Verifiable Relation Sharing and Multi-Verifier Zero-Knowledge in Two Rounds: Trading NIZKs with Honest Majority

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

Abstract

We introduce the problem of *Verifiable Relation Sharing* (VRS) where a client wishes to share a vector of secret data items among several servers (the verifiers) while proving in zero-knowledge that the shared data satisfies some properties. This combined task of sharing and proving generalizes notions like verifiable secret sharing and zero-knowledge proofs over secret-shared data. We study VRS from a theoretical perspective and focus on its round complexity.

As our main contribution, we show that every efficiently-computable relation can be realized by a VRS with an optimal round complexity of two rounds where the first round is input-independent (offline round). The protocol achieves full UC-security against an active adversary that is allowed to corrupt any t-subset of the parties that may include the client together with some of the verifiers. For a small (logarithmic) number of parties, we achieve an optimal resiliency threshold of $t = 0.5(k+1)$, and for a large (polynomial) number of parties, we achieve an almost-optimal resiliency threshold of $t = 0.5(k+1)(1-\epsilon)$ for an arbitrarily small constant $\epsilon > 0$. Both protocols can be based on sub-exponentially hard injective one-way functions. If the parties have an access to a collision resistance hash function, we can derive *statistical everlasting security*, i.e., the protocols are secure against adversaries that are computationally bounded during the protocol execution and become computationally unbounded after the protocol execution.

Previous 2-round solutions achieve smaller resiliency thresholds and weaker security notions regardless of the underlying assumptions. As a special case, our protocols give rise to 2-round offline/online constructions of multi-verifier zero-knowledge proofs (MVZK). Such constructions were previously obtained under the same type of assumptions that are needed for non-interactive zero-knowledge proofs (NIZK), i.e., public-key assumptions or random-oracle type assumptions (Abe et al., Asiacrypt 2002; Groth and Ostrovsky, Crypto 2007; Boneh et al., Crypto 2019; Yang, and Wang, Eprint 2022). Our work shows, for the first time, that in the presence of an honest majority these assumptions can be replaced with more conservative "Minicrypt"-type assumptions like injective one-way functions and collision-resistance hash functions. Indeed, our MVZK protocols provide a round-efficient substitute for NIZK in settings where honest-majority is present. Additional applications are also presented.

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

Contents

1 Introduction **4**
1.1 Our Contribution . . . 5
1.1.1 Round-Optimal VRS and MVZK in Minicrypt . . . 5
1.1.2 VRS and MVZK with Everlasting Security in Minicrypt . . . 7
1.1.3 Round-Optimal Linear Function Computation in Minicrypt . . . 7
1.1.4 Applications . . . 8
1.2 Related Works and Comparison . . . 9

2 Preliminaries **10**

3 Technical Overview **11**
3.1 SIF for Few Parties . . . 12
3.2 SIF for any Number of Parties . . . 14
3.3 Replacing the Idealised VSS with 1.5-Round Protocols . . . 16
3.3.1 Supporting Linear Operations . . . 17

4 Secure Linear Function Computation **19**
4.1 Verifiable Secret Sharing . . . 19
4.2 Secure Computation with a Guard . . . 21
4.3 Guided Linear Function Computation . . . 22
4.4 Linear Function Computation . . . 23

5 Single Input Functionalities **24**
5.1 Triple Secret Sharing . . . 24
5.2 Public Single Input Functionality . . . 25
5.3 General Single Input Functionality . . . 26

A Appendix: Security Model, Useful Facts and Standard Primitives **30**
A.1 Security model . . . 30
A.2 Standard Useful Facts . . . 33
A.2.1 Polynomials . . . 33
A.2.2 Statistical Distance . . . 34
A.3 Non-Interactive Commitment Schemes (NICOM) . . . 35

B Appendix: Applications of VRS and SIF **36**

C Appendix: A common strategy for the proofs **38**

D Verifiable Secret Sharing **38**
D.1 The VSS functionality and protocol . . . 40
D.2 Proof of security . . . 41
D.2.1 Honest Dealer . . . 41
D.2.2 Corrupt Dealer . . . 43

E Secure Partial Computation with a Guard **44**
E.1 PSM Protocols 44
E.2 The Protocol 45
E.3 Proof of security 46
E.3.1 Honest Alice and Bob – Simulator 47
E.3.2 Honest Alice and Bob – Statistically-Hiding Commitment Scheme 47
E.3.3 Honest Alice and Bob - Computationally-Hiding Commitment Scheme . . . 49
E.3.4 Corrupt Alice, Honest Bob 51
E.3.5 Honest Alice, Corrupt Bob 53
E.3.6 Corrupt Alice and Bob 53
E.3.7 Complexity Analysis 53

F Guided Linear Function Computation **54**
F.1 The Protocol 54
F.2 Proof of security 55
F.2.1 Honest Guide 55
F.2.2 Corrupt Guide 56

G General Linear Function Computation **58**

H Triple Secret Sharing for a Small Number of Parties **58**
H.1 Proof of security 61
H.1.1 Honest Dealer 61
H.1.2 Corrupt Dealer 67

I Public Single-Input Functionality for a Small Number of Parties **70**
I.1 Proof of security 72
I.1.1 Honest Dealer 72
I.1.2 Corrupt Dealer 76
I.1.3 Sufficient Conditions for Efficient Optimal-Resiliency SIF 78

J Public SIF for any Number of Parties **78**
J.1 Execution of psiflog with a Subset of Parties 78
J.2 Public SIF 80
J.3 Proof of security 82
J.3.1 Honest Dealer 82
J.3.2 Corrupt Dealer 84

K General Single Input Functionality **85**
K.1 Proof of Security 86
K.1.1 Honest Dealer 87
K.1.2 Corrupt Dealer 88

1 Introduction

In recent years, a large amount of research was dedicated to the study of zero-knowledge proofs in *distributed settings*, such as zero-knowledge proofs with multiple verifiers [41, 58, 10] and zero-knowledge proofs over secret-shared data [17, 26, 18, 25]. Those variants of zero-knowledge proofs have applications both in theory and practice, in round-optimal multiparty computation [2], private data aggregation [25], and anonymous communication [26].

A typical scenario of interest consists of a client $\mathcal{P}$ (the prover) that holds a vector of secret data items s, together with several servers $\mathcal{V}_1, \dots, \mathcal{V}_k$ (the verifiers). The client wishes to share s among the servers, and also prove in zero-knowledge that the shared data satisfies some properties. Previous works usually let $\mathcal{P}$ send each $\mathcal{V}_i$ its share, and then perform a zero-knowledge proof on the shared data. A natural question is whether considering the sharing and the proving as a single task could result in a protocol with better round-complexity and better security guarantees. To capture this joint task of sharing-and-proving, we present the notion of *verifiable relation sharing* (VRS).

Verifiable relation sharing. The VRS functionality of a public relation R receives from the prover an input $\mathbf{x} = (x_0, x_1, \dots, x_k)$, where we think of x_0 as a private information of the prover, and of x_i as the share of $\mathcal{V}_i$. The functionality verifies that $R(\mathbf{x}) = 1$, and if the verification fails, then it returns a failure-symbol $\perp$ to all the verifiers. If the verification succeeds, the functionality returns x_i to $\mathcal{V}_i$. Observe that the VRS functionality captures the typical scenario discussed above, as well as several cryptographic primitives, including verifiable secret sharing [24], verifiable function secret sharing [18], secure multicast [35], and zero-knowledge proofs with multiple verifiers.

We formalize the VRS functionality under the definitions of secure multiparty computation (MPC) in the universal-composability (UC) framework of [22]. We strive for full-security, including guaranteed output delivery, at the presence of an honest majority in the plain model. We note that honest-majority is necessary due to impossibility of UC-secure Zero-knowledge proofs in the plain model [23]. The active (aka Byzantine or malicious) adversary is allowed to corrupt any minority subset of the $k + 1$ parties $\{\mathcal{P}, \mathcal{V}_1, \dots, \mathcal{V}_k\}$ that may include the prover together with some of the verifiers. The use of MPC-based "full-security" definitions provides strong guarantees that are not supported by related notions of distributed zero-knowledge. Specifically, when the prover $\mathcal{P}$ is honest, we get *correctness*, i.e., every honest $\mathcal{V}_i$ outputs x_i even in the presence of corrupt active verifiers, as well as simulation-based *privacy*, which implies that the adversary only learns the outputs of the corrupt verifiers. For a corrupt $\mathcal{P}$, we get *soundness* and *knowledge extraction* even when $\mathcal{P}$ colludes with some of the verifiers. In contrast, previous works on weaker notions, such as zero-knowledge proofs over secret-shared data, achieve correctness only for semi-honest verifiers [17, 18, 26, 25], and in some cases (e.g., [25, 26]) provide soundness only when all the verifiers are honest.

We study the VRS problem from a theoretical perspective while focusing on the best-achievable *round complexity*. It is known that VRS cannot be realized in 1 round even for relatively simple relations (e.g., VSS [7]). Looking for the second best, we ask:

> **Q1:** Can VRS be realized by a 2-round protocol? Moreover, can we make the first round input-independent ("offline round")? If so, under what assumptions?

The question of obtaining a 2-round protocol in the plain model is open even for weaker notions like distributed zero-knowledge over secret-shared data.

Multi-verifier zero-knowledge. It is useful to consider the somewhat degenerate version of VRS in which all the verifiers get the same information except for some private witness that is kept by $\mathcal{P}$. This variant essentially corresponds to *multi-verifier zero-knowledge* proofs (MVZK) [21]. When modeled as an ideal functionality, MVZK is parameterized by a public relation R, it receives from $\mathcal{P}$ a statement x and a witness w, and verifies that $R(x, w) = 1$. If the verification fails, then the functionality returns a failure-symbol $\perp$ to all the verifiers $\mathcal{V}_1, \ldots, \mathcal{V}_k$, and if the verification succeeds, the functionality returns x to all the verifiers. Again, we strive for a 2-round offline/online solution in the plain model.

Observe that the single verifier case (where the adversary can either corrupt the verifier or the prover) corresponds to the standard notion of zero-knowledge proofs. Classical impossibility results [40] show that a plain-model protocol that consists of a single message from the prover to the verifier, also known as *non-interactive zero-knowledge* (NIZK), exist only for languages in BPP, even when one considers only stand-alone security. Assuming a minimal trusted setup in the form of a common reference string (CRS), one can achieve NIZK for every language in NP from public-key assumptions [16, 32, 42, 56, 14, 54], or, alternatively, in the random oracle model [33, 12]. In a related notion, called *Zaps* [30], the CRS is replaced with a preprocessing round in which only the verifier communicates by broadcasting its random coins, at the expense of downgrading zero-knowledge to witness-indistinguishability. Assuming the existence of one-way functions, it is known that Zaps are equivalent to NIZK [30].

Let us move back to the setting of multiple verifiers. Striving for a 2-round simulation-based zero-knowledge, we make the necessary assumption of an honest majority among the set of all parties (including the prover).[1] To the best of our knowledge, the only known solution in this setting follows from the work of Groth and Ostrovsky on Multi-string NIZK Proofs. Specifically, their work implicitly give rise to a 2-round offline/online honest-majority MVZK that achieves simulation-based security based on Zaps and public-key encryption [41, Theorem 3]. These assumptions are as strong (or even stronger) than the ones needed for NIZK protocols in the seemingly "harder" 2-party settings. We therefore ask:

> **Q2:** Are NIZK/Zaps assumptions inherently needed for an MVZK protocol with 1-offline and 1-online round in the honest-majority setting? Is it possible to replace these assumptions with weaker assumptions?

1.1 Our Contribution

1.1.1 Round-Optimal VRS and MVZK in Minicrypt

We answer Questions 1 and 2 in the affirmative. Our main result is a protocol with 1-offline round and 1-online round for VRS in the UC-framework, assuming the existence of perfectly-binding non-interactive commitment scheme (NICOM) with sub-exponential privacy. Such a NICOM scheme can be based on injective one-way functions with sub-exponential hardness or even on standard one-way function with sub-exponential hardness assuming worst-case complexity-theoretic derandomization assumptions [51, 9].[2] Throughout, we assume that the parties commu-

[1]Without an honest majority, a 2-round plain-model MVZK protocol (where in each round both the verifiers and prover can talk simultaneously) implies a 2-step ZK protocol (where the verifier sends a message and gets a response from the prover) which is ruled-out by [40] for non-trivial languages outside BPP.

[2]For technical reasons, the NICOM should satisfy some level of security against selective opening that, by "complexity leveraging", follows from the assumption that the underlying one-way function (or injective one-way function)

nicate over pairwise secure and authenticated point-to-point channels, as well as over a common broadcast channel, which allows each party to send a message to all parties and ensures that the received message is identical.

Theorem 1.1. *Assuming the existence of injective one-way functions with sub-exponential hardness, for every $\epsilon > 0$ the VRS functionality of every efficiently computable relation R can be realized in 1-offline round and 1-online round, with full security against an active rushing adversary, in any of the following settings.*

- (Optimal resiliency for small number of verifiers) *The number of verifiers k is at most logarithmic in the security parameter, and the adversary corrupts less than $(k + 1)/2$ parties.*
- (Almost-optimal resiliency for polynomially-many verifiers) *The number of verifiers k grows polynomially with the security parameter and the adversary corrupts less than $(k+1) \cdot (\frac{1}{2} - \epsilon)$ parties.*

Since MVZK is a special case of VRS, we obtain the following corollary.

Corollary 1.2. *Assuming the existence of injective one-way functions with sub-exponential hardness, the MVZK functionality of every efficiently computable relation R can be realized in 1-offline round and 1-online round, with full security against an active rushing adversary, in the same settings of Theorem 1.1.*

For optimal resiliency, we obtain a protocol with complexity polynomial in the security parameter, but exponential in the number of verifiers k. On the other hand, for every $\epsilon > 0$ we obtain a protocol with resiliency $(k + 1) \cdot (\frac{1}{2} - \epsilon)$, whose complexity is polynomial both in the security parameter and in k. (In fact, we can push ϵ to be as small as $\epsilon = \Omega(\frac{1}{\sqrt{\log k}})$; see Remark J.3.)

The difference between optimal resiliency and "almost-optimal resiliency" is mostly relevant when the number of verifiers is small, e.g., constant. In this setting, the first protocol provides an efficient solution. Specifically, we highlight the case of 3-party computation, with a single prover and two verifiers, and we note that by adding just a single verifier to the standard zero-knowledge settings, we can obtain a protocol with 1-offline round and 1-online round for the case of a single corruption from Minicrypt-type assumptions. (In contrast, general-purpose 3-party MPC for honest majority requires 3 rounds [53].)

Still, the existence of a strict-honest-majority 2-round VRS protocol whose complexity scales polynomially with the number of parties, remains an interesting open problem. We show that such a protocol can be constructed if one is willing to make stronger assumptions (e.g., random oracle or correlation-intractable functions) or if the adversary is non-rushing. In fact, we note that a weak limitation of the rushing capabilities of the adversary suffices, and present a new notion of *semi-rushing* adversary to model such a behavior.[3]

cannot be inverted in polynomial-time with more than sub-exponential probability. This seems to be a relatively mild assumption; See Remark A.9.

[3]The difference between rushing and non-rushing adversary boils down to the scheduling of the messages within a single round of a protocol. A *non-rushing* adversary must send the messages of the corrupt parties in a given round before receiving the messages of the honest parties in that round, whereas a *rushing* adversary may delay sending the messages of the corrupt parties until receiving the messages from the honest parties. Thus, the messages of the corrupt parties may depend on the messages of the honest parties in the same round. Our notion of *semi-rushing* adversary allows the adversary to see all the messages of the honest parties, except for one. For more about this model and its relevance, see the discussion in Section I.1.3.

1.1.2 VRS and MVZK with Everlasting Security in Minicrypt

It is known that if we do not put restriction on the round complexity, then, in the setting of honest-majority, one can obtain *unconditional* results and no assumptions are needed at all! Specifically, as shown by Rabin and Ben-Or [55], every efficiently computable function can be securely computed with statistical security against computationally-unbounded adversaries. While we do not know whether it is possible to achieve statistical security in 2 rounds, we show that VRS and MVZK can be implemented by a protocol that achieves *statistical everlasting security* assuming an access to a collision-resistant hash function h. The notion of statistical everlasting security [50] can be viewed as a hybrid version of statistical and computational security. During the run-time, the adversary is assumed to be computationally-bounded (e.g., cannot find collisions in the hash function) but after the protocol terminates, the adversary hands its view to a computationally-unbounded analyst who can apply arbitrary computations in order to extract information on the inputs of the honest parties (e.g., finding collisions or even reading the whole truth table of h).[4] This feature is one of the main advantages of information-theoretic protocols: after-the-fact secrecy holds regardless of technological advances and the time invested by the adversary.

Theorem 1.3. *Given an access to a collision-resistant hash function, the VRS and MVZK functionalities of efficiently computable relations can be realized in 1-offline round and 1-online round, with full security and* everlasting security *against an active rushing adversary, in the same settings (honest-majority with few verifiers or almost-honest majority with many verifiers) of Theorem 1.1.*

Remark 1.4 (On the use of hash function)**.** *Our protocol assumes that all parties are given an access to a collision resistance hash function h. Theoretically speaking, such a function should be chosen from a family of functions $\mathcal{H}$ in order to defeat non-uniform adversaries. One may assume that h is chosen once and for all by some simple set-up mechanism. In particular, by using the standard concatenation-based combiner for hash functions [46], this set-up mechanism may be realized distributively by a single round of public random coins where security holds against an active rushing adversary that may corrupt all the participants except for a single one. The choice of the hash function can be abstracted by a CRS functionality, or even, using the multi-string model of [41] with a single honestly-generated string. However, it should be emphasized that this CRS is being used in a very* weak *way: It is "non-programmable" (the simulator receives h as an input) and it can be sampled once and for all by using the above trivial public-coin mechanism. Even if one counts this extra set-up step as an additional round, to the best of our knowledge, everlasting security was not known to be achievable regardless of the underlying assumptions.*

The difference between everlasting and computational security is *fundamental* and is analogous to the difference between statistical commitments and computational commitments or statistical ZK vs. computational ZK (see, e.g., the discussions in [20, 52]). Indeed, Theorem 1.3 provides (UC-secure) MVZK with a *statistical zero-knowledge* property. As a side bonus, Theorem 1.3 does not require sub-exponential hardness assumptions.

1.1.3 Round-Optimal Linear Function Computation in Minicrypt

Using the machinery we develop for VRS and MVZK, we obtain a 3-round protocol for linear function computation. By the lower-bound of [37] our protocol has optimal round complexity. Like

[4]Technically, in the UC-framework we allow the environment to output its view and require statistical indistinguishability between the real and ideal experiments. For details, refer to Appendix A.1.

in previous results, we assume the existence of injective one-way functions with sub-exponential hardness in order to obtain a protocol with computational security in the plain model, or an access to a collision resistance hash-function in order to obtain a protocol with everlasting security. In contrast, previous works achieve only computational security by assuming public-key encryption and Zaps [2]. We emphasize that in Theorem 1.5 we obtain *optimal resiliency* even when the number of parties is polynomial in the security parameter.

Theorem 1.5. *Assuming the existence of injective one-way functions with sub-exponential hardness, every efficiently computable linear function can be realized in 3 rounds, with full security against an active rushing adversary, that corrupts a minority of the parties. If we replace the one-way function with an access to a collision resistance hash-function, we also obtain everlasting security.*

1.1.4 Applications

We present some applications of our protocols.

MVZK as a NIZK-substitute for honest majority. We notice that our MVZK protocol captures an important aspect of NIZK, its *minimal round complexity*, while using only Minicrypt-type assumptions. Indeed, our MVZK protocol implies that the CRS for NIZK is not required, and can be replaced with only a *single* offline-round of communication. Similar to NIZK, the proof itself requires only *one online round*. However, unlike NIZK, in our protocol all the parties have to communicate in the online round.

Round-efficient manipulation of non-homomorphic commitments. In a common scenario in multiparty computation, a party $\mathcal{P}$ holds openings to public commitments $C_1, \ldots, C_\ell$. $\mathcal{P}$ wishes to apply some function f on the committed values $z_1, \ldots, z_\ell$ and let the rest of the parties learn $y := f(z_1, \ldots, z_\ell)$, while proving in zero-knowledge that she used the committed values in the computation of f. Alternatively, $\mathcal{P}$ may want to generate another commitment C, that hides y, while proving in zero-knowledge that C was honestly generated. Both the tasks can be solved in 1-offline round and 1-online round by using our MVZK protocol. Since the offline round can be executed in parallel to the generation of $C_1, \ldots, C_\ell$, both tasks require only one additional round! (See Appendix B for details.)

Round-efficient GMW-type compilers in Minicrypt. Using MVZK one can obtain round-efficient GMW-type compilers in Minicrypt, for the case of honest majority. Given a protocol π which is secure against a semi-malicious adversary,[5] we obtain a protocol π' with unanimous abort against an active adversary at the expense of adding a single offline round. If π is secure against a passive (aka semi-honest) adversary, the overhead grows to 4 rounds. This is because, in order to extend our compiler to protocols with passive security, it is enough to be able to generate a random string for every party in 4 rounds. For this, we note that the coin-generation protocol of [38, Section 7.5.4.4] can be realized in 4 rounds using our protocol. Notably, unlike the GMW compiler, our transformation avoids the use of public-key encryption. We refer to Appendix B for elaborate details.

[5] A *semi-malicious* adversary is allowed to choose its input and randomness but otherwise follows the protocol. Many passively secure protocols (e.g., [13]) actually offer semi-malicious security.

Round-optimal honest-majority MPC in Minicrypt. A followup work by the same authors [57] shows that general secure multiparty computation with *full-security* (including guaranteed output delivery) in the presence of an honest majority can be achieved in an optimal number of 3 rounds based on Minicrypt-type assumptions (e.g., NICOMs). A main building block of the protocol is our 2-round offline/online VRS protocol.

Bibliographic Note. Previous unpublished version of [57] contained a weak form of some of the current results based on the Fiat-Shamir heuristic. These results were removed from the new version of [57] (that is under submission to this conference), and are fully subsumed by the current paper.

1.2 Related Works and Comparison

Related works. The VRS functionality was first studied in [37] under the framework of single-input functionalities that takes their input from a single party. The resiliency was improved to $(k+1)/3$ by Applebaum et al. [4], at the cost of degrading the perfect security to computational security, assuming the existence of NICOMs.

Boneh et al. [17] initiated the formal study of zero-knowledge proofs over secret-shared data. They considered information-theoretic security in the following models of corruptions: (1) the adversary corrupts the prover *or* up to $k-1$ verifiers, and (2) the adversary corrupts the prover *and* less than $k/2$ verifiers. In both corruption models, they only provide *security with abort*. Their protocols exploit PCP machinery to achieve low communication complexity (sub-linear in the description of the relation), but have a super-constant number of rounds. Based on a random oracle, the number of rounds can be collapsed to 2, assuming that the data is already secret-shared among the verifiers.

MVZKs were first introduced in [21]. The most relevant MVZK for us can be derived from [41] which provides a construction of NIZK in the *multi-string model* assuming the existence of Zaps. In the multi-string model, the CRS is replaced with several authorities, each providing the protocol with a public random string, and the protocol is secure as long as a majority of those authorities are honest (that is, if a majority of the strings are uniformly distributed). An MVZK protocol with an honest majority of parties can be obtained in the plain model by letting each party broadcast a random string in the offline round, so that a majority of the strings are uniformly distributed. Simulation-based security can be obtained via the additional help of public-key encryption [41, Theorem 3].

Other non-interactive variants of MVZK were presented in [1]. Translated to our model, their work yield 2-round MVZK for $t < k/3$ and a 3-round protocol for $t < n/2$. Both results hold under public-key (discrete-log) hardness assumptions. Recently, [58] and [10] constructed MVZK with practical real-world efficiency in honest and super-honest majority settings. However, their low round (2 or 3) variants rely on random oracle and achieve either selective abort or identifiable abort.

Comparison. We compare our results with the most relevant existing results.

Ref.	Primitive	Rounds	Threshold	Assumptions	Security[†]
[37]	SIF/VRS	2	$t < (k+1)/6$	–	it and full security
[4]	SIF/VRS	2	$t < (k+1)/3$	NICOM	cs and full security
[17]	ZK over shared data	2[⋆]	$t < k/2$[‡]	Random Oracle	it and abort
[41]	MVZK	2	$t < k/2$	PKE	cs and full security
[1]	MVZK	3	$t < k/2$	Discrete-log	cs and full security
[58]	MVZK	2	$t < k/2$	Random Oracle	it and abort
[10]	MVZK	2	$t < k/3$	Random Oracle	it and identifiable abort
This paper	SIF/VRS	2	$t < (k+1)(\frac{1}{2} - \epsilon)$[§]	NICOM[⋆⋆]	cs/es and full security

[†] it: information-theoretic, es: everlasting security, cs: computational security,
[‡] They assume (1) the adversary corrupts the prover *or* up to $k-1$ verifiers, and
(2) the adversary corrupts the prover *and* less than $k/2$ verifiers
[⋆] The round complexity does not include the rounds needed for data sharing.
[⋆⋆] Perfectly-binding and sub-exponentially hiding NICOM for cs security and
Computationally-binding and statistically-hiding NICOM for es security.
[§] We achieve $t < (k+1)/2$ when k is logarithmic in the security parameter.

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

2 Preliminaries

Single-Input Functionalities. We adopt an MPC-based notation and replace VRS with the following notion of *single-input functionalities* (SIF). We assume that there are n parties, $\mathsf{P} = \{P_1, \dots, P_n\}$, where one party (e.g., P_n) takes the role of a *Dealer* D. The SIF functionality $\mathcal{F}$ is parameterized with a function $f : \{0,1\}^* \to (\{0,1\}^*)^n$, it takes an input string $\mathbf{z}$ from the dealer, computes the outputs $(\mathbf{y}_1, \dots, \mathbf{y}_n) = f(\mathbf{z})$ and delivers $\mathbf{y}_i$ to the ith party P_i. It is not hard to see that VRS is a special case of SIF, and that VRS implies SIF in a round-preserving way. (Indeed, to realize $\mathcal{F}$ define the relation R that accepts a vector $(x_0, x_1, \dots, x_{n-1})$ if $x_i = \mathcal{F}_i(x_0)$ for $i \in [n-1]$, and let D invoke a VRS for R with the input $(z, \mathcal{F}_2(z), \dots, \mathcal{F}_n(z))$.) We will mostly focus on the special case of *public-SIF* that delivers the same output to all the parties, and show in Section K that a 2-round offline/online general-SIF reduces to 2-round offline/online public-SIF via the aid of NICOMs.

Security model. We consider an active static, rushing adversary that may corrupt up to t parties. We consider two main settings: the optimal resiliency setting where $n = 2t+1$ and the almost-optimal resiliency setting where $n = (2+\epsilon)t$ for some arbitrarily small constant $\epsilon > 0$. The parties are connected by pairwise secure channels and additionally a broadcast channel is available. We prove security of our protocols in the UC-framework [22] which is recalled in Appendix A.1. We identify the set of parties P with $\{1, \dots, n\}$, and denote the set of honest parties by $\mathsf{H} \subseteq \mathsf{P}$, and the set of corrupt parties by $\mathsf{C} \subseteq \mathsf{P}$. In our protocols, we follow the convention that the honest parties can "disqualify" the dealer whenever it is clear from broadcast messages that the dealer misbehaves. This does not violate "guaranteed output delivery" since in case of disqualification, the honest parties can always apply f on some predetermined default value and output the result. We denote by κ the security parameter and implicitly assume that all other parameters (e.g., the number of parties, and the complexity of the functionalities and protocols) depend in κ.

NICOM. A NICOM consists of two PPT algorithms $(\mathsf{commit}, \mathsf{open})$ where commit takes a security parameter κ, message x and random coins r, and outputs a commitment C and a corresponding opening information o. The open algorithm takes κ, and a commitment/opening pair (C, o) and outputs the message x or a failure message $\perp$. The algorithms should satisfy the standard properties of correctness, binding (i.e., it must be hard for an adversary to come up with two different openings of any C) and hiding (a commitment must not leak information about the underlying message) properties. (See Section A.3 for definition and more background.) NICOM comes in 2 main flavors: (1) with computational hiding and perfect binding, and (2) with statistical hiding and computational binding. Type (1) commitments can be based on injective one-way functions [15, 59, 39], and type (2) commitments can be based on collision resistance hash functions [29, 43]. In the latter case, a description of a collision resistance hash function h (that is sampled from a family $\mathcal{H}$) is given to the algorithms $(\mathsf{commit}, \mathsf{open})$ as an auxiliary public parameter. Our protocols make use of NICOM in a modular way such that a type (1) instantiation (with sub-exponential computational hiding) yield computational protocols and type (2) instantiation yield protocols with everlasting security. The proofs typically treat both notions in a unified way with minor adaptations when needed.

3 Technical Overview

On short interactions and long write-ups. Intuitively, a SIF protocol consists of the following sequential parts: (1) The dealer presents a statement; (2) The other parties challenge it via a random challenge; (3) The dealer sends a respond; and (4) The other parties decide whether to accept or reject. Compressing these steps into 2 rounds is highly challenging. For comparison, even the task of verifiable secret sharing (without revealing it) takes at least 2 rounds [36, 7]. To bypass this problem, we are forced to run sub-protocols in parallel and with some overlap. Specifically, we make an extensive use of (1) *tentative-output* protocols that prepare a tentative version of the output in an early round and only later, at the end, approve/reject/correct the tentative output; and (2) *offline-phase* protocols that begin with an *offline*, input-independent, round and only later receive the inputs. This allows us to save some rounds by allowing partial overlap between sub-protocols, but it also significantly complicates the presentation due to use of *reactive* MPC functionalities whose description is somewhat lengthy and tedious. In an attempt to make the paper more reader-friendly, we provide in this section a detailed overview of the protocols and the underlying ideas. The main body of the paper consists of formal statements and the appendices are devoted to the full proofs and additional background.

