)$ has degree more than t and $P_i \in B$.

A similar argument shows that $f_i^2(x) = F^2(x,i)$. Therefore, $f_i(x) = F^1(x,i) - F^2(x,i)$, as required.

Consider now every honest P_i that did not broadcast a complaint. If $P_i \in W^1$ and $P_i \in W^2$ then $f_i^1(x) = F^1(x,i)$ and $f_i^2(x) = F^2(x,i)$, so $f_i(x) = F^1(x,i) - F^2(x,i)$, as required. Otherwise, consider any $P_j \notin B$. We show that $f_{i,j} = F^1(j,i) - F^2(j,i)$.

- If P_j broadcasted a complaint then by the analysis above $f_{i,j} = f_j(i) = F^1(i,j) - F^2(i,j) = F^1(j,i) - F^2(j,i)$, as required.
- Otherwise P_j did not broadcast a complaint. If P_i is vss-conflicted with P_j in some vss instance, then the (i,j)-th shares are public in both executions, and by Lemma 6.5 it holds that $f_{i,j}^1 = F^1(j,i)$ and $f_{i,j}^2 = F^2(j,i)$, so $f_{i,j} = F^1(j,i) - F^2(j,i)$, as required. (Observe that this is true even if P_i or P_j are in W^1 or W^2.)
- Otherwise, if P_j is vss-conflicted with P_i in some vss instance,then the (j,i)-th shares are public in both executions, and by Lemma 6.5 it holds that $f_{i,j}^1 = F^1(j,i)$ and $f_{i,j}^2 = F^2(j,i)$, so $f_{i,j} = F^1(j,i) - F^2(j,i)$, as required.
- Otherwise, P_i and P_j are not vss-conflicted in any instance. Since P_i is honest and P_i did not complain, then the shares of P_i are equal to $f_i'(j)$ even if $P_i \in W^1$ or $P_i \in W^2$ (see Remark 6.6). In particular, the (i,j)-th share in vss^1 and vss^2 are equal to $b_{i,j}^1 - r_{j,i}^1$ and $b_{i,j}^2 - r_{j,i}^2$, respectively. Therefore, $f_{i,j} = (b_{i,j}^1 - b_{i,j}^2) - (r_{j,i}^1 - r_{j,i}^2) = (b_{i,j}^1 - r_{j,i}^1) - (b_{i,j}^2 - r_{j,i}^2) = F^1(j,i) - F^2(j,i)$, as required.

Consider now any corrupt $P_j \notin B$ that did not broadcast a complaint. If $P_i \in W^1$ and $P_i \in W^2$ then $f_i^1(x) = F^1(x,i)$ and $f_i^2(x) = F^2(x,i)$, so $f_i(x) = F^1(x,i) - F^2(x,i)$, as required. Otherwise, we show that $f_{i,j} = F^1(j,i) - F^2(j,i)$ for every honest P_j. Since there are at least $n - t \geq t + 1$ honest parties, this means that either $f_i(x) = F^1(x,i) - F^2(x,i)$ or $f_i(x)$ has degree more than t, in which case $P_j \in B$.

The same arguments as before show that if P_j broadcasts a complaint, or if P_i and P_j are vss-conflicted, then $f_{i,j} = F^1(j,i) - F^2(j,i)$. Therefore, we only analyse the case where P_i and P_j are not vss-conflicted in any instance of vss. Since P_j is honest and not vss-conflicted with P_i it must hold that $b_{i,j}^k = a_{j,i}^k = f_j^k(i) + r_{j,i}^k$ for every $k \in \{1,2\}$. In addition, since P_j is honest and did not broadcast a complaint, by Lemma 6.5 and Remark 6.6 it must hold that the (j,i)-th share in vss^k is $f_j^k(i) = F^k(i,j)$. But then $f_{i,j} = (b_{i,j}^1 - b_{i,j}^2) - (r_{j,i}^1 - r_{j,i}^2) = (b_{i,j}^1 - r_{j,i}^1) - (b_{i,j}^2 - r_{j,i}^2) = F^1(j,i) - F^2(j,i)$, as required.

Finally, since there are $n - t \geq t + 1$ honest parties not in B, and since for every $P_i \notin B$ it holds that $f_i(x) = F^1(x,i) - F^2(x,i)$, then the polynomial recovered by the parties is $F^1(x,y) - F^2(x,i)$, as required.

We proved the following lemma.

Lemma 8.2 (Correctness of linop_2)**.** *Consider any execution of* vss^1, vss^2 *and* linop_2 *in the* $\mathcal{F}_{\mathsf{iSig}}$*-hybrid model, where the parties are instructed according to the assumptions, and* $\mathcal{D}_1$ *and* $\mathcal{D}_2$ *are not discarded. Then the set* B *does not contain honest parties, and the output of the honest parties is* $F^1(x,y) - F^2(x,y)$*, where* $F^1(x,y)$ *and* $F^2(x,y)$ *are the polynomials defined by the shares of the honest parties in* vss^1 *and* vss^2*.*

8.2.2 Private Linear Operations over Tentative Shares from a Single Dealer with Verifiable Opening

Consider the case where the dealer P_d shares $F^1(x,y),\ldots,F^m(x,y)$ via vss and wants to let a receiver R learn a linear combination

$$H(x,y) := \beta_1 F^1(x,y) + \ldots \beta_m F^m(x,y),$$

in three rounds, where the linear coefficients $\beta_1,\ldots,\beta_m \in \mathbb{F}$ are known to all the parties at the beginning of the execution. In addition, R should be able to choose, based on external reasons, whether or not to open the polynomial $H(x,y)$ to the rest of the parties in the fourth round. If R chooses to open $H(x,y)$, then R should be able to convince the rest of the parties of the correctness of this polynomial.

In order to do so, we let P_d share another random polynomial $F^{m+1}(x,y)$ via vst with R as the receiver, and we would like the parties to compute the public value

$$\beta_1 F^1(x,y) + \ldots \beta_m F^m(x,y) + F^{m+1}(x,y) = H(x,y) + F^{m+1}(x,y).$$

In this way (1) the parties have no information about $H(x,y)$, as $F^{m+1}(x,y)$ is used as a one-time pad, (2) R knows $F^{m+1}(x,y)$ so he can compute $H(x,y)$, and (3) R can choose to open $F^{m+1}(x,y)$ via the reconstruction of vst and let all the parties learn the value of $H(x,y)$.

We denote the vss execution of $F^i(x,y)$ by vss^i, the vst execution of $F^{m+1}(x,y)$ by vst^{m+1}, and the internal vss execution by vss^{m+1}. In vss^k, we denote the random pad that P_i sent to P_j by $r^k_{i,j}$, and the broadcast messages of P_i by $(a^k_{i,j}, b^k_{i,j})_{j\in[n]}$. We denote the set of bad parties in vss^k by B^k for $k \in [m]$, and for vst^{m+1} by B^{k+1}. The protocol follows the same lines as linop_1, with the following modifications: (1) if P_d is not internally-vst-conflicted with P_i in the vst execution, but P_d raises flag^d_i in the vst execution, then P_d *is not* required to raise a flag in all vss instances, since it is possible that P_d raises the flag because of a corrupt R, and (2) if P_i is not internally-vst-conflicted with P_i in the vst execution, but P_i raises flag_i in the vst execution, then P_i *is not* required to raise a flag in all vss instances, since it is possible that P_i raises the flag because of a corrupt R. The protocol is described in Figure 14.

Protocol linop_3

Assumptions We assume that the first and the second round of all vss and vst instances were executed. In addition, we assume that in the third round of the outer protocol, the parties are instructed to execute the vss and vst instances in the following way. P_d does as follows.

- For every $i \in [n]$, if P_d is internally-vss-conflicted with P_i, or $\mathsf{flag}^d_i = 1$ in some vss^k execution for $k \in [m]$, or P_d is internally-vst-conflicted with P_i then P_d inputs $\mathsf{flag}^d_i = 1$ to all vss and vst executions. Otherwise, if P_d is internally-vst-conflicted with R or $\mathsf{flag}^d_R = 1$, then P_d inputs $\mathsf{flag}^d_i = 1$ to vst^{m+1} and vss^{m+1}.
- If P_d is internally-vst-conflicted with R then P_d inputs $\mathsf{flag}^d_R = 1$ to vst^{m+1}.

R does as follows.

- If R is internally-vst-conflicted with P_d then R inputs $\mathsf{flag}^R_d = 1$ to the vst executions.

- If R is internally-vst-conflicted with P_i, or if R is internally-vst-conflicted with P_d or $\mathsf{flag}_d^R = 1$, then R inputs $\mathsf{flag}_i^R = 1$ to vst.

Every P_i does as follows.

- If P_i is internally-vss-conflicted with P_d in any vss execution, or $\mathsf{flag}_i = 1$ in some vss^k execution for $k \in [m]$, or P_i is internally-vst-conflicted with P_d in the vst execution, then P_i inputs $\mathsf{flag}_i = 1$ to all vss and vst executions. Otherwise, if P_i is internally-vst-conflicted with R, or $\mathsf{flag}_{i,R} = 1$ in the vst execution, then P_i inputs $\mathsf{flag}_{i,d} = 1$ to vst^{m+1} and vss^{m+1}.
- If P_i is internally-vst-conflicted with R then P_i inputs $\mathsf{flag}_{i,R} = 1$ to the vst execution.
- If P_i is internally-vss-conflicted with P_j, or $\mathsf{flag}_{i,j} = 1$ in some vss or vst execution, then P_i sets $\mathsf{flag}_{i,j} = 1$ in all vss and vst executions.

Round 3 Every P_i inputs $(\mathsf{flag}_i^k, \mathsf{flag}_{i,j}^k)_{j\in[n],k\in[m]}$ and $(\mathsf{flag}_{i,d}^{m+1}, \mathsf{flag}_{i,R}^{m+1}, \mathsf{flag}_{i,j}^{m+1})_{j\in[n]}$. (Those are the flags to the vss and vst instances.)

Every P_i does as follows.

- If $\mathsf{flag}_i^1 = \ldots = \mathsf{flag}_i^m = 0$ and $\mathsf{flag}_{i,R}^{m+1} = 0$ then P_i executes $\mathsf{iSig.open}_1(P_j, P_i, \beta_1 \cdot r_{j,i}^1 + \ldots + \beta_m \cdot r_{j,i}^m + r_{j,i}^{m+1})$ for every P_j with $\mathsf{flag}_{i,j}^1 = \ldots = \mathsf{flag}_{i,j}^{m+1} = 0$.
- If $\mathsf{flag}_i^1 = \ldots = \mathsf{flag}_i^m = 0$ and $\mathsf{flag}_{i,R}^{m+1} = 1$ then P_i executes $\mathsf{iSig.open}_1(P_j, P_i, \beta_1 \cdot r_{j,i}^1 + \ldots + \beta_m \cdot r_{j,i}^m)$ for every P_j with $\mathsf{flag}_{i,j}^1 = \ldots = \mathsf{flag}_{i,j}^{m+1} = 0$.

Local computation At the end of the vss and vst execution, all flags are public. If some vss or vst execution ended with P_d discarded, or if P_d did not follow the assumed instructions, then P_d the parties output $\perp$ and terminate.

Otherwise, let $B := \cup_{k\in[m+1]} B^k$. Add to B any P_i that did not follow the assumed instructions, or that there exists an instance of public opening $\mathsf{iSig.open}_1(*, P_i, *)$ that failed (but did not have *iSign-conflict*), or P_i did not open the correct value (i.e. it used the wrong linear coefficients $\boldsymbol{\beta}$s).

For every $P_i \notin B$ the parties do as follows.

- If P_d is conflicted with P_i in all vss and vst executions, or P_i is conflicted with P_d in all vss and vst executions, let $f_i^1(x), \ldots, f_i^{m+1}(x)$ be the public shares of P_i. The parties compute $g_i(x) := \beta_1 f_i^1(x) + \ldots + \beta_m f_i^m(x) + F_i^{m+1}(x)$.
- Otherwise, if $\mathsf{flag}_{i,R}^{m+1} = 1$, then the polynomial $f_i^{m+1}(x)$ is public. For every $P_j \notin B$ do as follows.
  - If $\mathsf{flag}_{i,j} = 1$ or $\mathsf{flag}_{j,i} = 1$ or $\mathsf{flag}_j = 1$ or $\mathsf{flag}_j^d = 1$ in all vss instances, then all the (i,j)-th shares, denoted $f_{i,j}^1, \ldots, f_{i,j}^m, f_{i,j}^{m+1}$, are public. The parties locally compute $g_{i,j} := \beta_1 f_{i,j}^1 + \ldots + \beta_m f_{i,j}^m + f_{i,j}^{m+1}$.
  - Otherwise the parties compute $g_{i,j} := (\beta_1 \cdot b_{i,j}^1 + \ldots + \beta_m \cdot b_{i,j}^m) - (\beta_1 \cdot r_{j,i}^1 + \ldots + \beta_m \cdot r_{j,i}^m) + f_i^{m+1}(j)$, where $(\beta_1 \cdot r_{j,i}^1 + \ldots + \beta_m \cdot r_{j,i}^m)$ was opened by P_i.

  The parties interpolate over $(g_{i,j})_{P_j \notin B}$ to obtain a polynomial $g_i(x)$. If $g_i(x)$ is of degree more than t then the parties add P_i to B.
- Otherwise, if $\mathsf{flag}_{i,R} = 0$, then for every $P_j \notin B$ do as follows.
  - If $\mathsf{flag}_{i,j} = 1$ or $\mathsf{flag}_{j,i} = 1$ or $\mathsf{flag}_j = 1$ or $\mathsf{flag}_j^d = 1$ in all vss instances, then all the (i,j)-th, denoted $f_{i,j}^1, \ldots, f_{i,j}^m, f_{i,j}^{m+1}$, are public. The parties locally compute $g_{i,j} := \beta_1 f_{i,j}^1 + \ldots + \beta_m f_{i,j}^m + f_{i,j}^{m+1}$.
  - Otherwise, the parties compute $g_{i,j} := (\beta_1 \cdot b_{i,j}^1 + \ldots + \beta_m \cdot b_{i,j}^m + b_{i,j}^{m+1}) - (\beta_1 \cdot r_{j,i}^1 + \ldots + \beta_m \cdot r_{j,i}^m + r_{i,j}^{m+1})$, where $(\beta_1 \cdot r_{j,i}^1 + \ldots + \beta_m \cdot r_{j,i}^m + r_{i,j}^{m+1})$ was opened by P_i.

The parties interpolate over $(g_{i,j})_{P_j \notin B}$ to obtain a polynomial $g_i(x)$. If $g_i(x)$ is of degree more than t then the parties add P_i to B.

The parties interpolate over all $(g_i(x))_{P_i \notin B}$ in order to obtain a symmetric bivariate polynomial $G(x,y)$ of degree at most t in each variable. The parties output $G(x,y)$. The receiver R also outputs $F^{m+1}(x,y)$.

Round 4 (Opening phase) In order to reveal the unencrypted value $G(x,y) - F^{m+1}(x,y)$, R executes vrec_R with respect to vst^{m+1}. If the output is $\perp$ then all parties output $\perp$. Otherwise, let the output be $F^{m+1}(x,y)$ and all the parties output $G(x,y) - F^{m+1}(x,y)$.

Figure 14: Protocol linop_3

We continue by analysing the correctness of linop_3 in the $\mathcal{F}_{\mathsf{iSig}}$-hybrid model, in an execution in which the parties are instructed according to our assumptions. It is not hard to see that the honest parties are not in B, and that honest P_d and R are not discarded. Consider any execution in which P_d is not discarded, and by Lemma 6.5 the shares of the honest parties in the executions of $\mathsf{vss}^1, \ldots, \mathsf{vss}^{m+1}$ define polynomials $F^1(x,y), \ldots, F^{m+1}(x,y)$. We show that the output is $G(x,y) = \sum_{i=1}^{m} \beta_i F^i(x,y) + F^{m+1}(x,y)$ with probability 1.

Consider any P_i so that $\mathsf{flag}_i = 1$ or $\mathsf{flag}_i^d = 1$ in all vss and vst executions. Then the shares $f_i^1(x), \ldots, f_i^{m+1}(x)$ are public, and by Lemma 6.5 those shares are equal to $F^1(x,i), \ldots, F^{m+1}(x,i)$. Therefore $g_i(x) = G(x,i)$.

Consider now any honest P_i so that $\mathsf{flag}_i = \mathsf{flag}_i^d = 0$ in vss^k for all $k \in [m]$. Let $P_j \notin B$ be any party. We prove that $g_{i,j} = G(j,i)$. We split into cases.

- If $\mathsf{flag}_{i,j} = 1$ or $\mathsf{flag}_{j,i} = 1$ or $\mathsf{flag}_j = 1$ or $\mathsf{flag}_j^d = 1$ in all vss executions then the (i,j)-th share is public in all vss executions. Correctness now follows from Lemma 6.5.

- Otherwise, assume that $\mathsf{flag}_{i,R} = 1$. In this case the (i,j)-th share of vss^{m+1} is public, and by Lemma 6.5 it is equal to $F^{m+1}(j,i)$. In addition, by Lemma 6.5 for every $k \in [m]$ it holds that $b_{i,j}^k - r_{j,i}^k = F^k(j,i)$. Correctness follows.

- Otherwise $\mathsf{flag}_{i,R} = 0$. By Lemma 6.5 for every $k \in [m]$ it holds that $b_{i,j}^k - r_{j,i}^k = F^k(j,i)$, as required. If $\mathsf{flag}_i^d = 0$ in vst^{m+1} then by Lemma 6.5 it holds that $b_{i,j}^{m+1} - r_{j,i}^{m+1} = F^{m+1}(j,i)$ as well, and correctness follows.

  Otherwise, $\mathsf{flag}_i^d = 1$. Since P_d did not raise flags in all vss instances, then P_d is not internally-conflicted with P_i (or otherwise P_d is discarded). In particular, this means that P_i did not raise a complaint against P_d in any vss instance, and in every vss instance it holds that $a_{i,j}^d = a_{i,j}$ for all $j \in [n]$. Since P_i did not raise flags in all vss instances, then P_i is not *iSign-conflicted* with P_d. Therefore the tentative shares of P_i do not change (see Remark 6.6), and $b_{i,j}^{m+1} - r_{j,i}^{m+1} = F^{m+1}(j,i)$ as well, and correctness follows.

Consider now any corrupt $P_i \notin B$ so that $\mathsf{flag}_i = \mathsf{flag}_i^d = 0$ in vss^k for all $k \in [m]$. We show that for every honest P_j it holds that $g_{i,j} = G(i,j)$. Since there are at least $n - t \geq t + 1$ this means that $g_i(x) = G(x,i)$, or otherwise $g_i(x)$ has degree more than t and $P_i \in B$. We split into cases.

- If $\mathsf{flag}_{i,j} = 1$ or $\mathsf{flag}_{j,i} = 1$ or $\mathsf{flag}_j = 1$ or $\mathsf{flag}_j^d = 1$ in all vss executions then the (i,j)-th share is public in all vss executions. Correctness now follows from Lemma 6.5.

- Otherwise, assume that $\mathsf{flag}_{i,R} = 1$. In this case the (i,j)-th share of vss^{m+1} is public, and by Lemma 6.5 it is equal to $F^{m+1}(j,i)$. In addition, by Lemma 6.5 it holds that in vss^k the (i,j)-th share is $b^k_{i,j} - r^k_{j,i} = F^k(j,i)$, for every $k \in [m]$. Correctness follows.
- Otherwise $\mathsf{flag}_{i,R} = 0$. Since $\mathsf{flag}_{j,i} = \mathsf{flag}_{i,j} = 0$ in all vss instances, then for every $k \in [m+1]$ it holds that $b^k_{i,j} = a^k_{j,i} = (b^k_{j,i} - r^k_{i,j}) + r^k_{j,i}$, and we've already seen that $b^k_{j,i} - r^k_{i,j} = F^k(i,j)$. Correctness follows.

Since there are $n - t \geq t + 1$ honest parties not in B, and since for every $P_i \notin B$ it holds that $g_i(x) = G(x,i)$, then the polynomial recovered by the parties is indeed $G(x,y)$, as required. In addition, by the correctness of vst (Lemma 7.3), it follows that R outputs $F^{m+1}(x,y)$.

Finally, by Lemma 7.3, an honest R can open $F^{m+1}(x,y)$ by executing vrec_R, and all the parties can output $G(x,y) - F^{m+1}(x,y) = \sum_{i=1}^{m} \beta_i F^i(x,y)$. In addition, a corrupt R can either open $F^{m+1}(x,y)$ or $\perp$, so the output of the honest parties is either $\sum_{i=1}^{m} \beta_i F^i(x,y)$ or $\perp$, as required.

We proved the following lemma.

Lemma 8.3. *Consider any execution of* $\mathsf{vss}^1, \ldots, \mathsf{vss}^m$, vst^{m+1} *and* linop_3 *in the* $\mathcal{F}_{\mathsf{iSig}}$*-hybrid model, where the parties are instructed according to the assumptions, and* P_d *is not discarded. Let* $F^1(x,y), \ldots, F^{m+1}(x,y)$ *be the polynomials defined by the shares of the honest parties in the* vss *executions. Then the output of all honest parties at the end of the third round is* $\beta_1 F^1(x,y) + \ldots + \beta_m F^m(x,y) + F^{m+1}(x,y)$*, and* R *outputs the polynomial* $F^{m+1}(x,y)$*.*

In the opening phase, if R *is honest then all honest parties output* $\beta_1 F^1(x,y) + \ldots + \beta_m F^m(x,y)$*. If* R *is corrupt, then the output is either* $\beta_1 F^1(x,y) + \ldots + \beta_m F^m(x,y)$ *or* $\perp$*.*

8.2.3 The Protocol

We continue with the description of protocol sh-comp. We let every P_i to share $(G^{i,j}(x,y), G^{i,k}, H^{i,j,k}(x,y))$ via tss, thus proving that they satisfy the multiplicative relation. We also let P_i share $F^{i,v}(x,y)$ via vst with $P_{\phi(i,v)}$ as a receiver. The linear computation over tentative shares from a single dealer is done by using linop_3, while the linear computation between shares of two different dealers is done by linop_2. The protocol is presented in Figure 15.

Protocol sh-comp

Round 1 (Input phase) Every P_i does as follows.

- For every $j \in [m]$, P_i shares $G^{i,j}(x,y)$ via an instance of vss, denoted $\mathsf{vss}^{G^{i,j}}$.
- For every $j,k \in [m]$, P_i shares $H^{i,j,k}(x,y)$ via an instance of vss, denoted $\mathsf{vss}^{H^{i,j,k}}$.
- For every $j,k \in [m]$ the parties execute an instance of tss, denoted $\mathsf{tss}^{i,j,k}$, with P_i as a dealer and the instances $\mathsf{vss}^{G^{i,j}}$, $\mathsf{vss}^{G^{i,k}}$ and $\mathsf{vss}^{H^{i,j,k}}$ as the sharing of A^0, B^0 and C^0.
- For every $v \in [\ell]$, P_i shares $F^{i,v}(x,y)$ via an instance of vst, denoted $\mathsf{vst}^{F^{i,v}}$, with $P_{\phi(i,v)}$ as the receiver. Denote the internal vss execution by $\mathsf{vss}^{F^{i,v}}$.

Round 2 The parties continue with the execution of all vst, vss and tss instances.

Round 3 (Linear computation phase) At the beginning of the round all flags are initialized to 0. Every P_i sets its flags as follows.

- *(P_i as a dealer I)* For every $j \in [n]$, if P_i is internally-vss-conflicted with P_j in any vss execution where P_i is the dealer, or P_i is internally-vst-conflicted with P_j in any vst execution where P_i is the dealer, then P_i inputs $\mathsf{flag}_j^d = 1$ to every vss, tss and vst execution where P_i is the dealer.
- *(P_i as a dealer II)* If there exists an instance of vst in which P_i is the dealer and P_i is internally-vst-conflicted with the corresponding receiver, then P_i inputs $(\mathsf{flag}_j^d = 1)_{j\in[n]}$ and $\mathsf{flag}_R^d = 1$ into this vst instance.
- *(P_i as a receiver I)* If there exists an instance of vst in which P_i is the receiver, and P_i is internally-vst-conflicted with the corresponding dealer, then P_i inputs $(\mathsf{flag}_j^R = 1)_{j\in[n]}$ and $\mathsf{flag}_d^R = 1$ into this vst instance.
- *(P_i as a receiver II)* For every $j \in [n]$, if there exists an instance of vst in which P_i is the receiver, and P_i is internally-vst-conflicted with P_j, then P_i inputs $\mathsf{flag}_j^R = 1$ into this vst instance.
- *(Conflicts with dealer I)* For every $j \in [n]$, if P_i is internally-vss-conflicted with the dealer in any vss execution where P_j is the dealer, or P_i is internally-vst-conflicted with the dealer in any vst execution where P_j is the dealer, then P_i inputs $\mathsf{flag}_i = 1$ to every vss, tss and vst execution where P_j it the dealer, as well as $\mathsf{flag}_{i,R} = 1$ in the vst executions where P_j is the dealer.
- *(Conflicts with dealer II)* For every $j \in [n]$, if P_i thinks that its share might change in any vss execution in which P_j is the dealer (as per Observation 6.6), then P_i inputs $\mathsf{flag}_i = 1$ to every vss, tss and vst execution where P_j it the dealer, as well as $\mathsf{flag}_{i,R} = 1$ in the vst executions where P_j is the dealer.
- *(Conflicts with receiver)* If there exists some instance of vst where where P_i is internally-vst-conflicted with the receiver, then P_i inputs $\mathsf{flag}_{i,R} = 1$ and $\mathsf{flag}_{i,d} = 1$ into this vst instance, and $\mathsf{flag}_i = 1$ into the internal vss instance.
- *(Conflicts with other players)* For every $j, k \in [n]$, if P_i is internally-vss-conflicted with P_j in any vss execution where P_k is the dealer, then P_i inputs $\mathsf{flag}_{i,j} = 1$ to every vss, tss and vst execution where P_k it the dealer.

The parties do as follows.

- The parties execute all vss, tss and vst instances using the above flags.
- For every P_i and every $v \in [\ell]$, the parties compute

$$\sum_{j=1}^{m} \beta_{i,j}^v G^{i,j}(x,y) + \sum_{j,k\in[m]} \gamma_{i,j,k}^v H^{i,j,k}(x,y) + F^{i,v}(x,y)$$

  by executing linop_3 with respect to the instances $(\mathsf{vss}^{G^{i,j}}, \mathsf{vss}^{H^{i,j,k}})_{j,k\in[m]}$ and $\mathsf{vst}^{F^{i,v}}$, where P_i is the dealer, and $P_{\phi(i,v)}$ is the receiver, and with the same flags as in those instances.
- For every $((i_1,v_1),(i_2,v_2)) \in S$ the parties compute

$$F^{i_1,v_1}(x,y) - F^{i_2,v_2}(x,y)$$

  by executing linop_2 with respect to the instances $\mathsf{vss}^{F^{i_1,v_1}}$ and $\mathsf{vss}^{F^{i_2,v_2}}$ with P_{i_1} as $\mathcal{D}_1$ and P_{i_2} as $\mathcal{D}_2$, and with the same flags as in those instances.

- *(Local computation)* At the end of the round the parties execute the following local computation. Let B be the union of all sets of bad parties, discarded dealers, and discarded receivers, in all $\mathsf{vss}, \mathsf{tss}, \mathsf{vst}$ executions, as well as in linop_2 and linop_3 operations. The parties set $I = B$. The parties output the following values.
  - The set I.
  - For every $i \notin I$ and $j \in [m]$, every party outputs the degree-t polynomial defined by its shares in $\mathsf{vss}^{G^{i,j}}$. The parties act similarly in $\mathsf{vss}^{H^{i,j,k}}$ and $\mathsf{vst}^{F^{i,v}}$ for all $i \notin I$, $j,k \in [m]$ and $v \in [\ell]$.
  - For every $i \notin I$ and $v \in [\ell]$, the parties output $\sum_{j=1}^{m} \beta_{i,j}^{v} G^{i,j}(x,y) + \sum_{j,k\in[m]} \gamma_{i,j,k}^{v} H^{i,j,k}(x,y) + F^{i,v}(x,y)$. For every $i \notin I$ and every $v \in [\ell]$ so that $\phi(i,v) = j$, P_j outputs the bivariate polynomial defined by its shares as a receiver in $\mathsf{vst}^{F^{i,v}}$.
  - For every $((i_1,v_1),(i_2,v_2)) \in S$ so that $i_1, i_2 \notin I$, the parties output $((i_1,v_1),(i_2,v_2), F^{i_1,v_1}(x,y) - F^{i_2,v_2}(x,y))$. If $i_1 \in I$ or $i_2 \in I$ then the parties output $((i_1,v_1),(i_2,v_2),\perp)$.

Round 4 (Opening phase) Assume that P_i holds inputs $(\mu_i, (j_{i,k}, v_{i,k})_{k\in[\mu_i]}))$. Then P_i broadcasts $(\mu_i, (j_{i,k}, v_{i,k})_{k\in[\mu_i]}))$, and for every $k \in [\mu_i]$, P_i executes the open phase of the linop_3 execution that corresponds to $(\mathsf{vss}^{G^{j_{i,k},j'}}, \mathsf{vss}^{H^{j_{i,k},j',k'}})_{j',k'\in[m]}$ and $\mathsf{vst}^{F^{j_{i,k},v_{i,k}}}$.

In addition, the parties do the following local computation. Initialize $I' := I$, and add every P_i for which the opening phase of linop_3 where P_i is the receiver ended with $\perp$ to I'. Output

$$(I', (\mu_i, j_{i,k}, v_{i,k}, F^{j_{i,k},v_{i,k}}(x,y))_{i\notin I', k\in[\mu_i]}).$$

Figure 15: Protocol sh-comp

9 Augmented Single Input Functionalities

In this section we present our protocol for augmented SIF. First, we formalize our requirements by an ideal functionality $\mathcal{F}_{\mathsf{asif}}$. We start by providing an overview of the functionality. The functionality is parameterized by single input functionalities $\mathcal{F}_1, \ldots, \mathcal{F}_n$, where $\mathcal{F}_i$ takes as an input a vector $\mathbf{x}_i \in \mathbb{F}^m$ and returns $f_{i,j}(\mathbf{x}_i)$ to P_j, where $f_{i,j}(\mathbf{x}_i) = (f_{i,j,1}(\mathbf{x}_i), \ldots, f_{i,j,\ell}(\mathbf{x}_i))$ is a length-ℓ vector.

Input and computation phase. In the input phase, every P_i inputs $\mathbf{x}_i$ to $\mathcal{F}_{\mathsf{asif}}$, and in the computation phase every P_j receives $f_{i,j}(\mathbf{x}_i)$. In addition, in the computation phase, the functionality supports linear operations over the outputs from different dealers. Formally, the functionality is parameterized by a set S of tuples $(k,(i_1,v_1),(i_2,v_2)) \in [n] \times ([n] \times [\ell]) \times ([n] \times [\ell])$ and returns $f_{i_1,k,v_1}(\mathbf{x}_{i_1}) - f_{i_2,k,v_2}(\mathbf{x}_{i_2})$ to all the parties for each such tuple.

Communication with the adversary. Like in $\mathcal{F}_{\mathsf{sh\text{-}comp}}$, the functionality allows the adversary to choose the inputs of the corrupt parties, after seeing the following leakage: (1) the outputs of the corrupt parties in $(\mathcal{F}_i)_{i\in\mathsf{H}}$, (2) $f_{i_1,k,v_1}(\mathbf{x}_{i_1}) - f_{i_2,k,v_2}(\mathbf{x}_{i_2})$ for $(k,(i_1,v_1),(i_2,v_2)) \in S$ with honest P_{i_1}, P_{i_2}, (3) $f_{i_1,k,v_1}(\mathbf{x}_{i_1})$ for $(k,(i_1,v_1),(i_2,v_2)) \in S$ with corrupt P_{i_2}, and (4) $f_{i_2,k,v_2}(\mathbf{x}_{i_2})$ for $(k,(i_1,v_1),(i_2,v_2)) \in S$ with corrupt P_{i_1}. Looking forward, when using $\mathcal{F}_{\mathsf{asif}}$ as a subprotocol we will make sure that the outer protocol is secure even if the adversary can choose its inputs based on the leakage. We also allow the adversary to cause every corrupt P_i to abort the computation

of $\mathcal{F}_i$ by inputting $\mathsf{flag}_i = 1$ to the functionality. The functionality returns the set I, of all aborting corrupt parties, to all the parties.

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

The functionality is parameterized by an integer ℓ, and single input functionalities $\mathcal{F}_1, \ldots, \mathcal{F}_n$, where $\mathcal{F}_i$ takes as an input a vector $\mathbf{x}_i \in \mathbb{F}^m$ and returns $f_{i,j}(\mathbf{x}_i)$ to P_j, where $f_{i,j}(\mathbf{x}_i) = (f_{i,j,1}(\mathbf{x}_i), \ldots, f_{i,j,\ell}(\mathbf{x}_i))$. There is also a set $S \subseteq [n] \times ([n] \times [\ell]) \times ([n] \times [\ell])$ containing tuples $(k, (i_1, v_1), (i_2, v_2))$ where every pair (i, v) appears in at most one such tuple.

The functionality receives the set of corrupt parties C.

Input phase. Every honest party P_i inputs $\mathbf{x}_i$.

Computation phase.

- *(Leakage)* The values $(f_{i,j}(\mathbf{x}_i))_{i \in \mathsf{H}, j \in \mathsf{C}}$ are leaked to the adversary.
  In addition, for every $(k, (i_1, v_1), (i_2, v_2)) \in S$ so that P_{i_1} and P_{i_2} are honest, the adversary receives $f_{i_1,k,v_1}(\mathbf{x}_{i_1}) - f_{i_2,k,v_2}(\mathbf{x}_{i_2})$. For every $(k, (i_1, v_1), (i_2, v_2)) \in S$ so that P_{i_1} is honest and P_{i_2} is corrupt, the adversary receives $f_{i_1,k,v_1}(\mathbf{x}_{i_1})$. For every $(k, (i_1, v_1), (i_2, v_2)) \in S$ so that P_{i_2} is honest and P_{i_1} is corrupt, the adversary receives $f_{i_2,k,v_2}(\mathbf{x}_{i_2})$.
- *(Corrupt parties' inputs)* Every corrupt P_i inputs $\mathbf{x}_i$ and a bit flag_i. Let I be the set of all corrupt parties P_i with $\mathsf{flag}_i = 1$.
- *(Outputs)* Every honest P_i receives the private output

  $$(I, f_{j,i}(\mathbf{x}_j))_{j \notin I}.$$

  In addition, for every $(k, (i_1, v_1), (i_2, v_2)) \in S$, the functionality returns
  - *($i_1, i_2 \notin I$)* The tuple $(k, (i_1, v_1), (i_2, v_2), f_{i_1,k,v_1}(\mathbf{x}_{i_1}) - f_{i_2,k,v_2}(\mathbf{x}_{i_2}))$.
  - *(Otherwise)* The tuple $(k, (i_1, v_1), (i_2, v_2), \perp)$.

Opening phase.

- *(Inputs of honest parties)* Every honest P_i inputs an integer $\mu_i \in \mathbb{N}$, and pairs $(j_{i,k}, v_{i,k})_{k \in [\mu_i]}$ so that $j_{i,k} \in [n] \setminus I$, and $v_{i,k} \in [\ell]$.
- *(Leakage)* For every honest P_i the functionality returns $(\mu_i, (j_{i,k}, v_{i,k}, f_{j_{i,k},i,v_{i,k}}(\mathbf{x}_{j_{i,k}})_{k \in [\mu_i]}))$ to the adversary.
- *(Inputs of corrupt parties)* Every corrupt P_i inputs a bit abort_i, an integer $\mu_i \in \mathbb{N}$, and pairs $(j_{i,k}, v_{i,k})_{k \in [\mu_i]}$ so that $j_{i,k} \in [n] \setminus I$, and $v_{i,k} \in [\ell]$.
- *(Output)* Let I' be the set I together with every corrupt P_i with $\mathsf{abort}_i = 1$. The functionality returns

  $$(I', (\mu_i, j_{i,k}, v_{i,k}, f_{j_{i,k},i,v_{i,k}}(\mathbf{x}_{j_{i,k}}))_{i \notin I', k \in [\mu_i]})$$

  to all the parties.