Our protocol makes an extensive use of *verifiable secret sharing* (VSS) [24]. For now, let us think about a VSS protocol as an actively-secure realization of the ideal functionality that takes as an input a secret $s \in \mathbb{F}$ and randomness r from a dealer, and delivers to each party P_i a share s_i that is generated from s and r by using some threshold secret sharing scheme with threshold t. Here and throughout the paper, $\mathbb{F}$ is a finite field whose size is assumed to exponential in the security parameter κ, by default, $\mathbb{F} = \mathrm{GF}(2^\kappa)$. The underlying secret sharing scheme should be *binding* in the sense that a corrupted party cannot "lie" about its share. (This property implies that correct reconstruction is achievable even at the presence of an active adversary as long as we have $n - t$ honest parties.) To simplify the exposition, let us assume for now that the underlying seceret sharing is *linearly homomorphic* and that the VSS protocol takes a *single round*. We emphasize that both features are unrealistic and even impossible to achieve when $t > n/3$, let alone when t is close to $n/2$.[6] Jumping ahead, a considerable part of this work will be devoted to the removal of these assumption while preserving the round complexity; see Section 3.3.

3.1 SIF for Few Parties

Let us restrict our attention to the case where the number of parties n is small, i.e., $n = O(\log \kappa)$. Recall that our goal is to construct a 2-round protocol for a general SIF functionality whose first round is an offline round that does not depend on the input of the dealer. We will use standard techniques to reduce this problem to the problem of constructing a 2-round protocol for a specific SIF functionality known as *triple secret sharing* (TSS) where the dealer wishes to share a triple (a, b, c) such that $c = ab$. For TSS, let us strive for a "standard" 2 round protocol whose first round is allowed to depend on the input.

2-round TSS against non-rushing adversary. Our starting point is the following 2 round protocol that assumes that a corrupted dealer is non-rushing. In the first round, the dealer D, that holds a triple (a, b, c) with $c = ab$, picks three polynomials $A(x), B(x)$ and $C(x)$ of degree n, n and $2n$, respectively, such that $A(0) = a$, $B(0) = b$, $C(0) = c$ and $C(x) = A(x) \cdot B(x)$. Let A^i, B^i and C^i be the ith coefficient of $A(x), B(x)$ and $C(x)$, and note that $A^0 = a, B^0 = b$ and $C^0 = c$. The dealer shares all the coefficients $\{A^i, B^i\}_{i\in\{0,\ldots,n\}}$, and $\{C^i\}_{i\in\{0,\ldots,2n\}}$ via VSS. The parties now hold the shares of $a = A^0, b = B^0$ and $c = C^0$.

In order to ensure that $c = ab$, it suffices to verify that the polynomial $C(x)$ is equal to the polynomial $A(x) \cdot B(x)$. To this end, we want to compute $A(\alpha), B(\alpha)$ and $C(\alpha)$ for a random non-zero field element α, and verify that $C(\alpha) = A(\alpha)B(\alpha)$. Indeed, if $C(x) = A(x) \cdot B(x)$ then equality always holds, while if $C(x) \neq A(x) \cdot B(x)$ then the probability that the verification succeeds is at most $2n/(|\mathbb{F}| - 1) = \mathsf{negl}(\kappa)$. Therefore, in the first round, concurrently to the sharing of the dealer, we let every party P_i broadcast a random non-zero field element α_i.

In the second round, our goal is to compute $A(\alpha_i), B(\alpha_i), C(\alpha_i)$ for all $i \in \{1, \ldots, n\}$ and "disqualify the dealer" if for some α_i the test $A(\alpha_i) \cdot B(\alpha_i) = C(\alpha_i)$ fails. Recall that $A(x)$ and $B(x)$ are random polynomials of degree n conditioned on $A(0) = a$ and $B(0) = b$, and therefore one can safely release all these α_i evaluations without revealing any information on a, b and c. The actual computation of $A(\alpha_i), B(\alpha_i), C(\alpha_i)$ makes use of the linear-homomorphism of the secret-sharing. Specifically, observe that $A(\alpha)$ is just a linear function of $A^0, \ldots, A^n$ with coefficients $(\alpha^0, \ldots, \alpha^n)$ (and similarly for $B(\alpha)$ and $C(\alpha)$), and therefore each party can reveal in the second round its share of $A(\alpha_i)$ (resp., $B(\alpha_i), C(\alpha_i)$). The binding property of the VSS guarantees that a corrupted party cannot lie about its shares and the existence of $t + 1$ honest parties guarantees successful reconstruction. The protocol follows the standard commit-challenge-response template with a minor tweak: many challenges are generated (one for each "verifier") concurrently to the commitment stage, and each of the responses is being computed collectively by the "verifiers".

Coping with a rushing adversary. The above protocol is insecure against a rushing adversary since such an adversary can wait to see the selected challenges and then share triples that do not

[6] Even without homomorphism, computational VSS requires 2 rounds [7] when $n < 3t$. Moreover, even for such a large resiliency threshold, linear homomorphism is non-trivial to achieve. Specifically, for 2-round VSS, it is unknown how to achieve linear homomorphism without relying on strong primitives such as *homomorphic* NICOMs. The latter are typically contructed based on "structured" (public-key type) assumptions and are not known to follow from standard NICOMs.

satisfy the product relation and yet pass the tests. We solve this problem by hiding at least some of the challenges from the adversary while revealing them to enough parties so that the response (via reconstruction) can be computed in the second round. Details follow.

Consider all the possible $(t+1)$-subsets of the parties, $Q_1, \ldots, Q_N$ where $N = \binom{n}{t+1}$. In the first round, we let each subset Q_i generate a secret challenge α_i that is known only to the members of Q_i. Specifically, we define some canonical "leader" for Q_i (e.g., the party with the smallest index) and let her sample a random non-zero α_i and send it to the other members of Q_i over private channels. Concurrently, the dealer shares the coefficients of the polynomials A, B, C among the n parties as before, except that now the degree of A and B is taken to be $d = N(t+1)$ and the degree of C is taken to be $2d$. In the second round, each party P_j in Q_i broadcasts the value α_i and uses local linear operations to reveal to all the parties the jth share of $A(\alpha_i), B(\alpha_i)$ and $C(\alpha_i)$. After the second round, for each i, each party P (possibly outside Q_i) verifies that all the parties in Q_i broadcast the same point α_i and that their shares are valid. If one of these checks fail, we refer to the ith test as *bad* and ignore it; Otherwise, the i-th test is called *good*, and P can recover the points $A(\alpha_i), B(\alpha_i)$ and $C(\alpha_i)$. If these values satisfy the product relation, we say that the (good) test *passes*. Finally, P accepts the triple if all the good tests pass, and disqualifies the dealer otherwise.

The analysis is fairly simple. For a corrupt D, we note that there exists (at least) one set Q_i in which all the parties are honest, and that a corrupt dealer has no information about α_i in the first round. The parties in Q_i provide in the second round $t+1$ shares of $A(\alpha_i), B(\alpha_i)$ and $C(\alpha_i)$ and so these values can be publicly recovered, and the probability that $C(x) \neq A(x) \cdot B(x)$ and $C(\alpha_i) = A(\alpha_i) \cdot B(\alpha_i)$ is at most $2d/(|\mathbb{F}|-1) = 2N(t+1)/(|\mathbb{F}|-1) = \mathsf{negl}(\kappa)$. Thus, except with negligible probability, there will be at least one good test that fails to pass. On the other hand, an honest dealer will never be disqualified since, by the binding property of the secret sharing, even a fully corrupted set of verifiers Q_i cannot reveal incorrect shares. As for privacy, there are N sets, and from each set the adversary can learn information about at most $(t+1)$ points of $A(x), B(x)$ and $C(x)$ (a corrupt leader in a set Q can send different evaluation points to the parties in Q). Since the degree of $A(x)$ and $B(x)$ is d, and the adversary can learn information about at most $N(t+1) = d$ points, we conclude that the adversary learns no information about $A(0), B(0)$ and $C(0)$, as required. The complexity of the protocol is exponential in $t = \lceil n/2 \rceil - 1$ and so the protocol is efficient (polynomial in the security parameter κ) only when the number of parties n is logarithmic in κ. Indeed, this is the only place where we use the assumption $n = O(\log \kappa)$. (See Section 5.1 for more details.)

From TSS to public SIF. By the standard NP-completeness of quadratic equations, public SIF non-interactively reduces to public SIF where f computes a vector of degree-2 polynomials over an arbitrary finite field [37] and the same output is given to all the parties. One can easily adopt the TSS protocol to the case of general degree-2 SIF functionality (e.g., share the input vector $\mathbf{z}$ and the output vector $\mathbf{y}$, prove that they satisfy a degree-2 relation and ask the parties to publicly reconstruct $\mathbf{y}$.) However, this will not lead to an offline/online protocol. Instead, we use Beaver's trick [11] to transform random triple sharing (realized by TSS) into a degree-2 SIF. The standard transformation has an overhead of 2 additional rounds, and we avoid it by exploiting the SIF setting, i.e., the fact that a single dealer knows *all* the secrets. For details, we refer to Section 5.2. A reduction from general SIF to public SIF appears in Section K.

3.2 SIF for any Number of Parties

We move on to the case where the number of parties, n, is large (polynomial in κ) and the resiliency threshold t is almost optimal, i.e., $n = (2+\epsilon)t$ for some constant $\epsilon > 0$. Our goal is to construct a 2-round offline/online protocol Π for some public SIF functionality $\mathcal{F}$ that takes an input $\mathbf{z}$ from the dealer D and delivers the same output $\mathbf{y} = f(\mathbf{z})$ to all the parties.

We will handle this case by composing two protocols: (1) The aforementioned 2-round SIF protocol Π_s ("s" for small) that achieves an optimal resiliency for a small (logarithmic) number of parties; and (2) a perfectly-secure SIF protocol Π_b ("b" for big) with constant resiliency of, say 1/3, that works efficiently for polynomially many parties. The latter protocol can have many rounds and can be instantiated, for example, by the classical protocol of Ben-Or, Goldwasser and Wigderson (BGW) [13]. We will combine the 2 protocols into a single SIF protocol with almost-optimal threshold and $\mathrm{poly}(n)$ complexity via *player virtualization* technique. This idea goes back to the work of Bracha [19] in the context of Byzantine Agreement, and since then has been used several times in the MPC literature [34, 44, 28] culminating in the celebrated MPC-in-the head paradigm [48, 49]. Here we show how to apply this idea in the context of SIF. Unlike other contexts, we show that the combined protocol inherits the round complexity of the first ("internal") protocol, and therefore can be executed in 2 rounds! Details follow.

Let us partition the n parties to $M = \mathrm{poly}(n)$ committees $A_1, \ldots, A_M$ each of size n' for some constant n' that depends on the constant ϵ. Call a committee *good* if it contains at least $(n'+1)/2$ honest parties, and *bad* otherwise. We will make sure that the fraction of bad sets is at most $M/10$ no matter which subset of t parties the adversary decides to corrupt. Such a property can be guaranteed by taking all n' multisets (see Observation J.2) or, more efficiently, based on expander graphs (see, e.g., [28, Lemma 5]).[7] Let Π_b be the BGW protocol that realizes the SIF f among the dealer D and M "virtual" parties $Q_1, \ldots, Q_M$.

In our new protocol, Π, the dealer D executes the BGW protocol Π_b in her "head" with the input $\mathbf{z}$ and then broadcasts a commitment to the transcript. That is, D samples random tapes $r_1, \ldots, r_M$ for the virtual parties $Q_1, \ldots, Q_M$ and computes all the messages that are sent in Π_b, both over private channels and over broadcast channels. Then, D commits to each of these messages and to the randomness r_i of each party Q_i, and broadcasts the tuple of commitments G. In addition, D broadcasts the value $\mathbf{y} = f(\mathbf{z})$. Now, we let each committee A_i verify, with the aid of the small protocol Π_s, that the view of Q_i is *self-consistent*, i.e., that the (committed) randomness and incoming messages of Q_i yield the (committed) outgoing messages of Q_i and that the final output is indeed $\mathbf{y}$. More precisely, the committee A_i together with D, compute the following public-SIF functionality $\mathcal{G}_{\mathsf{zk}}$:

- (Dealer's input:) An index $i \in \{1, \ldots, M\}$, a vector of commitments G_i, supposedly to the randomness of Q_i and his incoming and outgoing messages, and the corresponding openings.
- (Public output:) the tuple $(v_i, \mathbf{y}_i, G_i, i)$ where v_i is a consistency bit that indicates whether the committed values are self-consistent, and the value $\mathbf{y}_i$ is the output that the virtual party

[7] In principle, n' should be taken to be $\Omega(1/\epsilon^2)$. Thus, in order to keep n' small (e.g., logarithmic in the security parameter), one has to assume that ϵ is not too small, e.g., at least $\Omega(1/\sqrt{\log \kappa})$. We limit the discussion to a constant ϵ only for the sake of simplicity.

Q_i outputs given the committed view.[8]

We realize this sub-computation by running the small SIF protocol Π_s among D and the sub-committee A_i while making sure that the final output is available to all parties including ones that do not belong to A_i. This can be done (without an extra round of communication) by passing all the broadcast messages of the small protocol Π_s over the external n-party broadcast channel. Indeed, we note that, for public-output SIF, the public output of Π_s can be fully recovered based on its broadcast messages. Getting back to Π, we conclude the protocol, by letting each party P_i accept the output $\mathbf{y}$ if at least $0.9M$ of the committees approve this output (i.e., if the output of the ith committee is $(1, \mathbf{y}, G_i, i)$ where G_i is consistent with G), and disqualify the dealer otherwise.

The protocol Π can be executed in 2 rounds where the first round is devoted to the offline round of all the instances of the Π_s protocol, and the second round is devoted to the commitment generation and to the second online-round of the Π_s instances. Note that the first round of Π remains input-independent. Let us briefly analyze the security of Π.

For an honest dealer, the verification Π_s succeeds for every good committee Q_i that contains an honest majority, and may fail for a bad committee Q_i that contains a dishonest majority. We conclude that at most $M/10$ of the verifications fail, and so an honest dealer will never be disqualified. As for privacy, a bad committee Q_i may completely learn the input of the dealer D in the corresponding SIF $\mathcal{G}_{\mathsf{zk}}$. This leakage is equivalent to learning the internal state of the virtual party Q_i in the external protocol Π_b. Since there are at most $M/10$ bad committees, the adversary can learn the state of at most $M/10$ parties of Π_b. The privacy of Π_b therefore protects us against such a leakage. (In fact, for this part we only use the privacy of Π_b against a passive corruption.)

A corrupt dealer can commit to an illegal transcript while being approved by all bad committees. So, in order to be approved, such a dealer must still get the votes of at least $0.8M$ good committees. Hence, cheating in Π reduces to cheating in Π_b while actively controlling at most $0.2M$ of the virtual parties, and while controlling the randomness of the honest virtual parties. Since Π_b is *perfectly correct* against $0.2M$ active corruptions, a cheating dealer will always be caught. (For this part, no privacy is needed and Π_b is only required to achieve "perfect correctness with abort" against an active adversary.) For more details see Section J.

Remark 3.1 (Comparison to the MPC-to-ZK transformation of [48])**.** *It is instructive to consider the following variant of the protocol. First, the dealer secret-shares its input $\mathbf{z}$ to $(\mathbf{z}_1, \ldots, \mathbf{z}_M)$ via some robust $M/3$-out-of-M secret sharing then it virtually runs an MPC protocol among the parties $Q_1, \ldots, Q_M$ for the public SIF $\mathcal{F}'$ that takes $(\mathbf{z}_1, \ldots, \mathbf{z}_M)$ from the parties, recovers $\mathbf{z}$ via robust reconstruction, and delivers the output $f(\mathbf{z})$. The dealer commits to the views and transcript and the committees $A_1, \ldots, A_M$ use the small SIF protocol to verify consistency for each virtual party. This description be can viewed as a special case of the protocol Π in which Π_b is realized by sharing $\mathbf{z}$ and computing $\mathcal{F}'$.*[9]

Under this choice, our transformation can be viewed as a multi-verifier version of the MPC-to-ZK transformation of [48]. The two versions differ with respect to the underlying secret sharing (M-out-of-M in [48] vs. $M/3$-out-of-M in our case), and, more importantly, with respect to the verification part. In [48] a single verifier opens few views (for soundness) while keeping other views unopened (for zero-knowledge), whereas in our case multiple verifiers distributively open (all) the views in a way that preserves soundness "globally", and secrecy for bounded-size coalitions. Furthermore, we show that verification can be realized with low round complexity based on an "internal" SIF protocol.

[8]The circuit that realizes $\mathcal{G}_{\mathsf{zk}}$ depends on the code of the NICOM, consequently, our final construction makes a non-black-box use of the NICOM.

[9]For technical reasons, we follow this structure in the formal description of the protocol in Section J.

3.3 Replacing the Idealised VSS with 1.5-Round Protocols

In the previous section, TSS and public SIF for logarithmic number of parties are the direct consumers of the idealized VSS. In both, the scenario is as follows: D has m inputs $s_1, \dots, s_m$ and the parties want to compute a linear combination of the inputs. The coefficients of the linear combination may be chosen by some other party, and the output should be delivered by the end of second round. For simplicity, we consider the somewhat degenerate case where the goal is to compute $z := s_1 + \ldots + s_m$. As mentioned earlier, two challenges arise: (a) VSS sharing itself requires 2 rounds, whereas our requirement is to complete sharing and reconstruction within 2 rounds and (b) the known 2-round VSS from Minicrypt-like assumptions is not homomorphic. In a nutshell, we solve the first issue by noting that the VSS of [7] is a "1.5-round" VSS in the sense that "tentative shares" are distributed already in the first round, and any update that may occur in the second round is *publicly known* to all parties. To solve the second issue, we construct a novel protocol that allows a party to reveal a "certified" linear combination of its shares. This protocol, glinear, has 2 rounds where the first round is an offline round. Since our protocols employ linear homomorphism during their second round, glinear forms a viable substitute. Related tools have been developed in [5] for a smaller resiliency threshold (e.g., $n \geq 3t + 1$), and we extend them to the challenging setting of $n = 2t + 1$ while maintaining efficiency for polynomially many parties $n = \mathrm{poly}(\kappa)$. Before describing our solutions in more detail, we present some background on the underlying secret sharing scheme.

The underlying secret sharing scheme. The secret sharing scheme is essentially the classical t-out-of-n Shamir-like scheme (extended to bivariate polynomials as in [13]) accompanied with public commitments to all the shares. To (honestly) share a secret $s \in \mathbb{F}$, one samples a random symmetric bivariate polynomial $F(x, y)$ of degree at most t in each variable conditioned on $F(0, 0) = s$, and hands to each party P_i the vector $(F(i, 0), \dots, F(i, n))$ which fully defines the degree-t univariate polynomial $f_i(x) = F(i, x)$. We embed these elements in an $(n + 1)$-by-$(n+1)$ matrix $\mathbf{F} = (F(i, j))_{i,j \in \{0,\dots,n\}}$, and note that this matrix is symmetric since $F(i, j) = F(j, i)$. The 0th row of this matrix is referred to as the *main* row and its ith entry $F(0, i) = F(i, 0)$ is referred to as the main share of party P_i. (The main row corresponds to the univariate polynomial $f_0(x) = F(0, x)$ which forms a standard Shamir sharing of s.) As part of the secret sharing, we publish a symmetric matrix, $\mathbf{C} = (C_{ij})_{i,j \in \{0,\dots,n\}}$ of commitments to each entry of $\mathbf{F}$, and hand the openings, $\mathbf{O}_i = (o_{ij})_{j \in \{0,\dots,n\}}$, of the ith row to party P_i. We let $\mathbf{O}$ denote the matrix of openings $(o_{ij})_{i,j \in \{0,\dots,n\}}$. It is well-known that this scheme is t-out-of-n secret sharing scheme. The commitment layer makes it impossible for a corrupted party to lie about its share (the scheme is "binding"), and so it enables robust reconstruction.[10] We point out that a statistically-hiding computationally-binding commitment leads to a secret sharing scheme with statistical privacy whose robustness holds only against computationally-bounded adversaries whereas a computationally-hiding statistically-binding commitment scheme yields with a secret sharing scheme with computational privacy and robustness against computationally-unbounded adversaries. Let us record the fact that the "polynomial part" of the secret sharing is linearly homomorphic but the "commitment part" is not.

[10]We, in fact, consider a weak variant of this sharing in which for a pair of corrupted parties, (P_i, P_j), the share $f_i(j)$ may be inconsistent with the commitment C_{ij}. Still, it can be shown that P_i and P_j cannot lie about their *main shares* and so this scheme still allows robust reconstruction. For details, refer to Section D.

1.5-round VSS. Backes et al. [7] describe a 2-round protocol for securely distributing a secret according to the above secret sharing scheme. We note that this protocol has the following structure. After the first ("sharing") round, the commitment matrix $\mathbf{C}$ is delivered to all the parties and each party holds a private *tentative share* that may be invalid. During the second ("verification") round of the protocol, each party P_i who may be "unhappy" for some reason, can form a "complaint" against the dealer D. At the end of this round, either some complaint turns to be "justified", or all the complaints are rejected as being "unjustified". In the former case, the dealer is being publicly disqualified, and in the latter case, the private shares of all unhappy parties are publicly revealed. (That is, all parties learn the openings $(\mathbf{O}_i)_{i \in \mathsf{W}}$ where W is the set of all unhappy parties.) By design, an honest party never complains about an honest dealer. We will make use of the fact that a tentative share either remains unchanged during the second round, or becomes publicly available to all parties.

We formalize these properties via a new 2-phase functionality $\mathcal{F}_{\mathsf{vss}}$ (a refined version of VSS), and prove that the protocol UC-realizes it. The choice of being unhappy is captured by an input $\mathsf{flag}_i \in \{0, 1\}$ that is given to P_i at the beginning of the verification phase. As a result P_i can ask to publicly reveal $\mathbf{O}_i$ even it is unhappy with D due to some external reason, that does not depend on the VSS execution (say, P_i thinks that D is corrupt in the outer-protocol). See Section 4.1.

3.3.1 Supporting Linear Operations

Let us now go back to our goal of computing $z := s_1 + \ldots + s_m$ in two rounds where the secrets $s_1, \ldots, s_m$ are given to D as inputs. We start by running the first round of the VSS to distribute tentative shares for $s_1, \ldots, s_m$ via the polynomials $F^1, \ldots, F^m$ and the commitments $\mathbf{C}^1, \ldots, \mathbf{C}^m$. Our goal now is to publicly reveal the value $z := s_1 + \ldots + s_m$ by using a single round of communication that will be carried in parallel to the verification phase of the VSS. Denote by $F^z(x, y)$ the bivariate polynomial $F^1(x, y) + \ldots + F^m(x, y)$. Observe that it suffices to design a single-round protocol that allows to each party P_i to publish the univariate polynomial $F^z(i, \cdot)$ while providing a certificate for correctness (and while hiding the original shares). Formally, for every "guide" P_i the parties engage in a subprotocol $\mathsf{glinear}$ ("guided linear computation") so that (1) if P_i is honest then all parties output $F^z(i, x)$, and (2) if P_i is corrupt then all parties output either $F^z(i, x)$ or an erasure $\perp$. Since there are $n - t \geq t + 1$ honest parties, and all non-$\perp$ shares are consistent with $F^z(x, y)$, the parties can recover the polynomial $F^z(x, y)$ and output $z = F^z(0, 0)$. Observe that we can restrict our attention to the case where the guide is "happy" with the dealer D, since the shares of a non-happy guide will be publicly released anyway in the end of the second round by the verification phase of the secret sharing.

Guided linear computation from SCG. To explain how $\mathsf{glinear}$ is implemented, let us focus, for concreteness, on the case where the guide is P_1. After the input sharing, the guide P_1 holds all the information regarding the first rows $F^1(1, x), \ldots, F^m(1, x)$, including the openings to the corresponding commitments. In addition, every P_j holds all the information regarding the j-th share of each first-row, $F^1(1, j), \ldots, F^m(1, j)$. The idea now is to let the guide P_1 and every P_j engage in a subprotocol for the computation of $F^z(1, j)$ where the role of P_j is to *guard* the computation, i.e., to make sure that P_1 uses the "correct" values as inputs. Formally, we construct such a subprotocol, called *secure computation with a guard* and denoted scg, that has essentially the following "patrial security" guarantees:

- If both, P_1 and P_j, are honest then the value $F^z(1,j)$ is given to all parties while the values, $\vec{F}(1,j) := (F^1(1,j), \ldots, F^m(1,j))$, remain hidden.
- If P_1 and P_j are both corrupt, there are no correctness or privacy guarantees.
- If exactly one party is corrupt (either P_1 or P_j) then there are no privacy guarantees and the public output is either $F^z(1,j)$ or an identifiable abort (i.e., $\perp$ symbol accompanied with the identity of the corrupt party).

We postpone the description of the scg protocol. For now, let us mention that the protocol is *publicly decodable* (all honest parties receive the same output that is computed based on broadcasted values), and has 2 rounds in the offline/online model. Since the first round is input-independent we can execute it in parallel to the first round of VSS. Now glinear can be reduced to n executions of scg between P_1 and each of the parties $P_1, \ldots, P_n$, where each P_j acts as the guard of the computation of $F^z(1,j)$. Given the scg outputs, we output a degree-t polynomial $f_1(\cdot)$ if and only if (1) P_1 was not disqualified by any of the scg calls, and (2) $f_1(\cdot)$ is consistent with all the revealed points. Otherwise, we disqualify P_1. The analysis is straightforward. If P_1 is honest, for every honest guard P_j all the parties learn $F^z(1,j)$ (without leaking information on $\vec{F}(1,j)$), while for every corrupt P_j the parties either learn $F^z(1,j)$ or an erasure $\perp$ (since the adversary already knows $\vec{F}(1,j)$ we do not care about leakage in this case). Since there are $n-t \geq t+1$ honest parties, the parties recover uniquely the polynomial $F^z(1,x)$. If P_1 is corrupt, then it is either being disqualified by one of the honest guards, or release at least $n-t \geq t+1$ points that are consistent with $F^z(1,\cdot)$. This means that the final outcome is either $F^z(1,\cdot)$ or $\perp$. (See Section 4.3.) Before delving into the scg construction, we mention that the VSS together with the guided linear computation lead to a protocol for general linear function evaluation in 3 rounds which is optimal by [37]. We refer to Section 4.4 for details.

Realizing scg. Roughly speaking, in an scg protocol, the guide Alice is given as an input a vector b^A and the guard Bob receives a copy, b^B, of this vector that supposedly agrees with b^A. Alice wishes to publicly reveal the value $f(b^A)$, for some public function f, and the guard Bob should make sure that f is computed consistently with respect to his input. This notion was introduced by [4] who constructed a 2-round offline/online protocol that statistically realizes the partial security properties defined above. However, their protocol works with a designated receiver, and so multiple invocations of this protocol (with different receivers) may lead to inconsistent outputs. (Such inconsistencies were tolerated in [4] by leveraging the existence of a strong honest majority, i.e., $t < n/3$.) We present a publicly decodable scg by exploiting the fact that all parties are given *external commitments* C to the input b^A and that the corresponding openings, o, are given to Alice as certificates. Moreover, we make use of NICOM internally in the scg itself, and so get only computational security. Details follow.

Thanks to the external commitments, it suffices to securely compute the functionality $\mathcal{F}$ that takes $x = (b^A, o)$ from Alice and $y = b^B$ from Bob, and outputs

$$y = \begin{cases} f(b^A), & \text{if } b^A = b^B, \\ (b^A, o) & \text{otherwise.} \end{cases}$$

Indeed, if Alice and Bob are honest the output will be $f(b^A)$. If the parties disagree (due to a single cheater) then the output reveals Alice's certified input, and one can check whether the released values (b^A, o) are consistent with the external commitments or not. In the former case, we can decode the output $f(b^A)$, and in the latter case, we conclude that Alice aborted the computation. While we will not be able to realize $\mathcal{F}$ with full security, we provide an instantiation that suffices for "partial security".

Our starting point is the following variant of private simultaneous message (PSM) protocol of [31]. Bob samples a random string r and sends it to Alice privately during the offline phase. Then, in the online phase, given the inputs, x and y, Alice and Bob publish messages, $A(x, r)$ and $B(y, r)$, that publicly reveal $\mathcal{F}$ and nothing else. Unfortunately, the standard PSM realization only works when both parties are honest, and a dishonest party, say Alice, can violate correctness by sending an invalid message a' that does not correspond to any input x (with respect to the chosen r).

Focusing on the case of corrupt Alice, we modify the protocol as follows. At the offline round, Bob broadcasts *internal commitments* to all the possible PSM online-messages. That is, for every possible Alice-input x (resp., every possible Bob-input y), Bob computes a commitment C'_x to the PSM message $A(x, r)$ (resp., C'_y to the PSM message $B(y, r)$). At the offline round, Bob broadcasts the (randomly permuted) list of commitments $(C'_x)_x$ and $(C'_y)_y$ and privately sends to Alice all the information: the PSM randomness r together with the corresponding openings $(o'_x)_x$ and $(o'_y)_y$. At the online round, Alice and Bob compute the PSM messages that correspond to their inputs, and certify them by opening the corresponding internal commitments. Now, assuming that Bob is honest, Alice is forced to behave honestly in the PSM and must send a "valid" PSM message that corresponds to an actual input x. This protocol achieves a similar guarantee against a cheating Bob and honest Alice, provided that Bob behaves *honestly* in the offline round. We handle the case where Bob misbehaves in the offline round (e.g., by committing to bad values or sending to Alice bad openings) by letting Alice fully expose her certified input. That is, if Alice sees that Bob misbehaved in the offline round, she simply broadcasts her inputs together with the external openings as certificates while ignoring the PSM execution. Here we exploit the fact that no privacy is required at the presence of a cheating Bob.