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

First we consider the special case where $\mathcal{F}_1, \ldots, \mathcal{F}_n$ are degree-2 single input functionalitiess over $\mathbb{F}$, and present a protocol asif for this special case. Later, we explain how to reduce general single input functionalities to degree-2 functionalities. Therefore, let us assume for now that for

every $i, i' \in [n]$ and $v \in [\ell]$, the function $f_{i,i',v}$ is a degree-2 function of the form

$$\sum_{j=1}^{m} \beta^v_{i,i',j} x_{i,j} + \sum_{j,k \in [m]} \gamma^v_{i,i',j,k} x_{i,j} \cdot x_{i,k}.$$

We use protocol sh-comp as a subprotocol with the following parameters. The input parameter is m and the output parameter is $n\ell$, and that the field $\mathbb{F}$ is sufficiently large, as per Theorem 8.1. For simplicity we identify each pair $(i, v) \in [n] \times [n\ell]$ with a tuple $(i, j, v') \in [n] \times [n] \times [\ell]$, where $j = \lfloor (v-1)/\ell \rfloor + 1$ and $v' = (v+1) \mod \ell$. That is, the ℓ pairs $(i, (j-1)\ell + 1), \ldots, (i, j\ell)$ are identified with the ℓ tuples $(i, j, 2), \ldots, (i, j, \ell), (i, j, 1)$, respectively. The coefficients in the definition of sh-comp are set to be $(\beta^v_{i,i',j}, \gamma^v_{i,i',j,k})_{i,i' \in [n], j,k \in [m], v \in [\ell]}$, and we define $\phi(i, j, v) = P_j$. Given the set S of $\mathcal{F}_{\mathsf{asif}}$, we define S' of sh-comp to include $((i_1, k, v_1), (i_2, k, v_2))$ for every $(k, (i_1, v_1), (i_2, v_2))$. We continue with a description of the protocol.

Protocol asif

Round 1 (Input phase) Every P_i does as follows.

- Samples a random symmetric bivariate polynomial $F^{i,i',v}(x, y)$ of degree at most t in each variable, for every $i' \in [n]$ and $v \in [\ell]$
- Samples a random symmetric bivariate polynomial $G^{i,j}(x, y)$ of degree at most t in each variable, conditioned on $G^{i,j}(0, 0) = x_{i,j}$, for every $j \in [m]$.
- Samples a random symmetric bivariate polynomial $H^{i,j,k}(x, y)$ of degree at most t in each variable, conditioned on $H^{i,j}(0, 0) = x_{i,j} \cdot x_{i,k}$, for every $j, k \in [m]$.

The parties execute the first round of sh-comp with those inputs.

Round 2 The parties execute the second round of sh-comp with those inputs.

Round 3 (Computation phase) The parties execute the third round of sh-comp. Observe that all the parties agree on the set I and on the bivariate polynomials

$$\left(\mathsf{Out}^{i,i',v}(x, y) := \sum_{j=1}^{m} \beta^v_{i,i',j} G^{i,j}(x, y) + \sum_{j,k \in [m]} \gamma^v_{i,i',j,k} H^{i,j,k}(x, y) + F^{i,i',v}(x, y) \right)_{i,i' \in [n] \setminus I, v \in [\ell]}$$

and

$$(\mathsf{Out}^{(k,(i_1,v_1),(i_2,v_2))}(x, y) := F^{i_1,k,v_1}(x, y) - F^{i_2,k,v_2}(x, y))_{(k,(i_1,v_1),(i_2,v_2)) \in S : i_1, i_2 \notin I}.$$

In addition, every $P_{i'}$ holds the polynomials $(F^{i,i',v}(x, y))_{i \in [n] \setminus I, v \in [\ell]}$, that were transferred as part of sh-comp. Every $P_{i'}$ does as follows.

- For every $i \in [n] \setminus I$ and $v \in [\ell]$, party $P_{i'}$ sets

$$f_{i,i',v} := (\mathsf{Out}^{i,i',v}(0, 0) - F^{i,i',v}(0, 0)).$$

- For every $(k, (i_1, v_1), (i_2, v_2)) \in S$ so that $i_1, i_2 \notin I$, party $P_{i'}$ sets $f_{(k,(i_1,v_1),(i_2,v_2))} := (\mathsf{Out}^{i_1,k,v_1}(0, 0) - \mathsf{Out}^{i_2,k,v_2}(0, 0)) - \mathsf{Out}^{(k,(i_1,v_1),(i_2,v_2))}(0, 0)$.

- $P_{i'}$ outputs
$$(I, (f_{i,i',v})_{i\in[n],v\in[\ell]}, (f_{(k,(i_1,v_1),(i_2,v_2))})_{(k,(i_1,v_1),(i_2,v_2))\in S:i_1,i_2\notin I}).$$

Round 4 (Opening phase) Every P_i on input $\mu_i \in \mathbb{N}$, and pairs $(j_{i,k}, v_{i,k})_{k\in[\mu_i]}$ broadcasts $(\mu_i, (j_{i,k}, v_{i,k})_{k\in[\mu_i]})$, and executes the opening phase of sh-comp with input $\mu_i \in \mathbb{N}$, and pairs $(j_{i,k}, i, v_{i,k})_{k\in[\mu_i]}$. At the end of the round the parties execute the local computation of sh-comp. Denote the outputs by
$$(I', (\mu_i, j_{i,k}, v_{i,k}, F^{j_{i,k},i,v_{i,k}}(x,y)_{i\notin I',k\in[\mu_i]})).$$

Then every P_i outputs
$$(I', (\mu_i, j_{i,k}, v_{i,k}, \mathsf{Out}^{j_{i,k},i,v_{i,k}}(0,0) - F^{j_{i,k},i,v_{i,k}}(0,0)_{i\notin I',k\in[\mu_i]})).$$

Figure 17: Protocol asif

Notation 7. *In the opening phase, instead of specifying all the indices that P_i opens, it would be convenient to say that P_i opens $f_{j,i,v}(\mathbf{x}_j)$ via* asif*, if one of the indices that P_i inputs is (j, v).*

In Section F we prove the following theorem. As always, we assume that $\mathbb{F}$ is a sufficiently large field of size $\exp(n, \kappa, s)$.

Theorem 9.1. *Let κ be a security parameter, let n be the number of parties and $t < n/2$ the number of corrupt parties. Let $\mathcal{F}_1, \ldots, \mathcal{F}_n$ be* degree-2 *single input functionalities with circuit size at most s, over a sufficiently large field $\mathbb{F}$ as a function of (n, κ, s). Then protocol* asif *is a UC-secure implementation of $\mathcal{F}_{\mathsf{asif}}$ against a static, active, rushing adversary corrupting up to t parties. The complexity of* asif *is* $\mathrm{poly}(\kappa, 2^n, s, \log|\mathbb{F}|)$.

Computing general single input functionalities. Let $\mathcal{F}_1, \ldots, \mathcal{F}_n$ be single input functionalities given by *boolean circuits* $C_1, \ldots, C_n$, respectively, where the size of each circuit is at most s. Following the standard reduction from circuit-satisfiability to quadratic equations over an arbitrary field $\mathbb{F}$, it is well known (see, e.g., [GIKR02, Theorem 1]) that the computation of every single input functionality $\mathcal{F}_i$ can be efficiently and non-interactively reduced to the computation of a degree-2 single input functionality $\mathcal{H}_i$ over an arbitrary finite field $\mathbb{F}$, where the circuit size of $\mathcal{H}_i$ is at most $s' = O(s)$, where s' is *independent of the choice of* $\mathbb{F}$. More concretely, the functionality $\mathcal{H}_i$ receives from the dealer the values of *all* wires of C_i, and (1) returns to every P_j the values of all output wires that belong to P_j, and (2) returns to all the parties a public vector that is equal to 0 if and only if the inputs are "well-formed", i.e., the values of all wires are binary and for every gate of C_i the values of the wires incident to the gate are consistent with the gate. Note that verifying that the inputs are well-formed can be computed by degree-2 functions.

We show that we can apply this reduction even for the case of augmented SIF. That is, we argue that the computation of $\mathcal{F}_{\mathsf{asif}}$ with $\mathcal{F}_1, \ldots, \mathcal{F}_n$ can be efficiently and non-interactively reduced to the computation of $\mathcal{H}_1, \ldots, \mathcal{H}_n$ over any field $\mathbb{F}$ that is an extension field of $\mathbb{F}_2$. Indeed, in the computation of $\mathcal{F}_{\mathsf{asif}}$ with $\mathcal{H}_1, \ldots, \mathcal{H}_n$, if the players find out in the computation phase that a corrupt P_i did not use "well-formed" inputs, they automatically add P_i to the set of bad parties I, and we observe that the use of not "well-formed" inputs reveals no additional information in the linear computation in the computation phase, as all the relevant information is already leaked to the adversary. We emphasize that we need to use an extension field of $\mathbb{F}_2$, since for every $(k, (i_1, v_1), (i_2, v_2)) \in S$, if the inputs of P_{i_1} and P_{i_2} are "well-formed" and P_{i_1} and P_{i_2} are not in

I, then $f_{i_1,k,v_1}(\mathbf{x}_{i_1})$ and $f_{i_2,k,v_2}(\mathbf{x}_{i_2})$ are in $\{0,1\}$, and the computation of $f_{i_1,k,v_1}(\mathbf{x}_{i_1}) - f_{i_2,k,v_2}(\mathbf{x}_{i_2})$ should be perforemd over $\mathbb{F}_2$, which is exactly what happens if we work over an extension field of $\mathbb{F}_2$.

We conclude that by choosing the field $\mathbb{F}$ to be a sufficiently large extension field of $\mathbb{F}_2$ as a function of (n, κ, s), we can use Theorem 9.1 with respect to $\mathcal{H}_1, \ldots, \mathcal{H}_n$ to obtain the following theorem.

Theorem 9.2. *Let κ be a security parameter, let n be the number of parties and $t < n/2$ the number of corrupt parties. Let $\mathcal{F}_1, \ldots, \mathcal{F}_n$ be single input functionalities with* boolean *circuit size at most s. Then protocol* asif *is a UC-secure implementation of $\mathcal{F}_{\mathsf{asif}}$ against a static, active, rushing adversary corrupting up to t parties. The complexity of* asif *is* $\mathrm{poly}(\kappa, 2^n, s)$.

In the rest of the paper, we always assume that the single input functionalities $\mathcal{F}_1, \ldots, \mathcal{F}_n$ are boolean functionalities.

10 General Multiparty Computation

10.1 Overview

Let $\mathcal{F}$ be an n-party functionality, with circuit size s and depth d. Our starting point is a 2-round perfectly-secure protocol Π^{sm} for general MPC against rushing *semi-malicious* adversaries. Such adversaries are allowed to choose their inputs and randomness, but other than that play honestly. For concreteness, we take the protocol of [ABT18] as Π^{sm}, that is executed over $\mathbb{F}_2$ and has complexity $\mathrm{poly}(n, s, 2^d)$. An important feature of [ABT18] is that for every player, given the input, randomness, and all incoming first-round messages, the second-round messages can be computed by a circuit with size $\mathrm{poly}(n, s, 2^d)$ and depth $O(\log(n \cdot s \cdot 2^d))$.

We follow the blueprints of [AKP21]. We start with the 2-round perfectly secure protocol Π^{sm} against rushing semi-malicious adversaries and compile this protocol into a 3-round protocol Π^{fs}, secure against *fail-stop* adversaries, that follow the protocol but can abort every corrupt P_i at any time. We then explain how to use augmented SIF in order to compile this protocol into a 4-round protocol, secure against *malicious* adversaries.

Throughout this section all computation is over $\mathbb{F}_2$. Therefore, all messages sent in the protocols are simply binary vectors, and by addition and subtraction we simply mean bitwise-XOR of vectors.

From semi-malicious to fail-stop. The work of [AKP21] provided compiler from semi-malicious security to fail-stop security, that preserves *information-theoretic* security. We briefly recall the details here. We note that protocol Π^{sm} consists of only private messages in the first round, and only broadcast messages in the second round,[16], and we denote the first-round private message from P_i to P_j in Π^{sm} by $a_{i,j}$ and the second round broadcast of P_i by b_i.

First-round abort of a party P_i causes two problems to the other parties: (1) P_i did not send her first round messages; and (2) the first-round messages that were directed to P_i were lost. The first issue is solved by letting each party to locally generate the outgoing messages of P_i by running

[16] In fact, using standard padding techniques [GIKR01], every protocol can be transformed into a protocol of this form.

P_i on the all-zero input and the all-zero random tape.[17] The second issue is solved by letting every party share its outgoing messages among the parties, using $(t+1)$-out-of-n secret sharing scheme. In this case, if P_i aborts during the first round, then in the second round the parties reconstruct all the first round incoming messages of P_i. After the second round, the parties have enough information to locally continue the emulation of P_i (with respect to the all-zero inputs) and generate her second round broadcast messages.

Handling second-round aborts is more subtle. The main idea is to let every P_i that did not abort in the first round to share enough information among the parties, so the parties could securely compute the second-round broadcast of P_i by themselves. In order to do so, we first make the communication in Π^{sm} public in the following way. We add a round at the beginning of the protocol, Round 0, in which every P_i sends a one-time pad $\rho_{i,j}$ to P_j. In Round 1, the parties execute the first round of Π^{sm} over the broadcast channel, where P_j encrypts its message to P_i with the one-time pad $\rho_{i,j}$ (i.e., P_j sends $a_{j,i} + \rho_{i,j}$ to P_i). Every P_i also generates a garbled circuit to a function G_i that takes (1) the input and randomness of P_i in Π^{sm}, (2) the one-time pads $(\rho_{i,j})_{j\in[n]}$, and (3) the encrypted incoming messages of P_i, i.e., $(a_{j,i} + \rho_{i,j})_{j\in[n]}$. The function returns the broadcast of P_i according to Π^{sm} given its input, randomness and all decrypted incoming messages. Observe that inputs (1)–(2) are known to P_i already in the first round, so P_i can share the label of those inputs among the parties. In addition, P_i shares the labels that correspond to every potential encrypted incoming message. Since all the encrypted messages to P_i are public by the end of Round 1, in Round 2 the parties can recover all the labels, and compute the output of G_i, that is, the broadcast of P_i in Round 2.

From fail-stop to malicious. The work of [AKP21] showed a round-preserving compiler from security against fail-stop adversaries to security against malicious adversaries assuming the existence of non-interactive commitments. In this work we show that this technique can be adapted to the statistical regime as well, without any cryptographic assumptions, at the expanse of having an additional round.

In the first three rounds, every P_i executes an augmented single input functionality, that takes the input and randomness of P_i to Π^{sm}, the inputs of G_i, and some additional random pads $(\eta_{i,j})_{j\in[n]}$ that P_i picked. The functionality returns to all parties the encrypted outgoing messages of P_i in Π^{sm}, where the message to P_j is $A_{i,j} := a_{i,j} + \eta_{i,j}$. In addition, it shares the (not-encrypted) outgoing messages $(a_{i,j})_{j\in[n]}$ of P_i among the parties (to handle first-round aborts), and also shares the labels of the garbled circuit of G_i (to handle second-round aborts). In the fourth round, every P_i can simply use the opening phase of the augmented SIF to open the labels of the garbled circuits of each party.

However, there is a little technicality to be handled: the function G_i was defined with respect to random pads $(\rho_{i,j})_{j\in[n]}$ *that are known to* P_i already in the first round. However, the encrypted message from P_j to P_i is encrypted with the random pad $\eta_{j,i}$ which is not known to P_i. To solve this mismatch, we use the linearity of our augmented SIF protocol, and publicly compute the value $\nu_{j,i} := \eta_{j,i} - \rho_{i,j}$ already in the third round, so that all the parties can locally compute the new encrypted message $A_{j,i} - \nu_{j,i} = a_{j,i} + \rho_{i,j}$, and open the corresponding label according to this encrypted message. The full details appear in the next section.

[17] Here, among other places, we use the fact that Π^{sm} is secure against a semi-malicious adversary.

10.2 The protocol

We continue with a formal presentation of the protocol.

The function G_i. For every $i \in \{1, \ldots, n\}$ we define the following function. (We slightly deviate from the high-level overview in order to simultaneously handle first-round aborts and second-round aborts.)

Function G_i

Inputs. The function receives (1) (x_i, r_i) the input and randomness of P_i in Π^{sm}, (2) random pads $(\rho_{i,j})_{j\in\{1,\ldots,n\}}$, (3) a set $L_i \subseteq \{1, \ldots, n\}$, represented as an n-bit string where the j-th bit is 1 if $j \in L_i$, (4) messages $(A_{j,i})_{j\in\{1,\ldots,n\}}$.

Outputs. Define $a_{j,i} := A_{j,i}$ if $j \in L_i$, and $a_{j,i} := A_{j,i} - \rho_{i,j}$ otherwise. Let b_i be the broadcast message of P_i in the second round of Π^{sm}, where P_i has input x_i, randomness r_i, and P_i received the private message $a_{j,i}$ from P_j in the first round. Output b_i.

Figure 18: Function G_i

Let ℓ be the bit-length of each input of G_i (by using padding, we assume without loss of generality that all inputs of G_i have the same length, and that every G_i and G_j the inputs have the same length of 4ℓ). Since for every player in Π^{sm} the second-round messages can be computed by a circuit of size $\mathrm{poly}(n, s, 2^d)$ and depth $O(\log(n \cdot s \cdot 2^d))$, the function G_i can be implemented by a circuit of size $\mathrm{poly}(n, s, 2^d)$ and depth $O(\log(n \cdot s \cdot 2^d))$. Therefore, by Theorem 3.4, G_i has a perfect decomposable randomized encoding of size $\mathrm{poly}(n, s, 2^d)$, which we denote by $\hat{G}_i := (\hat{G}_{i,1}, \ldots, \hat{G}_{i,4\ell})$.

The functionality $\mathcal{F}_i$. For every $i \in \{1, \ldots, n\}$ we define the functionality $\mathcal{F}_i$, presented in Figure 19. The functionality uses $(t+1)$-out-of-n secret sharing over $\mathbb{F}_2$: this is simply Shamir's secret sharing over an extension field $\mathbb{F}$ of the binary field $\mathbb{F}_2$ of size at least $n + 1$, where the computation can be performed by a *boolean circuit* of size $\mathrm{poly}(n)$. By computing *Shamir shares* of a secret s we always mean sampling a degree-t polynomial $p(x)$ over $\mathbb{F}$ whose free coefficient is s, and generating the shares $(p(1), \ldots, p(n))$.

Functionality $\mathcal{F}_i$

Inputs. The functionality receives from the dealer P_i the following inputs: (1) (x_i, r_i) the input and randomness of P_i in Π^{sm}, (2) random pads $(\rho_{i,j}, \eta_{i,j})_{j\in\{1,\ldots,n\}}$, and (3) auxiliary randomness, as described below.

Outputs. The functionality computes the first-round messages of P_i in Π^{sm} on input x_i and randomness r_i, denoted $(a_{i,1}, \ldots, a_{i,n})$. Using the auxiliary randomness, the functionality samples Shamir shares $(a_{i,j}[1], \ldots, a_{i,j}[n])$ for every messages $a_{i,j}$.

In addition, using the auxiliary randomness, the functionality samples a random string r_i^{RE} for a randomized encoding $\hat{G}_i$ of G_i and does as follows.

- *(For inputs* **(1)–(2)***.)* Inputs **(1)–(2)** of G_i correspond to (x_i, r_i) and the random pads $(\rho_{i,j})_{j\in\{1,\ldots,n\}}$. Consider the j-th input bit of G_i that correspond to inputs **(1)–(2)**, denoted β_j where $j \in \{1,\ldots,2\ell\}$. Compute $\hat{G}_{i,j}(\beta_j, r_i^{\mathsf{RE}})$, and use the auxiliary randomness to sample Shamir's shares of this value, denoted $s_{i,j}[1],\ldots,s_{i,j}[n]$.
- *(For inputs* **(3)–(4)***.)* Recall that inputs **(3)–(4)** of G_i correspond to a set L_i and messages $(A_{j,i})_{j\in\{1,\ldots,n\}}$. Consider the j-th input bit of G_i that correspond to inputs **(3)–(4)** where $j \in \{2\ell+1,\ldots,4\ell\}$. For every $\beta \in \{0,1\}$ compute $\hat{G}_{i,j}(\beta, r_i^{\mathsf{RE}})$, and use the auxiliary randomness to sample Shamir's shares of this value, denoted $s_{i,j}^{\beta}[1],\ldots,s_{i,j}^{\beta}[n]$.

For every $k \in [n]$, return the following values to P_k: (1) the message $a_{i,k}$ and the random pads $\eta_{i,k}$, (2) the shares $(a_{i,j}[k])_{j\in\{1,\ldots,n\}}$ of P_i's first round messages (3) the shares $(s_{i,j}[k])_{j\in\{1,\ldots,2\ell\}}$ corresponding to inputs **(1)–(2)** of G_i, (4) the shares $(s_{i,j}^{\beta}[k])_{\beta\in\{0,1\},j\in\{2\ell+1,\ldots,4\ell\}}$ corresponding to inputs **(3)–(4)** of G_i, (5) the encrypted messages $(a_{i,j}+\eta_{i,j})_{j\in\{1,\ldots,n\}}$.

In addition, P_i receives the values $(\rho_{i,j})_{j\in[n]}$ as an output.

Figure 19: Functionality $\mathcal{F}_i$

The protocol. We consider a asif protocol parameterized by $\mathcal{F}_1,\ldots,\mathcal{F}_n$, so that in the linear computation phase the values $(\eta_{k,j}-\rho_{j,k})_{j,k\in[n]}$ are revealed (note that P_j is the receiver for both $\eta_{k,j}$ and $\rho_{j,k}$). We continue with the protocol for general MPC.

Protocol mpc

Inputs. Every P_i holds an input $\mathbf{x}_i$. All parties share a statistical security parameter 1^κ.

Round 1. Every party P_i samples randomness r_i for Π^{sm}, random pads $(\rho_{i,j},\eta_{i,j})_{j\in\{1,\ldots,n\}}$, and the auxiliary randomness required for $\mathcal{F}_i$. The parties execute the first round of asif.

Round 2. The parties execute the second round of asif. This completes the input phase of asif.

Round 3. The parties execute the third round of asif, the linear-computation phase, where the values $(\nu_{j,k} := \eta_{j,k}-\rho_{k,j})_{j,k\in\{1,\ldots,n\}}$ are revealed. At the end of the round, let I be the set of corrupt parties in the linear-computation phase of asif. For every $j,k \notin I$ the parties hold $\nu_{j,k}$.

For $i \notin I$ denote the output of P_k from $\mathcal{F}_i$ by (1) the message $a_{i,k}$ and the random pads $\rho_{i,k},\eta_{i,k}$, (2) the shares $(a_{i,j}[k])_{j\in\{1,\ldots,n\}}$ of P_i's first round messages (3) the shares $(s_{i,j}[k])_{j\in\{1,\ldots,2\ell\}}$ corresponding to inputs **(1)–(2)** of G_i, (4) the shares $(s_{i,j}^{b}[k])_{b\in\{0,1\},j\in\{1,\ldots,2\ell\}}$ corresponding to inputs **(3)–(4)** of G_i, (5) the encrypted messages $(A'_{i,j})_{j\in\{1,\ldots,n\}}$.

Round 4. Every P_k does as follows in the fourth round of asif.

For every $i \in I$:

- For every $j \notin I$, open the value $a_{j,i}[k]$ via asif.

For every $i \notin I$:

- *(For inputs* **(1)–(2)***.)* For every $j \in \{1,\ldots,2\ell\}$, open the value $s_{i,j}[k]$ via asif.
- *(For inputs* **(3)–(4)***.)* Let $L_i := I$. For $j \notin I$ locally compute $A_{j,i} := A'_{j,i}-\nu_{j,i}$. For $j \in I$, compute $A_{j,i} = a_{j,i}$, where $a_{j,i}$ is the message that P_j sends according to Π^{sm} when x_j and r_j are the all-zero string.

Consider the length-2ℓ binary string $\boldsymbol{\beta}_i = (\boldsymbol{\beta}_i[1], \ldots, \boldsymbol{\beta}_i[2\ell])$ that corresponds to $(L_i, (A_{j,i})_{j\in\{1,\ldots,n\}})$. For $j \in \{2\ell+1, 4\ell\}$, open the value $s_{i,j}^{\boldsymbol{\beta}_i[j-2\ell]}[k]$ via asif.

Local computation. The parties execute the local computation of asif. The parties do as follows.

- For every $i \in I$, set $\mathbf{x}_i$ and r_i to be the all zero string. For every $j \in I$ compute the message $a_{j,i}$ based on the input and randomness of P_j (that are set to the all zero string). For $j \notin I$, use all opened shares $a_{j,i}[k]$ in order to recover $a_{j,i}$. Compute the second-round broadcast b_i of P_i in Π^{sm} according to $\mathbf{x}_i$, r_i, and $(a_{j,i})_{j\in\{1,\ldots,n\}}$.
- For every $i \notin I$, and $j \in \{1, \ldots, 4\ell\}$, use all valid opened shares to recover the value of $\hat{G}_{i,j}$, denoted $g_{i,j}$. Given $(g_{i,j})_{j\in\{1,\ldots,4\ell\}}$ use the decoding algorithm of the randomized encoding to obtain the output b_i.

Finally, every P_i computes its output in Π^{sm} based on $\mathbf{x}_i, r_i, (a_{j,i})_{j\in\{1,\ldots,n\}}$ and the broadcast messages $(b_j)_{j\in\{1,\ldots,n\}}$.

Figure 20: Protocol mpc

In Section G we prove the following theorem.

Theorem 10.1. *Let κ be a security parameter, let $\mathcal{F}$ be a degree-2 functionality with boolean circuit size s and depth d. Protocol* mpc *is a UC-secure implementation of $\mathcal{F}$, against a static, active, rushing adversary corrupting up to t parties. The complexity of the protocol is* $\mathrm{poly}(s, 2^d, 2^n, \kappa)$.

11 Lower Bound: Single Input Functionality

Theorem 11.1 (Lower Bound for SIF)**.** *Let $n \geq 3$ and $t \geq n/3$ be positive integers. Then there exists an n-party single input functionality that cannot be computed in two rounds with resiliency t and error $1/12$. This holds even for a non-rushing adversary.*

Let F be the 3-party single input functionality that takes $x, y \in \{0,1\}$ from D, and returns x to P_1 and the tuple $(y, x \wedge y)$ to P_2. D itself gets nothing. Formally,

$$F(x, y) := (x, (y, x \wedge y), \perp).$$

Let π be a 2-round protocol that provides ϵ-statistical security for $\epsilon = 1/12$.

Adversary $\mathcal{A}_{x,y}$. Adversary $\mathcal{A}_{x,y}$ corrupts D and acts as follows.

- In the first round D samples randomness r_D, computes $(a_D^{x,y}, a_{D,1}^{x,y}, a_{D,2}^{x,y}) := \pi_{D,1}((x, y); r_D)$, broadcasts $a_D^{x,y}$, sends $a_{D,2}^{x,y}$ to P_2, and sends $\perp$ to P_1.
- At the end of the round $\mathcal{A}_{x,y}$ receives the messages $(a_1, a_{1,D})$ from P_1, and $(a_2, a_{2,D})$ from P_2. $\mathcal{A}_{x,y}$ samples $\bar{r}_1$ conditioned on the event $\bar{a}_1 = a_1$, where $(\bar{a}_1, \bar{a}_{1,D}, \bar{a}_{1,2}) = \pi_{1,1}(\bar{r}_1)$.
- In the second round $\mathcal{A}$ computes $b_D^{x,y} := \pi_{D,1}((a_1, \bar{a}_{1,D}, a_2, a_{2,D}), (x, y); r_D)$ and broadcasts $b_D^{x,y}$.

Adversary $\mathcal{B}$. Adversary $\mathcal{B}$ corrupts P_1 and acts as follows.

- In the first round P_1 samples randomness r_1, and computes $(a_1, a_{1,D}, a_{1,2}) := \pi_{1,1}(r_1)$. P_1 samples $\bar{r}_1$ conditioned on $\bar{a}_1 = a_1$, where $(\bar{a}_1, \bar{a}_{1,D}, \bar{a}_{1,2}) = \pi_{1,1}(\bar{r}_1)$. P_1 broadcasts a_1, sends $a_{1,2}$ to P_2, and $\bar{a}_{1,D}$ to D.
- At the end of the first round $\mathcal{B}$ receives the messages $(a_D, a_{D,1})$ from D, and $(a_2, a_{2,1})$ from P_2. The adversary sets $\bar{a}_{D,1} := \perp$.
- In the second round $\mathcal{A}$ computes $b_1 := \pi_{1,2}((a_D, \bar{a}_{D,1}, a_2, a_{2,1}); r_1)$ and broadcasts b_1.
- At the end of the round $\mathcal{B}$ receives the second-round broadcasts b_D and b_2.
- We define the *imaginary view* of $\mathcal{B}$ to be $(r_1, a_D, \bar{a}_{D,1}, a_2, a_{2,1}, b_D, b_2)$.

Adversary $\mathcal{C}$. Adversary $\mathcal{C}$ corrupts P_2 and plays honestly.

Analysis. We start with the following simple observation.

Observation 11.2. *The view of P_2 in an execution of $\pi_{\mathcal{A}_{x,y}}$ has the same distribution as in an execution of $\pi_{\mathcal{B}}(x, y)$. In addition, the view of P_1 in an execution of $\pi_{\mathcal{A}_{x,y}}$ has the same distribution as the imaginary view of $\mathcal{B}$ in an execution of $\pi_{\mathcal{B}}(x, y)$.*

We continue with the following lemma.

Lemma 11.3. *The output of P_2 in $\pi_{\mathcal{A}_{x,y}}$ is $(y, x \wedge y)$ with probability at least $1 - \epsilon$.*

Proof. Consider an execution $\pi_{\mathcal{A}_{x,y}}$. By Observation 11.2, the view of P_2 has the same distribution as in an execution of $\pi(x, y)$ with adversary $\mathcal{B}$. By the correctness of π the output of P_2 in $\pi_{\mathcal{B}}(x, y)$ is $(y, x \wedge y)$ with probability $1 - \epsilon$. Therefore, the output of P_2 in $\pi_{\mathcal{A}_{x,y}}$ is $(y, x \wedge y)$ with probability at least $1 - \epsilon$. □

Case I ($x = y = 1$). We show that the output of P_1 in $\pi_{\mathcal{A}_{1,1}}$ is 1 with probability $1 - 3\epsilon$. By Lemma 11.3 the output of P_2 in $\pi_{\mathcal{A}_{1,1}}$ is $(1, 1)$ with probability at least $1 - \epsilon$. Therefore, in the ideal-world with $\mathcal{A}_{1,1}$, the output of P_2 is $(1, 1)$ with probability at least $1 - 2\epsilon$. This means that the inputs of $\mathcal{A}_{1,1}$ must be $(x = 1, y = 1)$ with probability at least $1 - 2\epsilon$. We conclude that in the real-world, the output of P_1 and P_2 is $(1, (1, 1))$ with probability at least $1 - 3\epsilon$.

Case II ($x = 1, y = 0$). We show that the output of P_1 in $\pi_{\mathcal{A}_{1,0}}$ is 1 with probability at least $1 - 5\epsilon$. By security against $\mathcal{B}$, we conclude that the imaginary-view of $\mathcal{B}$ in $\pi_{\mathcal{B}}(1, 1)$ is ϵ-close to the simulated view. Since the simulator sees the same output when $(x = 1, y = 1)$ and $(x = 1, y = 0)$, we conclude that the simulated view is ϵ-close to the imaginary-view in $\pi_{\mathcal{B}}(1, 0)$. Therefore, the imaginary-view of $\mathcal{B}$ in $\pi_{\mathcal{B}}(1, 1)$ is 2ϵ-close to the imaginary-view in $\pi_{\mathcal{B}}(1, 0)$.

By the analysis of Case I we know that the output of P_1 in $\pi_{\mathcal{A}_{1,1}}$ is 1 with probability at least $1 - 3\epsilon$. By Observation 11.2 the view of P_1 in $\pi_{\mathcal{A}_{1,1}}$ has the same distribution as the imaginary-view of $\mathcal{B}$ in $\pi_{\mathcal{B}}(1, 1)$, so the output of $\mathcal{B}$ in $\pi_{\mathcal{B}}(1, 1)$ is 1 with probability at least $1 - 3\epsilon$. Therefore, the output of P_1 in $\pi_{\mathcal{B}}(1, 0)$ is 1 with probability at least $1 - 5\epsilon$. By Observation 11.2 the view of P_1 in $\pi_{\mathcal{A}_{1,0}}$ has the same distribution as the imaginary-view of $\mathcal{B}$ in $\pi_{\mathcal{B}}(1, 0)$, so the output of P_1 in $\pi_{\mathcal{A}_{1,0}}$ is 1 with probability at least $1 - 5\epsilon$.

Case III ($x = 0, y = 0$). We show that the output of P_1 in $\pi_{\mathcal{A}_{0,0}}$ is 1 with probability at least $1 - 7\epsilon$. Denote the view of P_1 in an execution $\pi_{\mathcal{A}_{x,y}}$ by

$$(r_1^{x,y}, a_D^{x,y}, a_{D,1}^{x,y}, a_2^{x,y}, a_{2,1}^{x,y}, b_D^{x,y}, b_2^{x,y}).$$

Observe that $a_{D,1}^{x,y}$ is $\perp$ with probability 1.

First, we note that

$$(r_2^{0,0}, a_D^{0,0}, a_{D,2}^{0,0}, a_1^{0,0}, b_D^{0,0}) \quad \text{is } 2\epsilon\text{-close to} \quad (r_2^{1,0}, a_D^{1,0}, a_{D,2}^{1,0}, a_1^{1,0}, b_D^{1,0}). \tag{1}$$

Indeed, the LHS has the same distribution as the partial view of P_2 in an honest execution with $(x = 0, y = 0)$ and the RHS has the same distribution as the partial view of P_2 in an honest execution with $(x = 1, y = 0)$, and the claim follows by security against $\mathcal{C}$. We continue by showing that

$$(r_1^{0,0}, a_D^{0,0}, a_{D,1}^{0,0}, a_2^{0,0}, a_{2,1}^{0,0}, b_D^{0,0}, b_2^{0,0}) \quad \text{is } 2\epsilon\text{-close to} \quad (r_1^{1,0}, a_D^{1,0}, a_{D,1}^{1,0}, a_2^{1,0}, a_{2,1}^{1,0}, b_D^{1,0}, b_2^{1,0}). \tag{2}$$

Consider the following probabilistic process. Given $(r_2, a_D, a_{D,2}, a_1, b_D)$ as an input, the process (1) sets $a_{D,1} := \perp$, (2) samples r_1 conditioned on $(a_1, a_{1,D}, a_{1,2}) = \pi_{1,1}(r_1)$, for some $a_{1,D}, a_{1,2}$, (3) computes $(a_2, a_{2,D}, a_{2,1}) = \pi_{2,1}(r_2)$, (4) computes $b_2 = \pi_{2,2}(a_D, a_{D,2}, a_1, a_{1,2}; r_2)$, and (5) outputs $(r_1, a_D, a_{D,1}, a_2, a_{2,1}, b_D, b_2)$. It is not hard to see that given a sample from the LHS (resp., RHS) of Equation 1 the process outputs a sample from the LHS (resp., RHS) of Equation 2, and the claim follows. Finally, since the output of P_1 in $\pi_{\mathcal{A}_{1,0}}$ is 1 with probability at least $1 - 5\epsilon$, then the output of P_1 in $\pi_{\mathcal{A}_{0,0}}$ is 1 with probability at least $1 - 7\epsilon$.

Case IV ($x = 0, y = 1$). We show that the output of P_1 in $\pi_{\mathcal{A}_{1,0}}$ is 1 with probability at least $1 - 9\epsilon$. By security against $\mathcal{B}$, we conclude that the imaginary-view of $\mathcal{B}$ in $\pi_{\mathcal{B}}(0, 0)$ is ϵ-close to the simulated view. Since the simulator sees the same output when $(x = 0, y = 0)$ and $(x = 0, y = 1)$, we conclude that the simulated view is ϵ-close to the imaginary-view in $\pi_{\mathcal{B}}(0, 1)$. Therefore, the imaginary-view of $\mathcal{B}$ in $\pi_{\mathcal{B}}(0, 0)$ is 2ϵ-close to the imaginary-view in $\pi_{\mathcal{B}}(0, 1)$.

By the analysis of Case III we know that the output of P_1 in $\pi_{\mathcal{A}_{0,0}}$ is 1 with probability at least $1 - 7\epsilon$. By Observation 11.2 the view of P_1 in $\pi_{\mathcal{A}_{0,0}}$ has the same distribution as the imaginary-view of $\mathcal{B}$ in $\pi_{\mathcal{B}}(0, 0)$, so the output of $\mathcal{B}$ in $\pi_{\mathcal{B}}(0, 0)$ is 1 with probability at least $1 - 7\epsilon$. Therefore, the output of P_1 in $\pi_{\mathcal{B}}(0, 1)$ is 1 with probability at least $1 - 9\epsilon$. By Observation 11.2 the view of P_1 in $\pi_{\mathcal{A}_{0,1}}$ has the same distribution as the imaginary-view of $\mathcal{B}$ in $\pi_{\mathcal{B}}(0, 1)$, so the output of P_1 in $\pi_{\mathcal{A}_{0,1}}$ is 1 with probability at least $1 - 9\epsilon$.

Contradiction. By Lemma 11.3 the output of P_2 in $\pi_{\mathcal{A}_{0,1}}$ is $(1, 0)$ with probability at least $1 - \epsilon$. We've seen that the output of P_1 is 1 with probability $1 - 9\epsilon$, so the probability that the output of P_1 and P_2 is $(1, (1, 0))$ is at least $1 - 10\epsilon$. Note that $(1, (1, 0))$ is an output that cannot occur in the ideal-world, so the probability that it occurs is at most ϵ. Therefore, $1 - 10\epsilon \leq \epsilon$ so $1/12 = \epsilon \geq 1/11$, in contradiction. This concludes the proof.

References

[ABT18] Benny Applebaum, Zvika Brakerski, and Rotem Tsabary. Perfect secure computation in two rounds. In *Theory of Cryptography - 16th International Conference, TCC 2018, Panaji, India, November 11-14, 2018, Proceedings, Part I*, pages 152–174, 2018.

[ABT19] 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.

[ACGJ18] 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.

[ACGJ19] Prabhanjan Ananth, Arka Rai Choudhuri, Aarushi Goel, and Abhishek Jain. Two round information-theoretic MPC with malicious security. 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 532–561, 2019.

[ACJ17] P. Ananth, A. R. Choudhuri, and A. Jain. A new approach to round-optimal secure multiparty computation. In *CRYPTO*, 2017.

[AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. *SIAM Journal on Computing*, 36(4):845–888, 2006.

[AKP20a] Benny Applebaum, Eliran Kachlon, and Arpita Patra. The resiliency of MPC with low interaction: The benefit of making errors (extended abstract). In *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part II*, pages 562–594, 2020.

[AKP20b] 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, 2020.

[AKP21] Benny Applebaum, Eliran Kachlon, and Arpita Patra. Round-optimal honest-majority MPC in minicrypt and with everlasting security. *IACR Cryptol. ePrint Arch.*, 2021:346, 2021. To appear in TCC 2022.

[AKP22] Benny Applebaum, Eliran Kachlon, and Arpita Patra. Verifiable relation sharing and multi-verifier zero-knowledge in two rounds: Trading nizks with honest majority. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022*, pages 33–56. Springer, 2022.

[AL17] Gilad Asharov and Yehuda Lindell. A full proof of the BGW protocol for perfectly secure multiparty computation. *J. Cryptology*, 30(1):58–151, 2017.

[App17] Benny Applebaum. Garbled circuits as randomized encodings of functions: a primer. In *Tutorials on the Foundations of Cryptography.*, pages 1–44. 2017.

[BD91] Mike Burmester and Yvo Desmedt. Broadcast interactive proofs (extended abstract). In *Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings*, pages 81–95, 1991.

[Bea91] D. Beaver. Efficient Multiparty Protocols Using Circuit Randomization. In *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15*, pages 420–432, 1991.

[BGJ+18] Saikrishna Badrinarayanan, Vipul Goyal, Abhishek Jain, Yael Tauman Kalai, Dakshita Khurana, and Amit Sahai. Promise zero knowledge and its applications to round optimal mpc. In *Annual International Cryptology Conference*, pages 459–487. Springer, 2018.

[BGW88] 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.

[BHP17] Zvika Brakerski, Shai Halevi, and Antigoni Polychroniadou. Four round secure computation without setup. Cryptology ePrint Archive, Report 2017/386, 2017. http://eprint.iacr.org/2017/386.

[BJMS20] 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.

[BL18] Fabrice Benhamouda and Huijia Lin. k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, pages 500–532, 2018.

[BMR90] 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.

[Can01] 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.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 11–19, 1988.

[CDD+99] Ronald Cramer, Ivan Damgård, Stefan Dziembowski, Martin Hirt, and Tal Rabin. Efficient multiparty computations secure against an adaptive adversary. In *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, pages 311–326, 1999.

[CDF01] Ronald Cramer, Ivan Damgård, and Serge Fehr. On the cost of reconstructing a secret, or VSS with optimal reconstruction phase. In *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, pages 503–523, 2001.

[CDN15] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015.

[CFIK03] Ronald Cramer, Serge Fehr, Yuval Ishai, and Eyal Kushilevitz. Efficient multi-party computation over rings. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 596–613. Springer, 2003.

[CFOR12] Alfonso Cevallos, Serge Fehr, Rafail Ostrovsky, and Yuval Rabani. Unconditionally-secure robust secret sharing with compact shares. 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 195–208, 2012.

[CGMA85] Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*, pages 383–395, 1985.

[CK89] B. Chor and E. Kushilevitz. A zero-one law for boolean privacy. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, STOC '89, page 62–72, New York, NY, USA, 1989. Association for Computing Machinery.

[CKPR01] 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.

[Cle86] 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.

[DR85] Danny Dolev and Rüdiger Reischuk. Bounds on information exchange for byzantine agreement. *J. ACM*, 32(1):191–204, 1985.

[FGG+06] Matthias Fitzi, Juan A. Garay, Shyamnath Gollakota, C. Pandu Rangan, and K. Srinathan. Round-optimal and efficient verifiable secret sharing. In *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, pages 329–342, 2006.

[FM85] 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.

[FY20] Serge Fehr and Chen Yuan. Robust secret sharing with almost optimal share size and security against rushing adversaries. In *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part III*, pages 470–498, 2020.

[GIKR01] Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. The round complexity of verifiable secret sharing and secure multicast. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 580–589, 2001.

[GIKR02] 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.

[GIS18] Sanjam Garg, Yuval Ishai, and Akshayaram Srinivasan. Two-round MPC: information-theoretic and black-box. In *Theory of Cryptography - 16th International Conference, TCC 2018, Panaji, India, November 11-14, 2018, Proceedings, Part I*, pages 123–151, 2018.

[GK96] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM J. Comput.*, 25(1):169–192, 1996.

[GLS15] 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.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 218–229, 1987.

[GS17] Sanjam Garg and Akshayaram Srinivasan. Garbled protocols and two-round mpc from bilinear maps. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 588–599. IEEE, 2017.

[GS18] Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, pages 468–499, 2018.

[HHPV21] Shai Halevi, Carmit Hazay, Antigoni Polychroniadou, and Muthuramakrishnan Venkitasubramaniam. Round-optimal secure multi-party computation. *Journal of Cryptology*, 34(3):1–63, 2021.

[IK00] 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.

[IK02] 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.

[KKK09] Jonathan Katz, Chiu-Yuen Koo, and Ranjit Kumaresan. Improving the round complexity of VSS in point-to-point networks. *Inf. Comput.*, 207(8):889–899, 2009.

[KPR10] Ranjit Kumaresan, Arpita Patra, and C. Pandu Rangan. The round complexity of verifiable secret sharing: The statistical case. In *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, pages 431–447, 2010.

[LF82] Leslie Lamport and Michael Fischer. Byzantine generals and transaction commit protocols. Technical report, Technical Report 62, SRI International, 1982.

[MNS16] Tal Moran, Moni Naor, and Gil Segev. An optimally fair coin toss. *J. Cryptology*, 29(3):491–513, 2016.

[PCPR09] Arpita Patra, Ashish Choudhary, and Chandrasekharan Pandu Rangan. Simple and efficient asynchronous byzantine agreement with optimal resilience. In *Proceedings of the 28th ACM symposium on Principles of distributed computing*, pages 92–101, 2009.

[PCR08] Arpita Patra, Ashish Choudhary, and C Pandu Rangan. Round efficient unconditionally secure multiparty computation protocol. In *Progress in Cryptology-INDOCRYPT 2008: 9th International Conference on Cryptology in India, Kharagpur, India, December 14-17, 2008. Proceedings*, volume 5365, page 185. Springer, 2008.

[PCRR09] Arpita Patra, Ashish Choudhary, Tal Rabin, and C. Pandu Rangan. The round complexity of verifiable secret sharing revisited. In *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, pages 487–504, 2009.

[RB89] 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.

[RCCG+20] Arka Rai Choudhuri, Michele Ciampi, Vipul Goyal, Abhishek Jain, and Rafail Ostrovsky. Round optimal secure multiparty computation from minimal assumptions. In *Theory of Cryptography Conference*, pages 291–319. Springer, 2020.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 162–167, 1986.

A Appendix: SIF does not imply VSS

In this section we prove that SIF does not capture VSS when $n \leq 3t-1$. Closely related statements have appeared in [BGW88, CCD88, RB89], and the proof is mainly given here for the sake of completeness. In Section A.1 we provide a formal definition of VSS as a standalone primitive. Then, in Section A.2 we prove the impossibility result.

A.1 Definition of VSS

The following definition is taken verbatim from [AKP20a].

Definition A.1 (ϵ-secure VSS)**.** *Let Y be a finite domain, $|Y| \geq 2$, and let $\mathcal{P}$ be a set of parties that includes a distinguished* dealer *$D \in \mathcal{P}$. A VSS protocol consists of two phases, a* sharing phase *and a* reconstruction phase*, with the following syntax.*

- Sharing: *At the beginning, D holds a* secret *$s \in Y$ and each party including the dealer holds an independent random input r_i. The sharing phase may span over several rounds. At each round, each party can privately send messages to the other parties and it can also broadcast a message. Each message sent or broadcasted by P_i is determined by the view of P_i, consists of its input (if any), its random input and messages received from other parties in previous rounds.*

- Reconstruction: *At the beginning of the reconstruction, the parties are holding their view from the sharing phase. The reconstruction phase may span over several rounds, and at each round the parties send messages based on their view. At the end of the reconstruction, each party outputs a value.*

Let $\epsilon > 0$. A two-phase, n-party protocol as above is called an ϵ-secure (n,t)-VSS, if for any adversary $\mathcal{A} = (\mathcal{A}_{\mathsf{sh}}, \mathcal{A}_{\mathsf{rec}})$ corrupting at most t parties, the following holds:

- Correctness: *If D is honest then all honest parties output s at the end of the reconstruction phase, with probability at least $1-\epsilon$.*

- Privacy: *If D is honest then the adversary's view during the sharing phase reveals almost no information on s. Formally, let $\mathcal{D}_s$ is the view $\mathcal{A}$ in the sharing phase on secret s. Then, for any $s \neq s'$, the random variables $\mathcal{D}_s$ and $\mathcal{D}_{s'}$ are ϵ-close in statistical-distance.*

- Commitment: *If D is corrupt then, except with probability $1-\epsilon$, at the end of the sharing phase there is a value $s^* \in Y$ such that at the end of the reconstruction phase the output is s^*. More formally, we assume that an adversary $\mathcal{A}$ that corrupts D is a two-phase adversary $\mathcal{A} = (\mathcal{A}_{\mathsf{sh}}, \mathcal{A}_{\mathsf{rec}})$ where $\mathcal{A}_{\mathsf{sh}}$ takes randomness $r_{\mathcal{A}}$, plays the sharing phase and outputs a state Z. At the reconstruction phase $\mathcal{A}_{\mathsf{rec}}$ gets Z and, in addition, a bit σ and tries to flip the outcome depending on σ. Specifically, let H denote the set of honest parties. For $r = (r_i)_{i \in \mathsf{H}}$ and $\sigma \in \{0,1\}$ denote by $y_{r,r_{\mathcal{A}}}(i,\sigma)$ the final output of party P_i in an execution with $\mathcal{A}_{\mathsf{sh}}(r_{\mathcal{A}}), \mathcal{A}_{\mathsf{rec}}(Z,\sigma)$ where the random tape of an honest party P_j is set to r_j. Then the commitment property requires that*

$$\Pr_{r,r_{\mathcal{A}}}[\exists s^* \in Y : \forall i \in \mathsf{H}, \sigma \in \{0,1\}, s^* = y(i,\sigma)] > 1-\epsilon.$$

A.2 Impossibility Result

Consider an n-party VSS protocol Π with parties $\mathcal{D} = P_1, \ldots, P_n$ with the following structure.

- The sharing phase of Π consists of the computation of a single input functionality $\mathcal{F}$, that takes from $\mathcal{D}$ the secret s and randomness r, and returns a share s_i to P_i.
- In the reconstruction phase, we assume that the parties execute a (possibly multi-round) protocol rec where every P_i inputs s_i.

We prove that Π has error at least $1/10$.

Assume towards contradiction that Π is a VSS protocol with error at most $\epsilon \leq 1/10$. We construct two adversaries, $\mathcal{A}$ and $\mathcal{B}$, so that one of them violates the commitment property of Π with probability $1/10$, which is a contradiction.

Adversary $\mathcal{A}$. Let $\mathcal{A}$ be the adversary that corrupts $\mathcal{D} = P_1, \ldots, P_t$. In the sharing phase, $\mathcal{A}$ samples a random string r, and inputs $(0, r)$ to $\mathcal{F}$. Let $\mathcal{F}(0, r) = (s_1, \ldots, s_n)$. In the reconstruction phase, we present two strategies for $\mathcal{A}$.

- **Strategy $\mathcal{A}_0$:** The corrupt parties play the reconstruction phase honestly with inputs $s_1, \ldots, s_t$.
- **Strategy $\mathcal{A}_1$:** The adversary samples randomness r' conditioned on $\mathcal{F}(1, r')_i = s_i$ for every $i \in \{2t, \ldots, 3t-1\}$, where $\mathcal{F}(1, r')_i$ is the i-th output of $\mathcal{F}(1, r')$. Let $\mathcal{F}(1, r') = (s'_1, \ldots, s'_{2t-1}, s_{2t}, \ldots, s_{3t-1})$. $\mathcal{D}$ aborts and does not send any more messages. The corrupt parties $P_2, \ldots, P_t$ play honestly as if the shares they received are $s'_2, \ldots, s'_t$.

By the correctness property, when $\mathcal{A}$ picks strategy $\mathcal{A}_0$ the output is 0 with probability $1 - \epsilon$. Let

$$p := \Pr[\mathsf{rec}_{3t-1}(\perp, s'_2, \ldots, s'_t, s_{t+1}, \ldots, s_{3t-1}) = 1]$$

where $\mathsf{rec}_{3t-1}(\perp, s'_2, \ldots, s'_t, s_{t+1}, \ldots, s_{3t-1})$ denotes the output of P_{3t-1} in the reconstruction phase when $\mathcal{D}$ aborts, and $P_2, \ldots, P_n$ play honestly with inputs $s'_2, \ldots, s'_t, s_{t+1}, \ldots, s_{3t-1}$, respectively, sampled as in strategy $\mathcal{A}_1$.

Adversary $\mathcal{B}$. Let $\mathcal{B}$ be the adversary that corrupts $\mathcal{D} = P_1, P_{t+1} \ldots, P_{2t-1}$. In the sharing phase, $\mathcal{B}$ samples a random string $\bar{r}$, and inputs $(1, \bar{r})$ to $\mathcal{F}$. Let $\mathcal{F}(1, \bar{r}) = (\bar{s}_1, \ldots, \bar{s}_n)$. In the reconstruction phase, we present two strategies for $\mathcal{B}$.

- **Strategy $\mathcal{B}_0$:** The adversary samples randomness $\tilde{r}$ conditioned on $\mathcal{F}(0, \tilde{r})_i = \bar{s}_i$ for every $i \in \{2t, \ldots, 3t-1\}$. Let $\mathcal{F}(0, \tilde{r}) = (\tilde{s}_1, \ldots, \tilde{s}_{2t-1}, \bar{s}_{2t}, \ldots, \bar{s}_{3t-1})$. $\mathcal{D}$ aborts and does not send any more messages. The corrupt parties $P_{t+1}, \ldots, P_{2t-1}$ play honestly as if the shares they received are $\tilde{s}_{t+1}, \ldots, \tilde{s}_{2t-1}$.
- **Strategy $\mathcal{B}_1$:** The corrupt parties play the reconstruction phase honestly with inputs $\bar{s}_1, \bar{s}_{t+1} \ldots, \bar{s}_{2t-1}$.

By the correctness property, when $\mathcal{B}$ picks strategy $\mathcal{B}_1$ the output is 1 with probability $1-\epsilon$. Let

$$q := \Pr[\mathsf{rec}_{3t-1}(\perp, \bar{s}_2, \ldots, \bar{s}_t, \tilde{s}_{t+1}, \ldots, \tilde{s}_{2t-1}, \bar{s}_{2t}, \ldots, \bar{s}_{3t-1}) = 1]$$

where $\mathsf{rec}_{3t-1}(\perp, \bar{s}_2, \ldots, \bar{s}_t, \tilde{s}_{t+1}, \ldots, \tilde{s}_{2t-1}, \bar{s}_{2t}, \ldots, \bar{s}_{3t-1})$ denotes the output of P_{3t-1} in the reconstruction phase when $\mathcal{D}$ aborts, and $P_2, \ldots, P_n$ play honestly with inputs $\bar{s}_2, \ldots, \bar{s}_t, \tilde{s}_{t+1}, \ldots, \tilde{s}_{2t-1}, \bar{s}_{2t}, \ldots, \bar{s}_{3t-1}$, respectively, sampled as in strategy $\mathcal{B}_0$.

Analysis. First, we prove that $p \geq q - \epsilon$. Indeed, by the privacy property, the random variables $s_{2t}, \ldots, s_{3t-1}$ are ϵ-close in statistical distance to the random variables $\bar{s}_{2t}, \ldots, \bar{s}_{3t-1}$. Now, the random variables $(s'_2, \ldots, s'_t, s_{t+1}, \ldots, s_{3t-1})$ are ϵ-close to $(\bar{s}_2, \ldots, \bar{s}_t, \tilde{s}_{t+1}, \ldots, \tilde{s}_{2t-1}, \bar{s}_{2t}, \ldots, \bar{s}_{3t-1})$ because they can be sampled by the same randomized procedure given either $s_{2t}, \ldots, s_{3t-1}$ or $\bar{s}_{2t}, \ldots, \bar{s}_{3t-1}$. Concretely, given $s_{2t}, \ldots, s_{3t-1}$ the randomized procedure samples r and r' conditioned on $\mathcal{F}(0,r)_i = \mathcal{F}(1,r')_i = s_i$ for all $i \in \{2t, \ldots, 3t-1\}$, computes $s_i := \mathcal{F}(1,r')_i$ for all $i \in \{2, \ldots, t\}$ and $s_i := \mathcal{F}(0,r)_i$ for all $i \in \{t+1, \ldots, 2t-1\}$, and returns $(s_2, \ldots, s_{3t-1})$. It can be verified that the procedure perfectly samples $(s'_2, \ldots, s'_t, s_{t+1}, \ldots, s_{3t-1})$ (resp., $(\bar{s}_2, \ldots, \bar{s}_t, \tilde{s}_{t+1}, \ldots, \tilde{s}_{2t-1}, \bar{s}_{2t}, \ldots, \bar{s}_{3t-1})$) given $(s_{2t}, \ldots, s_{3t-1})$ (resp., $(\bar{s}_{2t}, \ldots, \bar{s}_{3t-1})$).

If $q \leq 1/2$ then the probability that the output of P_{3t-1} is 0 if $\mathcal{B}$ follows strategy $\mathcal{B}_0$ is at least $1 - q \geq 1/2$. This means that with probability at least $1 - \epsilon - 1/2 = 1/2 - \epsilon$, if $\mathcal{B}$ picks strategy 0 the output is 0, and if $\mathcal{B}$ picks strategy 1 the output is 1. Since $1/2 - \epsilon > \epsilon$, the commitment property is violated, in contradiction. Therefore, $q > 1/2$. This means that $p > 1/2 - \epsilon$. But now the same argument with respect to $\mathcal{A}$ shows that the commitment property is violated with probability $1/2 - 2\epsilon > \epsilon$, in contradiction. This completes the proof.

B Appendix: Standard Useful Facts

B.1 Polynomials

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., [AL17]).

Fact B.1. *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 B.2. *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 B.3. *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 $\mathcal{P}^{s',t}$.

B.2 Secret Sharing

The following fact is adopted from [CDN15, Theorem 6.6].

Fact B.4. *Let $A \in \mathbb{F}^{u \times v}$ be a $u \times v$ matrix so that the unit vector $(1, 0, \ldots, 0)$ is not spanned by the rows of A. Let $s, s' \in \mathbb{F}$ be two elements in $\mathbb{F}$, and let $\mathbf{f}, \mathbf{f}' \in \mathbb{F}^v$ be two random vectors conditioned on $\mathbf{f}[1] = s$ and $\mathbf{f}'[1] = s'$. Then the distribution of $A \cdot \mathbf{f}$ is identical to the distribution of $A \cdot \mathbf{f}'$.*

C Linear Private-Opening Signature Scheme

In this section, we prove that protocol poSig UC-emulates $\mathcal{F}_{\mathsf{poSig}}$ with statistical security. Let $\mathcal{A}$ be the dummy adversary. We define the simulator as follows. The simulator 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 parties C. We split into cases.

C.1 Honest $\mathcal{D}$ and $\mathcal{I}$

C.1.1 The Simulator

Simulation of poSig.dis**.** The simulator sets $\mathbf{s} = (0, \ldots, 0)$, takes the role of the honest parties, where $\mathcal{D}$ has input $\mathbf{s}$, and executes poSig.dis. The simulator computes the outgoing messages of $\mathcal{D}$ in poSig.dis, and gives them to the respective parties, where messages to honest parties are transferred to the simulated honest parties, and messages to corrupt parties are transferred to the adversary. Denote the polynomials that $\mathcal{D}$ picked by $f_1(x), \ldots, f_m(x), r_1(x), \ldots, r_m(x)$, and let $(\alpha_i, f_{1,i}, \ldots, f_{m,i}, r_{1,i}, \ldots, r_{m,i})$ be the message from $\mathcal{D}$ to P_i.

Simulation of poSig.ver**.** In poSig.ver the simulator continues by computing the broadcast message $(c, d_k(x))_{k \in [m]}$ of $\mathcal{I}$, and giving it to the adversary.

Simulation of poSig.open**.** In poSig.open, the simulator first receives the outputs of the corrupt parties, denoted $(\boldsymbol{\beta}_{i,j}, g_{i,j})_{i \in \mathsf{C}, j \in [\ell]}$, where $g_{i,j} \in \mathbb{F}$. The simulator does as follows.

- *($\mathcal{D}$ to corrupt parties)* The simulator broadcasts "accept" on behalf of $\mathcal{D}$.
- *($\mathcal{I}$ to corrupt parties)* The simulator finds any pre-image $\bar{\mathbf{s}} = (\bar{s}_1, \ldots, \bar{s}_m)$ such that $\boldsymbol{\beta}_{i,j} \cdot \bar{\mathbf{s}} = g_{i,j}$ for all $i \in \mathsf{C}$ and $j \in [\ell]$. (This can be done in polynomial time by Guassian elimination.)

  The simulator samples a random degree-d polynomial $\bar{f}_k(x)$ conditioned on (1) $\bar{f}_k(\alpha_i) = f_{k,i}$ for all $i \in \mathsf{C}$ and $k \in [m]$, and (2) $\bar{f}_k(0) = \bar{s}_k$ for every $k \in [m]$.

  The simulator computes $\overline{\mathsf{Out}}_{i,j}(x) := \boldsymbol{\beta}_{i,j}[1] \cdot \bar{f}_1(x) + \ldots + \boldsymbol{\beta}_{i,j}[m] \cdot \bar{f}_m(x)$ for every $i \in \mathsf{C}$ and $j \in [\ell]$, and sends $(\boldsymbol{\beta}_{i,j}, \overline{\mathsf{Out}}_{i,j}(x))_{j \in [\ell]}$ to every corrupt P_i on behalf of $\mathcal{I}$.

This concludes the simulation.

C.1.2 Analysis

The environment $\mathcal{Z}$ sees during the execution (1) the inputs $(s_1, \ldots, s_m)$ of $\mathcal{D}$, that are picked by the environment, (2) the outputs $(s_1, \ldots, s_m)$ of $\mathcal{I}$ in $\mathsf{poSig.dis}$, (3) the values $(\alpha_i, f_{1,i}, \ldots, f_{m,i}, r_{1,i}, \ldots, r_{m,i})_{i \in \mathsf{C}}$ received from $\mathcal{D}$, (4) the broadcast of $\mathcal{D}$ in $\mathsf{poSig.open}$, (5) the broadcast $(c, d_k(x))_{k \in [m]}$ of $\mathcal{I}$, (6) the inputs $(\boldsymbol{\beta}_{i,j})_{i \in [n], j \in [\ell]}$ to $\mathcal{I}$, (7) the values $(\boldsymbol{\beta}_{i,j}, \mathsf{Out}_{i,j}(x))_{i \in \mathsf{C}, j \in [\ell]}$ from $\mathcal{I}$, and (9) the outputs of the honest parties.

The inputs $(s_1, \ldots, s_m)$ are chosen by $\mathcal{Z}$ in the same way in both worlds, so (1) has the same distribution in both worlds. Fix (1). In the ideal world the output of $\mathcal{I}$ is always $(s_1, \ldots, s_m)$. It is not hard to see that this is also the case in the real world. This means that (2) is the same in both worlds.

Let $S \subseteq \mathbb{F} \setminus \{0\}$ be the set of all points α_i that appear in (3), and observe that $|S| \leq n = d$. All those points are chosen uniformly at random both in the real world and the ideal world, so they have the same distribution. Fix those points. Conditioned on those points, (3) consists of evaluations of each polynomial in $f_1(x), \ldots, f_m(x), r_1(x), \ldots, r_m(x)$ on the points of S. Observe that for every $k \in [m]$ it holds that (a) in the real world $f_k(x)$ is a random degree-d polynomial conditioned on $f_k(0) = s_k$, and in the ideal world $f_k(x)$ is a random degree-d polynomial conditioned on $f_k(0) = 0$, and (b) $r_k(x)$ is a random degree-d polynomial both in the real world and in the ideal world. Since $|S| \leq d$, by Fact B.1 those evaluations have the same distribution in both worlds. We conclude that (3) has the same distribution in both worlds, and we fix those values.

Since $\mathcal{D}$ and $\mathcal{I}$ are honest, the broadcast of $\mathcal{D}$ in the real-world is always "accept" just like in the ideal-world. Therefore, it remains to show that conditioned on (1)–(4), the partial view (5)–(8) has the same distribution in both worlds. In both worlds the random variable c is uniformly distributed in $\mathbb{F} \setminus \{0\}$, and we fix it. In both worlds $r_k(x)$ is a random degree-d polynomial conditioned on $r_k(\alpha_i) = r_{k,i}$ for every $i \in \mathsf{C}$. (Note that those are exactly the points in S.) Therefore, in both worlds, the random variables $d_1(x), \ldots, d_m(x)$ are uniformly distributed degree-d polynomial conditioned on $d_k(\alpha_i) = f_{k,i} + c r_{k,i}$ for every $k \in [m]$ and every i so that $i \in \mathsf{C}$. Therefore (5) has the same distribution in both worlds, and we fix it.

Conditioned on those values, observe that $(\boldsymbol{\beta}_{i,j})_{i \in [n], j \in [\ell]}$ are picked by the environment in the same way in both worlds. Fix those values as well. Observe that even conditioned on (1)–(6), the real-world polynomials $f_1(x), \ldots, f_m(x)$ are uniformly distributed degree-d polynomials conditioned on $f_k(0) = s_k$ and $f_k(\alpha_i) = f_{k,i}$ for every $k \in [m]$ and every i so that $i \in \mathsf{C}$. (Observe that those are exactly the points in S.) Similarly, the ideal-world polynomials $\bar{f}_1(x), \ldots, \bar{f}_m(x)$ are uniformly distributed degree-d polynomials conditioned on $\bar{f}_k(0) = \bar{s}_k$ and $\bar{f}_k(\alpha_i) = f_{k,i}$ for every i so that $i \in \mathsf{C}$. In addition, the real-world polynomials $(\mathsf{Out}_{i,j}(x))_{i \in \mathsf{C}, j \in [\ell]}$ and the ideal-world polynomials $(\overline{\mathsf{Out}}_{i,j}(x))_{i \in \mathsf{C}, j \in [\ell]}$ agree on all the points in $S \cup \{0\}$. Let $T \subseteq \mathbb{F} \setminus S$ be a set of $d - |S|$ non-zero points. Then, by Fact B.1 the real-world values $(f_k(\alpha))_{k \in [m], \alpha \in T}$ and the ideal-world values $(\bar{f}_k(\alpha))_{k \in [m], \alpha \in T}$ are uniformly distributed. This means that the real-world values $(\mathsf{Out}_{i,j}(\alpha) = \boldsymbol{\beta}_{i,j}[1] f_1(\alpha) + \ldots + \boldsymbol{\beta}_{i,j}[m] f_m(\alpha))_{i \in \mathsf{C}, j \in [\ell], \alpha \in T}$ and the ideal-world values $(\overline{\mathsf{Out}}_{i,j}(\alpha) = \boldsymbol{\beta}_{i,j}[1] \bar{f}_1(\alpha) + \ldots + \boldsymbol{\beta}_{i,j}[m] \bar{f}_m(\alpha))_{i \in \mathsf{C}, j \in [\ell], \alpha \in T}$ have the same distribution. Finally, for every fixing of those values, the real-world polynomial $\mathsf{Out}_{i,j}(x)$ and the ideal-world polynomial $\overline{\mathsf{Out}}_{i,j}(x)$ are both of degree-d and agree on $d + 1$ points (the points in $S \cup T \cup \{0\}$), which means that $\mathsf{Out}_{i,j}(x) = \overline{\mathsf{Out}}_{i,j}(x)$. Fix (7) as well.

It remains to show that the outputs of the honest parties are the same in both worlds. In the ideal world every honest P_i outputs $b = 0$ and $(\boldsymbol{\beta}_{i,j}, \boldsymbol{\beta}_{i,j} \cdot \mathbf{s})_{j \in [\ell]}$ with probability 1. In the real

world, since $\mathcal{D}$ and $\mathcal{I}$ are honest, it is not hard to see that $b = 0$ with probability 1, that every honest P_i accepts the opening, and that $\mathsf{Out}_{i,j}(0) = \boldsymbol{\beta}_{i,j} \cdot \mathbf{s}$ for all $i \in [n]$ and $j \in [\ell]$. This concludes the proof for honest $\mathcal{D}$ and $\mathcal{I}$.

C.2 Honest $\mathcal{D}$, Corrupt $\mathcal{I}$

C.2.1 The Simulator

Simulation of poSig.dis**.** At the beginning of poSig.dis, the simulator receives $(s_1, \ldots, s_m)$ from the ideal functionality. The simulator, that holds all the inputs of $\mathcal{D}$, simply initiates an execution of poSig.dis by taking the roles of the honest parties, where $\mathcal{D}$ inputs $(s_1, \ldots, s_m)$.

Simulation of poSig.ver**.** The simulator continues with the execution of poSig.ver, by receiving from the adversary the broadcast of the corrupt $\mathcal{I}$, and transfers it to all the simulated honest parties. The simulator computes the output $\mathsf{reveal}_{\mathcal{I}}$ of $\mathcal{D}$, and sends it to the ideal functionality as the input of the corrupt $\mathcal{I}$.

Simulation of poSig.open**.** The simulator continues with the execution of poSig.open, and computes the outputs of the honest parties.

If $\mathsf{reveal}_{\mathcal{I}} = 1$ then the inputs of $\mathcal{I}$ to the ideal functionality do not matter, and the simulator terminates. Otherwise, for every honest P_i the simulator does as follows. If the output of P_i is $\perp$ then the simulator inputs $\mathsf{abort}_i = 1$ to the ideal functionality. Otherwise, let $(\boldsymbol{\beta}_{i,j}, w_{i,j})_{j \in [\ell]}$ be the output of P_i, and the simulator inputs $\mathsf{abort}_i = 0$ and $(\boldsymbol{\beta}_{i,j})_{j \in [\ell]}$ to the ideal functionality.

C.2.2 Analysis

Observe that there is a 1-1 correspondence between real-world executions of the protocol and executions of the simulator, and that those executions agree on the output of $\mathcal{D}$ in poSig.ver, and on the output bit b in poSig.open. It remains to show that for $(1 - 2^{-\kappa})$-fraction of the executions, the real-world outputs of the honest parties in poSig.open is the same as the ideal-world output.

We say that an execution is "bad" if there exists some honest P_i so that (1) P_i received $(\boldsymbol{\beta}_{i,j}, \mathsf{Out}_{i,j}(x))$ from $\mathcal{I}$ for some $j \in [\ell]$, so that $\mathsf{Out}_{i,j}(x) \neq \boldsymbol{\beta}_{i,j}[1] \cdot f_1(x) + \ldots + \boldsymbol{\beta}_{i,j}[m] \cdot f_m(x)$, and (2) $\mathsf{Out}_{i,j}(\alpha_i) = \boldsymbol{\beta}_{i,j}[1] \cdot f_{1,i} + \ldots + \boldsymbol{\beta}_{i,j}[m] \cdot f_{m,i}$. Fix any honest P_i, any $j \in [\ell]$. Since $\mathcal{D}$ is honest, we have $f_{1,i} = f_1(\alpha_i), \ldots, f_{m,i} = f_m(\alpha_i)$. Since $\mathsf{Out}_{i,j}(x)$ and $f_1(x), \ldots, f_m(x)$ are all degree-d polynomials, and since α_i is uniformly distributed even conditioned on the view of the adversary, the probability that $\mathsf{Out}_{i,j}(x) \neq \boldsymbol{\beta}_{i,j}[1] \cdot f_1(x) + \ldots + \boldsymbol{\beta}_{i,j}[m] \cdot f_m(x)$ and $\mathsf{Out}_{i,j}(\alpha_i) = \boldsymbol{\beta}_{i,j}[1] \cdot f_1(\alpha_i) + \ldots + \boldsymbol{\beta}_{i,j}[m] \cdot f_m(\alpha_i)$ is at most $d/(|\mathbb{F}| - 1) \leq 2^{-\kappa}/(n\ell)$. By taking union bound over all P_i and $j \in [\ell]$, we conclude that the probability that the execution is bad is at most $2^{-\kappa}$.

Consider any good execution. If $\mathsf{reveal}_{\mathcal{I}} = 1$ then $\mathcal{D}$ broadcasts $\mathbf{s}$ in poSig.open then the output is the same in both worlds. Otherwise $\mathsf{reveal}_{\mathcal{I}} = 0$ and $\mathcal{D}$ broadcasts "accepts". This means that for every $k \in [m]$ it holds that $d_k(x) = f_k(x) + cr_k(x)$. Consider now any honest P_i. If there exists $j \in [\ell]$ so that $\mathsf{Out}_{i,j}(x) \neq \boldsymbol{\beta}_{i,j}[1] \cdot f_1(x) + \ldots + \boldsymbol{\beta}_{i,j}[m] \cdot f_m(x)$ then P_i outputs $\perp$ in both worlds. Otherwise, $\mathsf{Out}_{i,j}(x) = \boldsymbol{\beta}_{i,j}[1] \cdot f_1(x) + \ldots + \boldsymbol{\beta}_{i,j}[m] \cdot f_m(x)$ for every $j \in [\ell]$, so P_i outputs $(\boldsymbol{\beta}_{i,j}, \mathsf{Out}_{i,j}(0))_{j \in [\ell]} = (\boldsymbol{\beta}_{i,j}, \boldsymbol{\beta}_{i,j}[1] \cdot f_1(0) + \ldots + \boldsymbol{\beta}_{i,j}[m] \cdot f_m(0))_{j \in [\ell]} = (\boldsymbol{\beta}_{i,j}, \boldsymbol{\beta}_{i,j} \cdot \mathbf{s})_{j \in [\ell]}$, as required. This concludes the analysis of honest $\mathcal{D}$ and corrupt $\mathcal{I}$.

C.3 Corrupt $\mathcal{D}$, Honest $\mathcal{I}$

C.3.1 The Simulator

Simulation of poSig.dis**.** The simulator initiate an execution of poSig.dis by taking the roles of the honest parties. Let $(f_1(x), \ldots, f_m(x))$ be the polynomials that the corrupt $\mathcal{D}$ sent to $\mathcal{I}$. The simulator sets $s_i := f_i(0)$ for all $i \in [m]$, and inputs $(s_1, \ldots, s_m)$.