The above description is somewhat simplified and yields a solution whose complexity is linear in the domain of $\mathcal{F}$ which is too expensive. Moreover, when scg is modelled as a reactive functionality, simulation becomes somewhat subtle and the commitments should satisfy some level of security under a selective-openning attack. More details (including an efficient version based on multiparty PSM protocols and a refined definition of scg) appear in Section 4.2.

4 Secure Linear Function Computation

Here, we discuss VSS, SCG, guided linear and general linear computation.

4.1 Verifiable Secret Sharing

Following the discussion in Section 3.3, we present functionality $\mathcal{F}_{\mathsf{vss}}$. The functionality consists of two phases: a sharing phase and a verification phase. In the sharing phase, the dealer D inputs a commitment matrix $\mathbf{C} = (C_{ij})_{i,j \in \{0,\ldots,n\}}$ and an opening matrix $\mathbf{O} = (o_{ij})_{i,j \in \{0,\ldots,n\}}$. If D is honest, then the input $(\mathbf{C}, \mathbf{O})$ is "well-formed" in the following sense: the matrices $\mathbf{C}$ and $\mathbf{O}$ are symmetric, and there exists a symmetric bivariate polynomial $F(x, y)$ of degree at most t in each variable, such that the value committed by (C_{ij}, o_{ij}) is $F(i, j)$, for all $i, j \in \{0, \ldots, n\}$.[11] We think of $s := F(0, 0)$ as the *secret*. The functionality returns to each P_i the list of all commitments $\mathbf{C}$, as well as the list of the i-th row of openings $\mathbf{O}_i := (o_{ij})_{j\in\{0,\ldots,n\}}$. We call the pair $(\mathbf{C}, \mathbf{O}_i)$ the *tentative share* of P_i, and we say that it is *valid* if the matrix $\mathbf{C}$ is symmetric, and if there exists a degree-t polynomial $f_i(x)$, such that the value committed by (C_{ij}, o_{ij}) is $f_i(j)$ for all $j \in \{0, \ldots, n\}$. Note that the tentative shares produced by an honest dealer are always valid.

In the verification phase, each party P_i inputs a flag flag_i. We do not intend to keep the flags private, and so we leak them to the adversary. If D is honest then the flags of all honest parties are 0, and the functionality simply returns to all parties the set W of all (corrupt) parties that raised a flag, together with their openings $(\mathbf{O}_i)_{i\in\mathsf{W}}$. A corrupt dealer is allowed to have two additional inputs: a flag flag_D, and alternative openings $\bar{\mathbf{O}} = (\bar{o}_{ij})_{i,j\in\{0,\ldots,n\}}$ that will replace the tentative openings of unhappy parties. If $\mathbf{C}$ is not a symmetric matrix, or if $\mathsf{flag}_D = 1$ then the output of the verification phase is "D is corrupt".[12] Otherwise, let W be the set of all honest parties with invalid tentative shares, together with all parties that raised a flag. The functionality verifies that there exists a symmetric bivariate polynomial $F(x, y)$ of degree at most t for which (1) the value committed by (C_{ij}, o_{ij}) is $F(i, j)$ for all P_i in $\mathsf{H} \setminus \mathsf{W}$ and $j \in \{0, \ldots, n\}$, and (2) the value committed by $(C_{ij}, \bar{o}_{ij})$ is $F(i, j)$ for all P_i in W and $j \in \{0, \ldots, n\}$. If the verification fails then the functionality returns "D is corrupt" to all parties, and otherwise, it returns $(\mathsf{W}, (\bar{\mathbf{O}}_i)_{i\in\mathsf{W}})$ to all parties.[13] In Section D we formally define $\mathcal{F}_{\mathsf{vss}}$, present the protocol vss, and prove the following theorem.

Theorem 4.1. *Let κ be a security parameter, n be the number of parties with $t < n/2$, and let $\mathbb{F}$ be a field. Assuming the existence of perfectly-binding NICOM, there exists a 2-round protocol* vss *which is a UC-secure implementation of $\mathcal{F}_{\mathsf{vss}}$, against a static, active, rushing adversary corrupting up to t parties. The complexity of* vss *is* $\mathrm{poly}(n, \log|\mathbb{F}|, \kappa)$.

Alternatively, if we replace the perfectly-binding NICOM with statistically-hiding NICOM, we also obtain everlasting security.

On tentative shares and final shares. At the end of the first round each P_i holds the values $(\mathbf{C}, \mathbf{O}_i)$, which we call *tentative shares*. When D is honest then all parties hold valid tentative shares, and the tentative shares will become the *final shares* at the end of the second round. When D is corrupt, it may happen that only some of the parties hold valid tentative shares, and those may change only if D is found out to be corrupt in the verification phase.

[11]Note that, in the UC-framework, this means that *the environment* picks the sharing $(\mathbf{C}, \mathbf{O})$. When the environment gives the honest parties inputs that are not well-formed (for example, when an honest D receives $(\mathbf{C}, \mathbf{O})$ that are not exactly as mentioned), a *complete break-down* occurs (see Section A.1 for a formal definition) and we are able to simulate. We will ignore this issue when presenting the functionalities and protocols, but we do analyse it in the proof of security.

[12]If $\mathbf{C}$ is not symmetric then the parties know that D is corrupt already in the sharing phase. To simplify the functionality and have only one round of disqualification of the dealer, we postpone this case to the verification phase.

[13]For a corrupt dealer, we are only promised that the public shares and the shares of the honest parties are consistent with the sharing polynomial $F(x, y)$. However, there might be a corrupt party P_i, and an index $j \in \mathsf{C}$ such that the value committed by (C_{ij}, o_{ij}) is not $F(i, j)$. We call such a sharing a *weak-sharing*. In contrast, for an honest D, we are promised that for every $i, j \in \{0, \ldots, n\}$ the value committed by (C_{ij}, o_{ij}) is $F(i, j)$, and we call such a sharing a *strong-sharing*. See Section D for formal definitions. The distinction between strong sharing and weak-sharing will not be important in the following sections.

Public outputs. An external party, that can only listen to the broadcast channel, learns the following public outputs (for a formal discussion about public outputs, refer to Section A.1): (1) the commitments $\mathbf{C}$ in the sharing phase, and (2) the output of the verification phase (which is either "D is corrupt" or the set W together with the public openings of all parties in W).

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

4.2 Secure Computation with a Guard

We continue with a formal definition of the functionality $\mathcal{F}_{\mathsf{scg}}$. As discussed in Section 3.3, we first consider the function f_{scg}, which is defined below.

Function f_{scg}. The function

$$f_{\mathsf{scg}}\Big(\big(\mathbf{a}, \delta^A, (o_i)_{i=1}^m, \mathbf{b}^A\big), \big(\delta^B, \mathbf{b}^B\big)\Big). \tag{1}$$

takes as input two tuples: the first one, which we think of as Alice's inputs, consists of a vector of coefficients $\mathbf{a} \in \mathbb{F}^m$, a flag $\delta^A \in \{0, 1\}$, and a list of values $\mathbf{b}^A \in \mathbb{F}^m$ and corresponding openings $(o_i)_{i=1}^m$; the second tuple, which we think of as Bob's inputs, consists of a flag $\delta^B \in \{0, 1\}$, and a vector of values $\mathbf{b}^B \in \mathbb{F}^m$. The output of the function is defined as follows:

$$\begin{cases} \perp, & \text{if } \delta^A = 1, \\ \Big(\big(\mathbf{a}, \delta^A, (o_i)_{i=1}^m, \mathbf{b}^A\big), \big(\delta^B, \mathbf{b}^B\big)\Big), & \text{if } (\delta^A = 0 \text{ and } \delta^B = 1) \text{ OR} \\ & \quad (\delta^A = \delta^B = 0 \text{ and } \mathbf{b}^A \neq \mathbf{b}^B), \\ \Big(\mathbf{a}, \delta^A, \delta^B, \sum_{i \in \{1,\ldots,m\}} a_i \cdot b_i^A\Big), & \text{otherwise.} \end{cases}$$

That is, if Alice's flag is raised, i.e., $\delta^A = 1$, the function returns a failure symbol $\perp$. Intuitively, this means that for some external reason Alice does not want to participate in the computation. Otherwise, $\delta^A = 0$. In this case, if Bob's flag is raised, i.e., $\delta^B = 1$, or if $\delta^B = 0$ but there exists some inconsistency between the vectors $\mathbf{b}^A$ and $\mathbf{b}^B$, then the function simply returns the inputs of Alice and Bob. Intuitively, if Bob's flag is raised, or if there is an inconsistency between Alice and Bob, we do not care about privacy, and allow to reveal all the information. Finally, if none of the flags is raised, and if $\mathbf{b}^A = \mathbf{b}^B$, the functionality returns the vector of coefficients $\mathbf{a}$, the flags δ^A and δ^B, and the sum $\sum_{i \in \{1,\ldots,m\}} a_i \cdot b_i^A$. The functionality is presented in Figure 1.

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

The functionality receives the set of corrupt parties C.

Honest parties' inputs:

- All honest parties input the same commitments $(C_1, \ldots, C_m)$.
- An honest Alice inputs $\mathbf{a} = (a_1, \ldots, a_m) \in \mathbb{F}^m$, $\mathbf{b}^A = (b_1^A, \ldots, b_m^A) \in \mathbb{F}^m$, $\delta^A \in \{0,1\}$ and openings $(o_i)_{i=1}^m$. If $\delta^A = 0$ then it holds that $\mathsf{open}(C_i, o_i) = b_i^A$ for each $i \in \{1, \ldots, m\}$.
- An honest Bob inputs $\mathbf{b}^B = (b_1^B, \ldots, b_m^B) \in \mathbb{F}^m$ and $\delta^B \in \{0,1\}$. If both Alice and Bob are honest, and $\delta^A = \delta^B = 0$, then $\mathbf{b}^A = \mathbf{b}^B$.

Leakage: The adversary receives $(C_1, \ldots, C_m)$. In addition, if only Alice is corrupt, then the adversary receives Bob's input; if only Bob is corrupt, then the adversary receives Alice's input. If Alice and Bob are honest, the adversary receives the output of f_{scg}, defined in Equation 1, on the inputs of Alice and Bob.

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

Public outputs: If $\delta^A = 1$ return $\perp$ to all parties. Otherwise, we split into cases.

- *(Honest Alice and Bob)* Return $(\mathbf{a}, \sum_{i \in \{1,\ldots,m\}} a_i \cdot b_i^A)$ to all parties.
- *(Honest Alice, Corrupt Bob)* If $\mathsf{flag} = 1$, then return "Bob is corrupt" to all parties. Otherwise, return $(\mathbf{a}, \sum_{i \in \{1,\ldots,m\}} a_i \cdot b_i^A)$ to all parties.
- *(Corrupt Alice, Honest Bob)* If either (1) $\mathsf{flag} = 1$, or (2) $\delta^B = 1$ and there exists $i \in \{1, \ldots, m\}$ such that $b_i^A \neq \mathsf{open}(C_i, o_i)$, or (3) $\mathbf{b}^A \neq \mathbf{b}^B$ and there exists $i \in \{1, \ldots, m\}$ such that $b_i^A \neq \mathsf{open}(C_i, o_i)$ then return "Alice is corrupt" to all. Otherwise, return $(\mathbf{a}, \sum_{i \in \{1,\ldots,m\}} a_i \cdot b_i^A)$ to all parties.
- *(Corrupt Alice and Bob)* Return z to all parties.

Figure 1: Functionality $\mathcal{F}_{\mathsf{scg}}$

In Section E.2 we provide a protocol scg with 1-offline round and 1-online round, and prove the following lemma.

Lemma 4.2. *Let κ be a security parameter, let n be the number of parties, let $t < n/2$, and let $\mathbb{F}$ be a field. Assuming the existence of perfectly-binding sub-exponentially hiding NICOM, protocol* scg *is a UC-secure implementation of $\mathcal{F}_{\mathsf{scg}}$, against a static, active, rushing adversary corrupting up to t parties. The complexity of the protocol is* $\mathrm{poly}(n, \log|\mathbb{F}|, \kappa, m)$.

Alternatively, if we replace the perfectly-binding NICOM with statistically-hiding NICOM, we also obtain everlasting security.

Public outputs. An external party, that can only listen to the broadcast channel in an execution of scg, learns the output of the scg protocol as a public output.

4.3 Guided Linear Function Computation

Following on the motivation and discussion in Section 3.3, we present Functionality $\mathcal{F}_{\mathsf{glinear}}$ in Figure 2. In Section F.1 we present a protocol $\mathsf{glinear}$ with 1 offline round and 1 online round, and prove the following lemma.

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

The functionality receives the set of corrupt parties C.

Honest parties' inputs:

- All honest parties input the same commitments $(C_{ij})_{i\in\{1,\ldots,m\},j\in\{0,\ldots,n\}}$.
- If G is honest, G inputs (1) a list of coefficients $\mathbf{a} = (a_1,\ldots,a_m) \in \mathbb{F}^m$, (2) a list of values $\mathbf{b}^G = (b^G_{ij})_{i\in\{1,\ldots,m\},j\in\{0,\ldots,n\}} \in \mathbb{F}^{m(n+1)}$, (3) a list of openings $(o^G_{ij})_{i\in\{1,\ldots,m\},j\in\{0,\ldots,n\}}$, and (4) a flag $\delta^G \in \{0,1\}$.
  If $\delta^G = 0$ then for every $i \in \{1,\ldots,m\}$ it holds that $\mathsf{open}(C_{ij}, o^G_{ij}) = b^G_{ij}$ for every $j \in \{0,\ldots,n\}$, and the values $(b^G_{ij})_{j\in\{0,\ldots,n\}}$ correspond to a degree-t polynomial.
- An honest P_j inputs a flag $\delta^j \in \{0,1\}$, and values $(b^j_1,\ldots,b^j_m) \in \mathbb{F}^m$. If both G and P_j are honest and $\delta^G = \delta^j = 0$ then $(b^G_{1j},\ldots,b^G_{mj}) = (b^j_1,\ldots,b^j_m)$.

Leakage: The adversary receives $(C_{ij})_{i\in\{1,\ldots,m\},j\in\{0,\ldots,n\}}$. In addition,

- if G is honest and $\delta^G = 0$, the adversary receives $\mathbf{a}$, $(b^G_{ij}, o^G_{ij})_{i\in\{1,\ldots,m\},j\in\mathsf{C}}$ and δ^G, $(\delta^j)_{j\in\mathsf{H}}$, the sum $\sum_{i\in\{1,\ldots,m\}} a_i \cdot b^G_{ij}$ for any $j \in \{1,\ldots,n\}$, and $(b^G_{ij}, o^G_{ij})_{i\in\{1,\ldots,m\}}$ for every honest P_j with $\delta^j = 1$.
- if G is corrupt the adversary also receives $(b^j_1,\ldots,b^j_m)$ and δ^j for every $j \in \mathsf{H}$.

Adversary's inputs:

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

Public outputs: If $\delta^G = 1$ return $\perp$ to all parties. Otherwise, we split into cases.

- **Honest guide.** The functionality returns $(\mathbf{a}, \sum_{i:\delta^G_i=1} a_i \cdot b^G_{i,0})$ to all parties.
- **Corrupt guide.** The functionality returns "G is corrupt" if either (1) $\mathsf{flag} = 1$, or (2) there exists $j \in \mathsf{H}$ with $\delta^j = 1$, and $\mathsf{open}(C_{ij}, o_{ij}) \neq b^G_{ij}$ for some $i \in \{1,\ldots,m\}$, or (3) there exists $j \in \mathsf{H}$ with $\delta^j = 0$ and $(b^G_{1j},\ldots,b^G_{mj}) \neq (b^j_1,\ldots,b^j_m)$, and $\mathsf{open}(C_{ij}, o_{ij}) \neq b^G_{ij}$ for some $i \in \{1,\ldots,m\}$. Otherwise, for each $j \in \mathsf{H}$ let $v^j := \sum_{i\in\{1,\ldots,m\}} a_i \cdot b^G_{ij}$, and let $g(x)$ be the polynomial obtained by interpolating $(v^j)_{j\in\mathsf{H}}$. If the degree of $g(x)$ is more than t then the functionality returns "G is corrupt". Otherwise, the functionality returns $(\mathbf{a}, g(0))$ to all parties.

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

Lemma 4.3. *Let κ be a security parameter, let n be the number of parties, let $t < n/2$, and let $\mathbb{F}$ be a field. Assuming the existence of perfectly-binding sub-exponentially hiding NICOM, protocol* glinear *is a UC-secure implementation of $\mathcal{F}_{\mathsf{glinear}}$, against a static, active, rushing adversary corrupting up to t parties. The complexity of the protocol is* $\mathrm{poly}(n, \log|\mathbb{F}|, \kappa, m)$.

Alternatively, if we replace the perfectly-binding NICOM with statistically-hiding NICOM, we also obtain everlasting security.

Public outputs. An external party, that can only listen to the broadcast channel in an execution of glinear, learns the output of glinear as a public output.

4.4 Linear Function Computation

Using the machinery developed so far, we obtain a round-optimal protocol for linear function computation, with optimal resiliency, and under Minicrypt-type assumptions. The result is summarized in Theorem 4.4, and we sketch the proof in Section G.

Theorem 4.4. *Let κ be a security parameter, let n be the number of parties, let $t < n/2$, and let $\mathbb{F}$ be a field. Assuming the existence of perfectly-binding sub-exponentially hiding NICOM, there is a 3-round UC-secure protocol for the computation of general linear functionalities, against a static, active, rushing adversary corrupting up to t parties. The complexity of the protocol is* $\text{poly}(|\mathcal{C}|, n, \log|\mathbb{F}|, \kappa)$*, where $\mathcal{C}$ is the the size of the circuit computing the linear functionality.*

Alternatively, if we replace the perfectly-binding NICOM with statistically-hiding NICOM, we also obtain everlasting security.

5 Single Input Functionalities

Here, we discuss triple secret sharing, public SIF (for small and any number of parties) and general SIF.

5.1 Triple Secret Sharing

Following the discussion in Section 3.1, here we present the functionality $\mathcal{F}_{\mathsf{tss}}$. The goal of $\mathcal{F}_{\mathsf{tss}}$ is to make a dealer D share three values $a, b, c \in \mathbb{F}$ such that $c = ab$. Like $\mathcal{F}_{\mathsf{vss}}$, functionality $\mathcal{F}_{\mathsf{tss}}$ consists of two phases, the sharing phase, and the verification phase. In the sharing phase, the dealer inputs three pairs of matrices $(\mathbf{C}^a, \mathbf{O}^a), (\mathbf{C}^b, \mathbf{O}^b)$ and $(\mathbf{C}^c, \mathbf{O}^c)$. For an honest D, we are promised that each of the pairs is "well-formed" in the same sense as in the $\mathcal{F}_{\mathsf{vss}}$ functionality, i.e., all the matrices are symmetric, and the committed values correspond to bivariate polynomials $F^a(x, y), F^b(x, y)$ and $F^c(x, y)$ of degree at most t in each variable. In addition, when D is honest the secrets $a := F^a(0, 0)$, $b := F^b(0, 0)$ and $c := F^c(0, 0)$ satisfy the multiplicative relation $c = ab$. The output of P_i in the sharing phase is $(\mathbf{C}^a, \mathbf{O}^a_i)$, $(\mathbf{C}^b, \mathbf{O}^b_i)$ and $(\mathbf{C}^c, \mathbf{O}^c_i)$, and like in $\mathcal{F}_{\mathsf{vss}}$, we call those values the *tentative shares* of P_i.

In the verification phase, unlike in $\mathcal{F}_{\mathsf{vss}}$, the parties do not hold inputs, and we do not require the shares of all unhappy parties to be part of the output. Instead, for an honest D the output is always "verification succeeds". A corrupt dealer is allowed to have two additional inputs: a flag flag_D, and alternative openings $\bar{\mathbf{O}}^a = (\bar{o}^a_{ij})_{i,j\in\{0,\ldots,n\}}$, $\mathbf{O}^b = (\bar{o}^b_{ij})_{i,j\in\{0,\ldots,n\}}$ and $\mathbf{O}^c = (\bar{o}^c_{ij})_{i,j\in\{0,\ldots,n\}}$ that will replace the tentative openings of honest parties that received invalid shares in the sharing phase. If some matrix $\mathbf{C}^a, \mathbf{C}^b$ or $\mathbf{C}^c$ is not symmetric, or if $\mathsf{flag}_D = 1$, then the output of the verification phase is "D is corrupt". Otherwise, let W be the set of all honest parties with invalid tentative shares. The functionality verifies that for every $v \in \{a, b, c\}$ there exists a symmetric bivariate polynomial $F^v(x, y)$ of degree at most t for which (1) the value committed by (C^v_{ij}, o^v_{ij}) is $F^v(i, j)$ for all P_i in $\mathsf{H}\backslash\mathsf{W}$ and $j \in \{0, \ldots, n\}$, and (2) the value committed by $(C^v_{ij}, \bar{o}^v_{ij})$ is $F^v(i, j)$ for all P_i in W and $j \in \{0, \ldots, n\}$. In addition, the functionality verifies that the multiplicative relation $F^c(0, 0) = F^a(0, 0) \cdot F^b(0, 0)$ holds. If the verification fails then all parties output "D is corrupt". Otherwise, all the parties output "verification succeeds", and every honest P_i in W outputs the corrected shares $(\mathbf{C}^a, \bar{\mathbf{O}}^a_i)$, $(\mathbf{C}^b, \bar{\mathbf{O}}^b_i)$ and $(\mathbf{C}^c, \bar{\mathbf{O}}^c_i)$. In Section H we present the formal description of $\mathcal{F}_{\mathsf{tss}}$, together with a 2-round protocol tss that efficiently realizes $\mathcal{F}_{\mathsf{tss}}$ when the number of parties is small, i.e., $n = O(\log\kappa)$, and a proof of the following theorem.

Theorem 5.1. *Let κ be a security parameter, let n be the number of parties, let $t < n/2$, and let $\mathbb{F}$ be a field. Assuming the existence of perfectly-binding sub-exponentially hiding NICOM, protocol* tss *is a UC-*

secure implementation of $\mathcal{F}_{\mathsf{tss}}$, against a static, active, rushing adversary corrupting up to t parties. The complexity of the protocol is $\operatorname{poly}(2^n, \log|\mathbb{F}|, \kappa)$.

Alternatively, if we replace the perfectly-binding NICOM with statistically-hiding NICOM, we also obtain everlasting security.

Notation 2. *We say that D shares a random triple (a, b, c) with $c = ab$ via* tss*, if (1) D picks random symmetric bivariate polynomials $F^a(x, y)$, $F^b(x, y)$ and $F^c(x, y)$, of degree at most t in each variable, such that $F^c(0, 0) = F^a(0, 0) \cdot F^b(0, 0)$, (2) samples $(C^v_{ij}, o^v_{ij}) \leftarrow \mathsf{commit}(F^v(i, j); r^v_{ij})$ for every $v \in \{a, b, c\}$ and $i, j \in \{0, \dots, n\}$ such that $i \le j$, where r^v_{ij} is a fresh random string, (3) sets $C^v_{ji} := C^v_{ij}$ and $o^v_{ji} := o^v_{ij}$ for every $i < j$ and $v \in \{a, b, c\}$, and (4) initiates* tss *with $\mathbf{C}^a := (C^a_{ij})_{i,j\in\{0,\dots,n\}}$, $\mathbf{C}^b := (C^a_{ij})_{i,j\in\{0,\dots,n\}}$, $\mathbf{C}^c := (C^a_{ij})_{i,j\in\{0,\dots,n\}}$, and $\mathbf{O}^a := (o^a_{ij})_{i,j\in\{0,\dots,n\}}$, $\mathbf{O}^b := (o^b_{ij})_{i,j\in\{0,\dots,n\}}$, $\mathbf{O}^c := (o^c_{ij})_{i,j\in\{0,\dots,n\}}$.*

Public outputs. An external party, that can only listen to the broadcast channel, learns the following public outputs: (1) the commitments $\mathbf{C}^a, \mathbf{C}^b$ and $\mathbf{C}^c$ in the sharing phase, and (2) the output of the verification phase, which is either "verification succeeds" or "D is corrupt".

5.2 Public Single Input Functionality

We continue with public SIFs. Gennaro et al. [37] showed a reduction from secure computation of a single-input function to that of degree-2 polynomials and hence we implement $\mathcal{F}_{\mathsf{psif}}$ in Figure 3.

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

$\mathcal{F}_{\mathsf{psif}}$ is parameterized by a public degree-2 function $f : \mathbb{F}^\ell \to \mathbb{F}^m$. The functionality receives the set of corrupt parties C.

Input. $\mathcal{F}_{\mathsf{psif}}$ receives from an honest D inputs $\mathbf{z} = (z^1, \dots, z^\ell)$.

Public output. $\mathcal{F}_{\mathsf{psif}}$ returns $y(\mathbf{z}) = (y^1(\mathbf{z}), \dots, y^m(\mathbf{z}))$ to all parties, where $y^i(z^1, \dots, z^\ell) = \alpha^i_0 + \sum_{p\in\{1,\dots,\ell\}} \alpha^i_p z^p + \sum_{p,q\in\{1,\dots,\ell\}} \alpha^i_{pq} z^p z^q$ is a degree-2 polynomial in the variables $z^1, \dots, z^\ell$ and the coefficients $(\alpha^i_p, \alpha^i_{pq})_{i\in\{1,\dots,m\}p,q\in\{1,\dots,\ell\}}$ are given as part of the description of f.

Figure 3: Functionality $\mathcal{F}_{\mathsf{psif}}$

Public SIF for a small number of parties. In Section I, we realize $\mathcal{F}_{\mathsf{psif}}$ by a protocol psiflog with 1-offline round and 1-online round, that achieves optimal resiliency and prove the following theorem. Since psiflog uses tss as a sub-protocol, it is only efficient for a small number of parties, $n = O(\log \kappa)$.

Theorem 5.2. *Let κ be a security parameter, let n be the number of parties, let $t < n/2$, and let $\mathbb{F}$ be a field. Assuming the existence of perfectly-binding sub-exponentially hiding NICOM, protocol* psiflog *is a UC-secure implementation of $\mathcal{F}_{\mathsf{psif}}$, against a static, active, rushing adversary corrupting up to t parties. The complexity of the protocol is* $\operatorname{poly}(|\mathcal{C}|, 2^n, \log|\mathbb{F}|, \kappa)$*, where $\mathcal{C}$ is the circuit computing the function f.*

Alternatively, if we replace the perfectly-binding NICOM with statistically-hiding NICOM, we also obtain everlasting security.

Public SIF for a large number of parties. As discussed in Section 3.2, we use protocol psiflog in order to obtain a protocol psif, which is efficient even when the number of parties is polynomial in κ, at the expense of slightly relaxing the resiliency threshold. In Section J, we present protocol psif with 1-offline round and 1-online round, realizing $\mathcal{F}_{\mathsf{psif}}$, and prove the following theorem.

Theorem 5.3. *Let κ be a security parameter, let $\epsilon > 0$ be a constant, let n be the number of parties and t the number of corrupt parties such that $n = (2 + \epsilon)t$. Let $\mathbb{F}$ be a field. Assuming the existence of perfectly-binding sub-exponentially hiding NICOM, protocol* psif *is a UC-secure implementation of $\mathcal{F}_{\mathsf{psif}}$, against a static, active, rushing adversary corrupting up to t parties. The complexity of the protocol is* $\mathrm{poly}(|\mathcal{C}|, n, \log|\mathbb{F}|, \kappa)$*, where C is the size of the circuit computing the function f.*

Alternatively, if we replace the perfectly-binding NICOM with statistically-hiding NICOM, we also obtain everlasting security.

5.3 General Single Input Functionality

A general SIF $\mathcal{F}$ can be defined by a list of functions $f_1(\mathbf{z}), \ldots, f_n(\mathbf{z}) : \{0,1\}^* \to \{0,1\}^*$, where $f_i(\mathbf{z})$ specifies the output of P_i. That is, $\mathcal{F}$ receives an input $\mathbf{z}$ from the dealer, and returns $f_i(\mathbf{z})$ to every P_i. In Section K, we prove that general SIF reduces to public SIF in a round-preserving way and we prove the following theorems.

Theorem 5.4 (Optimal-resiliency SIF for a small number of parties)**.** *Let κ be a security parameter, let n be the number of parties and $t < n/2$. Let $\mathcal{F}$ be a single input functionality with binary circuit size s. Assuming the existence of perfectly-binding sub-exponentially hiding NICOM, there exists a protocol* sif *with 1-offline round and 1-online round which 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* $\mathrm{poly}(s, 2^n, \kappa)$.

Alternatively, if we replace the perfectly-binding NICOM with statistically-hiding NICOM, we also obtain everlasting security.

Theorem 5.5 (Almost-optimal resiliency SIF for a large number of parties)**.** *Let κ be a security parameter, let $\epsilon > 0$ be a constant, let n be the number of parties and let t the number of corrupt parties such that $n = (2 + \epsilon)t$. Let $\mathcal{F}$ be a single input functionality with binary circuit size s. Assuming the existence of perfectly-binding sub-exponentially hiding NICOM, there exists a protocol* sif *with 1-offline round and 1-online round which 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* $\mathrm{poly}(s, n, \kappa)$.

Alternatively, if we replace the perfectly-binding NICOM with statistically-hiding NICOM, we also obtain everlasting security.

References

[1] Masayuki Abe, Ronald Cramer, and Serge Fehr. Non-interactive distributed-verifier proofs and proving relations among commitments. In *Proc. of 8th ASIACRYPT*, pages 206–223, 2002.

[2] Prabhanjan Ananth, Arka Rai Choudhuri, Aarushi Goel, and Abhishek Jain. Round-optimal secure multiparty computation with honest majority. In *Proc. of 38th CRYPTO*, pages 395–424, 2018.

[3] Benny Applebaum. *Cryptography in Constant Parallel Time*. Information Security and Cryptography. 2014.