Simulation of poSig.ver**.** The simulator continues with the execution of poSig.ver.

Simulation of poSig.open**.** The simulator receives $(\boldsymbol{\beta}_{i,j})_{i\in\mathsf{C},j\in[\ell]}$ from the ideal functionality. In addition, the simulator computes the private opening of $\mathcal{I}$ to every corrupt P_i as follows: $\mathcal{I}$ sends $(\boldsymbol{\beta}_{i,j}, \mathsf{Out}_{i,j}(x))_{j\in[\ell]}$ to P_i, where $\mathsf{Out}_{i,j}(x) = \boldsymbol{\beta}_{i,j}[1] \cdot f_1(x) + \ldots + \boldsymbol{\beta}_{i,j}[m] \cdot f_m(x)$.

If $\mathcal{D}$ broadcasts $\mathbf{s}'$ in poSig.open, then the simulator inputs $\mathsf{reveal}_{\mathcal{D}} = 1$ and $\mathbf{s}'$ to the ideal functionality. Otherwise, the simulator inputs $\mathsf{reveal}_{\mathcal{D}} = 0$ and $\mathbf{s}$ to the ideal functionality. This concludes the simulation.

C.3.2 Analysis

Observe that there is a 1-1 correspondence between real-world executions of the protocol and executions of the simulator, and that those executions agree on the output of $\mathcal{D}$ in poSig.ver, and on the output bit b in poSig.open. It remains to show that for $(1 - 2^{-\kappa})$-fraction of the executions, the real-world outputs of the honest parties in poSig.open is the same as the ideal-world output.

We say that an execution is "bad" if there exists some honest P_i, and some $k \in [m]$ so that (1) $f_k(\alpha_i) \neq f_{k,i}$, and (2) $d_k(\alpha_i) = f_{k,i} + cr_{k,i}$. Since $d_k(\alpha_i) = f_k(\alpha_i) + cr_k(\alpha_i)$, (2) implies that $f_k(\alpha_i) + cr_k(\alpha_i) = f_{k,i} + cr_{k,i}$. Therefore (2) holds only if

$$f_k(\alpha_i) - f_{k,i} = c(r_{k,i} - r_k(\alpha_i)).$$

Since c is uniformly distributed over $\mathbb{F} \setminus \{0\}$ we conclude that if $f_k(\alpha_i) \neq f_{k,i}$ then (2) holds with probability at most $1/(|\mathbb{F}| - 1) \leq 2^{-\kappa}/(nm)$. By taking a union bound over all P_i and $k \in [m]$, we conclude that an execution is "bad" with probability at most $2^{-\kappa}$.

Consider any good execution of the protocol. If $b = 1$ then the output in both worlds is $\mathbf{s}'$, as required. Otherwise, $b = 0$. Fix any honest P_i, and observe that, since the execution is good, P_i outputs $(\boldsymbol{\beta}_{i,j}, \boldsymbol{\beta}_{i,j}[1] \cdot f_1(0) + \ldots + \boldsymbol{\beta}_{i,j}[1] \cdot f_m(0)) = (\boldsymbol{\beta}_{i,j}, \boldsymbol{\beta} \cdot \mathbf{s})$, just like in the ideal world. This concludes the analysis of corrupt $\mathcal{D}$ and honest $\mathcal{I}$.

C.4 Corrupt $\mathcal{D}$ and $\mathcal{I}$

C.4.1 The Simulator

Simulation of poSig.dis**.** The simulator initiate an execution of poSig.dis by taking the roles of the honest parties. The simulator sets $s_i := 0$ for all $i \in [m]$, and inputs $(s_1, \ldots, s_m)$.

Simulation of poSig.ver**.** The simulator continues with the execution of poSig.ver. The simulator inputs $\mathsf{reveal}_{\mathcal{I}} = 0$ to the functionality.

Simulation of poSig.open**.** The simulator continues with the execution of poSig.open. At the end of the execution the simulator computes the outputs of the honest parties.

If $\mathcal{D}$ broadcasts $\mathbf{s}'$ in poSig.open, then the simulator inputs $\mathsf{reveal}_{\mathcal{D}} = 1$ and $\mathbf{s}'$ to the ideal functionality. Otherwise, the simulator inputs $\mathsf{reveal}_{\mathcal{D}} = 0$ and $\mathbf{s}$ to the ideal functionality. In addition, if the output of P_i is $\perp$ then the simulator inputs $\mathsf{abort}_i = 1$ to the functionality (in this case the value of $(\boldsymbol{\beta}_{i,j}, z_{i,j})_{j\in[\ell]}$ doesn't matter). Otherwise, denote the output of P_i by $(\boldsymbol{\beta}_{i,j}, z_{i,j})_{j\in[\ell]}$. Then the simulator inputs $(\boldsymbol{\beta}_{i,j}, z_{i,j})_{j\in[\ell]}$ to the functionality. This concludes the simulation.

C.4.2 Analysis

It is straightforward to verify that the simulator perfectly simulates a real execution of the protocol. This concludes the analysis of protocol poSig.

D Linear Interactive Signature Scheme

In this section, we prove that protocol iSig UC-emulates $\mathcal{F}_{\mathsf{iSig}}$ with statistical security in the $\mathcal{F}_{\mathsf{poSig}}$-hybrid model. From the composition properties of UC-security, this implies that protocol iSig UC-emulates $\mathcal{F}_{\mathsf{iSig}}$. Let $\mathcal{A}$ be the dummy adversary. We define the simulator as follows. The simulator 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 parties C. We split into cases.

D.1 Honest $\mathcal{D}$ and $\mathcal{I}$

D.1.1 The Simulator

Simulation of iSig.dis**.** For every $k \in [m]$, the simulator picks a random vector $\mathbf{f}_k \in \mathbb{F}^v$ whose first entry is 0, and a random vector $\mathbf{r}_k \in \mathbb{F}^v$.

The first-round communication between $\mathcal{D}$ and a corrupt P_i consists only of the output of the distribution phase of $\mathcal{F}_{\mathsf{poSig}}$ where $\mathcal{D}$ is the dealer and P_i the intermediary. Those outputs correspond to the values $(A^S, (\mathbf{f}_k^S, \mathbf{r}_k^S)_{k\in[m]})$ for every set S that contains P_i. We simulate those values as follows. For every set S that contains a corrupt P_i,

- The simulator picks a random $u \times v$ matrix $A^S \in \mathbb{F}^{u\times v}$.
- The simulator computes $\mathbf{f}_k^S := A^S \cdot \mathbf{f}_k$ and $\mathbf{r}_k^S := A^S \cdot \mathbf{r}_k$.
- The simulator simulates the output of the distribution phase of $\mathcal{F}_{\mathsf{poSig}}$ corresponding to the set S by giving $(A^S, (\mathbf{f}_k^S, \mathbf{r}_k^S)_{k\in[m]})$ to every corrupt P_i in S.

We also let the simulator sample a random $u \times v$ matrix $A^S \in \mathbb{F}^{u\times v}$, and compute $\mathbf{f}_k^S := A^S \cdot \mathbf{f}_k$ and $\mathbf{r}_k^S := A^S \cdot \mathbf{r}_k$ for every set S that contains only honest parties (if such exists).

The first-round communication between an honest P_i and a corrupt P_j consists only of the output of the distribution phase of $\mathcal{F}_{\mathsf{poSig}}$, where P_i is the dealer and P_j is the intermediary. Those outputs correspond to the vectors $(\mathbf{r}_{i,j}^S)$ for every set S that contains both P_i and P_j. We simulate those values as follows. For every set S that contains P_i and P_j,

- The simulator picks a random pad $\mathbf{r}_{i,j}^S \in \mathbb{F}^{uv+2mu}$.

- The simulator simulates the output of $\mathcal{F}_{\mathsf{poSig}}$ corresponding to the set S by giving $\mathbf{r}_{i,j}^S$ to P_j.

We also let the simulator sample a random pad $\mathbf{r}_{i,j}^S \in \mathbb{F}^{uv+2mu}$ for every set S and every two honest parties P_i and P_j in S.

At the end of the round the simulator receives the inputs $\mathbf{r}_{i,j}^S$ of the corrupt parties to $\mathcal{F}_{\mathsf{poSig}}$, for every set S, corrupt P_i in S and any P_j in S.

Simulation of $\mathsf{iSig.ver}$**.** The simulator picks a random non-zero field element $c \in \mathbb{F} \setminus \{0\}$ and broadcasts $(c, \mathbf{f}_k + c\mathbf{r}_k)$. In addition, for every instance of $\mathcal{F}_{\mathsf{poSig}}$ where the dealer is corrupt and the intermediary is honest, the simulator returns $\mathsf{reveal}_{\mathcal{I}} = 0$ as the output of $\mathcal{F}_{\mathsf{poSig}}$ in the verification phase.

In addition, for every set S every honest P_i in S and every P_j in S, the simulator computes $\mathbf{a}_{i,j}^S := (A^S, (\mathbf{f}_k^S, \mathbf{r}_k^S)_{k\in[m]}) + \mathbf{r}_{i,j}^S$ and $\mathbf{b}_{i,j}^S := (A^S, (\mathbf{f}_k^S, \mathbf{r}_k^S)_{k\in[m]}) + \mathbf{r}_{j,i}^S$, and broadcasts $\mathbf{a}_{i,j}^S$ and $\mathbf{b}_{i,j}^S$ on behalf of P_i.

Finally, the simulator receives the following from the adversary.

- The broadcasts $\mathbf{a}_{i,j}^S$ and $\mathbf{b}_{i,j}^S$ for every set S, every corrupt P_i in S, and every P_j in S.
- The input bit $\mathsf{reveal}_{\mathcal{I}}$ for every instances of $\mathcal{F}_{\mathsf{poSig}}$ where the intermediary is corrupt.

Simulation of $\mathsf{iSig.open}_1$**.** The simulator receives the bit $b = 0$, as well as the outputs of the corrupt parties, denoted $(\boldsymbol{\beta}_j^{\mathsf{pub}}, w_j)_{j\in[\ell]}$ and $(\boldsymbol{\beta}_{i,j}^{\mathsf{pri}}, w_{i,j})_{i\in\mathsf{C},j\in[\ell]}$. The simulator continues by broadcasting "accept" on behalf of $\mathcal{D}$. In addition, for every instance of $\mathcal{F}_{\mathsf{poSig}}$ where both the dealer and the intermediary are honest, the simulator returns the bit $b = 0$ as the output of $\mathcal{F}_{\mathsf{poSig}}$ in the opening phase. In addition, for every instance of $\mathcal{F}_{\mathsf{poSig}}$ where the dealer is honest and $\mathcal{I}$ is corrupt, the simulator returns the bit $b = \mathsf{reveal}_{\mathcal{I}}$ as the output of poSig in the opening phase, where $\mathsf{reveal}_{\mathcal{I}}$ was received from the adversary in the simulation of $\mathsf{iSig.ver}$.

The simulator finds any vector $\bar{\mathbf{s}} = (\bar{s}_1, \ldots, \bar{s}_m)$ such that $\boldsymbol{\beta}_j^{\mathsf{pub}} \cdot \bar{\mathbf{s}} = w_j$ and $\boldsymbol{\beta}_{i,j}^{\mathsf{pri}} \cdot \bar{\mathbf{s}} = w_{i,j}$ for every $i \in \mathsf{C}$, $j \in [\ell]$. The simulator picks random vectors $\bar{\mathbf{f}}_1, \ldots, \bar{\mathbf{f}}_m \in \mathbb{F}^v$ conditioned on (1) $A^S \cdot \bar{\mathbf{f}}_k = \mathbf{f}_k^S$ for every $S \in \mathcal{S}$ and $k \in [m]$, and (2) $\bar{\mathbf{f}}_k[1] = \bar{s}_k$ for every $k \in [m]$. The simulator broadcasts $(\boldsymbol{\beta}_j^{\mathsf{pub}}, \sum_{k\in[m]} \boldsymbol{\beta}_j^{\mathsf{pub}}[k] \cdot \bar{\mathbf{f}}_k)_{j\in[\ell]}$ on behalf of $\mathcal{I}$. For every corrupt P_i, the simulator sends $(\boldsymbol{\beta}_{i,j}^{\mathsf{pri}}, \sum_{k\in[m]} \boldsymbol{\beta}_{i,j}^{\mathsf{pri}}[k] \cdot \bar{\mathbf{f}}_k)_{j\in[\ell]}$ to the adversary as the private message from $\mathcal{I}$ to P_i.

Consider any set S and any honest P_i in S. The simulator does as follows for every corrupt P_j.

- The simulator samples a row vector $\mathbf{t}_{i,j}^S \in \mathbb{F}^u$.
- The simulator returns $(\mathbf{t}_{i,j}^S \cdot A^S, \mathbf{t}_{i,j}^S \cdot \mathbf{f}_k^S)$ to the adversary as the output of P_j in the instance of poSig where $\mathcal{D}$ is the dealer and P_i is the intermediary.
- For every honest $P_{i'}$ in S, the simulator returns $(\mathbf{t}_{i,j}^S \cdot \mathbf{r}_{i',i}^S[A^S], \mathbf{t}_{i,j}^S \cdot \mathbf{r}_{i',i}^S[\mathbf{f}_k^S])$ to the adversary as the output of P_j in the instance of $\mathcal{F}_{\mathsf{poSig}}$ where $P_{i'}$ is the dealer and P_i is the intermediary.
- For every corrupt $P_{i'}$ in S, the simulator leaks the vector of coefficients corresponding to $\mathbf{t}_{i,j}^S$ to the adversary in the instance of $\mathcal{F}_{\mathsf{poSig}}$ where $P_{i'}$ is the dealer and P_i is the intermediary.

At the end of the round the simulator receives from the adversary the broadcasts of the corrupt parties, and their inputs to the $\mathcal{F}_{\mathsf{poSig}}$. For every instance of $\mathcal{F}_{\mathsf{poSig}}$ in which the dealer is corrupt or the intermediary is corrupt, the simulator holds all the inputs to $\mathcal{F}_{\mathsf{poSig}}$, and can compute the outputs and return them to the adversary.

Simulation of $\mathsf{iSig.open}_2$**.** The simulator receives from the functionality a list of length μ, denoted $(k_i, s_{k_i})_{i\in[\mu]}$, where $\mu \leq m$, $k_i \in [m]$ and $s_{k_i} \in \mathbb{F}$ for every $i \in [\mu]$. The simulator finds any vector $\tilde{\mathbf{s}} = (\tilde{s}_1, \ldots, \tilde{s}_m)$ such that (1) $\boldsymbol{\beta}_j^{\mathsf{pub}} \cdot \tilde{\mathbf{s}} = w_j$ for every $j \in [\ell]$, (2) $\boldsymbol{\beta}_{i,j}^{\mathsf{pri}} \cdot \tilde{\mathbf{s}} = w_{i,j}$ for every $i \in \mathsf{C}$, $j \in [\ell]$, and (3) $\tilde{s}_{k_i} = s_{k_i}$ for every $i \in [\mu]$.

The simulator samples random vectors $\tilde{\mathbf{f}}_1, \ldots, \tilde{\mathbf{f}}_m \in \mathbb{F}^b$ conditioned on (1) $A^S \cdot \tilde{\mathbf{f}}_k = \mathbf{f}_k^S$ for every $S \in \mathcal{S}$ and every $k \in [m]$, (2) $\sum_{k\in[m]} \boldsymbol{\beta}_j^{\mathsf{pub}}[k] \cdot \tilde{\mathbf{f}}_k = \sum_{k\in[m]} \boldsymbol{\beta}_j^{\mathsf{pub}}[k] \cdot \bar{\mathbf{f}}_k$ for every $j \in [\ell]$, (3) $\sum_{k\in[m]} \boldsymbol{\beta}_{i,j}^{\mathsf{pri}}[k] \cdot \tilde{\mathbf{f}}_k = \sum_{k\in[m]} \boldsymbol{\beta}_{i,j}^{\mathsf{pri}}[k] \cdot \bar{\mathbf{f}}_k$, for every $i \in \mathsf{C}$ and $j \in [\ell]$, and (4) $\tilde{\mathbf{f}}_i[1] = \tilde{s}_i$ for every $i \in [m]$. The simulator broadcasts $(k_i, \tilde{\mathbf{f}}_{k_i})_{i\in[m]}$ on behalf of $\mathcal{I}$. This concludes the simulation.

D.1.2 Analysis

The environment $\mathcal{Z}$ sees during the execution (1) the inputs $(s_1, \ldots, s_m)$ of $\mathcal{D}$, that are picked by the environment, (2) the outputs $(s_1, \ldots, s_m)$ of $\mathcal{I}$ in $\mathsf{poSig.dis}$, (3) the values $(A^S, (\mathbf{f}_k^S, \mathbf{r}_k^S)_{k\in[m]})$ for every set $S \in \mathcal{S}$ that contains a corrupt party, (4) the vector $\mathbf{r}_{i,j}^S$, for every set $S \in \mathcal{S}$, every honest P_i in S, and every corrupt P_j in S, (5) the broadcast $(c, \mathbf{d}_k)_{k\in[m]}$ of $\mathcal{I}$, (6) the broadcasts $\mathbf{a}_{i,j}^S$ and $\mathbf{b}_{i,j}^S$ for every set S, every honest $P_i \in S$ and every $P_j \in S$, (7) the output of $\mathcal{D}$ in the verification phase, (8) the inputs of $\mathcal{I}$ in $\mathsf{iSig.open}_1$ that are picked by the environment, (9) the broadcast of $\mathcal{D}$ in $\mathsf{iSig.open}_1$, (10) the broadcast of $\mathcal{I}$ in $\mathsf{iSig.open}_1$, (11) the output and leakage from the opening phase of all instances of $\mathcal{F}_{\mathsf{poSig}}$, (12) the outputs of the honest parties in $\mathsf{iSig.open}_1$, (13) the inputs of $\mathcal{I}$ in $\mathsf{iSig.open}_2$, that are picked by the environment, (14) the broadcast of $\mathcal{I}$ in $\mathsf{iSig.open}_2$, and (15) the outputs of the honest parties in $\mathsf{iSig.open}_2$.

Distribution phase. First, observe that the inputs $(s_1, \ldots, s_m)$ are picked by $\mathcal{Z}$ in the same way in both worlds, and that in both worlds the output of $\mathcal{I}$ is $(s_1, \ldots, s_m)$. Let A be the $\binom{n-2}{t}u \times v$ matrix consists of all matrices $(A^S)_{S\in\mathcal{S}}$ one on top of the other. In both worlds A is a random $\binom{n-2}{t}u \times v$, and the probability that its rows span the unit vector $(1, 0, \ldots, 0)$ is at most $|\mathbb{F}|^{\binom{n-2}{t}u}/|\mathbb{F}|^v \leq 2^{-\kappa}$. Here we require $v > u\binom{n-2}{t}$ which is guaranteed from our assumption that $v > u \cdot 2^n$. Fix any matrix A whose rows do not span the unit vector $(1, 0, \ldots, 0)$, and observe that this fixes the matrices $(A^S)_{S\in\mathcal{S}}$. Then, by Fact B.4 the vectors $(\mathbf{f}_{A,k} := A \cdot \mathbf{f}_k, \mathbf{r}_{A,k} := A \cdot \mathbf{r}_k)_{k\in[m]}$ have the same distribution in both worlds. Fix those vectors and note that this means that the vectors $(\mathbf{f}_k^S, \mathbf{r}_k^S)_{S\in\mathcal{S},k\in[m]}$ are fixed as well. Finally, it is not hard to see that for every set $\mathcal{S}$, and every honest P_i, the vectors $(\mathbf{r}_{i,j}^S)_{P_j\in S}$ are sampled in the same way in both worlds, and we fix them as well. This concludes the analysis of (1)–(4).

Verification phase. In both worlds the field element c that $\mathcal{I}$ picks is uniformly distributed over $\mathbb{F} \setminus \{0\}$. Fix c. In both worlds, the random variables $\mathbf{r}_1, \ldots, \mathbf{r}_m$ are uniformly distributed conditioned on $A^S \cdot \mathbf{r}_k = \mathbf{r}_k^S$ for every $S \in \mathcal{S}$. Therefore, in both worlds the vectors $\mathbf{d}_1, \ldots, \mathbf{d}_m$ are uniformly distributed conditioned on $A^S \cdot \mathbf{d}_k = \mathbf{f}_k^S + c\mathbf{r}_k^S$ for every $S \in \mathcal{S}$. Fix those vectors, which fixes (5).

Fix any set S and any honest P_i in S. For every P_j in S, it holds that $\mathbf{a}_{i,j}^S := (A^S, (\mathbf{f}_k^S, \mathbf{r}_k^S)_{k\in[m]}) + \mathbf{r}_{i,j}^S$ is fixed, since we've already fixed $A^S, \mathbf{f}_k^S, \mathbf{r}_k^S$ and $\mathbf{r}_{i,j}^S$. Similarly, it holds that $\mathbf{b}_{i,j}^S := (A^S, (\mathbf{f}_k^S, \mathbf{r}_k^S)_{k\in[m]}) + \mathbf{r}_{j,i}^S$ is fixed, since $A^S, \mathbf{f}_k^S, \mathbf{r}_k^S$ were already fixed, and if P_j is honest then $\mathbf{r}_{j,i}^S$ was fixed, and if P_j is corrupt then $\mathbf{r}_{j,i}^S$ was picked by the adversary in the distribution phase,

so it is also fixed. We conclude that (6) has the same distribution in both worlds, and we fix it. Finally, since both $\mathcal{D}$ and $\mathcal{I}$ are honest, in both worlds the output of $\mathcal{D}$ in the verification phase is $\mathsf{reveal}_{\mathcal{I}} = 0$, which fixes (7).

Opening phase 1. In both worlds the environment picks the inputs of $\mathcal{I}$ in the same way. Fix those inputs, which fixes (8). In both worlds, since both $\mathcal{D}$ and $\mathcal{I}$ are honest, $\mathcal{D}$ always broadcasts "accept", which fixes (9).

Consider the subspace spanned by the rows of the matrix A that we defined above, and Let $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_\nu \in \mathbb{F}^v$ be a basis for the subspace. Observe that for every $i \in [\nu]$ and $k \in [m]$ the values $\boldsymbol{\alpha}_i \cdot \mathbf{f}_k$ and $\boldsymbol{\alpha}_i \cdot \bar{\mathbf{f}}_k$ are fixed and equal, and we denote this value by $\gamma_{i,k}$. Recall that A does not span the unit vector $(1, 0, \ldots, 0)$, and let $\boldsymbol{\alpha}_{\nu+1}, \ldots, \boldsymbol{\alpha}_v \in \mathbb{F}^v$ be vectors so that $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_v$ is a basis of $\mathbb{F}^v$, where we assume that $\boldsymbol{\alpha}_v = (1, 0, \ldots, 0)$. Observe that the random variables $(\boldsymbol{\alpha}_i \cdot \mathbf{f}_k, \boldsymbol{\alpha}_i \cdot \bar{\mathbf{f}}_k)_{\nu+1 \le i \le v-1, k \in [m]}$ are uniformly distributed, where $\mathbf{f}_k$ is the real-world random variable, and $\bar{\mathbf{f}}_k$ is the ideal-world random variable defind by the simulator. Also, observe that every vector $\boldsymbol{\alpha} \in \mathbb{F}^v$ is fully determined by the values $\boldsymbol{\alpha}_1 \cdot \boldsymbol{\alpha}, \ldots, \boldsymbol{\alpha}_v \cdot \boldsymbol{\alpha}$.

Consider the real-world broadcasts of $\mathcal{I}$ and the private messages from $\mathcal{I}$ to the corrupt parties, $(\boldsymbol{\beta}_j^{\mathsf{pub}}, \mathsf{Out}_j := \sum_{k \in [m]} \boldsymbol{\beta}_j^{\mathsf{pub}}[k] \cdot \mathbf{f}_k)_{j \in [\ell]}$ and $(\boldsymbol{\beta}_{i,j}^{\mathsf{pri}}, \mathsf{Out}_{i,j} := \sum_{k \in [m]} \boldsymbol{\beta}_{i,j}^{\mathsf{pri}}[k] \cdot \mathbf{f}_k)_{i \in \mathsf{C}, j \in [\ell]}$, and the corresponding ideal-world messages $(\boldsymbol{\beta}_j^{\mathsf{pub}}, \overline{\mathsf{Out}}_j := \sum_{k \in [m]} \boldsymbol{\beta}_j^{\mathsf{pub}}[k] \cdot \bar{\mathbf{f}}_k)_{j \in [\ell]}$ and $(\boldsymbol{\beta}_{i,j}^{\mathsf{pri}}, \overline{\mathsf{Out}}_{i,j} := \sum_{k \in [m]} \boldsymbol{\beta}_{i,j}^{\mathsf{pri}}[k] \cdot \bar{\mathbf{f}}_k)_{i \in \mathsf{C}, j \in [\ell]}$. Consider the random variables

$$(\boldsymbol{\alpha}_k \cdot \mathsf{Out}_j, \boldsymbol{\alpha}_k \cdot \mathsf{Out}_{i,j})_{i \in \mathsf{C}, j \in [\ell], k \in [v]} \quad \text{and} \quad (\boldsymbol{\alpha}_k \cdot \overline{\mathsf{Out}}_j, \boldsymbol{\alpha}_k \cdot \overline{\mathsf{Out}}_{i,j})_{i \in \mathsf{C}, j \in [\ell], k \in [v]},$$

and note that they have the same distribution. Indeed, for every $i \in \mathsf{C}$, $j \in [\ell]$, and $k \in [\nu]$ it holds that

$$\boldsymbol{\alpha}_k \cdot \mathsf{Out}_j = \boldsymbol{\alpha}_k \cdot \sum_{k' \in [m]} \boldsymbol{\beta}_j^{\mathsf{pub}}[k'] \cdot \mathbf{f}_{k'} = \sum_{k' \in [m]} \boldsymbol{\beta}_j^{\mathsf{pub}}[k'] \gamma_{k,k'} = \boldsymbol{\alpha}_k \cdot \sum_{k' \in [m]} \boldsymbol{\beta}_j^{\mathsf{pub}}[k'] \cdot \bar{\mathbf{f}}_{k'} = \boldsymbol{\alpha}_k \cdot \overline{\mathsf{Out}}_j$$
$$\boldsymbol{\alpha}_k \cdot \mathsf{Out}_{i,j} = \boldsymbol{\alpha}_k \cdot \sum_{k' \in [m]} \boldsymbol{\beta}_{i,j}^{\mathsf{pri}}[k'] \cdot \mathbf{f}_{k'} = \sum_{k' \in [m]} \boldsymbol{\beta}_{i,j}^{\mathsf{pri}}[k'] \gamma_{k,k'} = \boldsymbol{\alpha}_k \cdot \sum_{k' \in [m]} \boldsymbol{\beta}_{i,j}^{\mathsf{pri}}[k'] \cdot \bar{\mathbf{f}}_{k'} = \boldsymbol{\alpha}_k \cdot \overline{\mathsf{Out}}_{i,j}.$$

In addition, for every $i \in \mathsf{C}$, $j \in [\ell]$ and $\nu + 1 \le k \le v - 1$ it holds that

$$\boldsymbol{\alpha}_k \cdot \mathsf{Out}_j = \sum_{k' \in [m]} \boldsymbol{\beta}_j^{\mathsf{pub}}[k'] \cdot (\boldsymbol{\alpha}_k \cdot \mathbf{f}_{k'}) \quad \text{and} \quad \boldsymbol{\alpha}_k \cdot \overline{\mathsf{Out}}_j = \sum_{k' \in [m]} \boldsymbol{\beta}_j^{\mathsf{pub}}[k'] \cdot (\boldsymbol{\alpha}_k \cdot \bar{\mathbf{f}}_{k'})$$
$$\boldsymbol{\alpha}_k \cdot \mathsf{Out}_{i,j} = \sum_{k' \in [m]} \boldsymbol{\beta}_{i,j}^{\mathsf{pri}}[k'] \cdot (\boldsymbol{\alpha}_k \cdot \mathbf{f}_{k'}) \quad \text{and} \quad \boldsymbol{\alpha}_k \cdot \overline{\mathsf{Out}}_{i,j} = \sum_{k' \in [m]} \boldsymbol{\beta}_{i,j}^{\mathsf{pri}}[k'] \cdot (\boldsymbol{\alpha}_k \cdot \bar{\mathbf{f}}_{k'})$$

Since the random variables $(\boldsymbol{\alpha}_i \cdot \mathbf{f}_k, \boldsymbol{\alpha}_i \cdot \bar{\mathbf{f}}_k)_{\nu+1 \le i \le v-1, k \in [m]}$ are uniformly distributed, we conclude that $(\boldsymbol{\alpha}_k \cdot \mathsf{Out}_j, \boldsymbol{\alpha}_k \cdot \mathsf{Out}_{i,j})_{i \in \mathsf{C}, j \in [\ell], \nu+1 \le k \le v-1}$ have the same distribution as $(\boldsymbol{\alpha}_k \cdot \overline{\mathsf{Out}}_j, \boldsymbol{\alpha}_k \cdot \overline{\mathsf{Out}}_{i,j})_{i \in \mathsf{C}, j \in [\ell], \nu+1 \le k \le v-1}$. In addition, by definition of $(\bar{\mathbf{f}}_k)_{k \in [m]}$ it holds that

$$\boldsymbol{\alpha}_v \cdot \overline{\mathsf{Out}}_j = \overline{\mathsf{Out}}_j[1] = \sum_{k \in [m]} \boldsymbol{\beta}_j[k] \cdot \bar{\mathbf{f}}_k[1] = \boldsymbol{\beta}_j^{\mathsf{pub}} \cdot \bar{\mathbf{s}} = w_j = \mathsf{Out}_j[1] = \boldsymbol{\alpha}_v \cdot \mathsf{Out}_j,$$
$$\boldsymbol{\alpha}_v \cdot \overline{\mathsf{Out}}_{i,j} = \overline{\mathsf{Out}}_{i,j}[1] = \sum_{k \in [m]} \boldsymbol{\beta}_{i,j}[k] \cdot \bar{\mathbf{f}}_k[1] = \boldsymbol{\beta}_{i,j}^{\mathsf{pub}} \cdot \bar{\mathbf{s}} = w_{i,j} = \mathsf{Out}_{i,j}[1] = \boldsymbol{\alpha}_v \cdot \mathsf{Out}_{i,j},$$

for every $i \in \mathsf{C}$ and $j \in [\ell]$. Since every vector $\boldsymbol{\alpha} \in \mathbb{F}^v$ is fully determined by the values $\boldsymbol{\alpha}_1 \cdot \boldsymbol{\alpha}, \ldots, \boldsymbol{\alpha}_v \cdot \boldsymbol{\alpha}$, we conclude that the real-world random variables $(\boldsymbol{\beta}^{\mathsf{pub}}_j, \mathsf{Out}_j)_{j\in[\ell]}$ and $(\boldsymbol{\beta}^{\mathsf{pri}}_{i,j}, \mathsf{Out}_{i,j})_{i\in\mathsf{C},j\in[\ell]}$, and the corresponding ideal-world messages $(\boldsymbol{\beta}^{\mathsf{pub}}_j, \overline{\mathsf{Out}}_j)_{j\in[\ell]}$ and $(\boldsymbol{\beta}^{\mathsf{pri}}_{i,j}, \overline{\mathsf{Out}}_{i,j})_{i\in\mathsf{C},j\in[\ell]}$ have the same distribution. We conclude that (10) has the same distribution in both worlds, and we fix it.

We continue by analysing the outputs of $\mathcal{F}_{\mathsf{poSig}}$. Consider any set $S \in \mathcal{S}$ that contains an honest party, any honest P_i in S, and any corrupt P_j (not necessarily in S).

- First, observe that in both worlds the vector $\mathbf{t}^S_{i,j} \in \mathbb{F}^u$ is uniformly distributed, and we fix this vector.
- Consider the instance of $\mathcal{F}_{\mathsf{poSig}}$ in which $\mathcal{D}$ is the dealer and P_i is the intermediary. Then in both worlds corresponding output is fixed to be $(\mathbf{t}^S_{i,j} \cdot A^S, \mathbf{t}^S_{i,j} \cdot \mathbf{f}^S_k)$.
- For an honest $P_{i'}$ in S, consider the instance of $\mathcal{F}_{\mathsf{poSig}}$ in which $P_{i'}$ is the dealer and P_i is the intermediary. Then the value $\mathbf{r}^S_{i',i}$ is fixed to be $\mathbf{a}^S_{i',i} - (A^S, (\mathbf{f}^S_k, \mathbf{r}^S_k)_{k\in[m]})$. Therefore, in both worlds the corresponding output is fixed to be $(\mathbf{t}^S_{i',i} \cdot \mathbf{r}^S_{i',i}[A^S], \mathbf{t}^S_{i',i} \cdot \mathbf{r}^S_{i',i}[\mathbf{f}^S_k])_{k\in[m]}$.
- For a corrupt $P_{i'}$ in S, consider the instance of $\mathcal{F}_{\mathsf{poSig}}$ in which $P_{i'}$ is the dealer and P_i is the intermediary. Then the leakage is fixed and computed in the same way in both worlds, as required.

At this point the corrupt parties send their inputs to the functionality $\mathcal{F}_{\mathsf{poSig}}$, and the outputs are computed in the same way in both worlds. This concludes the analysis of (11).

Outputs of honest parties in opening phase 1. In the ideal world the output of an honest P_i is $(\boldsymbol{\beta}^{\mathsf{pub}}_j, \boldsymbol{\beta}^{\mathsf{pub}}_j \cdot \mathbf{s})_{j\in[\ell]}$ and $(\boldsymbol{\beta}^{\mathsf{pri}}_{i,j}, \boldsymbol{\beta}^{\mathsf{pri}}_{i,j} \cdot \mathbf{s})_{j\in[\ell]}$ with probability 1. We show that this is also the case in the real world.

Consider any honest P_i and any $S \in \mathcal{G}$ (if $\mathcal{G}$ is empty then P_i accepts the opening with probability 1). We show that P_i does not reject the opening because of S. Fix any $P_k \in G^{S,i}$ (if $G^{S,i}$ is empty then P_i does not reject the opening because of S). Since $\mathcal{D}$ is honest, and since $P_k \in G^{S,i}$, it must hold that there exists some $\mathbf{t}^S_{k,i} \in \mathbb{F}^u$ so that $\boldsymbol{\alpha}^{S,i}_k = \mathbf{t}^S_{k,i} \cdot A^S$ and $\phi^{S,i}_{k,k'} = \mathbf{t}^S_{k,i} \cdot \mathbf{f}^S_{k'}$ for every $k' \in [m]$. In particular, for every $j \in [\ell]$ it holds that

$$\begin{aligned}
\boldsymbol{\alpha}^{S,i}_k \sum_{k'\in[m]} \boldsymbol{\beta}^{\mathsf{pub}}_j[k'] \cdot \mathbf{f}_{k'} &= \sum_{k'\in[m]} \boldsymbol{\beta}^{\mathsf{pub}}_j[k'] \cdot \mathbf{t}^S_{k,i} \cdot A^S \cdot \mathbf{f}_{k'} \\
&= \sum_{k'\in[m]} \boldsymbol{\beta}^{\mathsf{pub}}_j[k'] \cdot \mathbf{t}^S_{k,i} \cdot \mathbf{f}^S_{k'} \\
&= \sum_{k'\in[m]} \boldsymbol{\beta}^{\mathsf{pub}}_j[k'] \phi^{S,i}_{k,k'},
\end{aligned}$$

and

$$\begin{aligned}\boldsymbol{\alpha}_k^{S,i} \sum_{k' \in [m]} \boldsymbol{\beta}_{i,j}^{\mathsf{pri}}[k'] \cdot \mathbf{f}_{k'} &= \sum_{k' \in [m]} \boldsymbol{\beta}_{i,j}^{\mathsf{pri}}[k'] \cdot \mathbf{t}_{k,i}^{S} \cdot A^S \cdot \mathbf{f}_{k'} \\ &= \sum_{k' \in [m]} \boldsymbol{\beta}_{i,j}^{\mathsf{pri}}[k'] \cdot \mathbf{t}_{k,i}^{S} \cdot \mathbf{f}_{k'}^{S} \\ &= \sum_{k' \in [m]} \boldsymbol{\beta}_{i,j}^{\mathsf{pri}}[k'] \phi_{k,k'}^{S,i}.\end{aligned}$$

Therefore P_i does not reject the opening because of S. Since this holds for every $S \in \mathcal{G}$, the output of P_i in the real world is $(\boldsymbol{\beta}_j^{\mathsf{pub}}, \boldsymbol{\beta}_j^{\mathsf{pub}} \cdot \mathbf{s})_{j \in [\ell]}$ and $(\boldsymbol{\beta}_{i,j}^{\mathsf{pri}}, \boldsymbol{\beta}_{i,j}^{\mathsf{pri}} \cdot \mathbf{s})_{j \in [\ell]}$ with probability 1, as required. This concludes the analysis of (12).

Opening phase 2. The inputs of $\mathcal{I}$ are picked by $\mathcal{Z}$ in the same way in both worlds, and we fix them, which fixes (13). Let the inputs be $(k_i)_{i \in [\mu]}$. We show that the real-world random variables $(\mathbf{f}_{k_i})_{i \in [\mu]}$ have the same distribution as the ideal-world random variables $(\tilde{\mathbf{f}}_{k_i})_{i \in [\mu]}$. Observe that the real-world random variables $(\mathbf{f}_k)_{k \in [m]}$ are uniformly distributed conditioned on (a) $\mathbf{f}_k[1] = s_k$ for every $k \in [m]$, (b) $A^S \cdot \mathbf{f}_{k_i} = \mathbf{f}_k^S$ for every $S \in \mathcal{S}$ and $k \in [m]$, (c) $\sum_{k' \in [m]} \boldsymbol{\beta}_j^{\mathsf{pub}}[k'] \cdot \mathbf{f}_{k'} = \mathsf{Out}_j$ for every $j \in [\ell]$, and (d) $\sum_{k' \in [m]} \boldsymbol{\beta}_{i,j}^{\mathsf{pri}}[k'] \cdot \mathbf{f}_{k'} = \mathsf{Out}_{i,j}$ for every $i \in \mathsf{C}$ and $j \in [\ell]$. Similarly, the ideal-world random variables $(\tilde{\mathbf{f}}_k)_{k \in [m]}$ are uniformly distributed conditioned on (a) $\tilde{\mathbf{f}}_k[1] = \tilde{s}_k$ for every $k \in [m]$, (b) $A^S \cdot \tilde{\mathbf{f}}_{k_i} = \mathbf{f}_k^S$ for every $k \in [m]$, (c) $\sum_{k' \in [m]} \boldsymbol{\beta}_j^{\mathsf{pub}}[k'] \cdot \tilde{\mathbf{f}}_{k'} = \mathsf{Out}_j$ for every $j \in [\ell]$, and (d) $\sum_{k' \in [m]} \boldsymbol{\beta}_{i,j}^{\mathsf{pri}}[k'] \cdot \tilde{\mathbf{f}}_{k'} = \mathsf{Out}_{i,j}$ for every $i \in \mathsf{C}$ and $j \in [\ell]$.

As before, it is not hard to see that $\boldsymbol{\alpha}_j \cdot \mathbf{f}_{k_i} = \boldsymbol{\alpha}_j \cdot \tilde{\mathbf{f}}_{k_i}$, for every $j \in [\nu]$ and $i \in [\mu]$. Similarly, since $\boldsymbol{\alpha}_v$ is the unit vector $(1, 0, \ldots, 0)$ it holds that $\boldsymbol{\alpha}_v \cdot \mathbf{f}_{k_i} = \boldsymbol{\alpha}_v \cdot \tilde{\mathbf{f}}_{k_i}$, for every $i \in [\mu]$.

For every $\nu + 1 \leq j \leq v - 1$, observe that the random variables $(\boldsymbol{\alpha}_j \cdot \mathbf{f}_k)_{k \in [m]}$ are independent of $(\boldsymbol{\alpha}_j \cdot \mathbf{f}_k)_{k \in [m], j' \neq j}$. Fix any $\nu + 1 \leq j \leq v - 1$, and observe that $(\boldsymbol{\alpha}_j \cdot \mathbf{f}_k)_{k \in [m]}$ are uniformly distributed conditioned on (a) $\sum_{k' \in [m]} \boldsymbol{\beta}_{j'}^{\mathsf{pub}}[k'] \cdot \boldsymbol{\alpha}_j \cdot \mathbf{f}_{k'} = \boldsymbol{\alpha}_j \cdot \mathsf{Out}_{j'}$ for every $j' \in [\ell]$, and (b) $\sum_{k' \in [m]} \boldsymbol{\beta}_{i,j'}^{\mathsf{pri}}[k'] \cdot \mathbf{f}_{k'} = \mathsf{Out}_{i,j'}$ for every $i \in \mathsf{C}$ and $j' \in [\ell]$. Since the same argument holds with respect to $(\boldsymbol{\alpha}_j \cdot \tilde{\mathbf{f}}_k)_{k \in [m]}$, we conclude that $(\boldsymbol{\alpha}_j \cdot \mathbf{f}_{k_i})_{i \in [\mu], \nu+1 \leq j \leq v-1}$ has the same distribution as $(\boldsymbol{\alpha}_j \cdot \tilde{\mathbf{f}}_{k_i})_{i \in [\mu], \nu+1 \leq j \leq v-1}$. Since every vector $\boldsymbol{\alpha} \in \mathbb{F}^v$ is fully determined by the values $\boldsymbol{\alpha}_1 \cdot \boldsymbol{\alpha}, \ldots, \boldsymbol{\alpha}_v \cdot \boldsymbol{\alpha}$, we conclude that the real-world random variables $(\mathbf{f}_{k_i})_{i \in [\mu]}$ have the same distribution as the ideal-world random variables $(\tilde{\mathbf{f}}_{k_i})_{i \in [\mu]}$. This completes the analysis of (14).

Outputs of honest parties in opening phase 2. In the ideal world the output of an honest P_i is $(k_i, s_{k_i})_{i \in [\mu]}$ with probability 1. By the same analysis as in the outputs of honest parties in opening phase 1, we conclude that this is also true for the real world. This completes the analysis of an honest $\mathcal{D}$ and $\mathcal{I}$.

D.2 Honest $\mathcal{D}$, Corrupt $\mathcal{I}$

D.2.1 The Simulator

Simulation of iSig.dis**.** At the beginning of iSig.dis, the simulator receives $(s_1, \ldots, s_m)$ from the ideal functionality. The simulator, that holds all the inputs of $\mathcal{D}$, simply initiate an execution of

$\mathsf{iSig.dis}$ by taking the roles of the honest parties, where the inputs of $\mathcal{D}$ are $(s_1, \ldots, s_m)$.

Simulation of $\mathsf{iSig.ver}$. The simulator continues with the execution of $\mathsf{iSig.ver}$, first by sending the messages of the honest parties, and then by receiving from the adversary the broadcast of the corrupt $\mathcal{I}$, and sending it to all the simulated honest parties. The simulator computes the output $\mathsf{reveal}_{\mathcal{I}}$ of $\mathcal{D}$, and sends it to the ideal functionality.

Simulation of $\mathsf{iSig.open}_1$. The simulator continues with the execution of $\mathsf{iSig.open}_1$, and computes the outputs of the honest parties. If $\mathsf{reveal}_{\mathcal{I}} = 1$ then the inputs of $\mathcal{I}$ to $\mathcal{F}_{\mathsf{iSig}}$ do not matter, and the simulator terminates. Otherwise, fix any honest party P_i. If P_i outputs $\perp$ as the public output, then the simulator inputs $\mathsf{abort}^{\mathsf{pub}} = 1$ to $\mathcal{F}_{\mathsf{iSig}}$ and terminates (the rest of the inputs do not matter). Otherwise, the public output of P_i is $(\boldsymbol{\beta}_i^{\mathsf{pub}}, w_i)_{j\in[\ell]}$, and the simulator inputs $(\boldsymbol{\beta}_j^{\mathsf{pub}})_{j\in[\ell]}$ to the ideal functionality $\mathcal{F}_{\mathsf{iSig}}$.

In addition, for every honest P_i the simulator does as follows. If P_i outputs $\perp$ as the private output, then the simulator inputs $\mathsf{abort}_i^{\mathsf{pri}} = 1$ to $\mathcal{F}_{\mathsf{iSig}}$. Otherwise, the private output of P_i is $(\boldsymbol{\beta}_{i,j}^{\mathsf{pri}}, w_{i,j})_{j\in[\ell]}$, and the simulator inputs $(\boldsymbol{\beta}_{i,j}^{\mathsf{pri}})_{j\in[\ell]}$ to the ideal functionality $\mathcal{F}_{\mathsf{iSig}}$.

Simulation of $\mathsf{iSig.open}_2$. The simulator continues with the execution of $\mathsf{iSig.open}_2$, and computes the outputs of the honest parties. Let P_i be any honest party. If P_i outputs $\perp$ as the public output, then the simulator inputs $\mathsf{abort} = 1$ to $\mathcal{F}_{\mathsf{iSig}}$, and terminates (the rest of the inputs do not matter). Otherwise, the output of P_i is $(k_i, s_{k_i})_{i\in[\mu]}$, and the simulator inputs μ and $(k_i)_{i\in[\mu]}$ to the ideal functionality $\mathcal{F}_{\mathsf{iSig}}$. This concludes the simulation.

D.2.2 Analysis

Observe that there is a 1-1 correspondence between real-world executions and the simulated executions performed by the simulator up to the end of $\mathsf{iSig.open}_1$. It is not hard to see that $\mathsf{iSig.dis}$, $\mathsf{iSig.ver}$, the outputs of the honest parties in $\mathsf{iSig.ver}$ and $\mathsf{iSig.open}_1$ are perfectly simulated. We continue by considering the outputs of the honest parties in $\mathsf{iSig.open}_1$.

Output of honest parties in $\mathsf{iSig.open}_1$. Since $\mathcal{I}$ is corrupt, there exists some set $S^* \in \mathcal{S}$ that contains only honest parties. It is not hard to see that if $\mathcal{D}$ broadcasts "accept" then $S^* \in \mathcal{G}$, and that for every honest P_i it holds that $G^{S^*,i} = S^*$. We say that an execution is "bad" in $\mathsf{iSig.open}_1$ if $\mathcal{D}$ broadcasts "accept" in $\mathsf{iSig.open}_1$ and there exists an honest P_i and some $j \in [\ell]$ so that one of the following holds.

- $\mathcal{I}$ broadcasted $(\boldsymbol{\beta}_j^{\mathsf{pub}}, \mathsf{Out}_j)$ as the j-th public output so that (1) it holds that $\sum_{k\in[m]} \boldsymbol{\beta}_j^{\mathsf{pub}}[k] \cdot \mathbf{f}_k \neq \mathsf{Out}_j$, and (2) for every $P_{i'}$ in S^* it holds that $\boldsymbol{\alpha}_{i'}^{S^*,i} \cdot \mathsf{Out}_j = \sum_{k\in[m]} \boldsymbol{\beta}_j^{\mathsf{pub}}[k] \cdot \phi_{i',k}^{S,i}$.
- $\mathcal{I}$ sent $(\boldsymbol{\beta}_{i,j}^{\mathsf{pri}}, \mathsf{Out}_{i,j})$ as the j-th private output of P_i so that (1) it holds that $\sum_{k\in[m]} \boldsymbol{\beta}_{i,j}^{\mathsf{pri}}[k] \cdot \mathbf{f}_k \neq \mathsf{Out}_{i,j}$, and (2) for every $P_{i'}$ in S^* it holds that $\boldsymbol{\alpha}_{i'}^{S^*,i} \cdot \mathsf{Out}_{i,j} = \sum_{k\in[m]} \boldsymbol{\beta}_j^{\mathsf{pri}}[k] \cdot \phi_{i',k}^{S,i}$.

We continue by showing that an execution is bad in $\mathsf{iSig.open}_1$ with probability at most $2^{-\kappa}$.

Consider the $tn \times u$ matrix T^{S^*}, that consists of all the rows $(\mathbf{t}_{i,j}^{S^*})_{P_i\in S^*, j\in[n]}$. Observe that T^* has full rank with probability at least $1-|\mathbb{F}|^{tn}/|\mathbb{F}|^u \geq 1-2^{-\kappa}$. Here we require $u > tn$ which is what we

assume in our protocol. Fix any such matrix T^{S^*}, and note that this fixes the rows $(\mathbf{t}^{S^*}_{i,j})_{P_i \in S^*, j \in [n]}$. Consider the $tn \times v$ matrix $T^{S^*} \cdot A^{S^*}$ and observe that since T^{S^*} has full rank, and since A^{S^*} is uniformly distributed, then $T^{S^*} \cdot A^{S^*}$ is uniformly distributed. This means that even conditioned on the vectors $(\boldsymbol{\alpha}^{S^*,j}_{i'})_{P_{i'} \in S^*, j \in \mathsf{C}}$, the vectors $(\boldsymbol{\alpha}^{S^*,j}_{i'})_{P_{i'} \in S^*, j \in \mathsf{H}}$ are uniformly distributed.

Fix any P_i, any $j \in [\ell]$. Then for every $P_{i'} \in S^*$ the event $\boldsymbol{\alpha}^{S^*,i}_{i'} \cdot \mathsf{Out}_j = \sum_{k \in [m]} \boldsymbol{\beta}^{\mathsf{pub}}_j[k] \cdot \phi^{S,i}_{i',k}$ occurs if and only if $\boldsymbol{\alpha}^{S^*,i}_{i'} \cdot \mathsf{Out}_j = \sum_{k \in [m]} \boldsymbol{\beta}^{\mathsf{pub}}_j[k] \cdot \mathbf{t}^{S^*}_{i',i} A^{S^*} \mathbf{f}_k$, since $\phi^{S,i}_{i',k} = \mathbf{t}^{S^*}_{i',i} \cdot \mathbf{f}^S_k = \mathbf{t}^{S^*}_{i',i} A^{S^*} \mathbf{f}_k$; this event occurs if and only if $\boldsymbol{\alpha}^{S^*,i}_{i'} \cdot \mathsf{Out}_j = \boldsymbol{\alpha}^{S^*,i}_{i'} \cdot (\sum_{k \in [m]} \boldsymbol{\beta}^{\mathsf{pub}}_j[k] \cdot \mathbf{f}_k)$ since $\boldsymbol{\alpha}^{S^*,i}_{i'} = \mathbf{t}^{S^*}_{i',i} A^{S^*}$; therefore this event occurs if and only if $\boldsymbol{\alpha}^{S^*,i}_{i'} \cdot (\mathsf{Out}_j - \sum_{k \in [m]} \boldsymbol{\beta}^{\mathsf{pub}}_j[k] \cdot \mathbf{f}_k) = 0$, which occurs with probability $1/|\mathbb{F}| \leq 2^{-\kappa}/(2\ell n)$ if $\mathsf{Out}_j \neq \sum_{k \in [m]} \boldsymbol{\beta}^{\mathsf{pub}}_j[k] \cdot \mathbf{f}_k$ since the vectors the vectors $(\boldsymbol{\alpha}^{S^*,j}_{i'})_{P_{i'} \in S^*, j \in \mathsf{H}}$ are uniformly distributed even conditioned on the view of $\mathcal{Z}$. Since the same argument holds for the j-th private output, and by taking union bound over all honest P_i and $j \in [\ell]$, we conclude that the execution is bad in $\mathsf{iSig.open}_1$ with probability at most $2^{-\kappa}$. Note that here we require $|\mathbb{F}| \geq 2^\kappa \cdot 2\ell n$.

Consider any good execution of the protocol. If $\mathcal{D}$ did not broadcast "accept" in $\mathsf{iSig.open}_1$ then we're done. Otherwise, if $\mathsf{Out}_j \neq \sum_{k \in [m]} \boldsymbol{\beta}^{\mathsf{pub}}_j \cdot \mathbf{f}_k$ for some $j \in [\ell]$ then since the execution is good, all the parties output $\perp$ in both worlds. Otherwise $\mathsf{Out}_j = \sum_{k \in [m]} \boldsymbol{\beta}^{\mathsf{pub}}_j \cdot \mathbf{f}_k$ for every $j \in [\ell]$. In this case, since $\mathcal{D}$ is honest, for every P_i, every $S \in \mathcal{G}$ and every $P_{i'} \in \mathcal{G}^S_i$ it holds that $\boldsymbol{\alpha}^{S,i}_{i'} = \mathbf{t}^S_{i',i} A^S$ and $\phi^{S,i}_{i',k} = \mathbf{t}^S_{i',i} \cdot \mathbf{f}^S_k$ for some vector $\mathbf{t}^S_{i',i} \in \mathbb{F}^u$. Therefore in the real-world P_i does not reject the public output on behalf of $P_{i'}$. Since this is true for every P_i, every $S \in \mathcal{G}$ and every $P_{i'} \in G^{S,i}$, we conclude that in the real-world all honest parties output $(\boldsymbol{\beta}^{\mathsf{pub}}_j, \sum_{k \in [m]} \boldsymbol{\beta}^{\mathsf{pub}}_j \cdot \mathbf{f}_k)_{j \in [\ell]}$ as the public output. Therefore, in the ideal world the simulator inputs $(\boldsymbol{\beta}^{\mathsf{pub}}_j)_{j \in [\ell]}$ to the ideal functionality, and all the parties output $(\boldsymbol{\beta}^{\mathsf{pub}}_j, \sum_{k \in [m]} \boldsymbol{\beta}^{\mathsf{pub}}_j \cdot \mathbf{f}_k)_{j \in [\ell]}$ as the public output, as required.

Consider now the private output of an honest P_i. Then the same analysis shows that if the execution is good then the private output of P_i is the same in both worlds. This concludes the analysis of the outputs of the honest parties in $\mathsf{iSig.open}_1$.

Execution of $\mathsf{iSig.open}_2$. The only party that communicates in this round is the corrupt $\mathcal{I}$, so communication is done in the same way in both worlds. It remains to analyse the outputs of the honest parties in this round. We say that an execution is "bad" in $\mathsf{iSig.open}_2$ if $\mathcal{D}$ broadcasts "accept" in $\mathsf{iSig.open}_1$ and there exists some $j \in [\mu]$ so that $\mathcal{I}$ broadcasted $(k_j, \mathbf{g}_{k_j})$ in $\mathsf{iSig.open}_2$, and it holds that $\mathbf{f}_{k_j} \neq \mathbf{g}_{k_j}$, and there exists some honest P_i so that for every $P_{i'}$ in S^* it holds that $\boldsymbol{\alpha}^{S^*,i}_{i'} \cdot \mathbf{g}_{k_j} = \phi^{S,i}_{i',k_j}$.

By the above analysis, the random variables $(\boldsymbol{\alpha}^{S^*,j}_{i'})_{P_{i'} \in S^*, j \in \mathsf{H}}$ are $2^{-\kappa}$-close to uniform in statistical distance even conditioned on the view of $\mathcal{Z}$ and on the execution being good in $\mathsf{iSig.open}_1$. Therefore, conditioned on the execution being good in $\mathsf{iSig.open}_1$, the same analysis as before shows that an execution is bad in $\mathsf{iSig.open}_2$ with probability at most $mn/|\mathbb{F}| + 2^{-\kappa} \leq 2 \cdot 2^{-\kappa}$, as required. Note that here we require $|\mathbb{F}| \geq 2^\kappa mn$. Conditioned on the execution being good in $\mathsf{iSig.open}_2$ as well, it is not hard to see that the outputs are the same in both worlds. This concludes the analysis of honest $\mathcal{D}$ and corrupt $\mathcal{I}$.

D.3 Corrupt $\mathcal{D}$, Honest $\mathcal{I}$

D.3.1 The Simulator

Simulation of iSig.dis**.** The simulator initiate an execution of iSig.dis by taking the roles of the honest parties. Let $(\mathbf{f}_1, \ldots, \mathbf{f}_m)$ be the polynomials that the corrupt $\mathcal{D}$ sent to $\mathcal{I}$. The simulator sets $s_i := \mathbf{f}_i[1]$ for all $i \in [m]$, and inputs $(s_1, \ldots, s_m)$.

Simulation of iSig.ver**.** The simulator continues with the execution of iSig.ver.

Simulation of $\mathsf{iSig.open}_1$**.** The simulator receives $(\boldsymbol{\beta}_i^{\mathsf{pub}})_{j\in[\ell]}$ $(\boldsymbol{\beta}_{i,j}^{\mathsf{pri}})_{i\in\mathsf{C},j\in[\ell]}$ from the ideal functionality. The simulator continues by executing the honest parties that send the verification information to each other. In addition, the simulator computes the opening of $\mathcal{I}$ as follows.

- $\mathcal{I}$ broadcasts $(\boldsymbol{\beta}_j^{\mathsf{pub}}, \mathsf{Out}_j)_{j\in[\ell]}$, where $\mathsf{Out}_j = \boldsymbol{\beta}_j^{\mathsf{pub}}[1] \cdot \mathbf{f}_1 + \ldots + \boldsymbol{\beta}_j^{\mathsf{pub}}[m] \cdot \mathbf{f}_m$.
- For every corrutp P_i, $\mathcal{I}$ sends $(\boldsymbol{\beta}_{i,j}^{\mathsf{pri}}, \mathsf{Out}_{i,j})_{j\in[\ell]}$, where $\mathsf{Out}_{i,j} = \boldsymbol{\beta}_{i,j}^{\mathsf{pri}}[1] \cdot \mathbf{f}_1 + \ldots + \boldsymbol{\beta}_{i,j}^{\mathsf{pri}}[m] \cdot \mathbf{f}_m$.

If $\mathcal{D}$ broadcasts $\mathbf{s}'$ in poSig.open, then the simulator inputs $\mathsf{reveal}_{\mathcal{D}} = 1$ and $\mathbf{s}'$ to the ideal functionality and terminates. Otherwise, the simulator inputs $\mathsf{reveal}_{\mathcal{D}} = 0$ and $\mathbf{s}$ to the ideal functionality.

Simulation of $\mathsf{iSig.open}_2$**.** The simulator receives $(k_i, s_{k_i})_{i\in[\mu]}$ from the ideal functionality. The simulator broadcasts $(\mathbf{f}_{k_i})_{i\in[\mu]}$ on behalf of $\mathcal{I}$. This concludes the simulation.

D.3.2 Analysis

Observe that there is a 1-1 correspondence between real-world executions and the simulated executions performed by the simulator up to the end of $\mathsf{iSig.open}_1$. It is not hard to see that iSig.dis, iSig.ver, the outputs of the honest parties in iSig.ver and $\mathsf{iSig.open}_1$ are perfectly simulated. We continue by considering the outputs of the honest parties in $\mathsf{iSig.open}_1$.

Outputs of honest parties in $\mathsf{iSig.open}_1$**.** Since $\mathcal{D}$ is corrupt, every set $S \in \mathcal{S}$ contains at least on honest party. For every set $S \in \mathcal{S}$ and every honest P_i in S, denote the matrix that P_i from $\mathcal{D}$ by $A^{S,i}$, and the corresponding vectors by $(\mathbf{f}_k^{S,i}, \mathbf{r}_k^{S,i})_{k\in[m]}$. We say that an execution is "bad" if there exists a set $S \in \mathcal{S}$, an honest P_i in S, and some $k \in [m]$, so that (1) $A^{S,i} \cdot \mathbf{f}_k \neq \mathbf{f}_{S,k}^i$, and (2) P_i did not broadcast "not equal" in $\mathsf{iSig.open}_1$. Observe that P_i broadcasts "not equal" if $A^{S,i} \cdot \mathbf{d}_k \neq \mathbf{f}_k^{S,i} + c\mathbf{r}_k^{S,i}$ for some $S \in \mathcal{S}$ that contains P_i, and some $k \in [m]$. Since $\mathbf{d}_k = \mathbf{f}_k + c\mathbf{r}_k$ then P_i broadcasts "not equal" if $A^{S,i} \cdot (\mathbf{f}_k + c\mathbf{r}_k) \neq \mathbf{f}_k^{S,i} + c\mathbf{r}_k^{S,i}$. Therefore, P_i broadcasts "not equal" if

$$A^{S,i} \cdot \mathbf{f}_k - \mathbf{f}_k^{S,i} \neq c(\mathbf{r}_k^{S,i} - \mathbf{r}_k).$$

Since c is uniformly distributed over $\mathbb{F} \setminus \{0\}$ we conclude that if $A^{S,i} \cdot \mathbf{f}_k \neq \mathbf{f}_k^{S,i}$ then P_i does not broadcasts "not equal" with probability at most $1/(|\mathbb{F}| - 1) \leq 2^{-\kappa}/(nm\binom{n-2}{t})$. By taking a union bound over all $S \in \mathcal{S}$, honest P_i and $k \in [m]$, we conclude that an execution is "bad" with probability at most $2^{-\kappa}$. Note that here we require $|\mathbb{F}| > 2^{\kappa} nm\binom{n-2}{t}$.

Consider any good execution of the protocol. If $b = 1$ then the output in both worlds is $\mathbf{s}'$, as required. Otherwise, $b = 0$, and in the ideal world the output of any honest P_i is $(\boldsymbol{\beta}_j^{\mathsf{pub}} \cdot \mathbf{s})_{j\in[\ell]}$ and $(\boldsymbol{\beta}_{i,j}^{\mathsf{pri}} \cdot \mathbf{s})_{j\in[\ell]}$. We show that in the real-world every honest P_i accepts the openings of $\mathcal{I}$, so the outputs are the same in both worlds. Fix any honest P_i, and every set $S \in \mathcal{G}$. We show that P_i does not reject due to S. Recall that S contains at least one honest party, denoted P_{i^*}. Observe that all honest parties in S are in $G^{S,i}$, that they agree on the values of $A^S, (\mathbf{f}_k^S, \mathbf{r}_k^S)_{k\in[m]}$, and since the view is good it holds that $A^S \cdot \mathbf{f}_k = \mathbf{f}_k^S$ for every $k \in [m]$. Therefore, for every honest party $P_{i'}$ in S it holds that $\boldsymbol{\alpha}_{i'}^{S,i} \cdot \sum_{k\in[m]} \boldsymbol{\beta}_j^{\mathsf{pub}}[k] \cdot \mathbf{f}_k = \sum_{k\in[m]} \boldsymbol{\beta}_j^{\mathsf{pub}}[k] \cdot \phi_{i',k}^{S,i}$, and $\boldsymbol{\alpha}_{i'}^{S,i} \cdot \sum_{k\in[m]} \boldsymbol{\beta}_{i,j}^{\mathsf{pri}}[k] \cdot \mathbf{f}_k = \sum_{k\in[m]} \boldsymbol{\beta}_{i,j}^{\mathsf{pri}}[k] \cdot \phi_{i',k}^{S,i}$ for every $j \in [\ell]$.

Consider any corrupt $P_{i'}$ in $G^{S,i}$ and let $\mathbf{t}_{i',i}^S$ be the vector that P_i received from $P_{i'}$. We use the following claim.

Claim D.1. *It holds that $\boldsymbol{\alpha}_{i'}^{S,i} = \mathbf{t}_{i',i}^S \cdot A^S$, and $\phi_{i',k}^{S,i} = \mathbf{t}_{i',i}^S \cdot \mathbf{f}_k^S$ for all $k \in [m]$.*

Proof. Since $S \in \mathcal{G}$ then $\mathbf{b}_{i',i^*}^S = \mathbf{a}_{i^*,i'}^S$, and since $P_{i'}$ is in $G^{S,i}$ then the following holds.

- $\boldsymbol{\alpha}_{i'}^{S,i} + \boldsymbol{\rho}_{i^*,i'}^{S,i} = \mathbf{t}_{i',i}^S \cdot \mathbf{b}_{i',i^*}^S[A^S] = \mathbf{t}_{i',i}^S \cdot \mathbf{a}_{S,i^*,i}[A^S] = \mathbf{t}_{i',i}^S \cdot (A^S + \mathbf{r}_{i^*,i'}^S[A^S])$ and since P_{i^*} is honest then $\boldsymbol{\rho}_{i^*,i'}^{S,i} = \mathbf{t}_{i',i}^S \cdot \mathbf{r}_{i^*,i'}^S[A^S]$ so $\boldsymbol{\alpha}_{i'}^{S,i} = \mathbf{t}_{i',i}^S \cdot A^S$.
- For every $k \in [m]$ it holds that $\phi_{i',k}^{S,i} + \gamma_{i^*,i',k}^{S,i} = \mathbf{t}_{i',i}^S \cdot \mathbf{b}_{i',i^*}^S[\mathbf{f}_k^S] = \mathbf{t}_{i',i}^S \cdot \mathbf{a}_{i^*,i'}^S[\mathbf{f}_k^S] = \mathbf{t}_{i',i}^S \cdot (\mathbf{f}_k^S + \mathbf{r}_{i^*,i'}^S[\mathbf{f}_k^S])$ and since P_{i^*} is honest then $\boldsymbol{\rho}_{i^*,i'}^{S,i} = \mathbf{t}_{i',i}^S \cdot \mathbf{r}_{i^*,i'}^S[\mathbf{f}_k^S]$ so $\phi_{i',k}^{S,i} = \mathbf{t}_{i',i}^S \cdot \mathbf{f}_k^S$.

This concludes the proof of the claim. □

Therefore, the same argument as for an honest $P_{i'}$ shows that P_i does not reject because of $P_{i'}$, as required. This concludes the analysis of the outputs of the honest parties in $\mathsf{iSig.open}_1$.

Execution of $\mathsf{iSig.open}_2$. It is not hard to see that $\mathsf{iSig.open}_2$ is perfectly simulated. It remains to analyse the outputs of the honest parties. However, this follows as before for every good execution of the protocol. This concludes the analysis of a corrupt $\mathcal{D}$ and an honest $\mathcal{I}$.

D.4 Corrupt $\mathcal{D}$ and $\mathcal{I}$

D.4.1 The Simulator

Simulation of $\mathsf{iSig.dis}$. The simulator initiate an execution of $\mathsf{iSig.dis}$ by taking the roles of the honest parties. The simulator sets $s_i := 0$ for all $i \in [m]$, and inputs $(s_1, \ldots, s_m)$.

Simulation of $\mathsf{iSig.ver}$. The simulator continues with the execution of $\mathsf{iSig.ver}$. The simulator inputs $\mathsf{reveal}_{\mathcal{I}} = 0$ to the functionality.

Simulation of $\mathsf{iSig.open}_1$. The simulator continues with the execution of $\mathsf{iSig.open}_1$. At the end of the execution the simulator computes the outputs of the honest parties. If $\mathcal{D}$ broadcasts $\mathbf{s}'$ in $\mathsf{iSig.open}_1$, then the simulator inputs $\mathsf{reveal}_{\mathcal{D}} = 1$ and $\mathbf{s}'$ to the ideal functionality. Otherwise, the simulator inputs $\mathsf{reveal}_{\mathcal{D}} = 0$ and $\mathbf{s}$ to the ideal functionality.

Let P_i be an arbitrary honest party. If the public output of P_i is $\perp$ then the simulator inputs $\mathsf{abort}^{\mathsf{pub}} = 1$ and terminates (the rest of the inputs do not matter). Otherwise, let the public output of P_i be $(\boldsymbol{\beta}_j^{\mathsf{pub}}, z_j^{\mathsf{pub}})_{j\in[\ell]}$. Then the simulator inputs $(\boldsymbol{\beta}_j^{\mathsf{pub}}, z_j^{\mathsf{pub}})_{j\in[\ell]}$ and $\mathsf{abort}^{\mathsf{pub}} = 0$ to the ideal functionality for the public outputs.

For the private inputs of an honest P_i the simulator does as follows. If the private output is $\perp$ then the simulator inputs $\mathsf{abort}_i^{\mathsf{pri}} = 1$, and otherwise $\mathsf{abort}_i^{\mathsf{pri}} = 0$. In the latter case, let $(\boldsymbol{\beta}_{i,j}^{\mathsf{pri}}, z_{i,j}^{\mathsf{pri}})_{j\in[\ell]}$ be the private output of P_i. Then the simulator inputs $(\boldsymbol{\beta}_{i,j}^{\mathsf{pri}}, z_{i,j}^{\mathsf{pri}})_{j\in[\ell]}$ for the public outputs, for the private outputs of P_i.

Simulation of $\mathsf{iSig.open}_2$**.** The simulator continues with the execution of $\mathsf{iSig.open}_1$. At the end of the execution the simulator computes the outputs of the honest parties. Let P_i be an arbitrary honest party. If the output of P_i is $\perp$ then the simulator inputs $\mathsf{abort} = 1$ and terminates. Otherwise $\mathsf{abort} = 0$, and we denote the outputs of P_i by $(k_i, w_{k_i})_{i\in[\mu]}$. Then the simulator inputs $s''_{k_i} = w_{k_i}$ for every $i \in [\mu]$, inputs arbitrary values for the rest of s''_i, and inputs $\mu, (k_i)_{i\in[\mu]}$ to the functionality.

D.4.2 Analysis

Observe that there is a 1-1 correspondence between real-world executions and the simulated executions performed by the simulator up to the end of $\mathsf{iSig.open}_1$. It is not hard to see that $\mathsf{iSig.dis}$, $\mathsf{iSig.ver}$, the outputs of the honest parties in $\mathsf{iSig.ver}$ and $\mathsf{iSig.open}_1$ are perfectly simulated. We continue by considering the outputs of the honest parties in $\mathsf{iSig.open}_1$.

Outputs of honest parties in $\mathsf{iSig.open}_1$**.** We start by analysing the public output. Observe that if some honest party outputs $(\boldsymbol{\beta}_j^{\mathsf{pub}}, z_j^{\mathsf{pub}})_{j\in[\ell]}$ as the public output, then the rest of the honest parties can either output $(\boldsymbol{\beta}_j^{\mathsf{pub}}, z_j^{\mathsf{pub}})_{j\in[\ell]}$ or $\perp$. Therefore, we need to show that the honest parties agree on whether to accept the public output.

Denote the broadcast of $\mathcal{I}$ by $(\boldsymbol{\beta}_j^{\mathsf{pub}}, \mathsf{Out}_j)_{j\in[\ell]}$. Fix any set $S \in \mathcal{G}$, and observe that since $\mathcal{D}$ is corrupt, S contains at least one honest party, denoted P_{i^*}, and that for every honest P_i it holds that all honest parties in S are also in $G^{S,i}$. Also observe that all honest parties in S agree on the values of $A^S, (\mathbf{f}_k^S, \mathbf{r}_k^S)_{k\in[m]}$. We split into cases.

- Assume that $A^S \cdot \mathsf{Out}_j \neq \sum_{k\in[m]} \boldsymbol{\beta}_j[k] \cdot \mathbf{f}_k^S$ for some $j \in [\ell]$. Then, for every honest P_i it holds that $\boldsymbol{\alpha}_{i^*}^{S,i} \cdot \mathsf{Out}_j = \sum_{k\in[m]} \boldsymbol{\beta}_j[k] \cdot \phi_{i^*,k}^{S,i}$ if and only if $\mathbf{t}_{i^*,i}^S \cdot A^S \cdot \mathsf{Out}_j = \sum_{k\in[m]} \boldsymbol{\beta}_j[k] \cdot \mathbf{t}_{i^*,i}^S \cdot \mathbf{f}_k^S$ if and only if $\mathbf{t}_{i^*,i}^S \cdot (A^S \cdot \mathsf{Out}_j - \sum_{k\in[m]} \boldsymbol{\beta}_j[k] \cdot \mathbf{f}_k^S) = 0$. Since $A^S \cdot \mathsf{Out}_j \neq \sum_{k\in[m]} \boldsymbol{\beta}_j[k] \cdot \mathbf{f}_k^S$, and since $\mathbf{t}_{i^*,i}^S$ is uniformly distributed even conditioned on the view of the adversary so far, it follows that the probability that P_i accepts the public opening is at most $1/|\mathbb{F}|$. By taking union bound on every honest P_i, it holds that the probability that some honest P_i accepts is at most $n/|\mathbb{F}|$.

- Assume that $A^S \cdot \mathsf{Out}_j = \sum_{k\in[m]} \boldsymbol{\beta}_j[k] \cdot \mathbf{f}_k^S$ for all $j \in [\ell]$. Fix any honest P_i. It is not hard to see that P_i does not reject the public opening due to an honest party in S. Consider a corrupt $P_{i'}$ in $G^{S,i}$. Then the same proof of Claim D.1 shows that $\boldsymbol{\alpha}_{i'}^{S,i} = \mathbf{t}_{i',i}^S \cdot A^S$, and $\phi_{i',k}^{S,i} = \mathbf{t}_{i',i}^S \cdot \mathbf{f}_k^S$ for all $k \in [m]$. Therefore, it is not hard to see that P_i does not reject on behalf of $P_{i'}$ as well.

We conclude that for every set $S \in \mathcal{G}$, the parties agree whether to accept or reject the opening due to S with probability at least $1 - n/|\mathbb{F}|$. Taking a union bound over all sets $S \in \mathcal{S}$ we conclude that the probability of disagreement is at most $\binom{n-2}{t} n/|\mathbb{F}| \leq 2^{-\kappa}$, as required. Note that here we require $|\mathbb{F}| \geq \binom{n-2}{t} n 2^{\kappa}$. Finally, it is not hard to see that if there is an agreement on the public output, then the private outputs are the same in both worlds.