[4] Benny Applebaum, Eliran Kachlon, and Arpita Patra. The resiliency of MPC with low interaction: The benefit of making errors (extended abstract). In *Proc. of 18th TCC*, pages 562–594, 2020.

[5] Benny Applebaum, Eliran Kachlon, and Arpita Patra. The round complexity of perfect MPC with active security and optimal resiliency. In *Proc. of 61st FOCS*, pages 1277–1284, 2020.

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

[7] Michael Backes, Aniket Kate, and Arpita Patra. Computational verifiable secret sharing revisited. In *Proc. of 17th ASIACRYPT*, pages 590–609, 2011.

[8] Christian Badertscher, Ran Canetti, Julia Hesse, Björn Tackmann, and Vassilis Zikas. Universal composition with global subroutines: Capturing global setup within plain uc. In *Proc. of 18th TCC*, pages 1–30, 2020.

[9] Boaz Barak, Shien Jin Ong, and Salil P. Vadhan. Derandomization in cryptography. In *Proc. of 23rd CRYPTO*, pages 299–315, 2003.

[10] Carsten Baum, Robin Jadoul, Emmanuela Orsini, Peter Scholl, and Nigel P Smart. Feta: Efficient threshold designated-verifier zero-knowledge proofs. *Cryptology ePrint Archive*, 2022.

[11] D. Beaver. Efficient Multiparty Protocols Using Circuit Randomization. In *Proc. of 11th CRYPTO*, pages 420–432, 1991.

[12] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proc. of 1st CCS*, pages 62–73, 1993.

[13] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *Proc. of 20th STOC*, pages 1–10, 1988.

[14] Nir Bitansky and Omer Paneth. Zaps and non-interactive witness indistinguishability from indistinguishability obfuscation. In *Proc. of 12th TCC*, pages 401–427, 2015.

[15] Manuel Blum. Coin flipping by telephone. In *Proc. of 1st CRYPTO*, pages 11–15, 1981.

[16] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *Proc. of 20th STOC*, pages 103–112, 1988.

[17] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. Zero-knowledge proofs on secret-shared data via fully linear pcps. In *Proc. of 39th CRYPTO*, pages 67–97, 2019.

[18] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In *Proc. of 23rd CCS*, pages 1292–1303, 2016.

[19] Gabriel Bracha. An o(log n) expected rounds randomized byzantine generals protocol. *J. ACM*, 34(4):910–920, 1987.

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

[21] Mike Burmester and Yvo Desmedt. Broadcast interactive proofs (extended abstract). In *Proc. of EUROCRYPT*, pages 81–95, 1991.

[22] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proc. of 42nd FOCS*, pages 136–145, 2001.

[23] Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the limitations of universally composable two-party computation without set-up assumptions. In *Proc. of EUROCRYPT*, pages 68–86, 2003.

[24] Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In *Proc. of 26th FOCS*, pages 383–395, 1985.

[25] Henry Corrigan-Gibbs and Dan Boneh. Prio: Private, robust, and scalable computation of aggregate statistics. In *Proc. of 14th NSDI*, pages 259–282, 2017.

[26] Henry Corrigan-Gibbs, Dan Boneh, and David Mazières. Riposte: An anonymous messaging system handling millions of users. In *Proc. of 36th SSP*, pages 321–338, 2015.

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

[28] Ivan Damgård, Yuval Ishai, Mikkel Krøigaard, Jesper Buus Nielsen, and Adam D. Smith. Scalable multiparty computation with nearly optimal work and resilience. In *Proc. of 28th CRYPTO*, pages 241–261, 2008.

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

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

[31] Uriel Feige, Joe Kilian, and Moni Naor. A minimal model for secure computation (extended abstract). In *Proc. of 26th STOC*, pages 554–563, 1994.

[32] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple noninteractive zero knowledge proofs under general assumptions. *SIAM Journal on computing*, 29(1):1–28, 1999.

[33] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Proc. of 6th CRYPTO*, pages 186–194, 1986.

[34] Matthias Fitzi, Matthew K. Franklin, Juan A. Garay, and Harsha Vardhan Simhadri. Towards optimal and efficient perfectly secure message transmission. In *Proc. of 4th TCC*, pages 311–322, 2007.

[35] Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. The round complexity of verifiable secret sharing and secure multicast. In *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 580–589, 2001.

[36] Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. The round complexity of verifiable secret sharing and secure multicast. In *Proc. of 33rd STOC*, pages 580–589, 2001.

[37] Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. On 2-round secure multiparty computation. In *Proc. of 22nd CRYPTO*, pages 178–193, 2002.

[38] Oded Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.

[39] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Proc. of 21st STOC*, pages 25–32, 1989.

[40] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, 1994.

[41] Jens Groth and Rafail Ostrovsky. Cryptography in the multi-string model. *Journal of cryptology*, 27(3):506–543, 2014.

[42] Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for noninteractive zero-knowledge. *Journal of the ACM (JACM)*, 59(3):1–35, 2012.

[43] Shai Halevi and Silvio Micali. Practical and provably-secure commitment schemes from collision-free hashing. In *Proc. of 16th CRYPTO*, pages 201–215, 1996.

[44] Danny Harnik, Yuval Ishai, and Eyal Kushilevitz. How many oblivious transfers are needed for secure multiparty computation? In *Proc. of 27th CRYPTO*, pages 284–302, 2007.

[45] Alexander Healy and Emanuele Viola. Constant-depth circuits for arithmetic in finite fields of characteristic two. In *Proc. of 23rd STACS*, pages 672–683, 2006.

[46] Amir Herzberg. Folklore, practice and theory of robust combiners. *Journal of Computer Security*, 17(2):159–189, 2009.

[47] Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In *Proc. of 29th ICALP*, pages 244–256, 2002.

[48] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In *Proc. of 39th STOC*, pages 21–30, 2007.

[49] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In *Proc. of 28th CRYPTO*, pages 572–591, 2008.

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

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

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

[53] Arpita Patra and Divya Ravi. On the exact round complexity of secure three-party computation. In *Proc. of 38th CRYPTO*, pages 425–458, 2018.

[54] Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for np from (plain) learning with errors. In *Proc. of 39th CRYPTO*, pages 89–114, 2019.

[55] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *Proc. of 21st STOC*, pages 73–85, 1989.

[56] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *Proc. of 46th STOC*, pages 475–484, 2014.

[57] The same (anonymous) authors. Round-optimal honest-majority mpc in minicrypt and with everlasting security. *Under submission to this confrence*, 2021.

[58] Kang Yang and Xiao Wang. Non-interactive zero-knowledge proofs to multiple verifiers. *Cryptology ePrint Archive*, 2022.

[59] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *Proc. of 23th FOCS*, pages 80–91, 1982.

A Appendix: Security Model, Useful Facts and Standard Primitives

A.1 Security model

In this section we give a high-level description of the UC-framework, due to [22]. For more details, the reader is referred to [22]. We begin with a short description of the standard model, and then explain how the UC-framework augments it. At a high level, in the standard model, security of a protocol is argued by comparing the real-world execution to an ideal-world execution. In an ideal-world execution, the inputs of the parties are transferred to a trusted party $\mathcal{F}$ (called the *ideal functionality*) over a perfectly secure channel, the trusted party computes the function based on these inputs and sends to each party its respective output. Informally, a protocol π securely implements $\mathcal{F}$ if for any real-world adversary $\mathcal{A}$, there exists an ideal-world adversary $\mathcal{S}$ (called the *simulator*), that controls the same parties as $\mathcal{A}$, so that the global output of an execution of π with $\mathcal{A}$ (consisting of the honest parties' outputs and the output of $\mathcal{A}$), is indistinguishable from the global output of the ideal-world execution with $\mathcal{F}$ and $\mathcal{S}$ (consisting of the honest parties' outputs and the output of $\mathcal{S}$).

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

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

The dummy-adversary. Since the above definition quantifies over all environments, we can merge the adversary $\mathcal{A}$ with the environment $\mathcal{Z}$. That is, it is enough to require that the simulator $\mathcal{S}$ will be able to simulate, for any environment $\mathcal{Z}$, the *dummy adversary* that simply delivers messages from $\mathcal{Z}$ to the protocol machines. For more information, see [22].

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

Corruption-aware functionalities. Throughout, we assume that our functionalities are *corruption-aware*, which means that they might depend on the identities of the corrupt parties C. The notion of corruption aware functionalities was first introduced by [22], and the reader is referred to [22] for more information (see also [6, Section 6.2]).

We mention that our single-input functionality $\mathcal{F}_{\mathsf{sif}}$ is a *fictitiously corruption aware* functionality, which means that $\mathcal{F}_{\mathsf{sif}}$ receives the set of corrupt parties C, but does not depend on it. It is known (see, e.g., [6, Section 6.2]) that if a protocol securely computes a fictitiously corruption aware functionality, then it also securely computes it in the standard model, where the functionality does not receive the set of corrupt parties C. Therefore, our final functionality $\mathcal{F}_{\mathsf{sif}}$ is also secure in the standard model.

Reactive functionalities. In this work, we consider both single-phase functionalities and reactive (or multi-phase) functionalities. A single-phase functionality maps the inputs of the parties to the outputs in a single phase of computation. A multi-phase functionality consists of multiple phases of computation, where each phase depends on the internal state of the functionality. In each phase the functionality receives the inputs of the parties, computes the outputs based on the inputs and the internal state, and updates the internal state based on the inputs. We only consider functionalities with a single phase, or with two phases. For more information about reactive functionalities, see, e.g., [38, Chapter 7.7.1.3].

Functionalities with well-formed inputs and complete break-down. In many cases we assume that the ideal functionality receives well-formed inputs from the honest parties. (For example, we assume that all honest parties input *the same* commitments.) However, in order to prove UC-security, we actually need to consider arbitrary environments, including environments that pick inputs that are not well-formed. We solve this mismatch by adopting the (standard) convention that whenever the inputs of the honest parties are not well-formed, a *complete break-down* of the functionality occurs (see, e.g., [27, Chapter 4.4]). That is, whenever the inputs of the honest parties are not well-formed, then the ideal functionality (1) leaks all the private information (i.e., the inputs of the honest parties) to the simulator, and (2) allows the simulator to determine the outputs of the honest parties.

Complete break-down allows us to simulate even when the inputs are not well-formed, since the simulator has more power: it receives the inputs of the honest parties, and it is allowed to determine their outputs. Indeed, any protocol for a single-phase functionality can be perfectly simulated when complete break-down occurs, since the simulator, that holds all the inputs of the honest parties at the beginning of the simulation, can take the role of the honest parties in an execution of the protocol with $\mathcal{Z}$ and $\mathcal{A}$, and at the end of the execution determine the outputs of the honest parties accordingly. We emphasize that, when using the functionality in some other protocol, private information will not be leaked due to a complete break-down as long as we make sure that we always call the functionality with well-formed inputs.

In general, for 2-phase functionalities, complete break-down does not necessarily imply trivial simulation since the simulator might not be able to take the role of the honest parties when the bad inputs arrive only *in the second phase*. However, in all our 2-phase functionalities perfect simulation will be possible, because in the first phase (that consists of a single round) for every honest party the simulator does exactly what an honest party will do.[14] Therefore, even if the bad inputs arrive in the second phase, the simulator can take the role of the honest parties and complete an execution of the protocol. This issue is explicitely treated in the corresponding proofs.

Public outputs. Sometimes it will be useful to consider the case where there are external parties that listen to the broadcast channel but do not participate in the execution of the protocol. We would like to let those parties learn parts of the output, which we call *public outputs*. In order to argue that a protocol is secure even in those settings, we also need to consider an adversary that does not corrupt any party, but only listens to the broadcast channel. We prove that the same simulators can still simulate the execution that such an adversary sees, given the public outputs and the leakage specified by the functionality; that is, we show that no environment can distinguish the real-world from the ideal-world.

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

The notion of everlasting security was formalized in the UC-framework by [50]. In a nutshell, instead of considering environments that are unbounded after the execution, it is enough to consider only environments that are always polynomially-bounded, but are not limited to a single bit output. In particular, such environments can output their whole view. Using the same notation as before, $\text{REAL}_{\pi,\mathcal{Z}(\zeta),\mathcal{A}}(\kappa)$ and $\text{IDEAL}_{\mathcal{F},\mathcal{Z}(\zeta),\mathcal{S}}(\kappa)$, to denote the output distribution of $\mathcal{Z}$ in

[14]This is possible because the first phase is either an offline-phase, that has no inputs and no outputs, and the simulation of the first round consists merely of an execution of the first round of the protocol (like in protocol scg and protocol glinear, see Footnotes 21 and 24), or because in the first round only a single honest party has an input, and the communication from this honest party to the corrupt parties is fully determined by the outputs of the corrupt parties, which the simulator holds (like in protocol vss, see Footnote 19).

the real-world and in the ideal-world (where now the output may contain more than one bit), we say that a protocol π UC-emulates an ideal functionality $\mathcal{F}$ with everlasting security, if for every polynomial-time real-world adversary $\mathcal{A}$ there exists an ideal-world polynomial-time simulator $\mathcal{S}$ such that for any polynomial-time environment $\mathcal{Z}$ and any input ζ to $\mathcal{Z}$, the random variables $\{\text{REAL}_{\pi,\mathcal{Z}(\zeta),\mathcal{A}}(\kappa)\}_\kappa$, and $\{\text{IDEAL}_{\mathcal{F},\mathcal{Z}(\zeta),\mathcal{S}}(\kappa)\}_\kappa$ are *statistically* indistinguishable. Therefore, in general, in order to prove security it is enough to show that the view of the environment in the real-world is statistically-close to the view of the environment in the ideal-world.

We mention that the composition theorems of UC-security hold for protocols with everlasting security (i.e., the composition of two protocols with everlasting security results in a protocol with everlasting security). For a formal definition and statement of the composition theorem, the reader is referred to [50].

Global setup. In order to obtain protocols with everlasting security, we use non-interactive commitments which are *statistically-hiding* and computationally binding. Such commitments cannot be implemented in the plain model and they require an additional round of intereaction or some global setup. (Otherwise, a non-uniform adversary can "hardwire" an ambiguous commitment with 2 consisting openings). In our setting the setup consists the slection of a collision-resistance hash function h from a family $\mathcal{H}$. For simplicity, we capture this via the standard notion of common reference string (CRS). (Though, weaker notions suffice as discussed in Remark 1.4.) Throughout the paper, we assume that all functionalities and parties have access to the same global functionality $\mathcal{F}_{\mathsf{crs}}$, that, upon receiving a query, returns the common reference string. We mention that, since all our protocols are static systems, where all identities and connectivity is fixed beforehand, the composition theorems in this model follow immediately from the composition theorems guaranteed by UC-security, even when we consider everlasting security (see, e.g., [8, Section 1]).

A.2 Standard Useful Facts

A.2.1 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., [6]).

Fact A.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 A.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 A.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 $\mathcal{P}^{s',t}$.

Fact A.4. *For every $s, s' \in \mathbb{F}$, $\mathsf{C} \subseteq \{1, \ldots, n\}$ of size at most t, and two sets of degree-t polynomials $\{f_i(x)\}_{i \in \mathsf{C}}$ and $\{f'_i(x)\}_{i \in \mathsf{C}}$ such that $f_i(j) = f_j(i)$ and $f'_i(j) = f'_j(i)$ for all $i, j \in \mathsf{C}$, it holds that the support size of $F(x,y)$ is equal to the support size of $F'(x,y)$, where $F(x,y)$ is sampled from $\mathcal{P}^{s,t}$ conditioned on $F(x,i) = f_i(x)$ for every $i \in \mathsf{C}$, and $F'(x,y)$ is sampled from $\mathcal{P}^{s',t}$ conditioned on $F'(x,i) = f'_i(x)$ for every $i \in \mathsf{C}$.*

The following fact is due to Beaver [11].

Fact A.5 (Beaver's Trick over Univariate Polynomials)**.** *Let $\mathsf{C} \subseteq \{1, \ldots, n\}$ be a set of size at most t, and let $f^{\alpha}(x)$ and $f^{\beta}(x)$ be any degree-t polynomials. Then the following two experiments have the same distribution.*

- Experiment 1. *Sample three random degree-t polynomials $f^a(x), f^b(x)$ and $f^c(x)$ conditioned on $f^a(0) \cdot f^b(0) = f^c(0)$. Set $u := f^{\alpha}(0) - f^a(0)$ and $v := f^{\beta}(0) - f^b(0)$. Output*

$$\left((f^a(i), f^b(i), f^c(i))_{i \in \mathsf{C}}, f^{\alpha}(x) - f^a(x), f^{\beta}(x) - f^b(x), u f^b(x) + v f^a(x) + f^c(x) + uv \right).$$

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

$$\left((a_i, b_i, c_i)_{i \in \mathsf{C}}, f^u(x), f^v(x), f(x) \right).$$

A.2.2 Statistical Distance

We start with a basic fact, which can be found, e.g., in [3].

Fact A.6. *Let $\mathcal{W}$ be a set, and let $\{X_w\}_{w \in \mathcal{W}}$, $\{Y_w\}_{w \in \mathcal{W}}$ be distribution ensembles. Then, for every distribution W over $\mathcal{W}$, we have $\Delta((W, X_W), (W, Y_W)) = \mathbb{E}_{w \leftarrow W}[\Delta(X_w, Y_w)]$.*

Fact A.7. *Let $X = (X_1, X_2)$ and $Y = (Y_1, Y_2)$ be probability distributions on a set $A \times B$ such that $\Delta(X, Y) \leq \epsilon$. Let $\delta = \sqrt{2\epsilon}$. Then, with probability at least $1 - \delta$ over $z \leftarrow X_1$ it holds that*

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

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

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

By Fact A.6 it follows that

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

Finally, by Markov's inequality we conclude that

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

This conclude the proof. □

A.3 Non-Interactive Commitment Schemes (NICOM)

Definition A.8 (NICOM). *A NICOM is a pair of probabilistic algorithms* $(\mathsf{commit}, \mathsf{open})$ *that take as a common input the security parameter* 1^κ *and a some (possibly empty) random public parameters* $\mathsf{pp} \in \{0,1\}^{\ell(\kappa)}$ *for some polynomial* $\ell(\cdot)$ *and satisfy the following requirements:*

– *Syntax:* commit *takes as an input a message* $x \in \{0,1\}^*$ *and random tape* $r \in \{0,1\}^*$ *and outputs a commitment/openning pair* (C, o) *and the algorithm* open *takes as an input a commitment/opening pair* (C, o) *and outputs a message* $x' \in \{0,1\}^* \cup \{\perp\}$.

– *Correctness: For every* $\kappa, \mathsf{pp}, x, r$, *it holds that* $\mathsf{open}_{\mathsf{pp}}(1^\kappa, \mathsf{commit}_{\mathsf{pp}}(1^\kappa, x; r)) = x$.

– *Binding: For every family of polynomial-size non-uniform adversaries* $\mathcal{A} = \{\mathcal{A}_\kappa\}$ *and every security parameter* κ, *with probability at most* $\epsilon = \mathsf{negl}(\kappa)$ *over a uniform choice of* pp, *the tuple* $(C, o, o') := \mathcal{A}_\kappa(\mathsf{pp})$ *satisfies* $\mathsf{open}_{\mathsf{pp}}(1^\kappa, C, o) \neq \mathsf{open}_{\mathsf{pp}}(1^\kappa, C, o')$ *and* $\mathsf{open}_{\mathsf{pp}}(1^\kappa, C, o) \neq \perp$ *and* $\mathsf{open}_{\mathsf{pp}}(1^\kappa, C, o') \neq \perp$. *The scheme is* statistically binding, *if the above holds even for inefficient adversaries, and* perfectly binding *if, in addition,* $\epsilon = 0$.

– *Hiding: For every family of non-uniform adversaries* $\mathcal{A} = \{\mathcal{A}_\kappa\}$, *every polynomial* $p(\cdot)$, *every security parameter* κ, *every* pp, *and every pair of messages* $x, x' \in \{0,1\}^{p(\kappa)}$, *the distinguishing gap*

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

for some negligible $\epsilon(\cdot)$. *The scheme is* statistically hiding *if the above holds even for inefficient adversaries.*

For ease of reading, we typically omit the security parameter and the public parameters from the algorithms. By default, the security parameter is set according to the global security parameter that is being used by the system, and the public parameters are chosen once and for all before all protocols begin by a set-up phase as explained towards the end of Section A.1.

Remark A.9 (Sub-exponential hiding). *Assuming injective OWF over* m*-bit inputs that cannot be inverted by a PPT adversary with probability better than* 2^{-m^δ}, *it is possible to construct [15, 59, 39] a plain-model (with no public parameters) perfectly-binding NICOM whose computational hiding property holds for* $\epsilon \leq 2^{-\kappa}$. *We refer to such a commitment as* perfectly binding sub-exponentially hiding *NICOM. Moreover, under worst-case derandomization assumptions [9], such NICOMs can be based on general (not necessarily injective) sub-exponentially hard OWFs. Similar sub-exponential hardness assumptions are quite common in the literature and typical candidate one-way functions seem to achieve sub-exponential hardness. In fact, our variant of sub-exponential hardness is relatively mild compared to other notions, since we do not allow the adversary to run in sub-exponential time, but only allow it to succeed with sub-exponentially small probability.*

B Appendix: Applications of VRS and SIF

Here, we detail the applications briefed in Section 1.1.4. In order to simplify the presentation, we continue by presenting the applications using single input functionalities, and n parties, denoted $P_1, \dots, P_n$.

Round efficient manipulation of non-homomorphic encryption. Consider the case where there are public commitments $C_1, \dots, C_\ell$, and a distinguished party P_1 holds the corresponding openings $o_1, \dots, o_\ell$. Let the committed values be $z_1, \dots, z_\ell$ and think of them as private values of P_1. P_1 wishes to apply a public function f on the committed values, and reveal $y := f(z_1, \dots, z_\ell)$ to the rest of the parties $P_2, \dots, P_n$.

We note that P_1 can use a single input functionality $\mathcal{F}$ that (1) receives the commitments and openings $(C_i, o_i)_{i \in \{1,\dots,\ell\}}$ from P_1, (2) opens the i-th commitment C_i with o_i in order to reveal z_i, and (3) returns $(C_1, \dots, C_\ell, f(z_1, \dots, z_\ell))$ to all the parties (if the opening of some commitment fails then the functionality returns a failure-symbol $\perp$ to all the parties). Indeed, if the output of the functionality is $\perp$, the rest of the parties conclude that P_1 is corrupt. Otherwise, the rest of the parties hold the output $(\bar{C}_1, \dots, \bar{C}_\ell, y)$ of the functionality. Each party verifies that P_1 used the correct commitments as inputs to $\mathcal{F}$, that is, $\bar{C}_i = C_i$ for $i \in \{1, \dots, \ell\}$. If the verification succeeds the parties output y, and otherwise they conclude that P_1 is corrupt.

Round-efficient GMW-type compiler for protocols with semi-malicious security. We continue by describing a compiler that takes an r-round protocol which is secure against a semi-malicious adversary, and transforms it to an $(r+1)$-round protocol that provides security with unanimous abort against a malicious adversary. Let Φ be an r-round protocol, and let $\Phi^{i,j}$ be the next-message function of the i-th party in round j (that is, $\Phi^{i,j}$ is the function that computes the j-th round messages of P_i based on the input and randomness of P_i, as well as the messages that P_i received in rounds $1, \dots, j-1$). We assume without loss of generality that Φ only uses private-channel communication. The main idea is to emulate the execution of Φ with the help of single input functionalities. At the j-th round, we use SIF to make sure that P_i's behaviour is consistent with P_i's input, randomness and all messages that P_i received in the previous rounds, where we make sure that there are *public commitments* that hide those values. We also make sure that P_i provides public commitment for every message that she has to send in the j-th round, and that every other party P_k receives the opening of the respective commitment.

For every party P_i and round j we define the single input functionality $\mathcal{F}^{i,j}$ as follows.

- If $j = 1$ then $\mathcal{F}^{i,1}$ receives from P_i (1) input x_i for Φ, (2) randomness ρ_i for Φ, and (3) auxiliary randomness $\rho^{i,1}$.

  The functionality computes the first round messages of P_i in Round 1 by computing $\Phi^{i,1}(x_i, \rho_i) = (a_1^{i,1}, \dots, a_n^{i,1})$, where $a_k^{i,1}$ is the first round message from P_i to P_k. The functionality uses the auxiliary randomness $\rho^{i,1}$ to (1) sample a commitment and opening (C_{x_i}, o_{x_i}) of x_i, (2) sample a commitment and opening (C_{ρ_i}, o_{ρ_i}) of ρ_i, and (3) sample a commitment and opening $(C_k^{i,1}, o_k^{i,1})$ of $a_k^{i,1}$, for every $k \in \{1, \dots, n\}$.

  The functionality returns o_{x_i}, o_{ρ_i} to P_i. In addition, every P_k receives from the functionality all the commitments $C_{x_i}, C_{\rho_i}, C_1^{i,1}, \dots, C_n^{i,1}$ and the opening $o_k^{i,1}$.

- For $j > 1$, the functionality receives from P_i (1) commitments and openings (C_{x_i}, o_{x_i}) and (C_{ρ_i}, o_{ρ_i}), (2) commitments and openings $(C_i^{k,j'}, o_i^{k,j'})$ for all $k \in \{1, \ldots, n\}$ and $j' < j$, and (3) auxiliary randomness $\rho^{i,j}$.

  The functionality (1) opens C_{x_i} with o_{x_i} to obtain x_i, (2) opens C_{ρ_i} with o_{ρ_i} to obtain ρ_i, and (3) opens each $C_i^{k,j'}$ with $o_i^{k,j'}$ to obtain $a_i^{k,j'}$. If some opening fails then the functionality returns a failure-symbol $\perp$ to all the parties. Otherwise, the functionality computes $\Phi^{i,j}(x_i, \rho_i, (a_i^{k,j'})_{k\in\{1,\ldots,n\},j'<j}) = (a_1^{i,j}, \ldots, a_n^{i,j})$. The functionality uses the auxiliary randomness $\rho^{i,j}$ to sample a commitment and opening $(C_k^{i,j}, o_k^{i,j})$ of $a_k^{i,j}$, for every $k \in \{1, \ldots, n\}$.

  Every P_k receives from the functionality (1) all the commitments $C_1^{i,j}, \ldots, C_n^{i,j}$ and the opening $o_k^{i,j}$, and (2) the commitments $(C_i^{k,j'})_{k\in\{1,\ldots,n\},j'<j}$ and C_{x_i}, C_{ρ_i} that P_i inputs to $\mathcal{F}^{i,j}$.

The compiler does as follows.

- **(Offline round)** In this round the parties execute the offline round of all SIF executions that will be described below.

- **(Emulation of Round** 1**)** Every P_i holds its input x_i and samples randomness ρ_i for protocol Φ, and randomness $\rho^{i,1}$ for $\mathcal{F}^{i,1}$. For every P_i the parties execute the online round of the SIF execution of $\mathcal{F}^{i,1}$, where P_i inputs x_i, ρ_i and $\rho^{i,1}$. At the end of this round each P_i holds the commitments $C_k^{i',1}$ for all $i' \in \{1, \ldots, n\}$ and $k \in \{1, \ldots, n\}$, as well as the openings $o_i^{i',1}$ for all $i' \in \{1, \ldots, n\}$.[15]

- **(Emulation of Round** $j > 1$**)** For every $j = 2, \ldots, r$ the parties do as follows.

  – Every P_i samples randomness $\rho^{i,j}$ for $\mathcal{F}^{i,j}$.

  – For every P_i the parties execute the online round of the SIF execution of $\mathcal{F}^{i,j}$, where P_i inputs (1) commitments and openings (C_{x_i}, o_{x_i}) and (C_{ρ_i}, o_{ρ_i}), (2) commitments and openings $(C_i^{k,j'}, o_i^{k,j'})$ for all $k \in \{1, \ldots, n\}$ and $j' < j$, and (3) the randomness $\rho^{i,j}$.

  – At the end of the round, every P_k does as follows. If some SIF execution ended with $\perp$ then the parties abort.

  – Otherwise, the output of P_k in $\mathcal{F}^{i,j}$ has the form (1) commitments $C_1^{i,j}, \ldots, C_n^{i,j}$ and the opening $o_k^{i,j}$, and (2) the commitments $(\bar{C}_i^{k,j'})_{k\in\{1,\ldots,n\},j'<j}$ and $\bar{C}_{x_i}, \bar{C}_{\rho_i}$ that P_i inputs to $\mathcal{F}^{i,j}$. If there exists $i \in \{1, \ldots, n\}$ such that the commitments in the second part of the output are not equal to the public commitments $(C_i^{k,j'})_{k\in\{1,\ldots,n\},j'<j}$ and C_{x_i}, C_{ρ_i} that were generated in previous round, then P_k aborts.[16]

  – Otherwise, continue to the next iteration.

- **(Output computation)** After the emulation of Round r, each P_i holds input x_i, randomness ρ_i and incoming messages $(a_i^{i',j})_{i'\in\{1,\ldots,n\},j\in\{1,\ldots,r\}})$, where $a_i^{i',j}$ is the value hidden in $C_i^{i',j}$. P_i uses those values in order to compute its output in protocol Φ.

[15]For simplicity, we assume that the execution of $\mathcal{F}^{i,1}$ cannot end with failure, since if a corrupt P_i does not provide the functionality with x_i, ρ_i and $\rho^{i,1}$ then the parties can simply set those values to be the all-zero string.

[16]Observe that all the parties receive the same commitments, so all the parties abort in this case.

C Appendix: A common strategy for the proofs

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

In order to unify the proofs, we also assume that the parties and functionalities have access to a global functionality $\mathcal{F}_{\mathsf{crs}}$ that returns the CRS string. In the case of computational security we always assume that $\mathcal{F}_{\mathsf{crs}}$ does nothing, and that the CRS string is empty.

Throughout, we denote by View the tuple consists of the randomness of the environment, the messages that the corrupt parties sent and received, and the inputs of the honest parties (which are picked by the environment). We denote by ϵ the error term of the commitment scheme, where $\epsilon = \mathsf{negl}(\kappa)$. We always assume that the adversary is the dummy adversary (see Section A.1).

D Verifiable Secret Sharing

In this section we provide formal definitions to notions discussed in Section 4.1. Fix some common reference string crs (if the underlying commitment scheme is statistically-hiding), and consider a pair $(\mathbf{C}, \mathbf{O})$ of commitments $\mathbf{C} = (C_{ij})_{i,j \in \{0,\ldots,n\}}$ and openings $\mathbf{O} = (o_{ij})_{i,j \in \{0,\ldots,n\}}$. For $i \in \{0, \ldots, n\}$, let $\mathbf{C}_i = (C_{ij})_{j \in \{0,\ldots,n\}}$ and $\mathbf{O}_i = (o_{ij})_{j \in \{0,\ldots,n\}}$. We begin with the definitions of valid shares and strong sharing. We emphasize that the definitions consider the CRS only if the underlying commitment scheme is statistically-hiding.

Definition D.1 (validity)**.** *We say that $(\mathbf{C}, \mathbf{O}_i)$ is* valid *(with respect to crs) if the following conditions hold:*

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

Definition D.2 (Strong double t-sharing aka $\langle\langle \cdot \rangle\rangle$-sharing)**.** *A pair $(\mathbf{C}, \mathbf{O})$ is a* strong double t-sharing of s *(with respect to crs), denoted as $\langle\langle s \rangle\rangle$, if the following conditions hold:*

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

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

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

Definition D.3 (Weak double t-sharing aka $[\![\cdot]\!]$-sharing)**.** *A tuple* $(\mathsf{W}, \mathbf{C}, \mathbf{O}_{\mathsf{W}}, \mathbf{O}_{\mathsf{H}\setminus\mathsf{W}})$ *of parties* $\mathsf{W} \subseteq \{1, \ldots, n\}$*, public commitments* $\mathbf{C} = (C_{ij})_{i,j\in\{0,\ldots,n\}}$*, public openings* $\mathbf{O}_{\mathsf{W}} = (o_{ij})_{i\in\mathsf{W},j\in\{0,\ldots,n\}}$*, and private openings* $\mathbf{O}_{\mathsf{H}\setminus\mathsf{W}} = (o_{ij})_{i\in\mathsf{H}\setminus\mathsf{W},j\in\{0,\ldots,n\}}$ *for the set of honest parties* H *is a* weak double t-sharing *of* s *(with respect to* crs*), denoted as* $[\![s]\!]$*, if the following conditions holds:*

1. *(Partial validity) For every* $i \in \mathsf{W} \cup \mathsf{H}$ *it holds that* $(\mathbf{C}, \mathbf{O}_i)$ *is valid.*
2. *(Weakly consistent opening)* $\mathsf{open}(C_{ij}, o_{ij}) = \mathsf{open}(C_{ji}, o_{ji})$[17] *for* $i, j \in \mathsf{W} \cup \mathsf{H}$
3. *(Weak sharing of s) The values* $(f_{i0} := \mathsf{open}(C_{i0}, o_{i0}))_{i\in\mathsf{W}\cup\mathsf{H}}$ *correspond to a degree t polynomial* $f(x)$ *such that* $f(i) = f_{i0}$ *and* $f(0) = s$*.*

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

Observation D.4 (Weak-sharing as a robust sharing)**.** *We observe that unless the binding property of the commitment scheme is violated, weak-sharing is also a robust sharing. Indeed, for every* $i \in \mathsf{W}$ *the openings* $\mathbf{O}_i$ *are already public, and consistent with* $F(x, y)$*. In addition, every* $i \in \mathsf{H} \setminus \mathsf{W}$ *provides its openings* $\mathbf{O}_i$ *which are also consistent with* $F(x, y)$*. Finally, since there are at least* $t + 1$ *honest parties, for every corrupt* P_i *that reveals valid openings* $\mathbf{O}_i$*, the committed polynomial* $f_i(x)$ *is either consistent with* $F(x, y)$*, or has degree more than* t*, in which case the parties think of it as an erasure.*

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

Definition D.5 (Rows of Sharing)**.** *Let* $(\mathbf{C}, \mathbf{O})$ *and* $(\mathsf{W}, \mathbf{C}, \mathbf{O}_{\mathsf{W}}, \mathbf{O}_{\mathsf{H}\setminus\mathsf{W}})$ *denote a* $\langle\!\langle s \rangle\!\rangle$ *and respectively* $[\![s]\!]$*. We refer* $(\mathbf{C}_i, \mathbf{O}_i)$ *as the* ith row *and denote by* $\langle\!\langle s \rangle\!\rangle_i$ *and* $[\![s]\!]_i$ *for respective sharing. We refer* $(\mathbf{C}_0, \mathbf{O}_0)$ *as the* main row *of the sharings and denote by* $\langle s \rangle$ *and* $[s]$ *for respective sharing.*

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

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

D.1 The VSS functionality and protocol

We continue with a formal definition of the functionality $\mathcal{F}_{\mathsf{vss}}$.

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

The functionality $\mathcal{F}_{\mathsf{vss}}$ receives the set of corrupt parties C.

Sharing phase.

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

Verification phase.

- **Honest parties' inputs:** Each honest party P_i inputs a bit flag_i, such that if D is honest then $\mathsf{flag}_i = 0$ for every honest P_i.
- **Leakage:** For any honest P_i, the bit flag_i is leaked to the adversary.
- **Adversary's inputs:** Each corrupt P_i inputs a bit flag_i. If D is corrupt, then D has two additional inputs, $\bar{\mathbf{O}} := (\bar{o}_{ij})_{i,j\in\{0,\ldots,n\}}$ and a bit flag_D.
- **Public outputs:** We split into two cases.
  - **Honest** D. Let W be the set of all corrupt parties P_i with $\mathsf{flag}_i = 1$. $\mathcal{F}_{\mathsf{vss}}$ returns $\big(\mathsf{W}, (o_{ij})_{i\in\mathsf{W},j\in\{0,\ldots,n\}}\big)$ to all parties.
  - **Corrupt** D. Let W be the set containing all parties P_i with $\mathsf{flag}_i = 1$, together with all honest parties P_i with an invalid pair $(\mathbf{C}, \mathbf{O}_i)$, where $\mathbf{O}_i := (o_{ij})_{j\in\{0,\ldots,n\}}$. Let $\bar{\mathbf{O}}_{\mathsf{W}} := (\bar{o}_{ij})_{i\in\mathsf{W},j\in\{0,\ldots,n\}}$. If the tuple $(\mathsf{W}, \mathbf{C}, \bar{\mathbf{O}}_{\mathsf{W}}, \mathbf{O}_{\mathsf{H}\setminus\mathsf{W}})$ is a weak double t-sharing, and $\mathsf{flag}_D = 0$ then $\mathcal{F}_{\mathsf{vss}}$ returns $\big(\mathsf{W}, \bar{\mathbf{O}}_{\mathsf{W}}\big)$ to all parties. Otherwise, $\mathcal{F}_{\mathsf{vss}}$ returns "D is corrupt" to all parties.

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

A formal description of protocol vss, which is a slightly simplified version of the protocol of [7] appears in Figure 5.

Protocol vss

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

Sharing Phase (R1):

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

Verification Phase (R2):

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

Figure 5: Protocol vss

We continue with the proof of security of Theorem 4.1.

D.2 Proof of security

Proof of Theorem 4.1. In this section, we prove that protocol vss UC-emulates $\mathcal{F}_{\mathsf{vss}}$ (with everlasting security when the underlying commitment scheme is statistically-hiding). Let $\mathcal{A}$ be the dummy adversary. We define the simulator $\mathcal{S}$ as follows. $\mathcal{S}$ uses $\mathcal{A}$ in a black-box manner, and forwards all messages between $\mathcal{Z}$ and $\mathcal{A}$. $\mathcal{S}$ first receives the set of corrupt parties C. We split into two cases.

D.2.1 Honest Dealer

Sharing phase. First,[18] $\mathcal{S}$ receives from $\mathcal{F}_{\mathsf{vss}}$ the commitments and openings $(\mathbf{C}, \mathbf{O}_i)$ for every $i \in \mathsf{C}$. $\mathcal{S}$ sends $\mathbf{C}$ to $\mathcal{A}$ as the broadcast of D, and $\mathbf{O}_i$ as the private message from D to P_i, for every $i \in \mathsf{C}$. In addition, on behalf of every honest P_i, $\mathcal{S}$ sets $((G_{ij}, h_{ij}) := \mathsf{commit}(g_{ij}, r_{ij}))_{j\in\{0,\dots,n\}}$, where r_{ij} and g_{ij} are fresh random strings, and sends $\mathbf{G}_i := (G_{ij})_{j\in\{0,\dots,n\}}$ to $\mathcal{A}$ as the broadcast of P_i. This completes the communication from honest parties to corrupt parties in the first round. At this stage, $\mathcal{S}$ receives from $\mathcal{A}$ the messages that every corrupt party P_i sends, that is, the broadcast $\mathbf{G}_i$ and the private messages $(g_{ij}, h_{ij})_{j\in\{0,\dots,n\}}$ to D.

Verification phase. $\mathcal{S}$ receives the input bits of the honest parties, $\{\mathsf{flag}_i\}_{i\in\mathsf{H}}$ as a leakage from $\mathcal{F}_{\mathsf{vss}}$. Since D is honest, we are promised that all those bits are 0.[19] $\mathcal{S}$ simulates the honest parties

[18]If the inputs of the honest D are not well-formed (that is, $(\mathbf{C}, \mathbf{O})$ is not a strong double t-sharing), then a complete break-down occurs (see Section A.1). In this case simulation is trivial, because the simulator, that holds all the inputs of the honest parties, can simply take the role of the honest parties in an execution of the protocol, and at the end of the execution, it sends to $\mathcal{F}_{\mathsf{vss}}$ the outputs of the honest parties. This results in a *perfect* simulation of the protocol. Hence, we continue by assuming that the first-round inputs to the honest dealer are well-formed.

[19]If the honest parties inputs are not well-formed (that is, $\mathsf{flag}_i = 1$ for some honest P_i) then a complete break-down occurs (see Section A.1). In this case we can obtain perfect simulation. Indeed, the first round was simulated exactly like an execution of the protocol. Therefore the simulator, that now holds $(\mathbf{C}, \mathbf{O})$ can continue the execution of the protocol by computing the first round messages between D and the honest parties (which concludes the first round execution)

(except the dealer D) by not sending any message. $\mathcal{S}$ simulates D in the following way.

- For every honest P_i, $\mathcal{S}$ broadcasts $(\alpha_{ij})_{j\in\{0,\ldots,n\}}$, where each α_{ij} is a random string.

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

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

Fix a polynomial-time environment $\mathcal{Z}$ with input ζ, and assume without loss of generality that $\mathcal{Z}$ is deterministic. We begin by showing that the view of $\mathcal{Z}$ in the real-world is close to the view of $\mathcal{Z}$ in the ideal world.

We start by analysing (1) $\mathcal{Z}$'s view in the sharing phase, (2) the outputs of the honest parties in the sharing phase, and (3) $\mathcal{Z}$'s view in the verification phase. (That is, for now we ignore the outputs of the honest parties in the verification phase.) We consider the following hybrid-worlds, where we assume that the honest parties know the set H.

- In Hybrid 1, the honest parties act like in the real-world, except that in round 2, (a) an honest party P_i never sends a broadcast message, (b) for every $i \in \mathsf{H}$, D does not verify that $\mathsf{open}(G_{ij}, h_{ij}) \overset{?}{=} g_{ij}$ for $j \in \{0,\ldots,n\}$, but D is always happy with P_i, and (c) for every $i \in \mathsf{H}$, the computation of $\alpha_{i0},\ldots,\alpha_{in}$ by D is done using the values $g_{i0},\ldots,g_{in}$ that P_i sends to D (and not the values extracted from the commitments).

- In Hybrid 2, the honest parties act like in Hybrid 1, with the following modification: in the first round, an honest P_i samples random elements $(g_{i0},\ldots,g_{in})$ and $(g'_{i0},\ldots,g'_{in})$, samples $(G_{ij}, h_{ij}) \leftarrow \mathsf{commit}(g'_{ij}; r_{ij})$, where r_{ij} is a fresh random string, broadcasts $G_{i0},\ldots,G_{in}$, and sends $(g_{ij}, h_{ij})_{j\in\{0,\ldots,n\}}$ to D (so that value g_{ij} will be used in the computation of α_{ij}).

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

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

and then computing the second round messages of the honest parties, while using flag_i as the input of P_i. Finally, after receiving the corrupt parties messages from the adversary, the simulator can compute the outputs of the honest parties in the execution, and send them to $\mathcal{F}_{\mathsf{vss}}$ to be the outputs of the ideal-world honest parties. Therefore, we continue by assuming that the second-round inputs are well-formed as well.

from $(\mathsf{crs}, (\mathbf{G}_i)_{i \in \mathsf{H}}, (g_{i0}, \ldots, g_{in})_{i \in \mathsf{H}})$ by the same efficient process.[20] We conclude that the view in Hybrid 1 is $O(n^2\epsilon)$-close to the view in Hybrid 2.

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

We continue by analysing the honest parties' outputs in the verification phase. Let View denote the concatenation of (1) $\mathcal{Z}$'s view in the sharing phase, (2) the outputs of the honest parties in the sharing phase, and (3) $\mathcal{Z}$'s view in the verification phase. We say that View is "good" if for any commitment C from View, and for any two openings o, and o' that appear in View, it holds that either $\mathsf{open}(C, o) = \perp$ or $\mathsf{open}(C, o') = \perp$ or $\mathsf{open}(C, o) = \mathsf{open}(C, o)$. By the binding property, View is good with probability at least $1 - \epsilon$.

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

D.2.2 Corrupt Dealer

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

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

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

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

[20]We remind the reader that we consider the CRS string only in the case of statistically-hiding commitments. In the case of computationally-hiding commitments the CRS is simply an empty string.

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

Analysis. Since $\mathcal{S}$ always holds all the honest parties' inputs, and emulates the honest parties in an execution of vss, then the view of the sharing phase, together with the honest parties' outputs in the sharing phase and the view of the verification phase, have the same distribution in both worlds. It remains to analyse the output of the honest parties in the verification phase.

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

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

This process clearly works for a good View obtained from the real-world. For a good View obtained from the ideal-world, note that whenever D is discarded according to View then $\mathcal{S}$ sets $\mathsf{flag}_D = 1$, so all honest parties output "D is corrupt". Otherwise, note that the set W defined by the process is the same as the set W defined by $\mathcal{F}_{\mathsf{vss}}$. Indeed, all honest parties with invalid shares or with raised flag are in conflict with D, so they are in W according to the process and $\mathcal{F}_{\mathsf{vss}}$. In addition, $\mathcal{S}$ sets $\mathsf{flag}_i = 1$ for every corrupt P_i which is in W according to the process. In addition, it is not hard to verify that the set $\mathbf{O}_{\mathsf{W}}$ defined by the simulator is the same as the corresponding set defined by the simulator. Hence, it remains to show that the tuple $(\mathsf{W}, \mathbf{C}, \bar{\mathbf{O}}_{\mathsf{W}}, \mathbf{O}_{\mathsf{H}\setminus\mathsf{W}})$ is a weak double t-sharing of some value s.

Observe that for every $i \in \mathsf{W}$ the pair $(\mathbf{C}, \bar{\mathbf{O}}_i)$ is valid, or otherwise the honest parties would output "D is corrupt". In addition, for any $i \in \mathsf{H} \setminus \mathsf{W}$ the pair $(\mathbf{C}, \mathbf{O}_i)$ is also valid, or otherwise P_i would be unhappy with D, so P_i will be in W. In addition, since View is good, and D is not discarded, then $\mathsf{open}(C_{ij}, o'_{ij}) = \mathsf{open}(C_{ji}, o'_{ji})$ for every $i, j \in \mathsf{W} \cup \mathsf{H}$, where $o'_{ij} = \bar{o}_{ij}$ if $i \in \mathsf{W}$, and $o'_{ij} = o_{ij}$ otherwise. We conclude that $(\mathsf{W}, \mathbf{C}, \bar{\mathbf{O}}_{\mathsf{W}}, \mathbf{O}_{\mathsf{H}\setminus\mathsf{W}})$ is a weak double t-sharing of some value s. This completes the analysis of a corrupt dealer, and the proof of security of the protocol. $\square$

E Secure Partial Computation with a Guard

E.1 PSM Protocols

In this section we provide a formal definition of PSM, discussed in Section 3.3, together with a security statement.

Definition E.1 (PSM Protocols). *Let $X_1, \ldots, X_\ell, Z$ be finite sets, and let $X = X_1 \times \ldots \times X_\ell$. An ℓ-party PSM protocol* psm*, computing a ℓ-argument function $f : X \to Z$ consists of:*

- *A message computation function $\mathsf{psm}_i : X_i \times R \to M_i$, for every party $i \in \{1, \ldots, \ell\}$, where R is a finite set of common random inputs and M_i is a finite message domain.*

- *A reconstruction function* $\mathsf{rec} : M_1 \times \ldots \times M_\ell \to Z$ *that will be computed by the evaluator* E.

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

1. **(Correctness)** *For every* $(x_1, \ldots, x_\ell) \in X$ *and* $r \in R$, $\mathsf{rec}(\mathsf{psm}_1(x_1, r), \ldots, \mathsf{psm}_m(x_\ell, r)) = f(x_1, \ldots, x_\ell)$.

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

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

Lemma E.2 (Polynomial-time PSM Protocols [47])**.** *For every* ℓ*-argument functionality* f *that admits a Boolean NC*^{1} *circuit of size* s*, there exists a PSM protocol with complexity of* $\mathrm{poly}(s)$*. In particular, if* $s = \mathrm{poly}(\ell)$*, then there exists a PSM protocol with complexity* $\mathrm{poly}(\ell)$*.*

E.2 The Protocol

In this section we present the scg protocol. As discussed in Section 3.3, our goal is to let all parties learn the output of the function f_{scg} on the inputs of Alice and Bob, while learning nothing else about the inputs of Alice and Bob. (See Equation 1 for a formal definition of f_{scg}.) We think of f_{scg} as a binary-function, $f_{\mathsf{scg}} : \{0,1\}^\ell \to \{0,1\}^{\ell'}$, so that the first ℓ_A bits correspond to Alice's inputs, and the last $\ell_B = \ell - \ell_A$ bits correspond to Bob's inputs. We use an ℓ-party PSM protocol for the computation of f_{scg}, so that Alice simulates the first ℓ_A senders in the protocol, and Bob simulates the last ℓ_B senders (each sender holds a single input bit). In the offline-round, we let Bob pick the randomness r for the psm protocol and send it to Alice. In order to make sure that both Alice and Bob follow the psm protocol in the online round, we also let Bob commit all possible messages $\mathsf{psm}_i(x, r)$ for $x \in \{0,1\}$ and $i \in \{1, \ldots, \ell\}$ and send the openings to Alice. The commitments are published under a random shift, so that no information is leaked about the inputs of Alice and Bob. In the online-round, if Alice or Bob sends a PSM message $\mathsf{psm}_i(x, r)$ on behalf of the i-th sender, then they also have to provide the corresponding commitment's opening, so that the parties can verify that a valid message was sent. Every party then acts on the opening and PSM messages to either conclude Alice/Bob to be corrupt or obtain the output of f_{scg}. The protocol is efficient, because f_{scg} can be implemented as an NC^1 circuit. Protocol scg is given in Figure 6.

Protocol scg= (scg.off,scg.on)

Primitives: A PSM $\mathsf{psm} = (\mathsf{psm}_1, \ldots, \mathsf{psm}_\ell, \mathsf{rec})$ for $f_{\mathsf{scg}} : \{0,1\}^\ell \to \{0,1\}^{\ell'}$.

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

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

$\mathsf{scg.on}$**(R2):** The parties do as follows.

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

     Otherwise, if the output of f_{scg} is $((\mathbf{a}, \delta^A, (o_i, b_i^A)_{i=1}^m), (\delta^B, (b_i^B)_{i=1}^m))$, verify that $b_i^A \overset{?}{=} \mathsf{open}(C_i, o_i)$ for every $i \in \{1,\ldots,m\}$, and output "Alice is corrupt" if the verification fails. If the verification did not fail, output $(\mathbf{a}, \sum_{i\in\{1,\ldots,m\}} a_i \cdot b_i)$.

     Otherwise the output of f_{scg} is $(\mathbf{a}, \delta^A, \delta^B, \sum_{i\in\{1,\ldots,m\}} a_i \cdot b_i^A)$. Output $(\mathbf{a}, \sum_{i\in\{1,\ldots,m\}} a_i \cdot b_i^A)$.

Figure 6: Protocol scg= (scg.off,scg.on)

We continue with the proof of Lemma 4.2.

E.3 Proof of security

Proof of Lemma 4.2. In this section we prove that protocol scg UC-emulates $\mathcal{F}_{\mathsf{scg}}$ (with everlasting security when the underlying commitment scheme is statistically-hiding). Let $\mathcal{A}$ be the dummy adversary against scg. We define the simulator $\mathcal{S}$ as follows. The simulator $\mathcal{S}$ uses $\mathcal{A}$ in a black-box manner, and forwards all messages between $\mathcal{Z}$ and $\mathcal{A}$. The simulator first receives the set of corrupt parties C.

We split into cases, and begin by considering the case of honest Alice and Bob. In this case, we provide a single simulator (Section E.3.1), but two different analyses: one for the the case where the underlying commitment scheme is statistically-hiding and everlasting security is achieved (Section E.3.2), and one for the case where the underlying commitment scheme is only computationally-hiding, and computational security is achieved (Section E.3.3). Then, we present the rest of the cases (where at least one of Alice and Bob is corrupt), together with their analysis, which applies both to the case of statistically-hiding commitments and computationally-hiding commitments.

E.3.1 Honest Alice and Bob – Simulator

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

Online round. First,[21] $\mathcal{S}$ receives as leakage (1) the commitments $(C_1,\ldots,C_m)$, and (2) the output of f_{scg} on the inputs of Alice and Bob. We split into cases according the output of f_{scg}. We begin by treating the simple cases, where either $\delta^A = 1$, or $\delta^B = 1$.

- If the output of f_{scg} is $\bot$ then $\delta^A = 1$. In this case, $\mathcal{S}$ sends the messages of an honest Alice with the default values $\mathbf{a} = (0,\ldots,0)$, $\mathbf{b}^A = (0,\ldots,0)$, $(o_1,\ldots,o_m) = (0,\ldots,0)$ and with the flag $\delta^A = 1$. For Bob, $\mathcal{S}$ plays like in the protocol, assuming that the inputs of Bob are $\delta^B = 0$ and $\mathbf{b}^B = (0,\ldots,0)$.
- Otherwise, the output of f_{scg} is not $\bot$, so $\delta^A = 0$. If the output of f_{scg} is of the form $((\mathbf{a},\delta^A,(o_i,b^A_i)_{i=1}^m),(\delta^B,(b^B_i)_{i=1}^m))$, then $\delta^B = 1$. In this case the simulator holds the inputs of Alice and Bob and can perfectly simulate their messages in execution of scg.

We continue with the case where $\delta^A = \delta^B = 0$. In this case, since Alice and Bob are both honest, the output of f_{scg} is of the form $(\mathbf{a},0,0,\chi)$, where χ is the sum $\sum_{i\in\{1,\ldots,m\}} a_i \cdot b^A_i$. $\mathcal{S}$ finds $(b'_i)_{i\in\{1,\ldots,m\}}$ such that $\sum_{i\in\{1,\ldots,m\}} a_i \cdot b'_i = \chi$. This can be done by finding the first $i^* \in \{1,\ldots,m\}$ such that $a_{i^*} \neq 0$, and setting b'_{i^*} to $a_{i^*}^{-1}\cdot\chi$ while the rest of the b'_i's are set to 0 (if no such a_i exists then $\chi = 0$, and we can simply set $b'_i = 0$ for every i).

For each $i \in \{1,\ldots,m\}$ set $\bar{b}^A_i := b'_i$ and $\bar{b}^B_i := b'_i$. In addition, for $i \in \{1,\ldots,m\}$ set $\bar{o}_i := 0$. $\mathcal{S}$ then computes the binary string $x_A := (\mathbf{a},\delta_A,(\bar{o}_i,\bar{b}^A_i)_{i=1}^m)$, computes $\bar{s}_i := \mathsf{psm}_i(x^A_i, r)$ for each $i \in \{1,\ldots,\ell_A\}$ and broadcasts $(\sigma_i(x^A_i), o'_{i,x^A_i}, \bar{s}_i)_{i\in\{1,\ldots,\ell_A\}}$ on behalf of Alice. $\mathcal{S}$ also computes the binary string $x_B := (\delta_B,(\bar{b}^B_i)_{i=1}^m)$, computes $\bar{s}_i := \mathsf{psm}_i(x^B_{i-\ell_A}, r)$ for each $i \in \{\ell_A+1,\ldots,\ell\}$ and broadcasts $(\sigma_i(x^B_{i-\ell_A}), o'_{i,x^B_{i-\ell_A}}, \bar{s}_i)_{i\in\{\ell_A+1,\ldots,\ell\}}$ on behalf of Bob.[22] This concludes the simulation.

E.3.2 Honest Alice and Bob – Statistically-Hiding Commitment Scheme

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

[21] If the inputs of the honest parties are not well-formed (for example, if not all honest parties have the same commitments as inputs), then a complete break-down occurs (see Section A.1). In this case we note that we can obtain a perfect simulation. Indeed, the offline-round was simulated exactly like a real-world execution. In addition, the simulator, that now holds all the inputs of the honest parties, can continue following the protocol with those inputs in order to perfectly simulate the online-round. Finally, the simulator can compute the outputs of the honest parties in the execution, and determine them as the outputs of the honest parties in $\mathcal{F}_{\mathsf{scg}}$. Since the same reasoning also applies to the other cases (honest Alice and corrupt Bob; corrupt Alice and honest Bob; and corrupt Alice and Bob), the rest of the section assumes that the inputs of the honest parties are well-formed.

[22] Notice that in all three cases the simulation is done by computing some pre-image that gives the desired output and using it as the input of the PSM (in fact, when $\delta^B = 1$ we actually hold the inputs of Alice and Bob). Looking ahead, in the indistinguishability proof, we use PSM privacy that guarantees that those messages have (almost) the same distribution as in the real-world as long as the output remains the same in both cases.

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

We begin by showing that (1) has same distribution in both worlds. This would imply that (2) has the same distribution in both worlds, since given the first-round view, the environment picks the inputs of Alice and Bob in the same way in both worlds. Then, we consider the random variables $(r, (\sigma_i)_{i\in\{1,\dots,\ell\}})$, where where r is the PSM randomness sampled by Bob, and $(\sigma_i)_{i\in\{1,\dots,\ell\}}$ are the random shifts sampled by Bob. We show that even conditioned on (1), i.e $(\mathsf{crs}, (C'_{i,\sigma_i(x)})_{i\in\{1,\dots,\ell\},x\in\{0,1\}})$, in both worlds the random variables $(r, (\sigma_i)_{i\in\{1,\dots,\ell\}})$ are $O((\ell\epsilon)^{1/2})$-close to uniform, and there are overwhelmingly many such $(\mathsf{crs}, (C'_{i,\sigma_i(x)})_{i\in\{1,\dots,\ell\},x\in\{0,1\}})$ for which the above is true. This will allow us to extend the argument from $O((\ell\epsilon)^{1/2})$-closeness between distributions $(r, (\sigma_i)_{i\in\{1,\dots,\ell\}})$ and uniform to $O((\ell\epsilon)^{1/2})$-closeness between real and ideal world distributions of (3) and (4).[23]

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

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

We say that a fixing of $(\mathsf{crs}, (C'_{i,\sigma_i(x)})_{i\in\{1,\dots,\ell\},x\in\{0,1\}})$ is "good", if conditioned on this fixing, the random variables $(r, (\sigma_i)_{i\in\{1,\dots,\ell\}})$ are $O((\ell\epsilon)^{1/2})$-close to $(U_{|r|}, U_\ell)$. By Fact A.7 it follows that a fixing is good with probability at least $1 - O((\ell\epsilon)^{1/2})$. Condition on any good fixing of $(C'_{i,\sigma_i(x)})_{i\in\{1,\dots,\ell\},x\in\{0,1\}}$. At this stage the honest parties inputs are picked by $\mathcal{Z}$, and so they have the same distribution in both worlds. Fix those inputs as well.

It remains to show that

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

have the same distribution in both worlds, where i_j, o'_{j,i_j} and s_j are the indices, openings and PSM messages broadcasted by Alice and Bob in the online round. Since the simulator finds a preimage to the output of f_{scg}, and in both worlds $(r, (\sigma_i)_{i\in\{1,\dots,\ell\}})$ is $O((\ell\epsilon)^{1/2})$-close to $(U_{|r|}, U_\ell)$, and by the security of the PSM protocol, we conclude that in both worlds $((s_1,\dots,s_\ell), (i_1,\dots,i_\ell))$ are $O((\ell\epsilon)^{1/2})$-close to $(\mathcal{S}_{\mathsf{psm}}(\mathsf{val}), U_\ell)$, where $\mathcal{S}_{\mathsf{psm}}$ is the simulator of the PSM protocol, and val is the output of f_{scg}. Note that val is fixed, as we've already fixed the honest parties inputs. Finally, fix

[23]When $\delta^B = 1$ the simulator actually holds the inputs of Alice and Bob and we obtain perfect simulation. However, since the analysis captures this case as well, we don't need to split into cases.

$((s_1,\dots,s_\ell),(i_1,\dots,i_\ell))$ as well, and observe that the random variables $(o'_{1,i_1},\dots,o'_{\ell,i_\ell})$ have the same distribution in both worlds. We conclude that the real-world view is $O((\ell\epsilon)^{1/2})$-close to the ideal-world view.