Execution of $\mathsf{iSig.open}_2$**.** Even conditioned on the view of the adversary so far, the random variables $(\mathbf{t}_{i,j}^S)_{S \in \mathcal{G}, P_i \in S \cap \mathsf{H}, P_j \in \mathsf{H}}$ are $2^{-\kappa}$-close to uniform. Therefore the same argument as in the analysis of the outputs in $\mathsf{iSig.open}_1$ shows that the honest parties agree on the outputs in $\mathsf{iSig.open}_2$. This concludes the analysis of iSig.

E The Sharing Functionality

In this section, we prove that protocol sh-comp UC-emulates $\mathcal{F}_{\mathsf{sh\text{-}comp}}$ with statistical security in the $\mathcal{F}_{\mathsf{iSig}}$-hybrid model. From the composition properties of UC-security, this implies that protocol sh-comp UC-emulates $\mathcal{F}_{\mathsf{sh\text{-}comp}}$. We do so by presenting a simulator $\mathsf{Sim}_{\mathsf{sh\text{-}comp}}$ for the protocol sh-comp.

We begin by presenting some helper simulators, that will be used by the $\mathsf{Sim}_{\mathsf{sh\text{-}comp}}$ to simulate the sub-protocols of sh-comp. Those helper simulators receive inputs from $\mathsf{Sim}_{\mathsf{sh\text{-}comp}}$ and simulate the view of the adversary in an execution of a sub-protocol of sh-comp.

E.1 Verifiable Secret Sharing

We begin by presenting a helper simulator $\mathsf{Sim}_{\mathsf{vss}}$ for the vss protocol. We consider only the case of an honest P_d, and denote the simulator by $\mathsf{Sim}_{\mathsf{vss}}^{\mathsf{H}}$.

Round 1. On input $(i, f_i(x))_{i \in \mathsf{C}}$, the simulator gives $(f_i(j))_{j \in [n]}$ as the output of the distribution phase of $\mathcal{F}_{\mathsf{iSig}}$ in the instance where P_d is the dealer and P_i is the intermediary. In addition, for every honest P_i and every corrupt P_j, the simulator picks a random pad $r_{i,j} \in \mathbb{F}$ and gives $r_{i,j}$ as the output of the distribution phase of $\mathcal{F}_{\mathsf{iSig}}$ in the instance where P_i is the dealer and P_j is the intermediary. At this stage the simulator receives the messages from the corrupt parties to the honest parties, and the inputs of the corrupt parties to the $\mathcal{F}_{\mathsf{iSig}}$ instances in which the dealer is corrupt.

Round 2. For every honest P_i the simulator does as follows.

- For every honest P_j where $j \geq i$ the simulator picks random field elements $a_{i,j}$ and $b_{i,j}$ and broadcasts $a_{i,j}$ and $b_{i,j}$ on behalf of P_i.
- For every honest P_j where $j < i$ the simulator already picked $a_{j,i}$ and $b_{j,i}$. The simulator sets $a_{i,j} := b_{j,i}$ and $b_{i,j} := a_{j,i}$ and broadcasts $a_{i,j}$ and $b_{i,j}$ on behalf of P_i.
- For every corrupt P_j the simulator picked in the first round the random pad $r_{i,j}$, and also received from the adversary the pad $r'_{j,i}$. The simulator sets $a_{i,j} := f_j(i) + r_{i,j}$ and $b_{i,j} := f_j(i) + r'_{j,i}$ and broadcasts $a_{i,j}$ and $b_{i,j}$ on behalf of P_i.

In addition, the simulator does as follows for P_d.

- For every honest P_i, the simulator sets $a^d_{i,j} := a_{i,j}$ for every $j \in [n]$, and broadcasts $(a^d_{i,j})_{j\in[n]}$ on behalf of P_d.
- For every corrupt P_i, the simulator received a pad $r^d_{i,j}$ from P_i, for all $j \in [n]$. The simulator sets $a^d_{i,j} := f_i(j) + r^d_{i,j}$ for every $j \in [n]$, and broadcasts $(a^d_{i,j})_{j\in[n]}$ on behalf of P_d.

At this stage the simulator receives the messages from the corrupt parties to the honest parties, and the inputs of the corrupt parties to the $\mathcal{F}_{\mathsf{iSig}}$ instances in which the intermediary is corrupt.

We say that P_d is internally-vss-conflicted with a corrupt P_i if either (1) P_i's input to the verification phase of the iSig instance in which P_d is the dealer and P_i is the intermediary is $\mathsf{reveal}_{\mathcal{I}} = 1$, or (2) P_i broadcasted a public complaint in the second round, or (3) $a_{i,j} \neq a^d_{i,j}$ for some $j \in [n]$, where $a_{i,j}$ was received from the adversary in the second round.

We say that an honest P_i is internally-vss-conflicted with a corrupt P_j if either (1) P_j's input to the verification phase of the iSig instance in which P_i is the dealer and P_j is the intermediary is $\mathsf{reveal}_{\mathcal{I}} = 1$, or (2) $a_{i,j} \neq b_{j,i}$, or (3) $a_{j,i} \neq b_{i,j}$.

Round 3. The simulator receives the flags $\mathsf{flag}_i, (\mathsf{flag}_{i,j})_{j\in[n]}$ for every honest P_i, as well as the flags $(\mathsf{flag}^d_j)_{j\in[n]}$, where we assume that $\mathsf{flag}_{i,j} = 0$ for every honest P_i, P_j, and that $\mathsf{flag}_j = 0$ and $\mathsf{flag}^d_j = 0$ for every honest P_j.

The simulator first broadcasts the flags of all honest parties. In addition, the simulator does as follows.

- Consider any corrupt P_i for which either: (1) P_d is internally-vss-conflicted with P_i, or (2) $\mathsf{flag}^d_i = 1$. In this case P_d is vss-conflicted with P_i, and for every honest P_k the simulator sets $r_{k,i}$ as the output of the output phase 1 of the $\mathcal{F}_{\mathsf{iSig}}$ instance in which P_k is the dealer and P_d is the intermediary. In addition, in the instance of $\mathcal{F}_{\mathsf{iSig}}$ where P_i is the dealer and P_d is the intermediary the simulator leaks that P_d intends to open $(r^d_{i,j})_{j\in[n]}$ in that $\mathcal{F}_{\mathsf{iSig}}$ instance. Similarly, for every corrupt P_k, the simulator leaks that P_d intends to open $(r^d_{k,i})$ in the $\mathcal{F}_{\mathsf{iSig}}$ instance where P_k is the dealer and P_d is the intermediary.
- Consider an honest P_i and a corrupt P_j so that one of the following holds: (1) P_i is internally-vss-conflicted with P_j, or (2) $\mathsf{flag}_{i,j} = 1$. In this case P_i is vss-conflicted with P_j, and the simulator sets $f_j(i)$ as the output of the output phase 1 of the $\mathcal{F}_{\mathsf{iSig}}$ instance in which P_d is the dealer and P_i is the intermediary. In addition,in the $\mathcal{F}_{\mathsf{iSig}}$ instance where P_j is the dealer and P_i is the intermediary, the simulator leaks that P_i intends to open $(r_{i,j})$ in that $\mathcal{F}_{\mathsf{iSig}}$ instance.

At this point the simulator receives the messages of the corrupt parties, and their inputs to the various $\mathcal{F}_{\mathsf{iSig}}$ instances. Consider the opening phase 1 of the $\mathcal{F}_{\mathsf{iSig}}$ instance in which a corrupt P_i is the dealer and P_d is the intermediary.

- If the input is $\mathsf{reveal}_{\mathcal{D}} = 1$, then the simulator also received a pad $(\bar{r}^d_{i,j})_{j\in[n]}$ from the dealer P_i, and the simulator simply returns $b = 1$ and $(\bar{r}^d_{i,j})_{j\in[n]}$ as the output of the open phase 1.
- Otherwise, $\mathsf{reveal}_{\mathcal{D}} = 0$ and the simulator returns $b = 0$ as the output of the functionality.

If P_d is vss-conflicted with P_i then the simulator also returns $(r^d_{i,j})_{j\in[n]}$ as the output of the functionality.

If P_d is vss-conflicted with some P_k, then the simulator also returns $r^d_{i,k}$ as the output of the functionality.

Consider the opening phase 1 of the $\mathcal{F}_{\mathsf{iSig}}$ instance in which a corrupt P_i is the dealer and an honest P_j is the intermediary.

- If the input is $\mathsf{reveal}_{\mathcal{D}} = 1$, then the simulator also received a pad $(\bar{r}^d_{i,j})_{j\in[n]}$ from the dealer P_i, and the simulator simply returns $b = 1$ and $(\bar{r}^d_{i,j})_{j\in[n]}$ as the output of the open phase 1.
- Otherwise, $\mathsf{reveal}_{\mathcal{D}} = 0$ and the simulator returns $b = 0$ as the output of the functionality.

  In addition, if P_j is vss-conflicted with P_i, then the simulator also returns $(r_{i,j})$ as the output of the functionality.

For every instance of $\mathcal{F}_{\mathsf{iSig}}$ in which the dealer is P_d and the intermediary P_i is corrupt, the simulator holds all the inputs of P_d, and therefore, given the inputs of the corrupt P_i can compute the output of the functionality and return it to the adversary. Similarly, for every instance of $\mathcal{F}_{\mathsf{iSig}}$ in which the dealer is an honest P_i and the intermediary a corrupt P_j, the simulator holds all the inputs of P_i, and therefore, given the inputs of the corrupt P_j can compute the output of the functionality and return it to the adversary. Finally, for every instance of $\mathcal{F}_{\mathsf{iSig}}$ in which both the dealer and the intermediary are corrupt, the simulator holds all the inputs, and therefore can compute the output and return it to the adversary. This completes the simulation for an honest P_d.

E.2 Triple Secret Sharing

We continue by presenting a helper protocol $\mathsf{Sim}_{\mathsf{tss}}$ for the tss protocol. We only consider the simulator for an honest P_d, denoted $\mathsf{Sim}^{\mathsf{H}}_{\mathsf{tss}}$.

Round 1. The simulator receives inputs $(i, f^a_i(x), f^b_i(x), f^c_i(x))_{i\in\mathsf{C}}$. The simulator picks random polynomials $A(x), B(x)$ and $C(x)$ of degree d, d and $2d$, respectively, conditioned on $A(0) = B(0) = C(0) = 0$. The simulator picks any symmetric bivariate polynomials $F^{a,0}(x, y)$, $F^{b,0}(x, y)$ and $F^{c,0}(x, y)$ of degree at most t in each variables, such that $F^{u,0}(x, i) = f^u_i(x)$ and $F^{u,0}(0, 0) = 0$, for every $u \in \{a, b, c\}$. In addition, for every $1 \le k \le d$, the simulator picks random symmetric bivariate polynomials $F^{a,k}(x, y)$ and $F^{b,k}(x, y)$ of degree at most t in each variables such that $F^{a,k}(0, 0) = A^k$ and $F^{b,k}(0, 0) = B^k$. Similarly, for every $1 \le k \le 2d$, the simulator picks a random symmetric bivariate polynomial $F^{c,k}(x, y)$ of degree at most t in each variables such that $F^{c,k}(0, 0) = C^k$.

For every $u \in \{a, b\}$ and $k \in \{0, \ldots, d\}$ the first round of $\mathsf{vss}^{u,k}$ is simulated by executing an instance of $\mathsf{Sim}^{\mathsf{H}}_{\mathsf{vss}}$, denoted $\mathsf{Sim}^{\mathsf{H}}_{\mathsf{vss}}[u, k]$, with P_d as the dealer, and with inputs $(i, F^{u,k}(x, i))_{i\in\mathsf{C}}$. Similarly, for every $k \in \{0, \ldots, 2d\}$ the first round of $\mathsf{vss}^{c,k}$ is simulated by executing an instance of $\mathsf{Sim}^{\mathsf{H}}_{\mathsf{vss}}$, denoted $\mathsf{Sim}^{\mathsf{H}}_{\mathsf{vss}}[c, k]$, with P_d as the dealer, and with inputs $(i, F^{c,k}(x, i))_{i\in\mathsf{C}}$.

Round 2. The second round of $\mathsf{vss}^{u,k}$ is simulated by continuing the execution of $\mathsf{Sim}^{\mathsf{H}}_{\mathsf{vss}}[u,k]$. In addition, for every set Q of size $t+1$ in which P_Q is honest, the simulator picks random non-zero field elements $(\beta_{Q,i})_{i\in\{1,\ldots,n\}}$, and sends them to all corrupt parties in Q as the message from P_Q. At this stage the simulator receives the messages from the corrupt parties to the honest parties.

Round 3. The simulator receives the flags $\mathsf{flag}_i, (\mathsf{flag}_{i,j})_{j\in[n]}$ for every honest P_i, as well as the flags $(\mathsf{flag}^d_j)_{j\in[n]}$, where we assume that $\mathsf{flag}_{i,j}=0$ for every honest P_i, P_j, and that $\mathsf{flag}^d_j = \mathsf{flag}_j = 0$ for every honest P_j. For every honest P_i the simulator sets

- $\widetilde{\mathsf{flag}}_i := 0$,
- $\widetilde{\mathsf{flag}}_{i,j} := 0$ for every honest P_j,
- $\widetilde{\mathsf{flag}}_{i,j} := 1$ for every corrupt P_j with $\mathsf{flag}_{i,j}=1$ or so that P_i is internally-vss-conflicted with P_j in $\mathsf{Sim}^{\mathsf{H}}_{\mathsf{vss}}[u,k]$ for some $u\in\{a,b,c\}$ and $k\in\{0,\ldots,2d\}$.

In addition, for every corrupt P_i the simulator sets $\widetilde{\mathsf{flag}}^d_i := 1$ if $\mathsf{flag}^d_i = 1$ or if P_d is internally-vss-conflicted with P_i in $\mathsf{Sim}^{\mathsf{H}}_{\mathsf{vss}}[u,k]$ for some $u\in\{a,b,c\}$ and $k\in\{0,\ldots,2d\}$.

The third round of $\mathsf{vss}^{u,k}$ is simulated by continuing the execution of $\mathsf{Sim}^{\mathsf{H}}_{\mathsf{vss}}[u,k]$ with inputs $\widetilde{\mathsf{flag}}_i, (\widetilde{\mathsf{flag}}_{i,j})_{j\in[n]}$ for every honest P_i, as well as the flags $(\widetilde{\mathsf{flag}}^d_j)_{j\in[n]}$ for the dealer.

The simulation of linop_1 is done as follows. For every set Q, every honest P_i in Q does as follows for every corrupt P_j. If P_Q is honest then we already sampled $\beta_{Q,j}$. Otherwise let $\beta_{Q,j}$ be the value that the corrupt P_Q sent to P_i as the challenge corresponding to P_j. For every honest $P_{j'}$, the private output of P_j in the instance of $\mathcal{F}_{\mathsf{iSig}}$ where $P_{j'}$ is the dealer and P_i the intermediary is $(\sum_{k=0}^{d}\beta^k_{Q,j}\cdot r^{a,k}_{j',i}, \sum_{k=0}^{d}\beta^k_{Q,j}\cdot r^{b,k}_{j',i}, \sum_{k=0}^{2d}\beta^k_{Q,j}\cdot r^{c,k}_{j',i})$ together with the vector of coefficients $(\beta^0_{Q,j},\ldots,\beta^d_{Q,j})$, $(\beta^0_{Q,j},\ldots,\beta^d_{Q,j})$, and $(\beta^0_{Q,j},\ldots,\beta^{2d}_{Q,j})$, respectively, where $r^{u,k}_{j',i} = b^{u,k}_{i,j'} - F^{u,k}(j',i)$, and $b^{u,k}_{i,j'}$ is the broadcast of P_i in the second round of $\mathsf{Sim}^{\mathsf{H}}_{\mathsf{vss}}[u,k]$. In addition, for every corrupt $P_{j'}$ for which $\widetilde{\mathsf{flag}}_{i,j'} = 0$, the simulator leaks the vectors $(\beta^0_{Q,j},\ldots,\beta^d_{Q,j})$, $(\beta^0_{Q,j},\ldots,\beta^d_{Q,j})$, and $(\beta^0_{Q,j},\ldots,\beta^{2d}_{Q,j})$ to the adversary in the $\mathcal{F}_{\mathsf{iSig}}$ instance in which $P_{j'}$ is the dealer and P_i is the intermediary.

At this point the simulator receives the messages of the corrupt parties, and their inputs to the various $\mathcal{F}_{\mathsf{iSig}}$ instances. Consider the opening phase 1 of the $\mathcal{F}_{\mathsf{iSig}}$ instance in which a corrupt $P_{j'}$ is the dealer and an honest P_i is the intermediary.

- If the input is $\mathsf{reveal}_{\mathcal{D}} = 1$, then the simulator does nothing (the simulation is done in the third round of the $\mathsf{Sim}^{\mathsf{H}}_{\mathsf{vss}}$ executions).
- Otherwise, $\mathsf{reveal}_{\mathcal{D}} = 0$ and the simulator returns $b=0$ as the output of the functionality.

  In addition, if P_i is $\widetilde{\mathsf{flag}}_{i,j'} = 1$ then the simulator does nothing. Otherwise, for every set Q the simulator does as follows for every corrupt P_j. If P_Q is honest then we already sampled $\beta_{Q,j}$. Otherwise let $\beta_{Q,j}$ be the value that P_Q sent to P_i as the challenge corresponding to P_j. The private output of P_j in the instance of $\mathcal{F}_{\mathsf{iSig}}$ where $P_{j'}$ is the dealer and P_i the intermediary is $(\sum_{k=0}^{d}\beta^k_{Q,j}\cdot r^{a,k}_{j',i}, \sum_{k=0}^{d}\beta^k_{Q,j}\cdot r^{b,k}_{j',i}, \sum_{k=0}^{2d}\beta^k_{Q,j}\cdot r^{c,k}_{j',i})$ together with the vector of coefficients $(\beta^0_{Q,j},\ldots,\beta^d_{Q,j})$, $(\beta^0_{Q,j},\ldots,\beta^d_{Q,j})$, and $(\beta^0_{Q,j},\ldots,\beta^{2d}_{Q,j})$, respectively, where $r^{u,k}_{j',i}$ is the input of

$P_{j'}$ to the instance of $\mathcal{F}_{\mathsf{iSig}}$ corresponding to $\mathsf{Sim}^{\mathsf{H}}_{\mathsf{vss}}[u,k]$, where $P_{j'}$ is the dealer and P_i is the intermediary.

For every instance of $\mathcal{F}_{\mathsf{iSig}}$ in which the dealer is an honest P_i and the intermediary a corrupt P_j, the simulator holds all the inputs of P_i, and therefore, given the inputs of the corrupt P_j can compute the output of the functionality and return it to the adversary. Finally, for every instance of $\mathcal{F}_{\mathsf{iSig}}$ in which both the dealer and the intermediary are corrupt, the simulator holds all the inputs, and therefore can compute the output and return it to the adversary. This completes the simulation for an honest P_d.

E.3 Verifiable Sharing and Transferring

We continue by presenting a helper protocol $\mathsf{Sim}_{\mathsf{vst}}$ for the vst protocol. We only consider the simulator for honest P_d, R, denoted $\mathsf{Sim}^{\mathsf{H},\mathsf{H}}_{\mathsf{vst}}$.

Round 1. The simulator receives inputs $(i, f_i(x))_{i\in\mathsf{C}}$. The simulator simulates the vss instance by executing $\mathsf{Sim}^{\mathsf{H}}_{\mathsf{vss}}$ with inputs $(i, f_i(x))_{i\in\mathsf{C}}$. In addition, the simulator picks random pads $(\rho_{R,i,j})_{i\in\mathsf{C},j\in[n]}$ and gives $(\rho_{R,i,j})_{j\in[n]}$ as the output of the distribution phase of $\mathcal{F}_{\mathsf{iSig}}$ in the instance where R is the dealer and a corrupt P_i is the intermediary. At this stage the simulator receives the messages from the corrupt parties to the honest parties, and the inputs of the corrupt parties to the $\mathcal{F}_{\mathsf{iSig}}$ instances in which the dealer is corrupt.

Round 2. The simulator continues the simulation of the vss instance by executing the second round of $\mathsf{Sim}^{\mathsf{H}}_{\mathsf{vss}}$. In addition, for every honest P_i, and every $j \in [n]$ the simulator picks random field elements $\alpha_{i,j,R}$ and $\beta_{i,j,R}$, and broadcasts $\alpha_{i,j,R}$ and $\beta_{i,j,R}$ on behalf of P_i.

For every honest P_i, and every $j \in [n]$, the simulator sets $\alpha_{R,i,j} := \beta_{i,j,R}$ and $\beta_{R,i,j} := \alpha_{i,j,R}$, and broadcasts $\alpha_{R,i,j}$ and $\beta_{R,i,j}$ on behalf of R. For every corrupt P_i, and every $j \in [n]$, let $\rho'_{i,j,R}$ be the pad that R received in the first round. Then the simulator sets $\alpha_{R,i,j} := f_i(j) + \rho_{R,i,j}$ and $\beta_{R,i,j} := f_i(j) + \rho'_{i,j,R}$, and broadcasts $\alpha_{R,i,j}$ and $\beta_{R,i,j}$ on behalf of R.

In addition, the simulator does as follows for P_d.

- For every $i, j \in [n]$ the simulator sets $\alpha^d_{R,i,j} := \alpha_{R,i,j}$, and broadcasts $\alpha^d_{R,i,j}$ on behalf of P_d.
- For every honest P_i and every $j \in [n]$, the simulator sets $\alpha^d_{i,j,R} := \alpha_{i,j,R}$, and broadcasts $\alpha^d_{i,j,R}$ on behalf of P_d.
- For every corrupt P_i, the simulator received a pad $\rho^d_{i,j,R}$ from P_i, for all $j \in [n]$. The simulator sets $\alpha^d_{i,j,R} := f_i(j) + \rho^d_{i,j,R}$ for every $j \in [n]$, and broadcasts $(\alpha^d_{i,j,R})_{j\in[n]}$ on behalf of P_d.

At this stage the simulator receives the messages from the corrupt parties to the honest parties, and the inputs of the corrupt parties to the $\mathcal{F}_{\mathsf{iSig}}$ instances in which the intermediary is corrupt.

Round 3. The simulator receives the flags $\mathsf{flag}_{i,d}, \mathsf{flag}_{i,R}, (\mathsf{flag}_{i,j})_{j\in[n]}$ for every honest P_i, as well as the flags $(\mathsf{flag}^d_j)_{j\in[n]}$ and $(\mathsf{flag}^R_j)_{j\in[n]}$, where we assume that $\mathsf{flag}_{i,j} = 0$ for every honest P_i, P_j, and that $\mathsf{flag}^d_j = \mathsf{flag}^R_j = \mathsf{flag}_{j,d} = \mathsf{flag}_{j,R} = 0$ for every honest P_j.

For every honest P_i the simulator sets

- $\widetilde{\mathsf{flag}}_{i,d} = \widetilde{\mathsf{flag}}_{i,R} = 0$,
- $\widetilde{\mathsf{flag}}_{i,j} = 0$ for every honest P_j,
- $\widetilde{\mathsf{flag}}_{i,j} = 1$ for every corrupt P_j with either (1) $\mathsf{flag}_{i,j} = 1$, or (2) P_i is internally-vss-conflicted with P_j in $\mathsf{Sim}^{\mathsf{H}}_{\mathsf{vss}}$.

For the dealer, the simulator sets $\widetilde{\mathsf{flag}}^{d}_{R} = 0$; the simulator sets $\widetilde{\mathsf{flag}}^{d}_{i} = 0$ for every honest P_i; for every corrupt P_i the simulator sets $\widetilde{\mathsf{flag}}^{d}_{i} := 1$ if either (1) $\mathsf{flag}^{d}_{i} = 1$ or (2) if P_d is internally-vss-conflicted with P_i in $\mathsf{Sim}^{\mathsf{H}}_{\mathsf{vss}}$, or (3) if $\alpha_{i,j,R} \neq \alpha^{d}_{i,j,R}$. For the receiver, the simulator sets $\widetilde{\mathsf{flag}}^{R}_{d} = 0$; the simulator sets $\widetilde{\mathsf{flag}}^{R}_{i} = 0$ for every honest P_i; for every corrupt P_i the simulator sets $\widetilde{\mathsf{flag}}^{d}_{i} := 1$ if either (1) $\mathsf{flag}^{d}_{i} = 1$, or (2) $\alpha_{R,i,j} \neq \beta_{i,j,R}$, or (3) $\beta_{R,i,j} \neq \alpha_{i,j,R}$.

The simulator continues the execution of the vss instance by executing the third round of $\mathsf{Sim}^{\mathsf{H}}_{\mathsf{vss}}$ with inputs $(\widetilde{\mathsf{flag}}_{i,d}, \widetilde{\mathsf{flag}}_{i,R}, \widetilde{\mathsf{flag}}_{i,j})_{i\in\mathsf{H},j\in[n]}$, $(\widetilde{\mathsf{flag}}^{d}_{j}, \widetilde{\mathsf{flag}}^{d}_{R})_{j\in[n]}$ and $(\widetilde{\mathsf{flag}}^{R}_{j}, \widetilde{\mathsf{flag}}^{R}_{d})_{j\in[n]}$. The simulator also broadcasts the flags of all honest parties. In addition, the simulator does as follows.

- Consider any corrupt P_i for which $\widetilde{\mathsf{flag}}^{d}_{i} = 1$. In this case P_d is vst-conflicted with P_i, and the simulator sets $(\rho^{d}_{R,i,j} := \alpha^{d}_{R,i,j} - f_i(j))_{j\in[n]}$ as the output of the output phase 1 of the $\mathcal{F}_{\mathsf{iSig}}$ instance in which R is the dealer and P_d is the intermediary. In addition, in the $\mathcal{F}_{\mathsf{iSig}}$ instance where P_i is the dealer and P_d is the intermediary, the simulator leaks that P_d intends to open $(\rho^{d}_{i,j,R})_{j\in[n]}$.

- Consider any corrupt P_i for which $\widetilde{\mathsf{flag}}^{R}_{i} = 1$. In this case R is vst-conflicted with P_i, then for every honest P_j the simulator sets $(\rho_{j,i,R} := \beta^{d}_{R,i,j} - f_i(j))_{j\in\mathsf{H}}$ as the output of the output phase 1 of the $\mathcal{F}_{\mathsf{iSig}}$ instance in which an honest P_j is the dealer and R is the intermediary. Moreover, for every corrupt P_j, in the $\mathcal{F}_{\mathsf{iSig}}$ instance where P_j is the dealer and R is the intermediary, the simulator leaks that R intends to open $(\rho'_{j,i,R})$.

  In addition, the simulator sets the values $(f_i(j))_{j\in[n]}$ as the output of the output phase 1 of the $\mathcal{F}_{\mathsf{iSig}}$ instance in which P_d is the dealer and R is the intermediary, both for $F'(j,i)$ and for $F'(i,j)$.

  Moreover, in the $\mathcal{F}_{\mathsf{iSig}}$ instance where P_i is the dealer and R is the intermediary, the simulator leaks that R intends to open $(\rho'_{i,j,R})_{j\in[n]}$.

At this point the simulator receives the messages of the corrupt parties, and their inputs to the various $\mathcal{F}_{\mathsf{iSig}}$ instances. Consider the opening phase 1 of the $\mathcal{F}_{\mathsf{iSig}}$ instance in which a corrupt P_i is the dealer and P_d is the intermediary.

- If the input is $\mathsf{reveal}_{\mathcal{D}} = 1$, then the simulator also received a pad $(\bar{\rho}^{d}_{i,j,R})_{j\in[n]}$ from the adversary as an input to $\mathcal{F}_{\mathsf{iSig}}$, and the simulator simply returns $b = 1$ and $(\bar{\rho}^{d}_{i,j,R})_{j\in[n]}$ as the output of the open phase 1.

- Otherwise, $\mathsf{reveal}_{\mathcal{D}} = 0$ and the simulator returns $b = 0$ as the output of the functionality.

  If P_d is vst-conflicted with P_i then the simulator also returns $(\rho^{d}_{i,j,R})_{j\in[n]}$ as the output of the functionality.

Consider the opening phase 1 of the $\mathcal{F}_{\mathsf{iSig}}$ instance in which a corrupt P_i is the dealer and R is the intermediary.

- If the input is $\mathsf{reveal}_{\mathcal{D}} = 1$, then the simulator also received a pad $(\bar{\rho}'_{i,j,R})_{j\in[n]}$ from the adversary as an input to $\mathcal{F}_{\mathsf{iSig}}$, and the simulator simply returns $b = 1$ and $(\bar{\rho}'_{i,j,R})_{j\in[n]}$ as the output of the open phase 1.
- Otherwise, $\mathsf{reveal}_{\mathcal{D}} = 0$ and the simulator returns $b = 0$ as the output of the functionality.

  In addition, if R is vst-conflicted with P_i then the simulator also returns $(\rho_{i,j,R})_{j\in[n]}$ as the output of the functionality.

  If R is vst-conflicted with P_k, then the simulator also returns $(\rho_{k,i,R})$ as the output of the functionality.

For every instance of $\mathcal{F}_{\mathsf{iSig}}$ in which the dealer is P_d and the intermediary P_i is corrupt, the simulator holds all the inputs of P_d, and therefore, given the inputs of the corrupt P_i can compute the output of the functionality and return it to the adversary. Similarly, for every instance of $\mathcal{F}_{\mathsf{iSig}}$ in which the dealer is R and the intermediary a corrupt P_j, the simulator holds all the inputs of R, and therefore, given the inputs of the corrupt P_j can compute the output of the functionality and return it to the adversary. Finally, for every instance of $\mathcal{F}_{\mathsf{iSig}}$ in which both the dealer and the intermediary are corrupt, the simulator holds all the inputs, and therefore can compute the output and return it to the adversary. This completes the simulation for an honest P_d and an honest R.

E.4 The sh-comp Simulator

In this section, we describe the simulator for protocol sh-comp.

E.4.1 Round 1 Simulation: Input Phase

For every honest P_i the simulator receives the values $(F^{i,v}(x,i'), G^{i,j}(x,i'), H^{i,j,k}(x,i'))_{i\in\mathsf{H},i'\in\mathsf{C},j,k\in[m],v\in[\ell]}$ from the ideal functionality, as well as the polynomial $F^{i,v}(x,y)$ for every corrupt $P_{i'}$, and every $i \in \mathsf{H}$ and $v \in [\ell]$ so that $\phi(i,v) = i'$.

vst **simulation.** The simulator simulates the vst executions as follows.

- For every $i \in \mathsf{H}$ and $v \in [\ell]$ so that $P_{\phi(i,v)}$ is honest, the simulator simulates $\mathsf{vst}^{F^{i,v}}$ by executing an instance of $\mathsf{Sim}^{\mathsf{H},\mathsf{H}}_{\mathsf{vst}}$, denoted $\mathsf{Sim}^{\mathsf{H},\mathsf{H}}_{\mathsf{vst}}[F^{i,v}]$, with inputs $(F^{i,v}(x,i'))_{i'\in\mathsf{C}}$. Denote the internal execution of $\mathsf{Sim}^{\mathsf{H}}_{\mathsf{vss}}$ by $\mathsf{Sim}^{\mathsf{H}}_{\mathsf{vss}}[F^{i,j}]$.
- For every $i \in \mathsf{H}$ and $v \in [\ell]$ so that $\mathrm{P}_{\phi(i,v)}$ is corrupt, the simulator simulates $\mathsf{vst}^{F^{i,v}}$ as follows. The simulator, that holds the input $F^{i,v}(x,y)$ of the honest dealer, take the roles of the honest parties, samples randomness for the execution of $\mathsf{vst}^{F^{i,v}}$ on their behalf, computes their messages, including the calls to $\mathcal{F}_{\mathsf{iSig}}$, and transfers them to the corrupt parties and to the simulated honest parties in $\mathsf{vst}^{F^{i,v}}$. We denote this sub-execution by $\mathsf{Sim}^{\mathsf{H},\mathsf{C}}_{\mathsf{vst}}[F^{i,v}]$.
- For every $i \in \mathsf{C}$ and $v \in [\ell]$ so that $P_{\phi(i,v)}$ is honest, the simulator simulates $\mathsf{vst}^{F^{i,v}}$ by taking the roles of the honest parties, that have no inputs, sampling randomness for the execution of $\mathsf{vst}^{F^{i,v}}$ on their behalf, computing their messages, including the calls to $\mathcal{F}_{\mathsf{iSig}}$, and transferring

them to the corrupt parties and to the simulated honest parties in $\mathsf{vst}^{F^{i,v}}$. We denote this sub-execution by $\mathsf{Sim}^{\mathsf{H},\mathsf{C}}_{\mathsf{vst}}[F^{i,v}]$. We denote this sub-execution by $\mathsf{Sim}^{\mathsf{C},\mathsf{H}}_{\mathsf{vst}}[F^{i,v}]$.

- For every $i \in \mathsf{C}$ and $v \in [\ell]$ so that $P_{\phi(i,v)}$ is corrupt, the simulator simulates $\mathsf{vst}^{F^{i,v}}$ by taking the roles of the honest parties, that have no inputs, sampling randomness for the execution of $\mathsf{vst}^{F^{i,v}}$ on their behalf, computing their messages, including the calls to $\mathcal{F}_{\mathsf{iSig}}$, and transferring them to the corrupt parties and to the simulated honest parties in $\mathsf{vst}^{F^{i,v}}$. We denote this sub-execution by $\mathsf{Sim}^{\mathsf{C},\mathsf{C}}_{\mathsf{vst}}[F^{i,v}]$.

tss **simulation.** We start by describing how to simulate the internal vss executions.

- For every $i \in \mathsf{H}$ and $j \in [m]$, the simulator simulates $\mathsf{vss}^{G^{i,j}}$ by executing an instance of $\mathsf{Sim}^{\mathsf{H}}_{\mathsf{vss}}$, denoted $\mathsf{Sim}^{\mathsf{H}}_{\mathsf{vss}}[G^{i,j}]$, with inputs $(G^{i,j}(x,i'))_{i' \in \mathsf{C}}$.
- For every $i \in \mathsf{H}$ and $j,k \in [m]$, the simulator simulates $\mathsf{vss}^{H^{i,j,k}}$ by executing an instance of $\mathsf{Sim}^{\mathsf{H}}_{\mathsf{vss}}$, denoted $\mathsf{Sim}^{\mathsf{H}}_{\mathsf{vss}}[H^{i,j,k}]$, with inputs $(H^{i,j,k}(x,i'))_{i' \in \mathsf{C}}$.
- For every $i \in \mathsf{C}$ and $j \in [m]$, the simulator simulates the execution of $\mathsf{vss}^{G^{i,j}}$ by taking the role of the honest parties, that have no inputs, sampling randomness for the execution of $\mathsf{vss}^{G^{i,j}}$ on their behalf, computing their messages, including the calls to $\mathcal{F}_{\mathsf{iSig}}$, and transferring them to the corrupt parties and to the simulated honest parties in $\mathsf{vss}^{G^{i,j}}$. We denote this sub-execution by $\mathsf{Sim}^{\mathsf{C}}_{\mathsf{vss}}[G^{i,j}]$.
- For every $i \in \mathsf{C}$ and $j,k \in [m]$, the simulator simulates the execution of $\mathsf{vss}^{H^{i,j,k}}$ by taking the role of the honest parties, that have no inputs, sampling randomness for the execution of $\mathsf{vss}^{H^{i,j,k}}$ on their behalf, computing their messages, including the calls to $\mathcal{F}_{\mathsf{iSig}}$, and transferring them to the corrupt parties and to the simulated honest parties in $\mathsf{vss}^{H^{i,j,k}}$. We denote this sub-execution by $\mathsf{Sim}^{\mathsf{C}}_{\mathsf{vss}}[H^{i,j,k}]$.

In order to simulate the tss executions, the simulator does as follows.