Honest parties' outputs. Fix any view View of the adversary. In the real world, since Alice and Bob are honest, the verification of Alice always succeeds, and Alice and Bob are never discarded. If $\delta^A = 1$ then the ideal-world output is $\perp$, and by the perfect correctness of the PSM scheme the real-world output is $\perp$ as well. We continue by assuming that $\delta^A = 0$. In the ideal world, the output of the honest parties is $(\mathbf{a}, \sum_{i\in\{1,\dots,m\}} a_i \cdot b_i^A)$ with probability 1. In the real-world, the perfect correctness of the PSM protocol implies that all honest parties correctly recover the output of f_{scg}, which is either $(\mathbf{a}, \delta^A, \delta^B, \sum_{i\in\{1,\dots,m\}} a_i\cdot b_i^A)$ or $((\mathbf{a}, \delta^A, (o_i, b_i^A)_{i=1}^m), (\delta^B, (b_i^B)_{i=1}^m))$. In the first case the parties output $(\mathbf{a}, \sum_{i\in\{1,\dots,m\}} a_i \cdot b_i^A)$, and in the second case, since Alice is honest it holds that $\mathsf{open}(C_i, o_i) = b_i^A$ for all $i \in \{1,\dots,m\}$, so the the parties output $(\mathbf{a}, \sum_{i\in\{1,\dots,m\}} a_i \cdot b_i^A)$ as well. This concludes the case of honest Alice and Bob with statistically-hiding commitments.

E.3.3 Honest Alice and Bob - Computationally-Hiding Commitment Scheme

We continue by proving the security of our protocol when the underlying commitment scheme is merely computationally-hiding. We show that for the simulator described in Section E, no environment can distinguish the real-world from the ideal-world.

Under sub-exponential hardness assumption (see Section A.3 and Remark A.9), we may assume that for every security parameter μ, there exists an efficient commitment scheme $(\mathsf{commit}, \mathsf{open})$ which is perfectly-binding and computationally-hiding, and has error $2^{-\mu}$. We assume that the commitments used for the inputs of Alice and Bob have security parameter κ, which is the security parameter of the protocol. Observe that there exists a constant c such that the total bit-length of the honest inputs, which we've denoted by ℓ, is at most $c \cdot m \cdot (\kappa \log |\mathbb{F}|)^c$. In the protocol itself, for the committed PSM messages, we let Bob use a commitment scheme with security parameter κ', such that $\kappa' = \ell + \log \ell + \kappa$. In this section we set $\epsilon := 2^{-\kappa}$ and $\epsilon' := 2^{-\kappa'}$.

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

We begin with the following claim.

Claim E.3. *Let $\mathcal{Z}'$ be an environment, and let z' be an advice string. Assume that $\mathcal{Z}'(z')$ never corrupts Alice and Bob, and always gives the honest parties the same inputs at the beginning of the online phase. Then,*

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

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

Consider the real-world random variables

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

and the ideal-world random variables

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

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

Finally, since both Alice and Bob are honest and by the perfect correctness of the psm scheme, it follows that, in both worlds, the output of the honest parties is always $\perp$ when $\delta^A = 1$, and $(\mathbf{a}, \sum_{i\in\{1,\ldots,m\}} a_i \cdot b_i^A)$ otherwise. This completes the proof of the claim. □

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

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

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

Claim E.4. *Assume that* $|\Pr[\mathrm{REAL}_{\mathsf{scg},\mathcal{Z}(z),\mathcal{A}} = 1] - \Pr[\mathrm{IDEAL}_{\mathcal{F}_{\mathsf{scg}},\mathcal{Z}(z),\mathcal{S}} = 1]| = \delta$. *Then there exists* $i \in \{1, \ldots, 2^\ell\}$ *such that* $|\Pr[\mathrm{REAL}_{\mathsf{scg},\mathcal{Z}_i(z_i),\mathcal{A}} = 1] - \Pr[\mathrm{IDEAL}_{\mathcal{F}_{\mathsf{scg}},\mathcal{Z}_i(z_i),\mathcal{S}} = 1]| \geq \delta/2^\ell$.

Proof. Assume without loss of generality that

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

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

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

It follows that

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

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

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

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

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

Finally, assume that $|\Pr[\text{REAL}_{\mathsf{scg},\mathcal{Z}(z),\mathcal{A}} = 1] - \Pr[\text{IDEAL}_{\mathcal{F}_{\mathsf{scg}},\mathcal{Z}(z),\mathcal{S}} = 1]| = \delta$. By Claim E.4 there exists $i \in \{1, \ldots, 2^\ell\}$ such that $|\Pr[\text{REAL}_{\mathsf{scg},\mathcal{Z}_i(z_i),\mathcal{A}} = 1] - \Pr[\text{IDEAL}_{\mathcal{F}_{\mathsf{scg}},\mathcal{Z}_i(z_i),\mathcal{S}} = 1]| \geq \delta/2^\ell$. By Claim E.3 it follows that $|\Pr[\text{REAL}_{\mathsf{scg},\mathcal{Z}_i(z_i),\mathcal{A}} = 1] - \Pr[\text{IDEAL}_{\mathcal{F}_{\mathsf{scg}},\mathcal{Z}_i(z_i),\mathcal{S}} = 1]| < \ell\epsilon'$. We conclude that $|\Pr[\text{REAL}_{\mathsf{scg},\mathcal{Z}(z),\mathcal{A}} = 1] - \Pr[\text{IDEAL}_{\mathcal{F}_{\mathsf{scg}},\mathcal{Z}(z),\mathcal{S}} = 1]| < 2^\ell \cdot \ell\epsilon' = 2^{-\kappa}$, where the last equality follows by our choice of ϵ'. This concludes the analysis of honest Alice and Bob with computationally-hiding commitments.

E.3.4 Corrupt Alice, Honest Bob

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

Online round. $\mathcal{S}$ receives

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

from the leakage of the ideal functionality $\mathcal{F}_{\mathsf{scg}}$. $\mathcal{S}$ holds the input of Bob, and continues the simulation of Bob by sending the messages that Bob sends, and then receiving from $\mathcal{A}$ the massages that the corrupt Alice sends. At the end of the simulation, $\mathcal{S}$ computes the honest parties output, denoted v. If v is $\perp$ then the simulator inputs $\delta^A = 1$ (the rest of the inputs do not matter). Otherwise $\delta^A = 0$. If v is "Alice is corrupt", then $\mathcal{S}$ inputs $\mathsf{flag} := 1$ to $\mathcal{F}_{\mathsf{scg}}$ (the rest of the inputs do not matter). Otherwise, when v is not "Alice is corrupt", we split into cases.

1. If the corrupt Alice broadcasted $(\mathbf{a}, \delta^A, (o_i, b_i^A)_{i \in \{1,\ldots,m\}})$, let $\bar{o}_i := o_i$ and $\bar{b}_i^A := b_i^A$ for each $i \in \{1, \ldots, m\}$, and note that $\bar{b}_i^A = \mathsf{open}(C_i, \bar{o}_i)$ (or otherwise the honest parties would output "Alice is corrupt"). $\mathcal{S}$ inputs $(\mathbf{a}, \delta^A, (\bar{o}_i, \bar{b}_i^A)_{i=1}^m)$ and $\mathsf{flag} := 0$ to $\mathcal{F}_{\mathsf{scg}}$. (The input z of the doesn't matter.)

2. Otherwise, the corrupt Alice has broadcasted $(i_j, o'_j, s_j)_{j \in \{1,\ldots,\ell_A\}}$. For each $j \in \{1, \ldots, \ell_A\}$, $\mathcal{S}$ takes the index i_j broadcasted by the corrupt Alice, and sets $x_j := \sigma_j^{-1}(i_j)$. $\mathcal{S}$ then sets $x^A := (x_1, \ldots, x_{\ell_A}) \in \{0,1\}^{\ell_A}$, and parses $x^A = (\mathbf{a}, \delta^A, (o_i, b_i^A)_{i=1}^m)$. $\mathcal{S}$ inputs $(\mathbf{a}, \delta^A, (o_i, b_i^A)_{i=1}^m)$, $\mathsf{flag} := 0$ and $\mathsf{flag} := 0$ to $\mathcal{F}_{\mathsf{scg}}$. (The input z of the doesn't matter.)

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

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

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

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

- If the output according to View is $\perp$ then $\mathcal{S}$ sets $\delta^A = 1$, so the ideal-world output is also $\perp$. Otherwise $\delta^A = 0$.

- If the output according to View is "Alice is corrupt", then $\mathcal{S}$ sets $\mathsf{flag} = 1$, so the ideal-world output is also "Alice is corrupt". Otherwise $\mathsf{flag} = 0$.

- Otherwise, if Alice broadcasts $(\mathbf{a}, \delta^A, (o_i, b_i^A)_{i \in \{1,\ldots,m\}})$ and the output according to View is $(\mathbf{a}, \sum_{i \in \{1,\ldots,m\}} a_i \cdot b_i^A)$, then it must hold that $b_i^A = \mathsf{open}(C_i, o_i)$ for all $i \in \{1, \ldots, m\}$ (or otherwise Alice would be discarded). It is not hard to see that the ideal-world output is also $(\mathbf{a}, \sum_{i \in \{1,\ldots,m\}} a_i \cdot b_i^A)$.

- Otherwise, Alice broadcasts $(i_j, o'_j, s_j)_{j \in \{1,\ldots,\ell_A\}}$ and the output according to View is $\left(\mathbf{a}, \sum_{i \in \{1,\ldots,m\}} a_i \cdot b_i^A\right)$. If $\delta^B = 1$, or $\mathbf{b}^A \neq \mathbf{b}^B$ then the output of f_{scg} is $((\mathbf{a}, \delta^A, (o_i, b_i^A)_{i=1}^m), (\delta^B, (b_i^B)_{i=1}^m))$, so it must hold that $b_i^A = \mathsf{open}(C_i, o_i)$ for all $i \in \{1, \ldots, m\}$ (Otherwise, Alice is found to be corrupt). Otherwise, the output of f_{scg} is

$(\mathbf{a}, \delta^A, \delta^B, \sum_{i\in\{1,\ldots,m\}} a_i \cdot b_i^A)$. In both cases, since Alice is not found to be corrupt, it must hold that $\mathsf{open}(C'_{j,i_j}, o'_j) = s_j$ for any $j \in \{1,\ldots,\ell_A\}$. Since View is good, and by the correctness of the underlying PSM protocol, we conclude that the output in the ideal-world is also $(\mathbf{a}, \sum_{i\in\{1,\ldots,m\}} a_i \cdot b_i^A)$.

This concludes the case of corrupt Alice and honest Bob.

E.3.5 Honest Alice, Corrupt Bob

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

Online round. The adversary receives the inputs of the honest Alice, $(\mathbf{a}, \delta^A, (o_i, b_i^A)_{i=1}^m)$, and computes the online round broadcast of the honest Alice, based on the offline-round message from Bob to Alice. $\mathcal{S}$ then receives from $\mathcal{A}$ the online round broadcast of the corrupt Bob.

At the end of the simulation $\mathcal{S}$ computes the output of the honest parties in the simulation. If the output is $\perp$ then the inputs to $\mathcal{F}_{\mathsf{scg}}$ doesn't matter. If the output is "Bob is corrupt" $\mathcal{S}$ inputs $\mathsf{flag} = 1$ to $\mathcal{F}_{\mathsf{scg}}$ (the rest of the inputs do not matter). Otherwise, the output is not "Bob is corrupt", and $\mathcal{S}$ inputs $\mathsf{flag} = 0$ to $\mathcal{F}_{\mathsf{scg}}$ (the rest of the inputs do not matter).

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

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

Honest parties' output. We say that a view View is "good" if for any commitment C from the view, and for any two openings o, and o' that appear in the view, it holds that either $\mathsf{open}(C, o) = \perp$ or $\mathsf{open}(C, o') = \perp$ or $\mathsf{open}(C, o) = \mathsf{open}(C, o)$. By the binding property, a view View is good with probability at least $1 - \epsilon$. Whenever a view is good, it is not hard to verify that the output in the ideal-world is equal to the output of the honest parties according to View. This concludes the case of honest Alice and corrupt Bob.

E.3.6 Corrupt Alice and Bob

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

E.3.7 Complexity Analysis

Here we sketch the complexity analysisof protocol scg. In order to show that the complexity of scg is $\mathrm{poly}(n, \log|\mathbb{F}|, \kappa, m)$, it is enough to show that the complexity of the underlying PSM protocol is

$\text{poly}(n, \log |\mathbb{F}|, \kappa, m)$. For this, according to Theorem E.2 it is enough to show that f_{scg} has a circuit with size polynomial in $(n, \log |\mathbb{F}|, \kappa, m)$ and depth logarithmic in $(n, \log |\mathbb{F}|, \kappa, m)$. This follows by standard use of multiplication and addition over extension fields of the binary field, that can be done in NC1 (see [45]), and by the fact the comparing two strings requires logarithmic depth.

This completes the complexity analysis, and completes the proof of security of the protocol. □

F Guided Linear Function Computation

F.1 The Protocol

In this section we present the glinear protocol, discussed in Section 4.3. The protocol simply invokes n instances of SCG with G as Alice and every P_j as Bob, computing the function $a_1 \cdot b^G_{1j} + \ldots + a_m \cdot b^G_{mj}$. If (a) G (aka Alice) gets identified as corrupt in any of the n SCG instances, or (b) it inputs different $\mathbf{a}$ or different flags to the SCG instances, or (c) the outputs of SCGs corresponding to instances where Bob is not identified as corrupt lead to a non-degree-t polynomial, then G is identified as corrupt. Otherwise, the the parties use the outputs of all SCGs whose output is not "Bob is corrupt" in order to interpolate the degree-t polynomial $f(x)$, and output $(\mathbf{a}, f(0))$.

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

Primitives: SCG scheme $\mathsf{scg} = (\mathsf{scg.off}, \mathsf{scg.on})$.

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

$\mathsf{glinear.on}$ **inputs:** All parties hold the same commitments $(C_{ij})_{i\in\{1,\ldots,m\},j\in\{0,\ldots,n\}}$.

G holds (1) a list of coefficients $\mathbf{a} = (a_1, \ldots, a_m) \in \mathbb{F}^m$, (2) a list of values $\mathbf{b}^G = (b^G_{ij})_{i\in\{1,\ldots,m\},j\in\{0,\ldots,n\}} \in \mathbb{F}^{m(n+1)}$, (3) a list of openings $(o^G_{ij})_{i\in\{1,\ldots,m\},j\in\{0,\ldots,n\}}$, and (4) a flag $\delta^G \in \{0, 1\}$.

Every P_j holds a flag $\delta^j \in \{0, 1\}$, and values $(b^j_1, \ldots, b^j_m) \in \mathbb{F}^m$.

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

- All parties input $(C_{1j}, \ldots, C_{mj})$.
- G, as Alice, inputs the coefficient vector $\mathbf{a}$, the vector of values $(b^G_{1j}, \ldots, b^G_{mj})$, the flag δ^G and the openings $(o^G_{ij})^m_{i=1}$ (if $\delta^G = 1$ then we assume that the guide inputs some default values for $\mathbf{a}$, $(b^G_{1j}, \ldots, b^G_{mj})$ and $(o^G_{ij})^m_{i=1}$);
- P_j, as Bob, inputs the flag δ^j and the values $(b^j_1, \ldots, b^j_m)$ (if $\delta^j = 1$, P_j inputs some default values for $(b^j_1, \ldots, b^j_m)$).

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

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

Notation 4. *For a set of m final j-th rows $[\![s_1]\!]_j, \ldots, [\![s_m]\!]_j$ we say that the parties participate in* glinear *to compute $\sum_{k=1}^{m} a_k [\![s_k]\!]_j$, with party P_j as a guide G, and with flag δ^G to the guide and δ^k to P_k, to mean an execution of* glinear *where (a) all parties input the commitments corresponding to the j-th rows $[\![s_1]\!]_j, \ldots, [\![s_m]\!]_j$, (b) the guide P_j inputs $\mathbf{a} = (a_1, \ldots, a_m)$, δ^G, and the commitments, openings and values corresponding to the j-th rows, and (c) every P_k inputs δ^k and the values corresponding to the k-th shares of each row. We also use a similar notation in the case where we only have m* tentative *j-th rows $[\![s_1]\!\rrbracket_j, \ldots, [\![s_m]\!\rrbracket_j$.*

We continue with the proof of Lemma 4.3.

F.2 Proof of security

Proof of Lemma 4.3. In this section we prove that protocol glinear *perfectly* UC-emulates $\mathcal{F}_{\mathsf{glinear}}$ in the $\mathcal{F}_{\mathsf{scg}}$-hybrid model. From the composition properties of UC-security, this implies that protocol glinear UC-emulates $\mathcal{F}_{\mathsf{glinear}}$ (with everlasting security if the underlying commitment scheme is statistically-hiding). Let $\mathcal{A}$ be the dummy adversary against glinear. We define the simulator $\mathcal{S}$ as follows. The simulator $\mathcal{S}$ uses $\mathcal{A}$ in a black-box manner, and forwards all messages between $\mathcal{Z}$ and $\mathcal{A}$. The simulator first receives the set of corrupt C parties from $\mathcal{Z}$. We split into cases.

F.2.1 Honest Guide

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

Online round. First[24], the simulator receives the output and leakage of $\mathcal{F}_{\mathsf{glinear}}$. We start with the simple case where the output is $\perp$ and the leakage is $(C_{ij})_{i\in\{1,\ldots,m\}, j\in\{0,\ldots,n\}}$. In this case the simulator sets the output and leakage of every $\mathcal{F}^j_{\mathsf{scg}}$ for an honest P_j to be $\perp$ and $(C_{1j}, \ldots, C_{mj})$, respectively. In addition, for every $\mathcal{F}^j_{\mathsf{scg}}$ for a corrupt P_j the simulator sets the output and leakage to be $\perp$ and the default values of $\mathbf{a}$, $(b^G_{1j}, \ldots, b^G_{mj})$ and $(o^G_{ij})_{i=1}^m$. This concludes the simulation when the output is $\perp$.

Otherwise, the output is not $\perp$, and the simulator obtains the following leakage from $\mathcal{F}_{\mathsf{glinear}}$: (1) the commitments $(C_{ij})_{i\in\{1,\ldots,m\}, j\in\{0,\ldots,n\}}$, (2) the honest parties' flags $(\delta^i)_{i\in\mathsf{H}}$ and δ^G, (3) parts of the dealer's inputs $\mathbf{a}$, $(b^G_{i,j}, o^G_{i,j})_{i\in\{1,\ldots,m\}, j\in\mathsf{C}}$, (4) $v_j := \sum_{i\in\{1,\ldots,m\}} a_i b^G_{ij}$ for any $j \in \{1, \ldots, n\}$, and (5) $(b^G_{i,j}, o^G_{i,j})_{i\in\{1,\ldots,m\}}$ for every honest P_j with $\delta^j = 1$. In addition, the simulator receives the output of $\mathcal{F}_{\mathsf{glinear}}$, $v := \sum_{i\in\delta^G} a_i b^A_{i0}$.

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

[24]If the inputs of the honest parties are not well-formed (for example, if not all honest parties have the same commitments as inputs), then a complete break-down occurs (see Section A.1). In this case we can obtain a perfect simulation by following the same approach as in Footnote 21. Note that in this case complete break-down might also occur also in the internal $\mathcal{F}_{\mathsf{scg}}$ calls, and that they can be simulated since the simulator holds all the inputs of the honest parties. A similar argument applies to the case of a corrupt guide, so in the rest of this section we assume that the honest parties' inputs are well-formed.

- If $j \in \mathsf{H}$ and $\delta^j = 0$, then the simulator gives $(C_{1j}, \dots, C_{mj})$, and $(\mathbf{a}, \delta^G, \delta^j, v_j)$, to $\mathcal{A}$ as the leakage from $\mathcal{F}^j_{\mathsf{scg}}$. Otherwise, if $\delta^j = 1$, the simulator gives $(C_{1j}, \dots, C_{mj})$, and $((\mathbf{a}, \delta^G, (b^G_{ij}, o^G_{ij})_{i \in \{1,\dots,m\}}), (\delta^j, \mathbf{b}^j))$ as the leakage from $\mathcal{F}^j_{\mathsf{scg}}$, where $\mathbf{b}^j$ is the default vector used by P_j when $\delta^j = 1$. The simulator also gives $(\mathbf{a}, v_j)$ as the output of $\mathcal{F}^j_{\mathsf{scg}}$.
- If $j \in \mathsf{C}$ then the simulator gives the adversary the values $(C_{1j}, \dots, C_{mj})$, $\mathbf{a}$, $(b^G_{ij})_{i \in \{1,\dots,m\}}$, δ^G, and $(o^G_{ij})_{i \in \{1,\dots,m\}}$ as leakage from $\mathcal{F}^j_{\mathsf{scg}}$. Later, the simulator receives P_j's inputs to $\mathcal{F}^j_{\mathsf{scg}}$, denoted $\mathbf{b}^B$, δ^B, flag, and z from $\mathcal{A}$. Upon receiving the inputs, the simulator, that holds all inputs to $\mathcal{F}^j_{\mathsf{scg}}$, computes the output of the functionality, and returns it to $\mathcal{A}$.

This completes the simulation.

Fix any polynomial time environment $\mathcal{Z}$ and input ζ to the environment. We show that the view of $\mathcal{Z}$ in the real world has the same distribution as the view of $\mathcal{Z}$ in the ideal world. It is not hard to verify that this is the case when the output of $\mathcal{F}_{\mathsf{glinear}}$ is $\perp$, so we only focus on the case where the output is not $\perp$.

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

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

F.2.2 Corrupt Guide

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

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

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

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

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

Honest parties' output. Fix any view View of the adversary. If the output of the honest parties in the real-world according to View is $\perp$, then the simulator sets $\delta^G = 1$, and the output in the ideal-world is $\perp$ as well. Otherwise, $\delta^G = 0$. If the output of the honest parties in the real-world according to View is "G is corrupt" then the simulator sets $\mathsf{flag} = 1$, and so the output of the honest parties in the ideal-world is "G is corrupt". Therefore, from now on we assume that the output of the honest parties according to View is not $\perp$ or "G is corrupt".

Denote by K the set of all indices $k \in \{1,\dots,n\}$ such that the output of $\mathcal{F}^{k}_{\mathsf{scg}}$ is not "Bob is corrupt", and observe that $\mathsf{H} \subseteq K$. Since the guide is not discarded then (1) there is no $\mathcal{F}^{j}_{\mathsf{scg}}$ whose output is "Alice is corrupt", (2) the dealer sent the same $\mathbf{a}$ and δ^G to any $\mathcal{F}^{k}_{\mathsf{scg}}$ for $k \in H$, (3) for every $k \in H$ with $\delta^k = 1$ it holds that $\mathsf{open}(C_{ik}, o^{A,k}_i) = b^{A,k}_i$ for all $i \in \{1,\dots,m\}$, (4) for every $k \in H$ with $\delta^k = 0$ where $(b^k_1,\dots,b^k_m) \neq (b^{A,k}_1,\dots,b^{A,k}_m)$ it holds that $\mathsf{open}(C_{ik}, o^{A,k}_i) = b^{A,k}_i$ for all $i \in \{1,\dots,m\}$, and (5) For every $k \in H$ the output of $\mathcal{F}^{k}_{\mathsf{scg}}$ is $(\mathbf{a}, v^k)$, where $v^k := \sum_{i\in\{1,\dots,m\}} a_i \cdot b^{A,k}_i$, (6) the shares $(v^k)_{k\in K}$ correspond to a degree-t polynomial, denoted $g(x)$, where v^k is the sum obtained from the output of $\mathcal{F}^{k}_{\mathsf{scg}}$. Sine $\mathsf{H} \subseteq K$ and $|\mathsf{H}| \geq t+1$, we conclude that in both worlds the output of the honest parties is $(\mathbf{a}, g(0))$, as required. This concludes the analysis of a corrupt dealer, and the proof of security of the protocol. $\square$

G General Linear Function Computation

In this section, we sketch the 3-round protocol for general linear function computation. The protocol is round-optimal due to the lower bound of [37].

First, we focus on the simple case where the parties compute a function $y = a_1 \cdot s_1 + \ldots + a_m \cdot s_m$, where $a_1, \ldots, a_m \in \mathbb{F}$ are some known coefficients, and $s_1, \ldots, s_m$ are the inputs of the parties, where we denote by $I_i \subseteq \{1, \ldots, m\}$ the set of all indices j such that s_j is an input of P_i. Later, we explain how to obtain a protocol for general linear functions.

The basic protocol. In rounds 1 and 2 each P_i shares its inputs via an instance of vss. In round 2 the parties also execute an instance of $\mathsf{glinear.off}^i$ with P_i as a guide, for all $i \in \{1, \ldots, n\}$. At the end of this round, for every P_i that was disqualified as a dealer in a vss execution, the parties set $s_j = 0$ for all $j \in I_i$, and assume some default sharing of 0, which is known to all parties. In round 3, for any $i \in \{1, \ldots, n\}$ the parties execute $\mathsf{glinear.on}^i$, computing the i-th row of the sum $a_1[\![s_1]\!]_i + \ldots + a_m[\![s_m]\!]_i$ with P_i as a guide,[25] over the *final rows* taken from the vss instances. The abort flag and the reveal-all flag should be set to 0. For every $i \in \{1, \ldots, n\}$, we say that the output of $\mathsf{glinear.on}^i$ is valid if it is of the form $(\mathbf{a}, \sigma_i)$, where $\mathbf{a} = (a_1, \ldots, a_m)$. The parties interpolate over all valid σ_i to obtain a degree-t polynomial $f(x)$, and output $f(0)$.

Extension to multiple public outputs. So far we have only considered a functionality with a single output which is given to all the parties. Consider now a functionality with multiple public outputs $(y_1, \ldots, y_k)$, where each y_i is a linear function of the inputs $s_1, \ldots, s_m$, i.e., $y_i = a_{i1} \cdot s_1 + \ldots + a_{im} \cdot s_m$, for some $a_{i1}, \ldots, a_{im} \in \mathbb{F}$. We modify the basic protocol as follows. In Round 2, instead of executing a single instance of $\mathsf{glinear.off}$ for every P_i, we let the parties execute k instances, denoted $\mathsf{glinear.off}^{i,j}$ for $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, k\}$. In Round 3, for every $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, k\}$ the parties execute $\mathsf{glinear.on}^{i,j}$, computing the i-th row of the sum $a_{j1}[\![s_1]\!]_i + \ldots + a_{jm}[\![s_m]\!]_i$ with P_i as a guide, over the final rows taken from the vss instances. Finally, for every $j \in \{1, \ldots, k\}$ the parties act like in the basic protocol and use all valid outputs from $(\mathsf{glinear.on}^{i,j})_{i \in \{1,\ldots,n\}}$ in order to recover y_j.

General linear computation. It is well known that general linear function, where some of the outputs of the functionality are private, can be reduced to linear function computation with public outputs by the use of one-time pads. In particular, if some output y_i should be learned only by some party P_j, we let P_j sample a random pad r_{ij}, and we let the parties publicly compute the value $y'_i = y_i(s_1, \ldots, s_m) + r_{ij}$ instead of $y_i(s_1, \ldots, s_m)$. In this way, the rest of the parties learn no information about y_i while P_j, who knows r_{ij}, can compute $y_i = y'_i - r_{ij}$.

H Triple Secret Sharing for a Small Number of Parties

In this section we present the functionality $\mathcal{F}_{\mathsf{tss}}$, protocol tss and the proof of Theorem 5.1. Functionality $\mathcal{F}_{\mathsf{tss}}$ is given in Figure 8. For a reminder on weak and strong sharing, and of the definition of valid shares, see Section D.

[25]See Notation 4 for an explanation about the notation when using the glinear protocol.

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

$\mathcal{F}_{\mathsf{tss}}$ receives the set of corrupt parties C.

Sharing phase.

- **Inputs.** D inputs three pairs $(\mathbf{C}^a, \mathbf{O}^a), (\mathbf{C}^b, \mathbf{O}^b)$ and $(\mathbf{C}^c, \mathbf{O}^c)$. If D is honest, then these correspond to strong sharings $\langle\!\langle a \rangle\!\rangle, \langle\!\langle b \rangle\!\rangle$, and $\langle\!\langle c \rangle\!\rangle$ such that $c = ab$.
- **Public outputs.** $\mathcal{F}_{\mathsf{tss}}$ returns $\mathbf{C}^a, \mathbf{C}^b$ and $\mathbf{C}^c$ as public outputs.
- **Private outputs.** $\mathcal{F}_{\mathsf{tss}}$ returns $\mathbf{O}^a_i, \mathbf{O}^b_i$ and $\mathbf{O}^c_i$ to P_i.

Verification phase.

- **Adversary's inputs.** A corrupt dealer has two inputs, a bit flag_D, and $(\bar{\mathbf{O}}^a, \bar{\mathbf{O}}^b, \bar{\mathbf{O}}^c)$, where $\bar{\mathbf{O}}^v = (\bar{o}^v_{ij})_{i,j\in\{0,\ldots,n\}}$ for $v \in \{a, b, c\}$.
- **Outputs.** We split into two cases.
  - **Honest** D**.** $\mathcal{F}_{\mathsf{tss}}$ returns "verification succeeded" to all parties as a public output.
  - **Corrupt** D**.** Let W be the set of all honest parties that received invalid shares in the sharing phase. The functionality returns "D is corrupt" as a public output if either (1) $\mathsf{flag}_D = 1$, (2) the tuple $(\mathsf{W}, \mathbf{C}^v, \bar{\mathbf{O}}^v_{\mathsf{W}}, \mathbf{O}^v_{\mathsf{H}\setminus\mathsf{W}})$ is not a weak double t-sharing, for some $v \in \{a, b, c\}$, or (3) it does not holds that $F^c(0,0) = F^a(0,0)\cdot F^b(0,0)$, where $F^v(x,y)$ is the sharing polynomial of $(\mathsf{W}, \mathbf{C}^v, \bar{\mathbf{O}}^v_{\mathsf{W}}, \mathbf{O}^v_{\mathsf{H}\setminus\mathsf{W}})$, for $v \in \{a, b, c\}$. Otherwise, $\mathcal{F}_{\mathsf{tss}}$ returns "verification succeeded" to all parties as a public output, and $(\mathbf{C}^a, \bar{\mathbf{O}}^a_i)$, $(\mathbf{C}^b, \bar{\mathbf{O}}^b_i)$ and $(\mathbf{C}^c, \bar{\mathbf{O}}^c_i)$ to every P_i in W as a private output.

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

Protocol tss is given in Figure 9. In the protocol, we use Notation 4 in order to specify the inputs to the sub-protocol $\mathsf{glinear}$.

Protocol tss

Primitives: Guided linear function evaluation $\mathsf{glinear} = (\mathsf{glinear.off}, \mathsf{glinear.on})$; VSS vss.

Sharing phase (R1):

- *(Inputs)* D inputs three pairs $(\mathbf{C}^a, \mathbf{O}^a), (\mathbf{C}^b, \mathbf{O}^b)$ and $(\mathbf{C}^c, \mathbf{O}^c)$, that are strong t-sharing of a, b and c, respectively, such that $ab = c$.
- *(Sharing phase of VSS)*
  - D picks three random polynomials $A(x), B(x)$, and $C(x)$ of degree 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 $d := \binom{n}{t+1}\cdot(t+1)$. Let A^k, B^k, and C^k denote the k-th coefficient of $A(x), B(x)$ and $C(x)$, respectively, for $k \in \{0, \ldots, 2n\}$, where $A^k = B^k = 0$ for $k > d$.
  - 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 $F^{b,k}(x,y)$.[a] 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 $(\mathbf{C}^a, \mathbf{O}^a), (\mathbf{C}^b, \mathbf{O}^b)$ and $(\mathbf{C}^c, \mathbf{O}^c)$, respectively. We denote the corresponding sharing polynomials by $F^{a,0}(x,y), F^{b,0}(x,y)$ and $F^{c,0}(x,y)$.

- *(Generating challenges)* 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 a non-zero random field element $\alpha_Q \leftarrow \mathbb{F} \setminus \{0\}$ and sends it to all parties in Q via private channels.
- *(*glinear.off *calls)* For every set Q of $t+1$ parties, and every $i \in Q$ and $v \in \{a,b,c\}$, the parties execute $\mathsf{glinear.off}^{Q,i,v}$, with P_i as the guide.
- *(Local computation)*
  - *(VSS sharing phase output)* For every $k \in \{0,\dots,2d\}$, the parties hold the tentative shares $[\![A^k]\!], [\![B^k]\!]$ and $[\![C^k]\!]$.
  - *(Happiness)* If P_i received an invalid pair $(\mathbf{C}^{v,k}, \mathbf{O}_i^{v,k})$ for some $v \in \{a,b,c\}$ and $k \in \{0,\dots,2d\}$, then P_i is "unhappy", and sets $\mathsf{happy}_i := 0$. Otherwise, P_i is "happy", and sets $\mathsf{happy}_i := 1$.
  - If P_i received $\alpha_Q = 0$ for some set Q with $i \in Q$, then P_i sets $\alpha_Q := 1$.
  - *(Output)* P_i outputs the tentative shares $(\mathbf{C}^{a,0}, \mathbf{O}_i^{a,0}), (\mathbf{C}^{b,0}, \mathbf{O}_i^{b,0}), (\mathbf{C}^{c,0}, \mathbf{O}_i^{c,0})$.

Verification phase (R2):

- Each P_i broadcasts happy_i.
- *(Verification phase of VSS)* Each unhappy P_i raises a flag in all vss executions, and each happy P_i does not raise a flag in the vss executions.
- *(Challenge computation)* For every subset Q of $t+1$ parties, and every $i \in Q$, (a) P_i broadcasts the challenge α_Q, which was received from P_Q; (b) all parties (inside and outside Q) participate in $\mathsf{glinear.on}^{Q,i,a}$, $\mathsf{glinear.on}^{Q,i,b}$ and $\mathsf{glinear.on}^{Q,i,c}$ to compute $\sum_{k=0}^{2d} \alpha_Q^k [\![A^k]\!]_i$, $\sum_{k=0}^{2d} \alpha_Q^k [\![B^k]\!]_i$ and $\sum_{k=0}^{2d} \alpha_Q^k [\![C^k]\!]_i$, respectively, with P_i as a guide, with flag $\delta^G := \overline{\mathsf{happy}_i}$ and every P_j, $j \in \{1,\dots,n\}$ with flag $\delta^j := \overline{\mathsf{happy}_m}$.[b]
- *(Local computation)*
  - *(VSS verification phase outputs)* The parties compute the output the verification phase of all vss executions. If the output of some execution is "D is corrupt" then they output "D is corrupt" and terminate. Otherwise, denote the output of $\mathsf{vss}^{v,k}$ by $(\mathsf{W}^{v,k}, (\bar{o}_{ij}^{v,k})_{i \in \mathsf{W}^{v,k}, j \in \{0,\dots,n\}})$. Let $\mathsf{W} := \left(\bigcap_{k \in \{0,\dots,d\}} \mathsf{W}^{a,k}\right) \cap \left(\bigcap_{k \in \{0,\dots,d\}} \mathsf{W}^{b,k}\right) \cap \left(\bigcap_{k \in \{0,\dots,2d\}} \mathsf{W}^{c,k}\right)$.
  - *(Challenge verification)* Denote the output of $\mathsf{glinear}^{Q,i,v}$ by $\mathsf{out}^{Q,i,v}$. A set Q is termed as *good* if (1) there exists $\alpha \in \mathbb{F} \setminus \{0\}$ such that all the parties in Q broadcasted the challenge α, and for all the parties P_i in Q who broadcasted $\mathsf{happy}_i = 1$, and for all $v \in \{a,b,c\}$, the output of $\mathsf{out}^{Q,i,v}$ is $(\boldsymbol{\alpha}, u_{Q,i,v})$, where $\boldsymbol{\alpha} = (\alpha^0, \dots, \alpha^{2d})$, and (2) every P_i in Q who broadcasted $\mathsf{happy}_i = 0$ is in W.
    For every good set Q with corresponding challenge α, the parties do as follows.
    * For each P_i in Q such that $\mathsf{happy}_i = 1$, and for $v \in \{a,b,c\}$, let $\mathsf{out}^{Q,i,v} = (\boldsymbol{\alpha}, u_{Q,i,v})$.
    * For each P_i in Q such that $\mathsf{happy}_i = 0$, and for each $v \in \{a,b,c\}$, let $f_i^{v,k}(x)$ be the i-th row extracted from the verification phase of $\mathsf{vss}^{v,k}$. The parties locally compute $f_{Q,i,v}(x) := \alpha^0 \cdot f_i^{v,0}(x) + \ldots + \alpha^{2d} \cdot f_i^{v,2d}(x)$, and set $u_{Q,i,v} := f_{Q,i,v}(0)$.
    * For every $v \in \{a,b,c\}$, the parties interpolate over $(u_{Q,i,v})_{i \in Q}$ in order to obtain a degree-t polynomial $f_{Q,v}(x)$. If $f_{Q,c}(0) = f_{Q,a}(0) \cdot f_{Q,b}(0)$ then the verification succeeds. Otherwise, all parties output "D is corrupt" and terminate.
  - *(Output)* If all verifications of good sets Q succeeded then all parties output "verification succeeded" In addition, any unhappy party P_i in W that did not receive valid shares in $\mathsf{vss}^{a,0}$, $\mathsf{vss}^{b,0}$ or $\mathsf{vss}^{c,0}$ outputs $(\mathbf{C}^a, \bar{\mathbf{O}}^{a,0})$, $(\mathbf{C}^b, \bar{\mathbf{O}}^{b,0})$ and $(\mathbf{C}^c, \bar{\mathbf{O}}^{c,0})$.

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

[b]Here $\overline{\mathsf{happy}} = 1 - \mathsf{happy}$.

Figure 9: Protocol tss

Remark H.1 (Reducing the field size)**.** *So far we assumed that the size of $\mathbb{F}$ is exponential in the security parameter. We can reduce the size of the field by increasing d to be $d := \kappa \cdot \binom{n}{t+1} \cdot (t+1) = \mathrm{poly}(\kappa)$, and letting each set Q generate κ random challenges $\alpha_Q^1, \ldots, \alpha_Q^\kappa$, instead of just one. We can now take the size of the field to be $|\mathbb{F}| > 4d+1 = \mathrm{poly}(\kappa)$, and the probability that for a good set Q it holds that $C(x) \neq A(x) \cdot B(x)$ and $C(\alpha_Q^i) = A(\alpha_Q^i) \cdot B(\alpha_Q^i)$ for all $i \in \{1,\ldots,\kappa\}$ is at most $(2d/4d)^\kappa = 2^{-\kappa}$.*

We continue with the proof of Theorem 5.1.

H.1 Proof of security

Proof of Theorem 5.1. In this section we prove that protocol tss UC-emulates $\mathcal{F}_{\mathsf{tss}}$ (with everlasting security when the underlying commitment scheme is statistically-hiding). From the composition properties of UC-security, it is enough to prove security in the $(\mathcal{F}_{\mathsf{vss}}, \mathcal{F}_{\mathsf{glinear}})$-hybrid model. Let $\mathcal{A}$ be the dummy adversary against tss. We define the simulator $\mathcal{S}$ as follows. The simulator $\mathcal{S}$ uses $\mathcal{A}$ in a black-box manner, and forwards all messages between $\mathcal{Z}$ and $\mathcal{A}$. The simulator first receives the set of corrupt C parties from $\mathcal{Z}$. We split into cases.

H.1.1 Honest Dealer

Sharing phase. First,[26] the simulator receives $(\mathbf{C}^a, \mathbf{O}_i^a)$, $(\mathbf{C}^b, \mathbf{O}_i^b)$, $(\mathbf{C}^c, \mathbf{O}_i^c)$, for every $i \in \mathsf{C}$, from $\mathcal{F}_{\mathsf{tss}}$. We sometimes denote the pairs by $(\bar{\mathbf{C}}^{a,0}, \bar{\mathbf{O}}_i^{a,0})$, $(\bar{\mathbf{C}}^{b,0}, \bar{\mathbf{O}}_i^{b,0})$, and $(\bar{\mathbf{C}}^{c,0}, \bar{\mathbf{O}}_i^{c,0})$, respectively.

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

The simulator picks random polynomials $\bar{A}(x), \bar{B}(x)$ and $\bar{C}(x)$ of degree d, d and $2d$, respectively, such that $\bar{C}(x) = \bar{A}(x) \cdot \bar{B}(x)$. For $k \in \{0,\ldots,2d\}$, We denote their k-th coefficient by $\bar{A}^k$, $\bar{B}^k$ and $\bar{C}^k$, respectively, where $\bar{A}^k = \bar{B}^k = 0$ for $k > d$. For each $k \in \{0,\ldots,2d\}$, the simulator picks random symmetric bivariate polynomials $\bar{F}^{a,k}(x,y)$, $\bar{F}^{b,k}(x,y)$ and $\bar{F}^{c,k}(x,y)$ of degree at most t in each variable, such that for $k \in \{0,\ldots,d\}$ we condition on (1) $\bar{F}^{v,k}(x,i) = \bar{f}_i^{v,k}(x)$ for

[26]If the inputs of the honest D are not well-formed, then a complete break-down occurs (see Section A.1). In this case perfect simulation is possible following the approach presented in Footnote 18.

every $v \in \{a,b,c\}$ and $i \in \mathsf{C}$, and (2) $\bar{F}^{a,k}(0,0) = \bar{A}^k$, $\bar{F}^{b,k}(0,0) = \bar{B}^k$, and $\bar{F}^{c,k}(0,0) = \bar{C}^k$; for $k \in \{d+1,\ldots,2d\}$ we condition on (1) $\bar{F}^{a,k}(x,y) = 0$, (2) $\bar{F}^{b,k}(x,y) = 0$, and (3) $\bar{F}^{c,k}(x,i) = \bar{f}_i^{c,k}(x)$ for every $i \in \mathsf{C}$, and $\bar{F}^{c,k}(0,0) = \bar{C}^k$.

We continue with the challenge generation. For every subset Q of exactly $t+1$ parties, that has an honest distinguished party P_Q, the simulator samples a random non-zero field element α_Q and sends it to all corrupt parties in Q.

This concludes the communication from honest parties to corrupt parties. At this stage the simulator receives the messages from the corrupt parties to the honest parties, that include, for each subset Q with a corrupt distinguished party P_Q, the challenge that P_Q sends to each P_i in Q, which we denote by $\alpha_{Q,i}$ (note that a corrupt P_Q might send different values to different honest parties). If some $\alpha_{Q,i}$ is 0, then $\mathcal{S}$ sets $\alpha_{Q,i}$ to 1. Consider all the different challenges which are held by at least one honest party, and denote them by $\alpha_1,\ldots,\alpha_m$, where $m \leq d$.

Verification phase. For any honest party P_i the simulator broadcasts $\mathsf{happy}_i = 1$. For each $\mathcal{F}_{\mathsf{vss}}$ call, the simulator leaks the flags $(\mathsf{flag}_i = 0)_{i\in\mathsf{H}}$ to the adversary. For each $j \in \{1,\ldots,m\}$ and $v \in \{a,b,c\}$ the simulator sets $\bar{G}^{v,j}(x,y) := \sum_{k=0}^{2d} \alpha_j^k \cdot \bar{F}^{v,k}(x,y)$.

For a set Q, an honest party P_i in Q with corresponding challenge α_j for $j \in \{1,\ldots,m\}$, the simulator broadcast α_j on behalf of P_i. In addition, for $v \in \{a,b,c\}$, the leakage of $\mathcal{F}_{\mathsf{glinear}}^{Q,i,v}$ is simulated in the following way. The adversary receives (1) the j-th rows $\bar{\mathbf{C}}_j^{v,0},\ldots,\bar{\mathbf{C}}_j^{v,2d}$, (2) the flags $(\delta^\ell = 0)_{\ell\in\mathsf{H}}$, (3) the coefficient vector $\mathbf{a} := (\alpha_j^0,\ldots,\alpha_j^{2d})$, (4) the openings $\bar{o}_{\ell i}^{v,k}$ and values $\mathsf{open}(\bar{C}_{\ell i}^{v,k},\bar{o}_{\ell i}^{v,k})$ for every $k \in \{0,\ldots,2d\}$ and $\ell \in \mathsf{C}$, (5) the flag $\delta^G := 0$, (6) the values $(\bar{G}^{v,j}(\ell,i))_{\ell\in\{1,\ldots,n\}}$. The simulator also gives $\mathcal{A}$ the vector $(\mathbf{a},\bar{G}^{v,j}(0,i))$ as the output of $\mathcal{F}_{\mathsf{glinear}}^{Q,i,v}$.

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

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

Before proving that the ideal-world's view is close to the real-world's view, we analyse the distribution of the polynomials $\{G^{v,j}(x,y)\}_{j\in\{1,\ldots,m\}}$ that the simulator generates.

Lemma H.2. *Let $\mathbb{F}$ be a field, let $m \leq d$ be some positive integers with $2d < |\mathbb{F}|$, and let $\alpha_1,\ldots,\alpha_m \in \mathbb{F}$ 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 $\{G^j(x,y)\}_{j\in\{1,\ldots,m\}}$, 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 $j \in \{1,\ldots,m\}$ and $i \in \mathsf{C}$.

Proof. We show that the claim holds for every fixing of $F^{m+1}(x,y),\ldots,F^d(x,y)$. Let $H^1(x,y),\ldots,H^m(x,y)$ be any symmetric bivariate polynomials of degree at most t in each variable. It is not hard to see that if $H^j(x,i) \neq \sum_{k=0}^{2d}\alpha_j^k f_i^k(x)$ or $H^j(0,0) \neq P(\alpha_j)$, for some $j \in \{1,\ldots,m\}$ and $i \in \mathsf{C}$, then the probability that $(G^1,\ldots,G^m)$ is equal to $(H^1,\ldots,H^m)$ is 0. Therefore, we assume that $H^1(x,y),\ldots,H^m(x,y)$ are in the support, i.e., $H^j(0,0) = P(\alpha_j)$ and $H^j(x,i) = \sum_{k=0}^{2d}\alpha_j^k f_i^k(x)$ for all $j \in \{1,\ldots,m\}$ and $i \in \mathsf{C}$. Observe that $(G^1,\ldots,G^m) = (H^1,\ldots,H^m)$ if and only if

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

which occurs if and only if

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

where V is the $m \times m$ invertible matrix whose i-th row is $(\alpha_i^1,\ldots,\alpha_i^m)$. Observe that the RHS is fixed. For every $i \in \mathsf{C}$, when assigning $y = i$, the RHS is equal to the LHS. Indeed, the RHS is equal to

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

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

$$V^{-1}\begin{bmatrix} H^1(0,0) - F^0(0,0) - \alpha_1^{m+1}F^{m+1}(0,0) - \ldots - \alpha_1^{2d}F^{2d}(0,0) \\ \vdots \\ H^m(0,0) - F^0(0,0) - \alpha_m^{m+1}F^{m+1}(0,0) - \ldots - \alpha_m^{2d}F^{2d}(0,0) \end{bmatrix} = V^{-1}\begin{bmatrix} \sum_{k=1}^m \alpha_1^k P^k \\ \vdots \\ \sum_{k=1}^m \alpha_m^k P^k \end{bmatrix}$$
$$= \begin{bmatrix} P^1 \\ \vdots \\ P^m \end{bmatrix}.$$

We conclude that the RHS is in the support of the random variables in the LHS. By Fact A.4 the random variables $F^1(x,y), \ldots, F^d(x,y)$ have the same support size, which we denote by S, and we conclude that $(G^1, \ldots, G^m) = (H^1, \ldots, H^m)$ with probability $1/S^d$. Since this is true for every $(H^1, \ldots, H^m)$ in the support, the claim follows. $\square$

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

$\mathcal{Z}$'s view. The adversary's view consists of (1) the crs string, (2) the inputs to the honest dealer, (3) the outputs $(\mathbf{C}^{v,k}, \mathbf{O}_i^{v,k})_{v\in\{a,b,c\},k\in\{0,\ldots,2d\},i\in\mathsf{C}}$ from the sharing phase of $\mathcal{F}_{\mathsf{vss}}$ (or the default sharing in the case of $v \in \{a,b\}$ and $k > d$), (4) the challenges generated in any set Q that contains at least one corrupt party, (5) the output of the honest parties in the sharing phase, (6) the broadcasts of the players in the verification phase, (7) the leakage and output of the verification phase of the $\mathcal{F}_{\mathsf{vss}}$ calls, and (8) the leakage and output of the $\mathcal{F}_{\mathsf{glinear}}$ calls. In order to prove that the real-world view is close to the ideal-world view, we define the following hybrid-worlds, where we assume that the honest parties know the set H.

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

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

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

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

– For every set Q, honest P_i in Q and $v \in \{a,b,c\}$ the functionality $\mathcal{F}^{Q,i,v}_{\mathsf{glinear}}$ has the following output: (1) a vector $\mathbf{a} = (\alpha^0, \ldots, \alpha^{2d})$, where α is the challenge that P_i received from P_Q, and (2) the sum

$$\sum_{k=0}^{2d} \alpha^i \cdot \bar{F}^{v,k}(0,i).$$

In addition, it has the following leakage: (1) the commitments $(\mathbf{C}^{v,k}_j)_{k\in\{0,\ldots,2d\}}$, (2) the flags $(\delta^k = 0)_{k\in\mathsf{H}}$ and $\delta^G = 0$ of the honest parties, (3) the vector of coefficients $\mathbf{a} = (\alpha^0, \ldots, \alpha^{2d})$, (4) the values and openings $(\bar{F}^{v,k}(j,i), \bar{o}^{v,k}_{ij})_{k\in\{0,\ldots,2d\},j\in\mathsf{C}}$, and (5) the values $\sum_{k=0}^{2d} \alpha^i \cdot \bar{F}^{v,k}(j,i)$ for $j \in \{1, \ldots, n\}$.

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

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

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

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

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

Finally, in both hybrids the rest of the view can be obtained from the above random variables by the same efficient process, that executes the protocol with $\mathcal{Z}$ and $\mathcal{A}$ in the following way: (1) execute $\mathcal{Z}$ to obtain the dealer's inputs $(\mathbf{C}^{v,0}, \mathbf{O}^{v,0})_{v\in\{a,b,c\}}$, (2) simulate the $\mathcal{F}_{\mathsf{vss}}$ calls using the values $(\mathbf{C}^{v,k}, \mathbf{O}^{v,k}_i)_{v\in\{a,b,c\},k\in\{0,\ldots,2d\},i\in\mathsf{C}}$, (3) for every set Q with an honest P_Q, sample a random non-zero challenge α_Q, (4) execute $\mathcal{Z}$ and $\mathcal{A}$ to obtain the challenges generated by corrupt P_Q's, (5) set the output of an honest P_i in the sharing phase to be $(\mathbf{O}^a, \mathbf{C}^a_i), (\mathbf{O}^b, \mathbf{C}^b_i)$ and $(\mathbf{O}^c, \mathbf{C}^c_i)$, (6) broadcast on behalf of an honest P_i the bit $\mathsf{happy}_i = 1$ and the challenges α_Q that P_i received, for every set Q that contains P_i, (7) simulate a call to $\mathcal{F}^{Q,i,v}_{\mathsf{glinear}}$ for a set Q, an honest P_i in Q and $v \in \{a,b,c\}$ by leaking (a) the i-th rows $\mathbf{C}^{v,0}_i, \ldots, \mathbf{C}^{v,2d}_i$, (b) the flags $(\delta^\ell = 0)_{\ell\in\mathsf{H}}$, (c) the coefficient vector $\mathbf{a} = (\alpha^0, \ldots, \alpha^{2n})$, where α is the challenge that P_i received from P_Q, (d) the openings $o^{v,k}_{i\ell}$ and values $\mathsf{open}(C^{v,k}_{i\ell}, o^{v,k}_{i\ell})$ for every $k \in \{0, \ldots, 2d\}$ and $\ell \in \mathsf{C}$, (e) the flag $\delta^G := 0$, (f) partial sums

$\sum_{k=0}^{2d} \alpha^k \cdot F^{v,k}(\ell, i)$ for all $\ell \in \{1, \ldots, n\}$ and (g) the output $(\mathbf{a}, \sum_{k=0}^{2d} \alpha^k \cdot F^{v,k}(0, i))$, (8) simulate $\mathcal{F}_{\mathsf{glinear}}^{Q,i,v}$ for a set Q, corrupt P_i in Q and $v \in \{a, b, c\}$ by leaking (a) the i-th rows $\mathbf{C}_i^{v,0}, \ldots, \mathbf{C}_i^{v,2d}$, (b) the flags $(\delta^\ell = 0)_{\ell \in \mathsf{H}}$, and (c) the vector $(F^{v,0}(\ell, i), \ldots, F^{v,2d}(\ell, i))$ for every $\ell \in \mathsf{H}$, (9) at this stage, obtain from $\mathcal{Z}$ and $\mathcal{A}$ the inputs of the corrupt parties to the $\mathcal{F}_{\mathsf{vss}}$ calls and remaining $\mathcal{F}_{\mathsf{glinear}}$ calls (where the guide is corrupt), and compute the outputs according to the functionality (note that in each $\mathcal{F}_{\mathsf{vss}}$ call only corrupt parties are in W, and that for $\mathcal{F}_{\mathsf{glinear}}$ with a corrupt guide, the leakage together with the corrupt guide's inputs determine the output of the functionality). This completes the case of Hybrid 1 vs. Hybrid 2.

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

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

have the same distribution in both worlds, where $\bar{f}_i^{v,k}(i)$ is the polynomial corresponding to $(\mathbf{C}^{v,k}, \mathbf{O}_i^{v,k})$. We claim that in both worlds, the rest of the view can be obtained from the above random variables by the following efficient process: (1) obtain the dealer's inputs $(\mathbf{C}^{v,0}, \mathbf{O}^{v,0})_{v \in \{a,b,c\}}$ from $\mathcal{Z}$, (2) set the output of an honest P_i in the sharing phase to be $(\mathbf{O}^a, \mathbf{C}_i^a), (\mathbf{O}^b, \mathbf{C}_i^b)$ and $(\mathbf{O}^c, \mathbf{C}_i^c)$, (3) picks random polynomials $\bar{A}(x), \bar{B}(x)$ and $\bar{C}(x)$ of degree d, d and $2d$, respectively, such that $\bar{C}(x) = \bar{A}(x) \cdot \bar{B}(x)$, (4) sample polynomials $(\bar{F}^{v,k}(x, y))_{v \in \{a,b,c\}, k \in \{0,\ldots,2d\}}$ just like the sharing phase of the simulator, using the fixed values $(\bar{f}_i^{v,k}(x))_{v \in \{a,b,c\}, k \in \{1,\ldots,2n\}, i \in \mathsf{C}}$ and $\bar{A}(x), \bar{B}(x)$ and $\bar{C}(x)$, (5) simulate the sharing phase and the verification phase just like the simulator, using the polynomials $(\bar{F}^{v,k}(x, y))_{v \in \{a,b,c\}, k \in \{0,\ldots,2n\}}$, where the sharing phase output of an honest P_i is $(\mathbf{C}^{a,0}, \mathbf{O}_i^{a,0}), (\mathbf{C}^{b,0}, \mathbf{O}_i^{b,0})$ and $(\mathbf{C}^{a,0}, \mathbf{O}_i^{a,0})$.

Clearly, when the random variables in Equation 2 are taken from the ideal-world, the output of the above process is distributed exactly like the view of $\mathcal{Z}$ in the ideal-world. To see that this is also true for Hybrid 2, note that by Fact A.1 the random variables

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

generated by the process have the same distribution as the corresponding random variables generated in Hybrid 2 (where $\alpha_1, \ldots, \alpha_m$ are challenges held by the honest parties). Conditioned on those values, and by Lemma H.2, the random variables

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

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

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

We claim that both in the real-world and the ideal-world, the outputs of the honest parties can be extracted from a good View by the following efficient deterministic process: in the verification phase, all honest parties output "verification succeeded".

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

First observe that in the real-world the honest parties never output "D is corrupt" according to a good View. This follows since (1) no call to $\mathcal{F}_{\mathsf{vss}}$ ends with "D is corrupt", and (2) since View is good, for each good set Q,[27] P_i in Q, and $v \in \{a, b, c\}$ for which the output of $\mathcal{F}^{Q,i,v}_{\mathsf{glinear}}$ is $(\boldsymbol{\alpha}, u_{Q,i,v})$, the value $u_{Q,i,v}$ is $\sum_{k=0}^{2d} \alpha^k \cdot F^{v,k}(0, i)$, where α is the challenge corresponding to Q. This is clearly true for an honest P_i and for a corrupt P_i in W. In addition, for a corrupt P_i not in W, since View is good, the output has to be consistent with every honest party P_j's shares $\sum_{k=0}^{2d} \alpha^k \cdot F^{v,k}(j, i)$. We conclude that for every good Q with challenge α the players recover the value $\sum_{k=0}^{2d} \alpha^k \cdot F^{v,k}(0, 0)$ which is $A(\alpha)$ when $v = a$, $B(\alpha)$ when $v = b$ and $C(\alpha)$ when $v = c$. Finally, since D is honest it holds that $C(\alpha) = A(\alpha) \cdot B(\alpha)$, so D is never disqualified. This concludes the analysis of an honest dealer.

H.1.2 Corrupt Dealer

Sharing phase. The simulator takes the role of the honest parties, that have no inputs, in order to simulate an execution of the protocol. In the sharing phase, this includes only generating a challenge on behalf of any honest distinguihsed player of a set Q. Then, the simulator receives from $\mathcal{A}$ the inputs to the various $\mathcal{F}_{\mathsf{vss}}$ calls. Denote those inputs by $((\mathbf{C}^{v,k}, \mathbf{O}^{v,k}))_{v\in\{a,b,c\},k\in\{0,\dots,2d\}}$. The simulator gives the simulated honest party P_i the shares $((\mathbf{C}^{v,k}, \mathbf{O}^{v,k}_i))_{v\in\{a,b,c\},k\in\{0,\dots,2d\}}$. The simulator inputs $(\mathbf{C}^{a,0}, \mathbf{O}^{a,0})$,$(\mathbf{C}^{b,0}, \mathbf{O}^{b,0})$, and $(\mathbf{C}^{c,0}, \mathbf{O}^{c,0})$ to $\mathcal{F}_{\mathsf{tss}}$.

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

Fix any polynomial time environment $\mathcal{Z}$ and input ζ to the environment, and assume without loss of generality that $\mathcal{Z}$ is deterministic. It is not hard to see that the sharing phase view, the honest parties' outputs in the sharing phase, and the verification phase view have the same distribution in both worlds. Therefore, it remains to analyse the outputs of the verification phase. We separate between the case where the commitment scheme is perfectly-binding and there is no CRS, and the case where the commitment scheme is computationally-binding and there is a CRS.

[27]Recall that a set Q is *good* if (1) there exists $\alpha \in \mathbb{F} \setminus \{0\}$ such that for all parties P_i in Q who broadcasted $\mathsf{happy}_i = 1$, and for all $v \in \{a, b, c\}$, the output of $\mathsf{out}^{Q,i,v}$ is $(\boldsymbol{\alpha}, u_{Q,i,v})$, where $\boldsymbol{\alpha} = (\alpha^0, \dots, \alpha^{2d})$, and (2) every party P_i in Q who broadcasted $\mathsf{happy}_i = 0$ is in W.