- For every $i \in \mathsf{H}$ and $j,k \in [m]$, the simulator simulates $\mathsf{tss}^{i,j,k}$ by executing an instance of $\mathsf{Sim}^{\mathsf{H}}_{\mathsf{tss}}$ with inputs $(G^{i,j}(x,i'), G^{i,k}(x,i'), H^{i,j,k}(x,i'))_{i' \in \mathsf{C}}$, where the internal $\mathsf{Sim}^{\mathsf{H}}_{\mathsf{vss}}$ executions corresponding to A^0, B^0 and C^0 are $\mathsf{Sim}^{\mathsf{H}}_{\mathsf{vss}}[G^{i,j}]$, $\mathsf{Sim}^{\mathsf{H}}_{\mathsf{vss}}[G^{i,k}]$ and $\mathsf{Sim}^{\mathsf{H}}_{\mathsf{vss}}[H^{i,j,k}]$ that were already executed. We denote this execution by $\mathsf{Sim}^{\mathsf{H}}_{\mathsf{tss}}[i,j,k]$.
- For every $i \in \mathsf{C}$ and $j,k \in [m]$, the simulator simulates $\mathsf{tss}^{i,j,k}$ by taking the role of the honest parties, that have no inputs, sampling randomness for the execution of $\mathsf{tss}^{i,j,k}$ on their behalf, computing their messages, including the calls to $\mathcal{F}_{\mathsf{iSig}}$, and transferring them to the corrupt parties and to the simulated honest parties in $\mathsf{tss}^{i,j,k}$, where the internal $\mathsf{Sim}^{\mathsf{H}}_{\mathsf{vss}}$ executions corresponding to A^0, B^0 and C^0 are $\mathsf{Sim}^{\mathsf{C}}_{\mathsf{vss}}[G^{i,j}]$, $\mathsf{Sim}^{\mathsf{C}}_{\mathsf{vss}}[G^{i,k}]$ and $\mathsf{Sim}^{\mathsf{C}}_{\mathsf{vss}}[H^{i,j,k}]$ that were already executed. We denote this execution by $\mathsf{Sim}^{\mathsf{C}}_{\mathsf{tss}}[i,j,k]$.

At this stage the simulator receives from the adversary the massages of all corrupt parties, as well as the inputs of all honest parties to the $\mathcal{F}_{\mathsf{iSig}}$ instances where the dealer is corrupt.

E.4.2 Round 2 Simulation

The second round simulation is done by continuing the execution of all $\mathsf{Sim}_{\mathsf{vss}}, \mathsf{Sim}_{\mathsf{vst}}$ and $\mathsf{Sim}_{\mathsf{tss}}$ in the same manner as in the first round. At the end of the round the simulator receives from the adversary the massages of all corrupt parties, as well as the inputs of all honest parties to the $\mathcal{F}_{\mathsf{iSig}}$ instances where the intermediary is corrupt.

E.4.3 Round 3 Simulation: Linear Computation Phase

For every honest P_i the simulator receives

$$\left(\sum_{j=1}^{m}\beta_{i,j}^{v}G^{i,j}(x,y)+\sum_{j,k\in[m]}\gamma_{i,j,k}^{v}H^{i,j,k}(x,y)+F^{i,v}(x,y)\right)_{v\in[\ell]}$$

from the ideal functionality. In addition, for every $((i_1,v_1),(i_2,v_2))\in S$ so that P_{i_1} and P_{i_2} are honest, the simulator receives $F^{i_1,v_1}(x,y)-F^{i_2,v_2}(x,y)$. Finally, for every or every $((i_1,v_1),(i_2,v_2))\in S$ so that P_{i_1} is honest and P_{i_2} is corrupt, the simulator receives $F^{i_1,v_1}(x,y)$ from the ideal functionality, and for every $((i_1,v_1),(i_2,v_2))\in S$ so that P_{i_2} is honest and P_{i_1} is corrupt, the simulator receives $F^{i_2,v_2}(x,y)$ from the ideal functionality.

Flag computation. At the beginning of the third round of simulation, for every honest P_i and every corrupt P_j, the simulator can compute whether P_i is *iSign-conflicted* with P_j by checking the input bit $\mathsf{reveal}_{\mathcal{I}}$ that the corrupt P_j sent to the $\mathcal{F}_{\mathsf{iSig}}$ instance where P_i is the dealer and P_j is the intermediary. For every honest P_i and P_j, we say that P_i and P_j are not *iSign-conflicted* . Therefore, for every honest P_i, and every P_j the simulator knows whether P_i is *iSign-conflicted* with P_j or not.

Based on this information, in every vss, tss and vst instance, the simulator can compute for every honest P_i and every P_j whether P_i is internally-vss-conflicted and internally-vst-conflicted with P_j. Based on this information, for every honest P_i the simulator can compute the input-flags of P_i to every vss, tss and vst instance just like in the protocol. Similarly, for every honest P_i and every corrupt P_j we know whether P_i thinks that its shares might change in a vss instance in which P_j is the dealer.

The simulator simulates the third round of vss, tss and vst by continuing the execution of all $\mathsf{Sim}_{\mathsf{vss}}, \mathsf{Sim}_{\mathsf{vst}}$ and $\mathsf{Sim}_{\mathsf{tss}}$ in the same manner, and using the computed flags as the inputs of the honest parties. In addition, for every instance of $\mathcal{F}_{\mathsf{iSig}}$ in which both the dealer and the intermediary are honest, the simulator returns the bit $b=0$.

linop_3 **simulation.** Consider any honest P_i and $v\in[\ell]$. Let $P_{i'}$ be an honest party, and for every $j,k\in[m]$, denote by $(b_{i',j'}^{G^{i,j}})_{j'\in[n]}, (b_{i',j'}^{H^{i,j,k}})_{j'\in[n]}, (b_{i',j'}^{F^{i,v}})_{j'\in[n]}$ the broadcast of $P_{i'}$ in the simulation of $\mathsf{vss}^{G^{i,j}}, \mathsf{vss}^{H^{i,j,k}}$ and $\mathsf{vss}^{F^{i,v}}$, respectively. Then the messages of $P_{i'}$ in the v-th linear computation of P_i are simulated as follows.

- If $\mathsf{flag}_{i',R}=0$ in $\mathsf{vst}^{F^{i,v}}$, then for every honest $P_{j'}$, let the corresponding output of $\mathcal{F}_{\mathsf{iSig}}$ where

$P_{j'}$ is the dealer and $P_{i'}$ is the intermediary be

$$\left(\sum_{j=1}^{m}\beta_{i,j}^{v}b_{i',j'}^{G^{i,j}}+\sum_{j,k\in[m]}\gamma_{i,j,k}^{v}b_{i',j'}^{H^{i,j,k}}+b_{i',j'}^{F^{i,v}}\right)-\left(\sum_{j=1}^{m}\beta_{i,j}^{v}G^{i,j}(j',i')+\sum_{j,k\in[m]}\gamma_{i,j,k}^{v}H^{i,j,k}(j',i')+F^{i,v}(j',i')\right),$$

where the first term can be computed from the broadcasts of $P_{i'}$, and the second term is given as part of the leakage.

In addition, for every corrupt $P_{j'}$ with $\mathsf{flag}_{i',j'}=0$ in $\mathsf{vss}^{G^{i,j}}$, $\mathsf{vss}^{H^{i,j,k}}$ and $\mathsf{vss}^{F^{i,v}}$, the simulator gives the vector of coefficients $(\beta_{i,j}^{v},\gamma_{i,j,k}^{v},1)_{j,k\in[m]}$ to the adversary, as the corresponding leakage in $\mathcal{F}_{\mathsf{iSig}}$ where $P_{j'}$ is the dealer and $P_{i'}$ is the intermediary. (Otherwise, the vector of coefficients is set to be the all-zero vector.)

- If $\mathsf{flag}_{i',R}=1$ in $\mathsf{vst}^{F^{i,v}}$, then R is necessarily corrupt, and for every honest $P_{j'}$, let the corresponding output of $\mathcal{F}_{\mathsf{iSig}}$ where $P_{j'}$ is the dealer and $P_{i'}$ is the intermediary be

$$\left(\sum_{j=1}^{m}\beta_{i,j}^{v}r_{j',i'}^{G^{i,j}}+\sum_{j,k\in[m]}\gamma_{i,j,k}^{v}r_{j',i'}^{H^{i,j,k}}\right).$$

  Observe that the simulator holds all the random pads since R is corrupt.

  In addition, for every corrupt $P_{j'}$ with $\mathsf{flag}_{i',j'}=0$ in $\mathsf{vss}^{G^{i,j}}$, $\mathsf{vss}^{H^{i,j,k}}$ and $\mathsf{vss}^{F^{i,v}}$, the simulator gives the vector of coefficients $(\beta_{i,j}^{v},\gamma_{i,j,k}^{v})_{j,k\in[m]}$ to the adversary, as the corresponding leakage in $\mathcal{F}_{\mathsf{iSig}}$ where $P_{j'}$ is the dealer and $P_{i'}$ is the intermediary. (Otherwise, the vector of coefficients is set to be the all-zero vector.)

Consider any corrupt P_i and $v\in[\ell]$. Let $P_{i'}$ be an honest party, and denote by $(r_{j',i'}^{G^{i,j}}),(r_{j',i'}^{H^{i,j,k}}),(r_{j',i'}^{F^{i,v}})$ the random pads that $P_{j'}$ sent to $P_{i'}$ in the simulation of $\mathsf{vss}^{G^{i,j}}$, $\mathsf{vss}^{H^{i,j,k}}$ and $\mathsf{vss}^{F^{i,v}}$, respectively. (Observe that the simulator holds all of those values when P_i is corrupt.) Then the messages of $P_{i'}$ in the v-th linear computation of P_i are simulated as follows.

- If $\mathsf{flag}_{i',R}=0$ in $\mathsf{vst}^{F^{i,v}}$ and $\mathsf{flag}_{i'}=0$ in $(\mathsf{vss}^{G^{i,j}},\mathsf{vss}^{H^{i,j,k}})_{j,k\in[m]}$, then for every honest $P_{j'}$, let the corresponding output of $\mathcal{F}_{\mathsf{iSig}}$ where $P_{j'}$ is the dealer and $P_{i'}$ is the intermediary be

$$\left(\sum_{j=1}^{m}\beta_{i,j}^{v}r_{j',i'}^{G^{i,j}}+\sum_{j,k\in[m]}\gamma_{i,j,k}^{v}r_{j',i'}^{H^{i,j,k}}+r_{j',i'}^{F^{i,v}}\right).$$

  In addition, for every corrupt $P_{j'}$ with $\mathsf{flag}_{i',j'}=0$ in $\mathsf{vss}^{G^{i,j}}$, $\mathsf{vss}^{H^{i,j,k}}$ and $\mathsf{vss}^{F^{i,v}}$, the simulator gives the vector of coefficients $(\beta_{i,j}^{v},\gamma_{i,j,k}^{v},1)_{j,k\in[m]}$ to the adversary, as the corresponding leakage in $\mathcal{F}_{\mathsf{iSig}}$ where $P_{j'}$ is the dealer and $P_{i'}$ is the intermediary. (Otherwise, the vector of coefficients is set to be the all-zero vector.)

- If $\mathsf{flag}_{i',R}=1$ in $\mathsf{vst}^{F^{i,v}}$ and $\mathsf{flag}_{i'}=0$ in $(\mathsf{vss}^{G^{i,j}},\mathsf{vss}^{H^{i,j,k}})_{j,k\in[m]}$, then for every honest $P_{j'}$, let the corresponding output of $\mathcal{F}_{\mathsf{iSig}}$ where $P_{j'}$ is the dealer and $P_{i'}$ is the intermediary be

$$\left(\sum_{j=1}^{m}\beta_{i,j}^{v}r_{j',i'}^{G^{i,j}}+\sum_{j,k\in[m]}\gamma_{i,j,k}^{v}r_{j',i'}^{H^{i,j,k}}\right).$$

In addition, for every corrupt $P_{j'}$ with $\mathsf{flag}_{i',j'} = 0$ in $\mathsf{vss}^{G^{i,j}}$, $\mathsf{vss}^{H^{i,j,k}}$ and $\mathsf{vss}^{F^{i,v}}$, the simulator gives the vector of coefficients $(\beta^v_{i,j}, \gamma^v_{i,j,k})_{j,k\in[m]}$ to the adversary, as the corresponding leakage in $\mathcal{F}_{\mathsf{iSig}}$ where $P_{j'}$ is the dealer and $P_{i'}$ is the intermediary. (Otherwise, the vector of coefficients is set to be the all-zero vector.)

linop_2 **simulation.** Let $((i_1, v_1), (i_2, v_2)) \in S$. Consider any honest P_i and denote by $b^{i_k,v_k}_{i,j}$ the broadcast message of P_i in $\mathsf{vss}^{F^{i_k,v_k}}$, $k \in \{1, 2\}$. We split into cases.

- Assume that P_{i_1}, P_{i_2} are honest. Then for every honest P_j, the output of $\mathcal{F}_{\mathsf{iSig}}$ where P_j is the dealer and P_i is the intermediary is $(b^{i_1,v_1}_{i,j} - b^{i_2,v_2}_{i,j}) - (F^{i_1,v_1}(j, i) - F^{i_2,v_2}(j, i))$.

- Assume that P_{i_1} is corrupt and P_{i_2} is honest. If $\mathsf{flag}_i = 1$ in $\mathsf{vss}^{F^{i_1,v_1}}$, then P_i broadcasts "complaint:1". In addition, for every honest P_j let the corresponding output of $\mathcal{F}_{\mathsf{iSig}}$ where P_j is the dealer and P_i is the intermediary be $b^{i_2,v_2}_{i,j} - F^{i_2,v_2}(j, i)$.

  Otherwise, $\mathsf{flag}_i = 0$ in $\mathsf{vss}^{F^{i_1,v_1}}$. For every honest P_j, if $\mathsf{flag}_{i,j} = 1$ in $\mathsf{vss}^{F^{i_1,v_1}}$ then let the corresponding output of $\mathcal{F}_{\mathsf{iSig}}$ where P_j is the dealer and P_i is the intermediary be $b^{i_2,v_2}_{i,j} - F^{i_2,v_2}(j, i)$. Otherwise, if $\mathsf{flag}_{i,j} = 0$ in $\mathsf{vss}^{F^{i_1,v_1}}$, let the corresponding output of $\mathcal{F}_{\mathsf{iSig}}$ where P_j is the dealer and P_i is the intermediary be $r^{i_1,v_1}_{j,i} - (b^{i_2,v_2}_{i,j} - F^{i_2,v_2}(j, i))$, where $r^{i_1,v_1}_{j,i}$ is the random pad sent from P_j to P_i in the simulation of $\mathsf{vss}^{F^{i_1,v_1}}$.

  The case where P_{i_1} is honest and P_{i_2} is corrupt is symmetric.

- Assume that P_{i_1} and P_{i_2} are corrupt. If $\mathsf{flag}_i = 1$ in $\mathsf{vss}^{F^{i_1,v_1}}$ and in $\mathsf{vss}^{F^{i_2,v_2}}$ then P_i broadcasts "complaint:1,2".

  Otherwise, if $\mathsf{flag}_i = 1$ in $\mathsf{vss}^{F^{i_1,v_1}}$ but not in $\mathsf{vss}^{F^{i_2,v_2}}$ then P_i broadcasts "complaint:1". In addition, for every honest P_j let the corresponding output of $\mathcal{F}_{\mathsf{iSig}}$ where P_j is the dealer and P_i is the intermediary be $r^{i_2,j_2}_{j,i}$, where $r^{i_2,j_2}_{j,i}$ is the random pad sent from P_j to P_i in the simulation of $\mathsf{vss}^{F^{i_2,j_2}}$.

  Otherwise, if $\mathsf{flag}_i = 1$ in $\mathsf{vss}^{F^{i_2,j_2}}$ but not in $\mathsf{vss}^{F^{i_1,j_1}}$ then P_i broadcasts "complaint:2". In addition, for every honest P_j let the corresponding output of $\mathcal{F}_{\mathsf{iSig}}$ where P_j is the dealer and P_i is the intermediary be $r^{i_1,j_1}_{j,i}$, where $r^{i_1,j_1}_{j,i}$ is the random pad sent from P_j to P_i in the simulation of $\mathsf{vss}^{F^{i_1,j_1}}$.

  Otherwise $\mathsf{flag}_i = 0$ in $\mathsf{vss}^{F^{i_1,j_1}}$ and in $\mathsf{vss}^{F^{i_2,j_2}}$. For every honest P_j, if $\mathsf{flag}_{i,j} = 1$ in $\mathsf{vss}^{F^{i_1,j_1}}$ and $\mathsf{vss}^{F^{i_2,j_2}}$ then do nothing. If $\mathsf{flag}_{i,j} = 1$ only in $\mathsf{vss}^{F^{i_1,j_1}}$ then let the corresponding output of $\mathcal{F}_{\mathsf{iSig}}$ where P_j is the dealer and P_i is the intermediary be $r^{i_2,j_2}_{j,i}$. Similarly, if $\mathsf{flag}_{i,j} = 1$ only in $\mathsf{vss}^{F^{i_2,j_2}}$ then let the corresponding output of $\mathcal{F}_{\mathsf{iSig}}$ where P_j is the dealer and P_i is the intermediary be $r^{i_1,j_1}_{j,i}$. Otherwise, if $\mathsf{flag}_{i,j} = 0$ in $\mathsf{vss}^{F^{i_1,j_1}}$ and in $\mathsf{vss}^{F^{i_2,j_2}}$, we let the corresponding output of $\mathcal{F}_{\mathsf{iSig}}$ where P_j is the dealer and P_i is the intermediary be $r^{i_1,j_1}_{j,i} - r^{i_2,j_2}_{j,i}$.

For every honest P_i and corrupt P_j, the leakage of the $\mathcal{F}_{\mathsf{iSig}}$ instance in which P_j is the dealer and P_i is the intermediary is computed as follows. If $\mathsf{flag}_i = 1$ in $\mathsf{vss}^{F^{i_1,v_1}}$ or in $\mathsf{vss}^{F^{i_2,v_2}}$ then the

leakage is the all-zero string. Otherwise, $\mathsf{flag}_i = 0$ in $\mathsf{vss}^{F^{i_1,v_1}}$ and in $\mathsf{vss}^{F^{i_2,v_2}}$. If $\mathsf{flag}_{i,j} = 1$ in $\mathsf{vss}^{F^{i_1,v_1}}$ and in $\mathsf{vss}^{F^{i_2,v_2}}$, then the leakage is the all-zero string. Otherwise, if $\mathsf{flag}_{i,j} = 1$ in $\mathsf{vss}^{F^{i_1,v_1}}$ but not in $\mathsf{vss}^{F^{i_2,v_2}}$ then the leakage is the vector of coefficients (1). Otherwise, if $\mathsf{flag}_{i,j} = 1$ in $\mathsf{vss}^{F^{i_2,v_2}}$ but not in $\mathsf{vss}^{F^{i_1,v_1}}$ then the leakage is the vector of coefficients (1). Otherwise, if $\mathsf{flag}_{i,j} = 0$ in $\mathsf{vss}^{F^{i_1,v_1}}$ and in $\mathsf{vss}^{F^{i_2,v_2}}$ then the leakage is the vector of coefficients $(1,-1)$.

Completing linop_3 and linop_2. At this stage the simulator receives the messages from the corrupt parties to the honest parties, as well as the inputs to the $\mathcal{F}_{\mathsf{iSig}}$ instances where either the dealer or the intermediary (or both) is corrupt. For all of those instances that correspond to linop_3 and linop_2 the simulator holds all the inputs the $\mathcal{F}_{\mathsf{iSig}}$, and can compute the output accordingly. This concludes the simulation of the third round.

E.4.4 Communication with $\mathcal{F}_{\mathsf{sh\text{-}comp}}$

The simulator computes the set I like in the protocol. Consider a corrupt P_i. If P_i is in I, then the simulator inputs $\mathsf{flag}_i = 1$ to $\mathcal{F}_{\mathsf{sh\text{-}comp}}$. Otherwise $\mathsf{flag}_i = 0$, and P_i is not discarded in any vss instance. Let $(G^{i,j}(x,y), H^{i,j,k}(x,y), F^{i,v}(x,y))_{j,k\in[m],v\in[\ell]}$ be the polynomials defined by the shares of the honest parties in $(\mathsf{vss}^{G^{i,j}}, \mathsf{vss}^{H^{i,j,k}}, \mathsf{vss}^{F^{i,v}})_{j,k\in[m],v\in[\ell]}$. The simulator inputs $\mathsf{flag}_i, (G^{i,j}(x,y), H^{i,j,k}(x,y), F^{i,v}(x,y))_{j,k\in[m],v\in[\ell]}$ as the input of P_i.

E.4.5 Round 4 Simulation: Opening Phase

For every honest P_i the simulator receives $(\mu_i, (j_{i,k}, v_{i,k}, F^{j_{i,k},v_{i,k}}(x,y))_{k\in[\mu_i]}))$. For every honest P_i the simulator broadcasts $(\mu_i, (j_{i,k}, v_{i,k})_{k\in[\mu_i]}))$ on behalf of P_i. In addition, for every honest P_i and every $k \in \mu_i$ the simulator does as follows.

Honest $P_{j_{i,k}}$. Assume that $P_{j_{i,k}}$ is honest. Then for every honest $P_{i'}$ so that $P_{i'}$ is not in $W \cup W' \cup U \cup B$ in $\mathsf{vst}^{F^{j_{i,k},v_{i,k}}}$, the simulator returns $(b_{i',j'} - F^{j_{i,k},v_{i,k}}(j',i'))_{j'\in[n]}$ as the output of the $\mathcal{F}_{\mathsf{iSig}}$ instance in which $P_{i'}$ is the dealer and P_i is the intermediary, where $b_{i',j'}$ is the broadcast of $P_{i'}$ in $\mathsf{vst}^{F^{j_{i,k},v_{i,k}}}$. In addition, for every corrupt $P_{i'}$ so that P_i is not in $W \cup W' \cup U \cup B$ in $\mathsf{vst}^{F^{j_{i,k},v_{i,k}}}$, the simulator gives the vector of coefficients (1) to the adversary as the leakage of the $\mathcal{F}_{\mathsf{iSig}}$ instance in which $P_{i'}$ is the dealer and P_i is the intermediary.

Corrupt $P_{j_{i,k}}$. Assume that $P_{j_{i,k}}$ is corrupt. Then for every honest $P_{i'}$ so that $P_{i'}$ is not in $W \cup W' \cup U \cup B$ in $\mathsf{vst}^{F^{j_{i,k},v_{i,k}}}$, the simulator returns $(\rho'_{i',j'})_{j'\in[n]}$ as the output of the $\mathcal{F}_{\mathsf{iSig}}$ instance in which $P_{i'}$ is the dealer and P_i is the intermediary, where $\rho'_{i',j',R}$ is the message that $P_{i'}$ sent to R in the execution of $\mathsf{vst}^{F^{j_{i,k},v_{i,k}}}$. In addition, for every corrupt $P_{i'}$ so that P_i is not in $W \cup W' \cup U \cup B$ in $\mathsf{vst}^{F^{j_{i,k},v_{i,k}}}$, the simulator gives the vector of coefficients (1) to the adversary as the leakage of the $\mathcal{F}_{\mathsf{iSig}}$ instance in which $P_{i'}$ is the dealer and P_i is the intermediary.

At this stage the simulator receives the messages from the corrupt parties to the honest parties, as well as the inputs to the $\mathcal{F}_{\mathsf{iSig}}$ instances where either the dealer or the intermediary (or both) is corrupt. For all of those instances the simulator holds all the inputs the $\mathcal{F}_{\mathsf{iSig}}$, and can compute the output accordingly.

E.4.6 Communication with $\mathcal{F}_{\mathsf{sh\text{-}comp}}$

For every corrupt P_i so that $i \notin I$ at the beginning of Round 4, let $(\mu_i, (j_{i,k}, v_{i,k}, F^{j_{i,k},v_{i,k}}(x,y))_{k\in[\mu_i]}))$ be the broadcast of P_i. If the output of the opening phase of some linop_3 execution where P_i is the receiver ended with $\perp$, then the simulator inputs $\mathsf{abort}_i = 1$. Otherwise, the simulator inputs $(\mu_i, (j_{i,k}, v_{i,k})_{k\in[\mu_i]}))$ as the input of P_i. This concludes the simulation.

E.5 Analysis of the sh-comp Simulator

E.5.1 Analysis: Round 1 Simulation

At the beginning of the round the environment picks the inputs to the honest parties in the same way in both worlds. Fix those inputs. We begin by analysing the internal vss executions in the various tss and vst executions. Fix any order to the vss instances and consider the i-th instance. Then it is not hard to see that the internal randomness used to generate the messages of the honest parties in the i-th vss instance is independent of the messages of the honest parties in the previous instances. Therefore we can analyse each instance separately. Consider any instance of vss, and let us split into cases.

Corrupt P_d. Assume that dealer P_d is corrupt. In this case, the vss instance is executed in the same way in both worlds, so the view is the same in both worlds, as required.

Honest P_d and corrupt receiver in vst. Assume that dealer P_d is honest, and the vss instance is executed as part of a vst execution in which the receiver is corrupt. Then the vss instance is executed in the same way in both worlds, so the view is the same in both worlds, as required.

Honest P_d. Assume that the dealer P_d is honest, and the vss execution is not part of a vst execution in which the receiver is corrupt. Denote the real-world sharing polynomial by $f(x,y)$. We start by analysing the inputs $(i, f_i(x))_{i\in\mathsf{C}}$ to $\mathsf{Sim}^{\mathsf{H}}_{\mathsf{vss}}$, and show that those inputs have the same distribution as the real-world values $(i, f(x,i))_{i\in\mathsf{C}}$. Indeed, if the execution corresponds to the sharing of $F^{i,j}(x,y)$, $G^{i,j}(x,y)$ or $H^{i,j,k}$ for some $i \in [n]$ and $j,k \in [m]$, then $f(x,y)$ is fixed and equal to $F^{i,j}(x,y)$, $G^{i,j}(x,y)$ or $H^{i,j,k}$, respectively, and it is not hard to verify that the claim holds. Otherwise the execution is an internal vss-simulation inside the simulation of tss. In this case, the claim holds by Fact B.3. Fix the inputs $(i, f_i(x))_{i\in\mathsf{C}}$. Then, in both worlds the corresponding outputs of $\mathcal{F}_{\mathsf{iSig}}$ where P_d is the dealer and a corrupt P_i is the intermediary are $(f_i(j))_{j\in\mathsf{C}}$. In addition, for every honest P_i and corrupt P_j the random pads $r_{i,j}$ are uniformly distributed, so the output of $\mathcal{F}_{\mathsf{iSig}}$ where P_d is the dealer and a corrupt P_i is the intermediary is $r_{i,j}$.

Fix the view of the adversary in all vss instances. Since the first round of tss consists only of vss calls, we conclude that the adversary's view in tss is fixed as well. We continue by analysing the vst instances. Fix any order to the vst instances and consider the i-th instance. Then it is not hard to see that the internal randomness used to generate the messages of the honest parties in the i-th vst instance is independent of the messages of the honest parties in the previous instances. Therefore we can analyse each instance separately. Consider any instance of vst, and let us split into cases.

Honest P_d and R. Assume that the dealer P_d and the receiver R are honest. Denote the real-world sharing polynomial by $f(x, y)$. Then the same analysis as in the vss cast shows that the inputs $(i, f_i(x))_{i\in\mathsf{C}}$ to $\mathsf{Sim}^{\mathsf{H}}_{\mathsf{vss}}$ are fixed and equal to $(i, f(x, i))_{i\in\mathsf{C}}$. In addition, the random pads $(\rho_{R,i,j})_{i\in\mathsf{C},j\in[n]}$ are uniformly distributed in both worlds, so the output of $\mathcal{F}_{\mathsf{iSig}}$ where R is the dealer and a corrupt P_i is the intermediary, that is $(\rho_{R,i,j})_{j\in[n]}$, has the same distribution in both worlds.

Honest P_d, corrupt R. Assume that P_d is honest and R is corrupt. Denote the real-world sharing polynomial by $f(x, y)$, and observe that $f(x, y)$ is the input to vst. Therefore, it is not hard to verify that the vst instance is executed in the same way in both worlds, so the view is the same in both worlds, as required.

Corrupt P_d. Assume that the dealer P_d is corrupt. In this case, the vst instance is executed in the same way in both worlds, so the view is the same in both worlds, as required.

This concludes the analysis of the first round.

E.5.2 Analysis: Round 2 Simulation

Fix any first-round view of the adversary. We begin by analysing the second-round simulation of the internal vss executions in the various tss and vst executions. As before, the messages of the honest parties in the i-th vss instance are independent of the messages of the honest parties in previous instances. Therefore we can analyse each instance separately. Consider any instance of vss, and let us split into cases.

Corrupt P_d. Assume that dealer P_d is corrupt. In this case, the vss instance is executed in the same way in both worlds, so the view is the same in both worlds, as required.

Honest P_d and corrupt receiver in vst. Assume that dealer P_d is honest, and the vss instance is executed as part of a vst execution in which the receiver is corrupt. Then the vss instance is executed in the same way in both worlds, so the view is the same in both worlds, as required.

Honest P_d. Assume that the dealer P_d is honest, and the vss execution is not part of a vst execution in which the receiver is corrupt. As before, we denote the real-world sharing polynomial by $f(x, y)$ and the inputs to the simulator by $(i, f_i(x))_{i\in\mathsf{C}}$, where we saw that $f_i(x) = f(x, i)$ for every $i \in \mathsf{C}$. For every honest P_i and P_j, the random variables $a_{i,j}$ and $b_{i,j}$ are uniformly distributed, and it holds that $a_{j,i} = b_{i,j}$ and $b_{j,i} = a_{i,j}$. Therefore, the broadcast messages $(a_{i,j}, a_{j,i}, b_{i,j}, b_{j,i})$ have the same distribution in both worlds. Fix those messages. In addition, in both worlds the broadcast messages $(a^d_{i,j})_{i\in\mathsf{H},j\in[n]}$ are fixed and equal to $(a_{i,j})_{i\in\mathsf{H},j\in[n]}$.

For every honest P_i and corrupt P_j, the real-world broadcast message $a_{i,j}$ is set to be $f(j, i) + r_{i,j} = f_j(i) + r_{i,j}$, so it is equal to the ideal-world broadcast message. Similarly, the real-world broadcast message $b_{i,j}$ is set to be $f(j, i) + r'_{j,i} = f_j(i) + r'_{j,i}$, so it is equal to the ideal-world broadcast message. Finally, for every corrupt P_i and $j \in [n]$, the real-world broadcast message $a_{i,j}$ is set to be $f(j, i) + r^d_{i,j} = f_j(i) + r^d_{i,j}$, so it is equal to the ideal-world broadcast message.

Fix the view of the adversary in all vss instances. We continue by analysing the second round of all tss instances, and by the same reasoning as before, we can analyse each instance separately. Observe that the second round of tss consists only of vss calls and challenge generation. Therefore, the only additional messages sent by the honest parties are the challenges $(\beta_{Q,i})_{Q,i}$, where we enumerate over all sets Q where P_Q is honest and Q contains a corrupt party, and all $i \in [n]$. In both worlds $(\beta_{Q,i})_{Q,i}$, are uniformly distributed. Fix those challenges, so the view in the tss execution is fixed as well. We continue by analysing the vst instances and by the same reasoning as before, we can analyse each instance separately. We split into cases.

Honest P_d and R. Assume that P_d and R are honest. As before, denote the real-world sharing polynomial by $f(x, y)$, and the inputs to the simulator by $(i, f_i(x))_{i\in\mathsf{C}}$, where we saw that $f_i(x) = f(x, i)$ for every $i \in \mathsf{C}$. For every honest P_i and every $j \in [n]$, the random variables $\alpha_{i,j,R}$ and $\beta_{i,j,R}$ are uniformly distributed, and it holds that $\alpha_{R,i,j} = \beta_{i,j,R}$ and $\beta_{R,i,j} = \alpha_{i,j,R}$. Fix those messages. In addition, in both worlds the broadcast messages $(\alpha^d_{i,j,R}, \alpha^d_{R,i,j})_{i\in\mathsf{H},j\in[n]}$ are fixed and equal to $(\alpha_{i,j,R}, \alpha_{R,i,j})_{i\in\mathsf{H},j\in[n]}$.

For every corrupt P_i, the real-world broadcast message $\alpha_{R,i,j}$ is set to be $f(j, i) + \rho_{R,i,j} = f_j(i) + \rho_{R,i,j}$, so it is equal to the ideal-world broadcast message. Similarly, $\beta_{R,i,j}$ is set to be $f(j, i) + \rho'_{i,j,R} = f_j(i) + \rho'_{i,j,R}$, so it is equal to the ideal-world broadcast message. Moreover, the broadcast messages $(\alpha^d_{R,i,j})_{i\in\mathsf{C},j\in[n]}$ are fixed and equal to $(\alpha_{R,i,j})_{i\in\mathsf{C},j\in[n]}$. Finally, in both worlds, for every corrupt P_i and every $j \in [n]$ the broadcast message $\alpha^d_{i,j,R}$ is fixed and equal to $f(j, i) + \rho^d_{i,j,R} = f_i(j) + \rho^d_{i,j,R}$.

Other cases. In all other cases, the vst instance is executed in the same way in both worlds, so the view is the same in both worlds, as required.

This concludes the analysis of the second round.

E.5.3 Analysis: Round 3 Simulation

Fix any second-round view of the adversary. It is not hard to see that the flags of the honest parties to each vss, tss and vst instance are computed in the same way in both worlds. We continue by analysing the third-round simulation of the internal vss executions in the various tss and vst executions, and by the same reasoning as before, we can analyse each instance separately. We split into cases.

Corrupt P_d. Assume that dealer P_d is corrupt. In this case, the vss instance is executed in the same way in both worlds, so the view is the same in both worlds, as required.

Honest P_d and corrupt receiver in vst. Assume that dealer P_d is honest, and the vss instance is executed as part of a vst execution in which the receiver is corrupt. Then the vss instance is executed in the same way in both worlds, so the view is the same in both worlds, as required.

Honest P_d. Assume that the dealer P_d is honest, and the vss execution is not part of a vst execution in which the receiver is corrupt. As before, we denote the real-world sharing polynomial by $f(x,y)$ and the inputs to the simulator by $(i, f_i(x))_{i\in\mathsf{C}}$, where we saw that $f_i(x) = f(x,i)$ for every $i \in \mathsf{C}$. For an honest P_i we denote the flags by $\mathsf{flag}_i, (\mathsf{flag}_{i,j})_{j\in[n]}$ and we denote the flags of P_d by $(\mathsf{flag}_i^d)_{i\in[n]}$.

First, we observe that for every honest P_i it holds that $\mathsf{flag}_i^d = 0$. Indeed, P_d is not *iSign-conflicted* with P_i, or internally-vss-conflicted with P_i in any vss execution in which P_d is the dealer, or internally vst-conflicted with P_i in any vst execution in which P_d is the dealer, since both are honest. A similar reasoning shows that for every honest P_i it holds that $\mathsf{flag}_i = 0$, and that for every pair of honest parties P_i, P_j it holds that $\mathsf{flag}_{i,j} = \mathsf{flag}_{j,i} = 0$. In addition, observe that if P_d is internally-vss-conflicted with some corrupt P_i then $\mathsf{flag}_i^d = 1$, and that if an honest P_j is internally-vss-conflicted with P_i then $\mathsf{flag}_{j,i} = 1$. Therefore, the broadcasts of the flags of the honest parties are the same in both worlds.

Consider any corrupt P_i with $\mathsf{flag}_i^d = 1$. Then in both worlds for every honest P_j the value $r_{j,i}$ is opened in any $\mathcal{F}_{\mathsf{iSig}}$ instance in which P_j is the dealer and P_d is the intermediary; if $\mathsf{flag}_i^d = 0$ then no value is opened. Similarly, for every honest P_i and corrupt P_j so that $\mathsf{flag}_{i,j} = 1$ the value $f_j(i) = f_i(j)$ is opened in the $\mathcal{F}_{\mathsf{iSig}}$ instance in which P_d is the dealer and P_i is the intermediary; if $\mathsf{flag}_{i,j} = 0$ then no value is opened.

In the rest of the $\mathcal{F}_{\mathsf{iSig}}$ instances, where either the dealer or the intermediary or both are corrupt, the simulator holds all the inputs to $\mathcal{F}_{\mathsf{iSig}}$ and the computation of the outputs is done in the same way in both worlds.

Fix the view of the adversary in all vss instances. We continue with the analysis of the tss. By the same reasoning as before, we can analyse each instance separately. We split into cases.

Honest P_d. First, observe that the values $(F^{v,k}(x,i))_{v\in\{a,b,c\},k\in\{0,\ldots,2d\},i\in\mathsf{C}}$ are fixed and the same in both worlds, where $F^{a,k}(x,y) = F^{b,k}(x,y) = 0$ for $k > d$.

Like in the case of vss, it is not hard to verify that for every honest P_i it holds that $\mathsf{flag}_i^d = \mathsf{flag}_i = 0$, that for every pair of honest parties P_i, P_j it holds that $\mathsf{flag}_{i,j} = \mathsf{flag}_{j,i} = 0$. In addition, if P_d is internally-vss-conflicted with some corrupt P_i then $\mathsf{flag}_i^d = 1$, and that if an honest P_j is internally-vss-conflicted with P_i then $\mathsf{flag}_{j,i} = 1$.

Consider any set Q and honest P_i in Q. If P_Q is corrupt than the challenges that P_Q sent to P_i were chosen by the adversary in Round 2. If P_Q is honest and Q contains a corrupt party, then the challenges $(\beta_{Q,j})_{j\in\mathsf{C}}$ were already fixed in the second round. Otherwise, if P_Q is honest and Q does not contain a corrupt party, then the challenges $(\beta_{Q,j})_{j\in\mathsf{C}}$ are uniformly distributed in both worlds, and we fix them. This fixes all the challenges that the adversary sees, and we denote those challenges by $\alpha_1, \ldots, \alpha_q$ for $q \le d$. We need the following lemma, that was proved in [AKP22, Lemma H.2].

Lemma E.1. *Let $\mathbb{F}$ be a field, let $q \le d$ be some positive integers with $2d < |\mathbb{F}|$, and let $\alpha_1, \ldots, \alpha_q \in \mathbb{F}$ be distinct non-zero elements. Let n and $t < n/2$ be positive integers, and let $\mathsf{C} \subseteq \{1, \ldots, n\}$ be a set of size at most t. Let $P(x)$ be a degree-$2d$ polynomial, and let $\bar{F}^0(x,y)$ and $\bar{F}^{d+1}(x,y), \ldots, \bar{F}^{2d}(x,y)$ be symmetric bivariate polynomials of degree at most t in each variable, such that $\bar{F}^k(0,0) = P^k$ for $k \in \{0, d+1, \ldots, 2d\}$, where P^k is the k-th coefficient of $P(x)$. For $k \in \{0, d+1, \ldots, 2d\}$, and $i \in \mathsf{C}$ let $f_i^k(x) := \bar{F}^k(x,i)$. Let $\{f_i^k(x)\}_{k\in\{1,\ldots,d\},i\in\mathsf{C}}$ be a set of degree-t polynomials, such that $\bar{f}_i^k(j) = \bar{f}_j^k(i)$ for all $i, j \in \mathsf{C}$ and $k \in \{1, \ldots, d\}$.*

Let $F^0(x,y),\ldots,F^{2d}(x,y)$ 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 $k\in\{d+1,\ldots,2d\}$, (2) $F^k(x,i)=f_i^k(x)$ for all $k\in\{1,\ldots,d\}$ 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 $k\in\{1,\ldots,d\}$.

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

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

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

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

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

Consider the real-world random variables $(A(\alpha_i),B(\alpha_i),C(\alpha_i))_{i\in[q]}$ and observe that $(A(\alpha_i),B(\alpha_i))_{i\in[q]}$ are uniformly distributed, and that $C(\alpha_i)=A(\alpha_i)\cdot B(\alpha_i)$ for every $i\in[q]$. Since $q\leq d$, and by Fact B.1 the ideal-world random variables $(A(\alpha_i),B(\alpha_i),C(\alpha_i))_{i\in[q]}$ have the same distribution. Fix those random variables. By Lemma E.1 the random variables

$$\left(G^{v,j}(x,y):=\sum_{k=0}^{2d}\alpha_j^k\cdot F^{v,k}(x,y)\right)_{v\in\{a,b,c\},j\in[q]}$$

have the same distribution in both worlds, where $F^{a,k}(x,y)=F^{b,k}(x,y)=0$ for $k>d$. Conditioned on those values, it is not hard to verify that the computation in linop_1 is done in the same way in both worlds.

Corrupt P_d. Assume that P_d is corrupt. In this case, the tss instance is executed in the same way in both worlds, so the view is the same in both worlds, as required.

Fix the view of the adversary in all tss instances as well. We continue with the analysis of the vst. Again, we can analyse each instance separately. We split into cases.

Honest P_d,R. Assume that P_d and R are honest. Like in the case of vss and tss, one can verify that $\mathsf{flag}_d^R=\mathsf{flag}_R^d=0$, that for every honest P_i it holds that $\mathsf{flag}_i^d=\mathsf{flag}_i^R=\mathsf{flag}_{i,d}=\mathsf{flag}_{i,R}=0$, and that for every pair of honest parties P_i,P_j it holds that $\mathsf{flag}_{i,j}=\mathsf{flag}_{j,i}=0$. In addition, if P_d is internally-vss-conflicted with some corrupt P_i then $\mathsf{flag}_i^d=1$, and that if an honest P_j is internally-vss-conflicted with P_i then $\mathsf{flag}_{j,i}=1$. Therefore, the broadcasts of the flags of the honest parties are the same in both worlds.

Consider a corrupt P_i with $\mathsf{flag}_i^d=1$. In the real world the values $(\rho_{R,j,i}^d)_{j\in[n]}$ are opened in the $\mathcal{F}_{\mathsf{iSig}}$ instance where R is the dealer and P_d is the intermediary, and since $\beta_{R,i,j}^d=f_i(j)+\rho_{R,j,i}^d$ it is not hard to see that the same values are opened in the ideal world as well. Similarly, consider a corrupt P_i with $\mathsf{flag}_i^R=1$. The for every honest P_j, in the real world the values $(\rho_{j,i,R}^d)$ are opened in the $\mathcal{F}_{\mathsf{iSig}}$ instance where P_j is the dealer and R is the intermediary, and since $\beta_{R,i,j}=f_i(j)+\rho_{j,i,R}$,

the same values are opened in the ideal world as well. In addition, in both worlds the values $(f_i(k))_{k\in[n]}$ are opened in both worlds in the $\mathcal{F}_{\mathsf{iSig}}$ instance where P_d is the dealer and R is the intermediary.

In the rest of the $\mathcal{F}_{\mathsf{iSig}}$ instances, where either the dealer or the intermediary or both are corrupt, the simulator holds all the inputs to $\mathcal{F}_{\mathsf{iSig}}$ and the computation of the outputs is done in the same way in both worlds.

Other cases. In all other cases, the vst instance is executed in the same way in both worlds, so the view is the same in both worlds, as required.

We continue with the analysis of linop_3. Consider any honest P_i and $v \in [\ell]$. Then in the real world, for every honest $P_{i'}$ with $\mathsf{flag}_{i',R} = 0$ in $\mathsf{vst}^{F^{i,v}}$, and every $j' \in \mathsf{H}$ it holds that

$$\left(\sum_{j=1}^{m} \beta_{i,j}^{v} b_{i',j'}^{G^{i,j}} + \sum_{j,k\in[m]} \gamma_{i,j,k}^{v} b_{i',j'}^{H^{i,j,k}} + b_{i',j'}^{F^{i,v}}\right)$$
$$= \left(\sum_{j=1}^{m} \beta_{i,j}^{v} (G^{i,j}(j',i') + r_{j',i'}^{G^{i,j}}) + \sum_{j,k\in[m]} \gamma_{i,j,k}^{v} (H^{i,j,k}(j',i') + r_{j',i'}^{H^{i,j,k}}) + (F^{i,v}(j',i') + r_{j',i'}^{F^{i,v}})\right).$$

Therefore, the simulator perfectly simulates the output of $\mathcal{F}_{\mathsf{iSig}}$ for every honest dealer $P_{j'}$ and honest intermediary $P_{i'}$. If $\mathsf{flag}_{i',R} = 1$ then R is corrupt, and then the output the $\mathcal{F}_{\mathsf{iSig}}$ is computed in the same way in both worlds. Similarly, if P_i is corrupt then the output the $\mathcal{F}_{\mathsf{iSig}}$ is computed in the same way in both worlds. Finally, one can verify that in all cases the leakage to the adversary is computed exactly like in the real world. This concludes the analysis of linop_3. The analysis of linop_2 follows in a similar way, by noting that, in the real world, for every honest P_i, every $v \in [\ell]$, and every honest P_j, P_k it holds that $b_{j,k}^{F^{i,v}} = F^{i,v}(k,j) + r_{k,j}^{F^{i,v}}$. This concludes the analysis of the third round of the simulation.

E.5.4 Analysis: Output of Honest Parties

Consider an honest P_i. Then, by Lemma 6.5 P_i is never discarded as a dealer in vss, by the analysis in Section 7.1.2 P_i is not discarded as a dealer in tss, and by Lemma 7.3 P_i is not discarded as a dealer in vst. In addition, it is not hard to verify that all the assumptions of $\mathsf{linop}_1, \mathsf{linop}_2$ and linop_3 are satisfied, so by the above lemmas, together with Lemmas 7.1, Lemma 8.2 and Lemma 8.3 it holds that the set B contains no honest parties. Therefore, by Lemma 6.5, in the real world every honest $P_{i'}$ outputs $(F^{i,v}(x,i'), G^{i,j}(x,i'), H^{i,j,k}(x,i'))_{j,k\in[m],v\in[\ell]}$, just like in the ideal world. In addition, by the correctness of linop_3 (Lemma 8.3), the recovered values that correspond to the linear operations on the inputs of P_i are exactly $(\sum_{j=1}^{m} \beta_{i,j}^{v} G^{i,j}(x,y) + \sum_{j,k\in[m]} \gamma_{i,j,k}^{v} H^{i,j,k}(x,y) + F^{i,v}(x,y))_{v\in[\ell]}$, as required.

Consider now a corrupt P_i. By the analysis in Section 7.1.2, the probability that there exists a tss instance in which P_i is the dealer, P_i is not discarded by some honest party, and $F^{a,0}(0,0) \cdot F^{b,0}(0,0) \neq F^{c,0}(0,0)$ is at most $2m^2nd/(|\mathbb{F}|-1)$. Taking a union bound over all corrupt parties, we conclude that the probability that there exists a tss instance in which a corrupt party is the dealer, the dealer is not discarded by some honest party, and $F^{a,0}(0,0) \cdot F^{b,0}(0,0) \neq F^{c,0}(0,0)$ is at most $2m^2n^2d/(|\mathbb{F}|-1) \leq 2^{-\kappa}$. Fix any execution for which this even does not occur. Then it

is not hard to see that in both worlds the set I is the same. In addition, by Lemma 6.5, for every corrupt P_i so that $i \notin I$ it holds that the shares of the honest parties in P_i's polynomials are the same in both worlds. Finally, by the correctness of linop_2 (Lemma 8.2) it follows that for every $((i_1, v_1), (i_2, v_2)) \in S$ so that $i_1, i_2 \notin I$, the output of linop_2 is $F^{i_1,v_1}(x, y) - F^{i_2,v_2}(x, y)$, as required.

E.5.5 Analysis: Round 4 Simulation

By Lemma 7.3, in every vst execution where R is honest, and every $P_i, P_j \notin B$, the (i, j)-th share is $F'(j, i)$, where $F'(x, y)$ is the polynomial defined by the shares of the honest parties in the underlying vss execution (and $F'(x, y) = F(x, y)$ if the dealer is honest). Therefore, both in the real-world and in the simulation the shares are revealed in the same way, as required. In addition, by Lemma 7.3 in every vst execution where R is corrupt, R either opens $F'(x, y)$ or $\perp$, where $F'(x, y)$ is the polynomial defined by the shares of the honest parties in the underlying vss execution. Therefore, the outputs of the honest parties are the same in both worlds. This concludes the analysis of the simulator of sh-comp.

F Augmented Single Input Functionality

In this section, we prove that protocol asif UC-emulates $\mathcal{F}_{\mathsf{asif}}$ with statistical security in the $\mathcal{F}_{\mathsf{sh\text{-}comp}}$-hybrid model. From the composition properties of UC-security, this implies that protocol asif UC-emulates $\mathcal{F}_{\mathsf{asif}}$.

Let $\mathcal{A}$ be the dummy adversary. We define the simulator as follows. The simulator 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 parties C.

F.1 The Simulator

Round 1. The simulator does as follows on behalf of every honest P_i.

- Samples a random symmetric bivariate polynomial $F^{i,i',v}(x, y)$ of degree at most t in each variable, for every $i' \in [n]$ and $v \in [\ell]$
- Samples a random symmetric bivariate polynomial $G^{i,j}(x, y)$ of degree at most t in each variable, conditioned on $G^{i,j}(0, 0) = 0$, for every $j \in [m]$.
- Samples a random symmetric bivariate polynomial $H^{i,j,k}(x, y)$ of degree at most t in each variable, conditioned on $H^{i,j}(0, 0) = 0$, for every $j, k \in [m]$.

The simulator gives the adversary the values $(F^{i,i',v}(x, j'), G^{i,j}(x, j'), H^{i,j,k}(x, j'))_{i\in\mathsf{H},i'\in[n],v\in[\ell],j,k\in[m],j'\in\mathsf{C}}$ as well as $(F^{i,i',v}(x, y))_{i\in\mathsf{H},i'\in\mathsf{C},v\in[\ell]}$ as the leakage from $\mathcal{F}_{\mathsf{sh\text{-}comp}}$.

Round 2. The simulator does nothing.

Round 3. The simulator receives the values $(f_{i,j}(\mathbf{x}_i))_{i\in\mathsf{H},j\in\mathsf{C}}$ as a leakage from $\mathcal{F}_{\mathsf{asif}}$, as well as (1) $f_{i_1,k,v_1}(\mathbf{x}_{i_1}) - f_{i_2,k,v_2}(\mathbf{x}_{i_2})$ for every $(k,(i_1,v_1),(i_2,v_2)) \in S$ so that P_{i_1} and P_{i_2} are honest, (2) $f_{i_1,k,v_1}(\mathbf{x}_{i_1})$ for every $(k,(i_1,v_1),(i_2,v_2)) \in S$ so that P_{i_1} is honest and P_{i_2} is corrupt, and (3) $f_{i_2,k,v_2}(\mathbf{x}_{i_2})$ for every $(k,(i_1,v_1),(i_2,v_2)) \in S$ so that P_{i_2} is honest and P_{i_1} is corrupt.

The simulator samples symmetric bivariate polynomials of degree at most t in each variables $(\bar{F}^{i,i',v}(x,j'), \bar{G}^{i,j}(x,j'), \bar{H}^{i,j,k}(x,j'))_{i\in\mathsf{H},i'\in[n],v\in[\ell],j,k\in[m],j'\in\mathsf{C}}$ conditioned on

1. $(\bar{F}^{i,i',v}(x,j') = F^{i,i',v}(x,j'))_{i\in\mathsf{H},i'\in[n],v\in[\ell],j'\in\mathsf{C}}$,
2. $(\bar{G}^{i,j}(x,j') = G^{i,j}(x,j'))_{i\in\mathsf{H},j\in[m],j'\in\mathsf{C}}$,
3. $(\bar{H}^{i,j,k}(x,j') = H^{i,j,k}(x,j'))_{i\in\mathsf{H},j,k\in[m],j'\in\mathsf{C}}$,
4. $(\bar{F}^{i,i',v}(x,y) = F^{i,i',v}(x,y))_{i\in\mathsf{H},i'\in\mathsf{C},v\in[\ell]}$,
5. For every $(k',(i_1,v_1),(i_2,v_2)) \in S$ so that P_{i_1} and P_{i_2} are honest, the value
$$\left(\sum_{j=1}^{m}\beta^{v_1}_{i_1,k',j}\bar{G}^{i_1,j}(0,0) + \sum_{j,k\in[m]}\gamma^{v_2}_{i_1,k',j,k}\bar{H}^{i_1,j,k}(0,0)\right) - \left(\sum_{j=1}^{m}\beta^{v_2}_{i_2,k',j}\bar{G}^{i_2,j}(0,0) + \sum_{j,k\in[m]}\gamma^{v_2}_{i_2,k',j,k}\bar{H}^{i_2,j,k}(0,0)\right)$$
is equal to $f_{i_1,k,v_1}(\mathbf{x}_{i_1}) - f_{i_2,k,v_2}(\mathbf{x}_{i_2})$,
6. For every $(k',(i_1,v_1),(i_2,v_2)) \in S$ so that P_{i_1} is honest and P_{i_2} is corrupt, the value
$$\left(\sum_{j=1}^{m}\beta^{v_1}_{i_1,k',j}\bar{G}^{i_1,j}(0,0) + \sum_{j,k\in[m]}\gamma^{v_2}_{i_1,k',j,k}\bar{H}^{i_1,j,k}(0,0)\right)$$
is equal to $f_{i_1,k,v_1}(\mathbf{x}_{i_1})$,
7. For every $(k',(i_1,v_1),(i_2,v_2)) \in S$ so that P_{i_1} is corrupt and P_{i_2} is honest, the value
$$\left(\sum_{j=1}^{m}\beta^{v_2}_{i_2,k',j}\bar{G}^{i_2,j}(0,0) + \sum_{j,k\in[m]}\gamma^{v_2}_{i_2,k',j,k}\bar{H}^{i_2,j,k}(0,0)\right)$$
is equal to $f_{i_2,k,v_2}(\mathbf{x}_{i_2})$.

The simulator gives the following values to the adversary as the leakage from $\mathcal{F}_{\mathsf{sh\text{-}comp}}$: the values
$$\left(\mathsf{Out}^{i,i',v}(x,y) := \sum_{j=1}^{m}\beta^{v}_{i,i',j}G^{i,j}(x,y) + \sum_{j,k\in[m]}\gamma^{v}_{i,i',j,k}H^{i,j,k}(x,y) + F^{i,i',v}(x,y)\right)_{i\in\mathsf{H},i'\in[n],v\in[\ell]}$$
and (1) the polynomial $F^{i_1,k,v_1}(x,y) - F^{i_2,k,v_2}(x,y)$ for every $(k,(i_1,v_1),(i_2,v_2)) \in S$ so that P_{i_1} and P_{i_2} are honest, (2) the polynomial $F^{i_1,k,v_1}(x,y)$ for every $(k,(i_1,v_1),(i_2,v_2)) \in S$ so that P_{i_1} is honest and P_{i_2} is corrupt, and (3) the polynomial $F^{i_2,k,v_2}(x,y)$ for every $(k,(i_1,v_1),(i_2,v_2)) \in S$ so that P_{i_1} is corrupt and P_{i_2} is honest.

Communication with $\mathcal{F}_{\mathsf{sh\text{-}comp}}$. At this stage the simulator receives the inputs of the honest parties to $\mathcal{F}_{\mathsf{sh\text{-}comp}}$. For every corrupt P_i the simulator does as follows. If $\mathsf{flag}_i = 1$, or if $(F^{i,i',v}(x,j'), G^{i,j}(x,j'), H^{i,j,k}(x,j'))_{i'\in[n],v\in[\ell],j,k\in[m],j'\in\mathsf{C}}$ are not all symmetric bivariate polynomials of degree at most t in each variable, or if there exist $j,k \in [m]$ so that $G^{i,j}(0,0) \cdot G^{i,k}(0,0) \neq H^{i,j,k}(0,0)$ then the simulator inputs $\mathsf{flag}_i = 1$ to $\mathcal{F}_{\mathsf{asif}}$ (the rest of the inputs do not matter). Otherwise the simulator sets $\mathbf{x}_i[j] = G^{i,j}(0,0)$ for all $j \in [m]$, and inputs $\mathsf{flag}_i = 0$ and $\mathbf{x}_i$ to $\mathcal{F}_{\mathsf{asif}}$.

Round 4. The simulator receives the leakage $(\mu_i, j_{i,k}, v_{i,k}, f^{j_{i,k},i,v_{i,k}}(\mathbf{x}_{j_{i,k}}))_{i\in\mathsf{H},k\in[\mu_i]}$ from $\mathcal{F}_{\mathsf{asif}}$. The simulator samples symmetric bivariate polynomials of degree at most t in each variables $(\tilde{F}^{i,i',v}(x,j'), \tilde{G}^{i,j}(x,j'), \tilde{H}^{i,j,k}(x,j'))_{i\in\mathsf{H},i'\in[n],v\in[\ell],j,k\in[m],j'\in\mathsf{C}}$ with the same conditioning (1)–(7) as in Round 3, together with

8. For every $i \in \mathsf{H}$ and $k \in [\mu_i]$ so that $j_{i,k} \in \mathsf{H}$ the value

$$\left(\sum_{j=1}^{m} \beta_{j_{i,k},i,j}^{v_{i,k}} \tilde{G}^{j_{i,k},j}(0,0) + \sum_{j,k\in[m]} \gamma_{j_{i,k},i,j,k}^{v_{i,k}} \tilde{H}^{j_{i,k},j,k}(0,0)\right)$$

   is equal to $f^{j_{i,k},i,v_{i,k}}(\mathbf{x}_{j_{i,k}})$.

The simulator gives $(\mu_i, j_{i,k}, v_{i,k}, F^{j_{i,k},i,v_{i,k}}(x,y))_{i\in\mathsf{H},k\in[\mu_i]}$ to the adversary as the leakage of $\mathcal{F}_{\mathsf{asif}}$.

Communication with $\mathcal{F}_{\mathsf{sh\text{-}comp}}$. The simulator receives the inputs of the corrupt parties to $\mathcal{F}_{\mathsf{sh\text{-}comp}}$. For every corrupt P_i with $\mathsf{abort}_i = 1$, the simulator inputs $\mathsf{abort}_i = 1$ to $\mathcal{F}_{\mathsf{asif}}$. Otherwise, on inputs μ_i and $(j_{i,k}, i, v_{i,k})_{k\in[\mu_i]}$ from P_i, the simulator inputs $\mathsf{abort}_i = 0$, μ_i and $(j_{i,k}, v_{i,k})_{k\in[\mu_i]}$ to $\mathcal{F}_{\mathsf{asif}}$. This conclude the simulation.

F.2 Analysis

The security proof of our protocol in the $\mathcal{F}_{\mathsf{sh\text{-}comp}}$-hybrid model follows the same lines as the security proof of the classic BGW protocol [BGW88] for linear computation in the settings honest-majority and perfect security against against a semi-honest adversary (see [AL17] for more information). We therefore omit the analysis.

G General MPC

In this section, we prove that protocol mpc UC-emulates $\mathcal{F}$ with statistical security in the $\mathcal{F}_{\mathsf{asif}}$-hybrid model. From the composition properties of UC-security, this implies that protocol mpc UC-emulates $\mathcal{F}$. As mentioned in Section 10, we assume without loss of generality that $\mathcal{F}$ has public output.

Recall that protocol Π^{sm} provides perfect security for the computation of $\mathcal{F}$ against semi-malicious adversaries. We note that all we actually need is a weak form of security against rushing semi-malicious adversaries, that first see the first-round messages of the honest parties, then generate input $\mathbf{x}_i$ and randomness r_i for every corrupt P_i, and write $(\mathbf{x}_i, r_i)_{i\in\mathsf{C}}$ on a special input-randomness tape, so the corrupt parties play according to $(\mathbf{x}_i, r_i)_{i\in\mathsf{C}}$ in the protocol Π^{sm}. For security, we assume that there exists a simulator Sim that receives the description of the semi-malicious

adversary and uses the adversary in a straight-line black-box way. That is, Sim first generates the messages of the honest parties in the first round, and then queries the adversary in order to obtain $(\mathbf{x}_i, r_i)_{i\in\mathsf{C}}$. Therefore, it will be convenient to assume that Sim actually receives $(\mathbf{x}_i, r_i)_{i\in\mathsf{C}}$ from an external party. In addition, we assume that at the end of the first-round simulation, Sim sends $(\mathbf{x}_i)_{i\in\mathsf{H}}$ to the ideal functionality $\mathcal{F}$ and receives $\mathcal{F}(\mathbf{x}_1, \ldots, \mathbf{x}_n)$.[18]

Let $\mathcal{A}$ be the dummy adversary. We define the simulator for mpc as follows. The simulator 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 parties C.

G.1 The Simulator

Rounds 1 and 2. The simulator does nothing in Round 1 and Round 2.

Round 3. The simulator executes Sim to get the messages from the honest parties to the corrupt parties, denoted $(a_{i,j})_{i\in\mathsf{H},j\in\mathsf{C}}$. For every $i \in \mathsf{H}$ and $j \in \mathsf{C}$, the simulator samples Shamir's shares of $a_{i,j}$, denoted $(a_{i,j}[1], \ldots, a_{i,j}[n])$. In addition, for every $i \in \mathsf{H}$ and $j \in \mathsf{H}$, the simulator samples random values for the shares $(a_{i,j}[k])_{k\in\mathsf{C}}$. Moreover, for every honest P_i the simulator samples random values for the shares $(s_{i,j}[k])_{j\in[2\ell],k\in\mathsf{C}}$ and $(s^b_{i,j}[k])_{j\in\{2\ell+1,\ldots,4\ell\},b\in\{0,1\},k\in\mathsf{C}}$. The simulator also samples random pads $\eta_{i,j}$ for every $i \in \mathsf{H}$ and $j \in \mathsf{C}$, and sets $A'_{i,j} := a_{i,j} + \eta_{i,j}$. For $i \in \mathsf{H}$ and $j \in \mathsf{H}$ the simulator sets $A'_{i,j}$ to be a random value.

For every honest P_i and very corrupt P_k, the simulator provides the adversary with the following values as the output of $\mathcal{F}_i$ that corresponds to P_k: (1) the message $a_{i,k}$ together with $\eta_{i,k}$, (2) the shares $(a_{i,j}[k])_{j\in[n]}$, (3) the shares $(s_{i,j}[k])_{j\in[2\ell]}$, (4) the shares $(s^b_{i,j}[k])_{j\in\{2\ell+1,\ldots,4\ell\},b\in\{0,1\}}$, (5) the encrypted messages $(A'_{i,j})_{j\in[n]}$.

In addition, for every honest P_i and P_j, the simulator samples $\rho_{i,j}$ and $\eta_{i,j}$ at random. For an honest P_i and corrupt P_j, the simulator samples $\rho_{i,j}$ at random. The simulator gives the adversary the following values as the additional leakage of $\mathcal{F}_{\mathsf{asif}}$: (1) $\eta_{j,k} - \rho_{k,j}$ for every honest P_j, P_k, and (2) $\eta_{j,k}, \rho_{j,k}$ for every honest P_j and corrupt P_k.

At this stage the simulator receives the input of the corrupt parties to $\mathcal{F}_{\mathsf{asif}}$. The simulator returns the set I as the set of all corrupt P_i's with $\mathsf{flag}_i = 1$. For every corrupt P_i and P_j so that $i \notin I$, the simulator holds all the inputs of $\mathcal{F}_i$ and can provide P_j with the corresponding outputs. This concludes the simulation of the third round.

Communication with $\mathcal{F}$. For every corrupt P_i with $\mathsf{flag}_i = 1$, the simulator resets $\mathbf{x}_i$ and r_i to be the all-zero string. Denote the messages For every corrupt P_i with $\mathsf{flag}_i = 0$, the simulator holds the values $\mathbf{x}_i$ and r_i that P_i sent to $\mathcal{F}_{\mathsf{asif}}$. The simulator inputs $(\mathbf{x}_i)_{i\in\mathsf{C}}$ to $\mathcal{F}$. The simulator receives from $\mathcal{F}$ the outputs of the corrupt parties, denoted $\mathcal{F}(\mathbf{x}_1, \ldots, \mathbf{x}_n)$.

Round 4. The simulator provides Sim with $(\mathbf{x}_i, r_i)_{i\in\mathsf{C}}$, receives the query $(\mathbf{x}_i)_{i\in\mathsf{C}}$ from Sim to $\mathcal{F}$ and provides Sim with the outputs $\mathcal{F}(\mathbf{x}_1, \ldots, \mathbf{x}_n)$. The simulator now receives from Sim the broadcasts $(b_i)_{i\in\mathsf{H}}$ of the honest parties in the second round of Π^{sm}.

For every $i \in \mathsf{H}$ the simulator sampled $\eta_{i,j}, \rho_{i,j}$ for every $j \in [n]$. For every corrupt P_i so that $i \notin I$, the simulator holds the inputs of $\mathcal{F}_i$ can compute the values $\eta_{i,j}, \rho_{i,j}$ for every $j \in [n]$. For every $i, j \notin I$ we set $\nu_{j,i} := \eta_{j,i} - \rho_{i,j}$.

[18] We note that the protocol of [ABT18] satisfies this property.

For every $i \in I$ and every $j \notin I$, the simulator broadcasts $a_{j,i}[k]$ on behalf of every honest P_k, where $a_{j,i}$ was sampled by the simulator in Round 3 if $j \in \mathsf{H}$, and $a_{j,i}[k]$ is given in the output of $\mathcal{F}_j$ if $j \in \mathsf{C}$.

In addition, for $i \notin I$ the simulator does as follows. The simulator sets $L_i := I$. In addition, for every $j \notin I$ the simulator sets $A_{j,i} := A'_{j,i} - \nu_{j,i}$. For every $j \in I$ let $A_{j,i} := a_{j,i}$, where $a_{j,i}$ is the message that P_j sends to P_i according to Π^{sm} when x_j and r_j are the all-zero string. Consider the length-2ℓ binary string $\boldsymbol{\beta}_i = (\boldsymbol{\beta}_i[1], \ldots, \boldsymbol{\beta}_i[2\ell])$ that corresponds to $(L_i, (A_{j,i})_{j\in[n]})$. We split into cases.

- Assume that P_i is honest. The simulator executes $\mathsf{Sim}_i^{\mathsf{RE}}(b_i)$ to obtain a binary string, denoted $s_{i,1}, \ldots, s_{i,4\ell}$, where $\mathsf{Sim}_i^{\mathsf{RE}}$ is the simulator of the randomized encoding of G_i. For every $j \in [2\ell]$ the simulator samples Shamir shares of $s_{i,j}$ conditioned on the k-th share being $s_{i,j}[k]$. Similarly, for every $j \in \{2\ell+1, \ldots, 4\ell\}$ the simulator samples Shamir shares of $s_{i,j}$ conditioned on the k-th share being $s_{i,j}^{\boldsymbol{\beta}_i[j-2\ell]}[k]$. For every honest P_k, the simulator gives the adversary the values $(s_{i,j}[k])_{j\in[2\ell]}$ and $(s_{i,j}^{\boldsymbol{\beta}_i[j-2\ell]}[k])_{j\in\{2\ell+1,\ldots,4\ell\}}$ as the output of $\mathcal{F}_{\mathsf{asif}}$ that corresponds to P_k.
- Assume that P_i is corrupt. Then for every honest For every honest P_k, the simulator gives the adversary the values $(s_{i,j}[k])_{j\in[2\ell]}$ and $(s_{i,j}^{\boldsymbol{\beta}_i[j-2\ell]}[k])_{j\in\{2\ell+1,\ldots,4\ell\}}$ as the output of $\mathcal{F}_{\mathsf{asif}}$ that corresponds to P_k, where $s_{i,j}[k]$ is computed by the simulator that holds all inputs to $\mathcal{F}_i$.

This concludes the simulation.

G.2 Analysis

The first and second round consist only of the input-phase of $\mathcal{F}_{\mathsf{asif}}$ so there is no communication in this round.

Third round simulation. In the third round, the adversary first receives the outputs of $\mathcal{F}_i$ for every honest P_i as leakage. By the perfect security of Π^{sm}, the messages $(a_{i,k})_{i\in\mathsf{H},k\in\mathsf{C}}$ have the same distribution in both worlds. Fix those messages. The random pads $(\eta_{i,k})_{i\in\mathsf{H},k\in\mathsf{C}}$ also have the same distribution in both worlds, and we fix them as well. By the perfect privacy of Shamir's secret sharing, the shares $(a_{i,j}[k])_{i,j\in\mathsf{H},k\in\mathsf{C}}$, $(s_{i,j}[k])_{i\in\mathsf{H},j\in[\ell],k\in\mathsf{C}}$ and $(s_{i,j}^{\beta}[k])_{i\in\mathsf{H},j\in\{2\ell+1,\ldots,4\ell\}\beta\in\{0,1\},k\in\mathsf{C}}$ have the same distribution in both worlds, and we fix them as well. In addition, the shares $(a_{i,j}[k])_{i\in\mathsf{H},j\in\mathsf{C},k\in[n]}$, are sampled in the same way in both worlds, and we fix them as well. Finally, for every $i \in \mathsf{H}$ and $k \in \mathsf{C}$ the encrypted message $A'_{i,k}$ is fixed and equal to $a_{i,j} + \eta_{i,k}$ in both worlds, while for every $i \in \mathsf{H}$ and $k \in \mathsf{H}$ the encrypted message $A'_{i,k}$ is uniformly distributed in both worlds. In addition, in the third round the adversary receives $\nu_{j,k}$ as a leakage for every honest P_j, P_k. In both worlds $\nu_{j,k}$ is uniformly distributed since $\rho_{k,j}$ is uniformly distributed. The adversary also receives $\eta_{j,k}$ and $\rho_{j,k}$ for every honest P_j and corrupt P_k, where $\eta_{j,k}$ was already fixed, and $\rho_{j,k}$ is uniformly distributed in both worlds. At the end of the round, the inputs of the corrupt parties to $\mathcal{F}_{\mathsf{asif}}$ are generated in the same way in both worlds. Fix those inputs. This concludes the analysis of the third round simulation.

Fourth round simulation. For every $i, j \notin I$, the values $\nu_{j,i}$ are fixed and the same in both worlds. In addition, for every $i \in I$ and $j \notin I$, the shares $(a_{i,j}[k])_{k\in\mathsf{H}}$ are fixed and the same in both worlds.

The first round messages of the corrupt parties in Π^{sm} are generated according to $(\mathbf{x}_i, r_i)_{i\in\mathsf{C}}$ and fully defines the messages $(a_{i,j})_{i\in\mathsf{C},j\in[n]}$. We continue by analysing the broadcasts regarding the i-th randomized encoding, for $i \notin I$.

- Assume that P_i is honest, denote the input of P_i by $\mathbf{x}_i$ and the randomness that P_i picked for Π^{sm} by r_i (this is a random variable). Observe that this two values fully determine the first-round messages $(a_{i,j})_{j\in[n]}$ of P_i in Π^{sm} (those are random variables as well).

  In the real-world, consider the second-round broadcast b_i of P_i, that is defined according to $\mathbf{x}_i, r_i, (a_{j,i})_{j\in[n]}$. By the perfect security of Π^{sm}, we conclude that b_i has the same distribution in both worlds. Fix those broadcasts. Note that the output of G_i on inputs (x_i, r_i), $(\rho_{i,j})_{j\in[n]}$, $L_i = I$ and $(A_{j,i})_{j\in[n]}$ is indeed the vector b_i.

  Observe that the vector $\boldsymbol{\beta}_i$ is fixed and the same in both worlds. Consider the real-world random variables $(g_{i,1}, \ldots, g_{i,4\ell})$ and the ideal-world random variables $(s_{i,1}, \ldots, s_{i,4\ell})$. by the perfect correctness of the randomized encoding, the decoding of $(g_{i,1}, \ldots, g_{i,4\ell})$ is b_i. Therefore, by the perfect privacy of the randomized encoding, the distribution of $(g_{i,1}, \ldots, g_{i,4\ell})$ is the same as the distribution of $(s_{i,1}, \ldots, s_{i,4\ell})$. Conditioned on those values, it is not hard to see that the broadcasts of the honest parties have the same distribution in both worlds.

- Assume that P_i is corrupt. Then broadcasts of every honest P_k are based on the same fixed values $(\boldsymbol{\beta}_i, (s_{i,j}[k])_{j\in[2\ell]}, (s_{i,j}^{\boldsymbol{\beta}_i[j-2\ell]}[k])_{j\in\{2\ell+1,\ldots,4\ell\}})$ in both worlds, and therefore they are the same.

This concludes the analysis of the fourth round simulation.

Outputs of honest parties. In the ideal world the output of every honest P_i is $\mathcal{F}(\mathbf{x}_1, \ldots, \mathbf{x}_n)$. It remains to show that this is also the output in the real world. We've seen that the first-round messages of Π^{sm} correspond to an execution with inputs $(\mathbf{x}_i, r_i)$ to P_i, and that for every honest P_i the second-round broadcast is also computed in the same way as in Π^{sm}. To complete the proof we need to show that for every corrupt P_i, the second-round broadcast of P_i is also computed as in Π^{sm}. This is clearly true for $i \in I$, since the parties locally compute the broadcast b_i according to $\mathbf{x}_i = 0$ and $r_i = 0$. In addition, for a corrupt P_i with $i \notin I$, this is true by the perfect correctness of the randomized encoding. Therefore, the messages that the honest parties see correspond to an execution of Π^{sm} with $(\mathbf{x}_i, r_i)_{i\in[n]}$. Correctness now follows from the perfect correctness of Π^{sm}. This concludes the proof.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il