Perfectly-binding commitment scheme. Assume that the commitment scheme is perfectly binding. First, observe that whenever D is discarded then the output in both worlds is "D is corrupt". Consider the commitments $\mathbf{C} := (\mathbf{C}^{v,k})_{v\in\{a,b,c\},k\in\{0,\ldots,2d\}}$ that are broadcasted in the first round by the dealer. Since the commitment scheme is perfectly binding, the commitments $\mathbf{C}$ fully determine the sharing polynomial $F^{v,k}(x,y)$ in $\mathcal{F}_{\mathsf{vss}}^{v,k}$ for every $v \in \{a,b,c\}$ and $k \in \{0,\ldots,2d\}$ (or otherwise D is discarded in some $\mathcal{F}_{\mathsf{vss}}$ call, and we are done). In particular, $\mathbf{C}$ determine the polynomials $A(x), B(x)$ and $C(x)$, whose k-th coefficient is $F^{a,k}(0,0)$, $F^{b,k}(0,0)$ and $F^{c,k}(0,0)$, respectively. Fix those commitments.

If $F^{c,0}(0,0) = F^{a,0}(0,0) \cdot F^{b,0}(0,0)$ then the outputs in both worlds is the same, so we assume that $F^{c,0}(0,0) \neq F^{a,0}(0,0) \cdot F^{b,0}(0,0)$. In this case the output of the honest players in the ideal-world is "D is corrupt", and we continue by showing that with probability $1-(2d/(|\mathbb{F}|-1))$ this is also the output of the honest parties in the real-world.

Let Q^* be a set of size $t+1$ that contains only honest parties. Note that the challenge α_{Q^*} is uniformly distributed even conditioned on $\mathbf{C}$ (since the rushing adversary does not know α_{Q^*} in the first round). Since $F^{c,0}(0,0) \neq F^{a,0}(0,0)\cdot F^{b,0}(0,0)$ then $C(x) \neq A(x)\cdot B(x)$, so with probability at least $1-(2d/(|\mathbb{F}|-1))$ it holds that $C(\alpha_{Q^*}) \neq A(\alpha_{Q^*}) \cdot B(\alpha_{Q^*})$. Since Q^* contains only honest parties it is always a good set, and with probability at least $1-(2d/(|\mathbb{F}|-1))$ the verification of Q^* fails and D is discarded. This concludes the analysis when the underlying commitment scheme is perfectly-binding.

Computationally-binding commitment scheme. We say that a view View is "good" if either (1) the output of the honest parties is "D is corrupt", or (2) the output is not "D is corrupt", and it holds that $F^{c,0}(0,0) = F^{a,0}(0,0) \cdot F^{b,0}(0,0)$, where $F^{v,0}$ is the sharing polynomial of the weak double t-sharing produced by $\mathcal{F}_{\mathsf{vss}}^{v,0}$, for $v \in \{a,b,c\}$ (observe that whenever the output is not "D is corrupt" then all $\mathcal{F}_{\mathsf{vss}}$ calls define weak double t-sharing). We note that if View is good then the output of the honest parties in the real world is fixed and equal to the output of the honest parties in the ideal world. Therefore, it is enough to prove that a view View is good with probability at least $1-\delta$ for $\delta = 2(\epsilon^{1/3} + (2d)/(|\mathbb{F}|-1)) = \mathsf{negl}(\kappa)$.

A view View is not good if and only if the output of the honest parties is not "D is corrupt", and

$$F^{c,0}(0,0) \neq F^{a,0}(0,0) \cdot F^{b,0}(0,0). \tag{3}$$

Let Q^* be a set of $t+1$ honest players. If View is not good then D is not discarded. Since Q^* contains only honest players it is a good set, so it must hold that the verification of Q^* succeeds, i.e., that

$$\Big(\sum_{k=0}^{2d} \alpha_{Q^*}^k F^{c,k}(0,0)\Big) = \Big(\sum_{k=0}^{2d} \alpha_{Q^*}^k F^{a,k}(0,0)\Big) \cdot \Big(\sum_{k=0}^{2d} \alpha_{Q^*}^k F^{b,k}(0,0)\Big), \tag{4}$$

where $F^{v,k}(x,y)$ is the sharing polynomial of the weak double t-sharing produced by $\mathcal{F}_{\mathsf{vss}}^{v,k}$, and α_{Q^*} is the challenge picked by the set Q^*.

Let E be the event that the dealer is not discarded, and that Inequality 3 holds and Equality 4 holds. We continue by showing that if event E occurs with probability at least δ then we can construct an adversary that violates the binding property of the commitment scheme.

Assume towards contradiction that E occurs with probability at least δ. This means that E occurs with probability at least δ for some fixed challenges α_Q^* for every set $Q \neq Q^*$ with an honest P_Q. Consider the adversary $\mathcal{A}'$ against the commitment scheme that, given the common reference string crs, picks two challenges $\beta_1, \beta_2 \leftarrow \mathbb{F} \setminus \{0\}$, and computes two executions of protocol tss with $\mathcal{Z}$, $\mathcal{A}$ and (the same) common reference string crs, with the fixed challenges α^*_Q for every set $Q \neq Q^*$ with an honest P_Q, and with challenge β_1 as the challenge of Q^* in the first execution and with challenge β_2 as the challenge of Q^* in the second execution. (Observe that the only randomness in the execution is $(\mathsf{crs}, \beta_1, \beta_2)$.) If $\mathcal{A}'$ finds in the two executions of tss a commitment C and two openings o, and o', such that $\mathsf{open}(C, o) \neq \perp$ and $\mathsf{open}(C, o') \neq \perp$ and $\mathsf{open}(C, o) \neq \mathsf{open}(C, o)$ then $\mathcal{A}'$ outputs (C, o, o'). Otherwise $\mathcal{A}'$ outputs $\perp$. Note that whenever $\mathcal{A}'$ outputs (C, o, o') then $\mathcal{A}'$ successfully violates the binding property of the commitment scheme.

We continue by showing that $\mathcal{A}'$ violates the binding property with probability greater than ϵ. This will contradict the security guarantees of the commitments scheme. First, observe that there are at least $\delta/2$-fraction of common reference strings crs such that event E occurs with probability at least $\delta/2$ conditioned on crs. Indeed, if this does not hold then

$$\delta \leq \Pr[E] = \frac{1}{2^\ell} \sum_{c \in \{0,1\}^\ell} \Pr[E \mid \mathsf{crs} = c] < \frac{1}{2^\ell} \cdot (\delta/2) 2^\ell \cdot 1 + \frac{1}{2^\ell} \cdot 2^\ell \cdot (\delta/2) = \delta,$$

in contradiction, where we denoted the length of the common reference string by ℓ.

Fix any common reference string c such that $\Pr[E \mid \mathsf{crs} = c] \geq \delta/2$, and observe that the only randomness remains in the execution of $\mathcal{A}'$ is the challenges β_1 and β_2. Fix any β_1 such that event E occurs in the first execution, and note that there are at least $(|\mathbb{F}| - 1)\delta/2$ such values. Let $(F^{v,k}(x,y))_{v \in \{a,b,c\}, k \in \{0,\ldots,2d\}}$ be the corresponding sharing polynomials of the first execution, and let $A(x), B(x)$ and $C(x)$ be the polynomials whose k-th coefficients are $F^{a,k}(0,0)$, $F^{b,k}(0,0)$ and $F^{c,k}(0,0)$, respectively. Since E occurs then $F^{c,0}(0,0) \neq F^{a,0}(0,0) \cdot F^{b,0}(0,0)$ and $C(\beta_1) = A(\beta_1) \cdot B(\beta_1)$.

Since β_2 is independent of β_1, the probability that event E occurs in the second execution is $\delta/2$, even conditioned on β_1. In addition, since $C(x) \neq A(x) \cdot B(x)$ then the probability that $C(\beta_2) = A(\beta_2) \cdot B(\beta_2)$ is at most $2d/(|\mathbb{F}| - 1)$. Therefore, the probability that both E occurs in the second execution and $C(\beta_2) \neq A(\beta_2) \cdot B(\beta_2)$ is at least $(\delta/2) - (2d/(|\mathbb{F}| - 1)) > 0$. Condition on such β_2. We claim that in an execution with such $(\mathsf{crs}, \beta_1, \beta_2)$ the adversary $\mathcal{A}'$ successfully violates the binding property.

Indeed, let $(\tilde{F}^{v,k}(x,y))_{v \in \{a,b,c\}, k \in \{0,\ldots,2d\}}$ and $\tilde{A}(x), \tilde{B}(x)$ and $\tilde{C}(x)$ be the polynomials corresponding to the second execution. Since E occurs then $\tilde{C}(\beta_2) = \tilde{A}(\beta_2) \cdot \tilde{B}(\beta_2)$. Since $C(\beta_2) \neq A(\beta_2) \cdot B(\beta_2)$ then necessarily $(A(x), B(x), C(x)) \neq (\tilde{A}(x), \tilde{B}(x), \tilde{C}(x))$, which means that for some $v \in \{a, b\}$ and $k \in \{0, \ldots, d\}$, or $v = c$ and $k \in \{0, \ldots, 2d\}$, $F^{v,k}(x,y) \neq \tilde{F}^{v,k}(x,y)$. Let $\mathbf{O}_\mathsf{H}$ be the openings of the honest parties in the first execution, and let $\tilde{\mathbf{O}}_\mathsf{H}$ be the openings of the honest parties in the second execution. Since, by Fact A.2, the honest parties shares fully determine the polynomials $F^{v,k}(x,y)$ and $\tilde{F}^{v,k}(x,y)$, there must exist $\ell \in \mathsf{H}$ and $r \in \{0, \ldots, n\}$ such that $\mathsf{open}(C^{v,k}_{\ell,r}, o^{v,k}_{\ell,r}) \neq \perp$, $\mathsf{open}(C^{v,k}_{\ell,r}, \tilde{o}^{v,k}_{\ell,r}) \neq \perp$, and $\mathsf{open}(C^{v,k}_{\ell,r}, o^{v,k}_{\ell,r}) \neq \mathsf{open}(C^{v,k}_{\ell,r}, \tilde{o}^{v,k}_{\ell,r})$.

We conclude that $\mathcal{A}'$ breaks the binding of the NICOM scheme with probability $(\delta/2)(\delta/2)((\delta/2) - (2d)/(|\mathbb{F}| - 1))) > \epsilon$, in contradiction to the security guarantees of the NICOM scheme. This concludes the analysis of a corrupt dealer, and the proof of security of the protocol. $\square$

I Public Single-Input Functionality for a Small Number of Parties

In this section, we provide a formal description of protocol psiflog. For a reminder on weak and strong sharing, and of tentative share, see Section D. We also use Notation 4 to specify inputs to the sub-protocol glinear. We begin with a short reminder about Beaver's trick.

Beaver's trick. Suppose that we have a random triple (a, b, c) such that $c = ab$ which is already shared among the parties. Say that the parties also hold sharings of two values x and y, and they want to compute the multiplication xy. Then the parties do as follows: (1) reconstruct the values $\langle\langle u\rangle\rangle := \langle\langle x\rangle\rangle - \langle\langle a\rangle\rangle$ and $\langle\langle v\rangle\rangle := \langle\langle y\rangle\rangle - \langle\langle b\rangle\rangle$, and (2) reconstruct the value $\langle\langle z\rangle\rangle := v \cdot \langle\langle a\rangle\rangle + u \cdot \langle\langle b\rangle\rangle + uv + \langle\langle c\rangle\rangle$. Note that (1) reveals no information about x and y because a and b are used as a one-time pad. In addition, one can show that $z = xy$, and that z is shared via a fresh sharing that reveals no information about x and y. We emphasize that, after the reconstruction of u and v, in order to compute z it is enough to compute a linear function of a, b and c with coefficients that depend on u and v which are known to the parties. For a formal statement, see Fact A.5.

Protocol overview. In the offline round, the dealer D picks random values $a^1, \ldots, a^\ell$ (where ℓ the number of its inputs), and shares them via vss. In addition, for every $p, q \in \{1, \ldots, \ell\}$, the dealer sets $a^{pq} := a^p \cdot a^q$, and shares the triple (a^p, a^q, a^{pq}) via tss. Note that, all the tss instances use the same sharings for a^p and a^q as in their vss executions. In the online round, the dealer receives the inputs $z^1, \ldots, z^\ell$. For every $p \in \{1, \ldots, \ell\}$, the dealer broadcasts a "correction" $\Delta^p := z^p - a^p$. In addition, for every $i \in \{1, \ldots, m\}$ (m is the number of terms in y, see Figure 3) and $j \in \{1, \ldots, n\}$, *the dealer*, that holds all shares and knows the values of all Δ's, guides an execution of glinear computing the j-th row of

$$\alpha_0^i \cdot [\![1]\!] + \sum_{p=1}^{\ell} \alpha_p^i([\![a^p]\!] + \Delta^p \cdot [\![1]\!]) + \sum_{p,q\leq\ell} \alpha_{pq}^i(\Delta^q[\![a^p]\!] + \Delta^p[\![a^q]\!] + \Delta^p \cdot \Delta^q \cdot [\![1]\!] + [\![a^{pq}]\!]) + [\![\eta_i]\!]. \quad (5)$$

We assume that $[\![1]\!]$ is some default double strong t-sharing of the value 1, via the constant sharing polynomial 1, which is locally computable by the parties. In the first sum, the p-th term computed is $a^p + \Delta^p = z^p$, and in the second sum, by Beaver's trick it follows that the (p, q)-th term computed is $z^p \cdot z^q$. Therefore, correctness and privacy for an honest dealer follow from Beaver's trick. For a corrupt dealer which is not discarded, one can prove that the output is consistent with the inputs $z^p := a^p + \Delta^p$. In order to simplify the simulation, we randomized each y^i (summand of y) using a random sharing of 0, which is the last term in Equation 5, denoted $[\![\eta_i]\!]$. The zero-sharing is generated by letting D generate three sharings of 0 via tss, and then open the first two sharings in order to prove that the last sharing also corresponds to 0.

Protocol psiflog appears in Figure 10.

Protocol psiflog

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

Offline phase. The parties do as follows.

- *(Sharing random elements)* For every $p \in \{1, \ldots, \ell\}$, D picks a random field element $a^p \leftarrow \mathbb{F}$ and shares it via an instance of vss, denoted as vss^p. Denote the corresponding strong degree-t sharing by $(\mathbf{C}^p, \mathbf{O}^p)$ and the sharing polynomial by $F^p(x, y)$.
- *(Completing a triple)* For every $p, q \in \{1, \ldots, \ell\}$ the dealer sets $a^{pq} := a^p \cdot a^q$ and shares a^{pq} via an instance of vss, denoted vss^{pq}. Denote the corresponding strong degree-t sharing by $(\mathbf{C}^{pq}, \mathbf{O}^{pq})$ and the sharing polynomial by $F^{pq}(x, y)$.
- *(Triple verification)* For every $p, q \in \{1, \ldots, \ell\}$, the parties execute an instance of tss, denoted tss^{pq}, where D inputs $(\mathbf{C}^p, \mathbf{O}^p)$, $(\mathbf{C}^q, \mathbf{O}^q)$ and $(\mathbf{C}^{pq}, \mathbf{O}^{pq})$.
- *(Zero sharing)* For each $i \in \{1, \ldots, m\}$, the parties execute an instance of tss, denoted tss^i, where D inputs three random strong sharings of zero, denoted $(\mathbf{C}^{\gamma_i}, \mathbf{O}^{\gamma_i})$, $(\mathbf{C}^{\rho_i}, \mathbf{O}^{\rho}_i)$ and $(\mathbf{C}^{\eta_i}, \mathbf{O}^{\eta_i})$. In addition, D broadcasts $(\mathbf{C}^{\gamma_i}, \mathbf{O}^{\gamma_i})$ and $(\mathbf{C}^{\rho_i}, \mathbf{O}^{\rho_i})$.
- (glinear.off *calls)* For $i \in \{1, \ldots, m\}$, $j \in \{1, \ldots, n\}$, the parties execute glinear.off^{ij} with D as guide.
- *(Local computation)* If one of the following happens then D is discarded:
  - If for some $i \in \{1, \ldots, m\}$, either $(\mathbf{C}^{\gamma_i}, \mathbf{O}^{\gamma_i})$ or $(\mathbf{C}^{\rho_i}, \mathbf{O}^{\rho_i})$ broadcasted by D as part of the 0 sharing, are not a strong sharing of zero, or $\mathbf{C}^{\gamma_i}$ and $\mathbf{C}^{\rho_i}$ are not the commitments used in tss^i.
  - If D did not use $\mathbf{C}^p$, $\mathbf{C}^q$ and $\mathbf{C}^{pq}$ as the commitments in the execution of tss^{pq}.

  If D is discarded then all parties output $y(0, \ldots, 0)$ in the online phase (there is no need in any other communication).

  Otherwise D is not discarded. Denote the output of party P_j in the execution of vss^p by $(\mathbf{C}^p, \mathbf{O}^p_j)$, and in the execution of vss^{pq} by $(\mathbf{C}^{pq}, \mathbf{O}^{pq}_j)$. If (1) P_j received an *invalid pair* (see Section 4.1 and Definition D.1) in some vss or tss call, or (2) for some $p, q \in \{1, \ldots, \ell\}$, the output of party P_j in tss^{pq} is not $(\mathbf{C}^p, \mathbf{O}^p_j)$, $(\mathbf{C}^q, \mathbf{O}^q_j)$, $(\mathbf{C}^{pq}, \mathbf{O}^{pq}_j)$, or (3) for some $i \in \{1, \ldots, m\}$, the output of party P_j in tss^i is not $(\mathbf{C}^{\gamma_i}, \mathbf{O}^{\gamma_i}_j)$, $(\mathbf{C}^{\rho_i}, \mathbf{O}^{\rho_i}_j)$, $(\mathbf{C}^{\eta_i}, \mathbf{O}^{\eta_i}_j)$, where $(\mathbf{C}^{\gamma_i}, \mathbf{O}^{\gamma_i})$ and $(\mathbf{C}^{\rho_i}, \mathbf{O}^{\rho_i})$ were broadcasted by D as part of the 0 sharing, then P_j sets $\mathsf{flag}_j = 1$. Otherwise P_j sets $\mathsf{flag}_j = 0$.

Online phase inputs. D has input $z^1, \ldots, z^\ell$.

Online phase. The parties do the following.

- *(Correction broadcast)* For every $p \in \{1, \ldots, \ell\}$, D broadcasts $\Delta^p := z^p - a^p$.
- (vss *verification phase)* The parties execute the verification phase of vss^p and vss^{pq} for every $p, q \in \{1, \ldots, \ell\}$, with a flag equals to 0.
- (tss *verification phase)* For each $p, q \in \{1, \ldots, \ell\}$ the parties execute the verification phase of tss^{pq}. In addition, for each $i \in \{1, \ldots, m\}$, the parties execute the verification phase of tss^i.
- (glinear.on *calls)* For $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, n\}$, in glinear.on^{ij} the parties partially compute Equation 5, with D as the guide with input $\delta^G = 0$ and every P_k with flag $\delta^k = \mathsf{flag}_k$.[a]
- *(Local computation)*
  - *(VSS/TSS verification phase outputs)* The parties compute the output of all vss and tss executions. If the output of some execution is "D is corrupt" then they output $y(0, \ldots, 0)$ and terminate.
  - (glinear *outputs)* If the output of some glinear execution is "G is corrupt", or if it is of the form $(\boldsymbol{\alpha}, v)$ and the vector of coefficients $\boldsymbol{\alpha}$ is not consistent with the coefficients in Equation 5, the parties output $y(0, \ldots, 0)$.
  - *(Output computation)* Let $(\boldsymbol{\alpha}_{ij}, v_{ij})$ be the output of glinear^{ij} for $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, n\}$. Let $f_i(x)$ be the polynomial obtained by interpolating $(v_{ij})_{j \in \{1, \ldots, n\}}$. If $f_i(x)$ is not of degree-t then all parties output $y(0, \ldots, 0)$. Otherwise, the parties output $(f_1(0), \ldots, f_m(0))$.

[a]We assume that $[\![1]\!]$ is some default double strong t-sharing of the value 1, via the constant sharing polynomial 1.

Figure 10: Protocol psiflog

We continue with a proof of Theorem 5.2.

I.1 Proof of security

Proof of Theorem 5.2. In this section we prove that protocol psiflog UC-emulates $\mathcal{F}_{\mathsf{psif}}$ (with everlasting security if the underlying commitment-scheme is statistically-hiding). From the composition properties of UC-security, it is enough to prove security in the $(\mathcal{F}_{\mathsf{vss}}, \mathcal{F}_{\mathsf{glinear}}, \mathcal{F}_{\mathsf{tss}})$-hybrid model. Let $\mathcal{A}$ be the dummy adversary against psiflog. We define the simulator $\mathcal{S}$ as follows. The simulator $\mathcal{S}$ uses $\mathcal{A}$ in a black-box manner, and forwards all messages between $\mathcal{Z}$ and $\mathcal{A}$. The simulator first receives the set of corrupt C parties from $\mathcal{Z}$. We split into cases.

I.1.1 Honest Dealer

Offline phase. The simulator simulates the offline phase exactly like an honest dealer. In more details, the simulator does as follows.

- *(VSS simulation)* For every $p \in \{1, \dots, \ell\}$ the dealer picks a random field element a^p, and samples a strong double-t sharing of a^p, denoted $(\bar{\mathbf{C}}^p, \bar{\mathbf{O}}^p)$. The simulator gives to the adversary the shares $(\bar{\mathbf{C}}^p, \bar{\mathbf{O}}^p_j)$ for every corrupt P_j, as the output of $\mathcal{F}^p_{\mathsf{vss}}$. Denote the sharing polynomial by $\bar{F}^p(x, y)$.

  For every $p, q \in \{1, \dots, \ell\}$ the dealer sets $a^{pq} := a^p \cdot a^q$, and samples a strong double-t sharing of a^{pq}, denoted $(\bar{\mathbf{C}}^{pq}, \bar{\mathbf{O}}^{pq})$. The simulator gives to the adversary the shares $(\bar{\mathbf{C}}^{pq}, \bar{\mathbf{O}}^{pq}_j)$ for every corrupt P_j, as the output of $\mathcal{F}^{pq}_{\mathsf{vss}}$. Denote the sharing polynomial by $\bar{F}^{pq}(x, y)$

- *(TSS simulation)* For every $p, q \in \{1, \dots, \ell\}$ the simulator gives to the adversary the shares $(\bar{\mathbf{C}}^p, \bar{\mathbf{O}}^p_j)$, $(\bar{\mathbf{C}}^q, \bar{\mathbf{O}}^q_j)$ and $(\bar{\mathbf{C}}^{pq}, \bar{\mathbf{O}}^{pq}_j)$ for every corrupt P_j, as the output of $\mathcal{F}^{pq}_{\mathsf{tss}}$.

- *(Zero sharing)* For every $i \in \{1, \dots, m\}$ the dealer picks three random strong sharings of zero, denoted $(\mathbf{C}^{\gamma_i}, \mathbf{O}^{\gamma_i})$, $(\mathbf{C}^{\rho_i}, \mathbf{O}^{\rho_i})$, and $(\mathbf{C}^{\eta_i}, \mathbf{O}^{\eta_i})$. The simulator gives to the adversary the shares $(\mathbf{C}^{\gamma_i}, \mathbf{O}^{\gamma_i}_j)$, $(\mathbf{C}^{\rho_i}, \mathbf{O}^{\rho_i}_j)$, and $(\mathbf{C}^{\eta_i}, \mathbf{O}^{\eta_i}_j)$ for every corrupt P_j, as the output of $\mathcal{F}^i_{\mathsf{tss}}$. Denote the sharing corresponding to η_i by $\bar{F}^{\eta_i}(x, y)$. In addition, the simulator broadcasts $(\mathbf{C}^{\gamma_i}, \mathbf{O}^{\gamma_i}_j)$ and $(\mathbf{C}^{\rho_i}, \mathbf{O}^{\rho_i}_j)$.

Online phase. The simulator receives the output $(y^1, \dots, y^m)$ from $\mathcal{F}_{\mathsf{psif}}$. For each $p \in \{1, \dots, \ell\}$ the simulator broadcasts a random field element Δ^p. For every $\mathcal{F}_{\mathsf{vss}}$ call, the simulator leaks the to the adversary the flags of the honest parties, where all of them are set to 0. The simulator also returns "verification succeeded" for every $\mathcal{F}_{\mathsf{tss}}$ call.

For each $i \in \{1,\dots,m\}$, the simulator picks a random symmetric bivariate polynomial $\bar{F}^{y^i}(x,y)$, of degree at most t in each variable, conditioned on $\bar{F}^{y^i}(0,0) = y^i$, and

$$\begin{aligned}\bar{F}^{y^i}(x,j) = \alpha_0^i + &\sum_{p\in\{1,\dots,\ell\}} \alpha_p^i(\bar{F}^p(x,j) + \Delta^p)\\ &+ \sum_{p,q\in\{1,\dots,\ell\}} \alpha_{pq}^i(\Delta^q \bar{F}^p(x,j) + \Delta^p \bar{F}^q(x,j) + \Delta^p \cdot \Delta^q + \bar{F}^{pq}(x,j)) + \bar{F}^{\eta_i}(x,j),\end{aligned}$$

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

For every $i \in \{1,\dots,m\}$ and $j \in \{1,\dots,n\}$ the leakage of $\mathcal{F}^{ij}_{\mathsf{glinear}}$ is simulated as follows. The simulator sends $\mathcal{A}$, (1) the commitments $(\mathbf{C}^1_j,\dots,\mathbf{C}^\ell_j,\mathbf{C}^{1,1}_j,\dots,\mathbf{C}^{\ell,\ell}_j,\mathbf{C}^{\eta_i}_j)$, as well as the commitments corresponding to the default strong double t-sharing of 1, (2) the flags $(\delta^k = 0)_{k\in\mathsf{H}}$ and $\delta^G = 0$ of the honest parties, (3) the vector of coefficients $\mathbf{a}$ that corresponds to the computation in Equation 5, (4) the values and openings $(\bar{F}^p(k,j), \bar{o}^p_{kj})_{p\in\{1,\dots,\ell\},k\in\mathsf{C}}$, $(\bar{F}^{pq}(k,j), \bar{o}^{pq}_{kj})_{p,q\in\{1,\dots,\ell\},k\in\mathsf{C}}$, and $(\bar{F}^{\eta_i}(k,j), \bar{o}^{\eta_i}_{kj})_{k\in\mathsf{C}}$, (5) the values $\bar{F}^{y^i}(k,j)$ for $k \in \{1,\dots,n\}$. In addition, upon receiving the inputs of the corrupt parties to $\mathcal{F}^{ij}_{\mathsf{glinear}}$, the simulator also sends $(\mathbf{a}, \bar{F}^{y^i}(0,j))$ as the output of the functionality.

At this stage the simulator also receives the inputs of the adversary to $\mathcal{F}_{\mathsf{vss}}$, which include the flags of the corrupt parties. For $\mathcal{F}^p_{\mathsf{vss}}$ with flags $(\mathsf{flag}_i)_{i\in\mathsf{C}}$, the simulator sets W to be the set of all corrupt parties P_i with $\mathsf{flag}_i = 1$ and returns $(\mathsf{W}, (o^p_{ij})_{i\in\mathsf{W},j\in\{0,\dots,n\}})$. The simulator acts similarly with $\mathcal{F}^{pq}_{\mathsf{vss}}$. This concludes the simulation of an honest dealer.

Fix a polynomial-time environment $\mathcal{Z}$ with input ζ. Assume without loss of generality that $\mathcal{Z}$ is deterministic. We begin by showing that the view of $\mathcal{Z}$ in the real-world is close to the view of $\mathcal{Z}$ in the ideal world.

$\mathcal{Z}$'s view. The adversary's view consists of (1) the crs, (2) the output of the sharing phase of $\mathcal{F}_{\mathsf{vss}}$ and $\mathcal{F}_{\mathsf{tss}}$, (3) the broadcast of the dealer in the offline round, (4) the inputs to the honest dealer, (5) the leakage and output of the verification phase of $\mathcal{F}_{\mathsf{vss}}$ and $\mathcal{F}_{\mathsf{tss}}$ and of the online phase of $\mathcal{F}_{\mathsf{glinear}}$, and (6) the broadcast of the dealer in the online round. We consider the following two hybrid worlds, where we assume that the honest parties know the set H.

- In Hybrid 1, the honest parties act like in the real-world, with the following modifications. In the offline round,
  - an honest P_i does not check the validity of the rows received from D, but is always happy with D (so the flag is always 0),
  - for every $p,q \in \{1,\dots,\ell\}$ and $k \in \mathsf{H}$, the dealer also sends $F^p(x,k)$ and $F^{pq}(x,k)$ to P_k, where $F^p(x,y)$ and $F^{pq}(x,y)$ are the sharing polynomials of a^p and a^{pq}, and
  - for every $i \in \{1,\dots,m\}$ and $k \in \mathsf{H}$, the dealer also sends $F^{\eta_i}(x,k)$ to P_k, where $F^{\eta_i}(x,y)$ is the sharing polynomial of η_i.

  Also, in the online round, for every $i \in \{1,\dots,m\}$ and $j \in \{1,\dots,n\}$, the honest dealer inputs to $\mathsf{glinear}^{ij}$ the vector $\mathbf{a}$ according to Equation 5, the values $(F^p(k,j), F^{pq}(k,j))_{p,q\in\{1,\dots,\ell\},k\in\{0,\dots,n\}}$ and $(F^{\eta_i}(k,j))_{k\in\{0,\dots,n\}}$, the openings

$(o^p_{k,j}, o^{pq}_{k,j})_{p,q\in\{1,\dots,\ell\},k\in\{0,\dots,n\}}$ and $(o^{\eta_i}_{k,j})_{k\in\{0,\dots,n\}}$, and the flag $\delta^G = 0$; In addition, an honest P_k inputs to $\mathcal{F}^{ij}_{\mathsf{glinear}}$ the values $(F^p(k,j), F^{pq}(k,j))_{p,q\in\{1,\dots,\ell\}}$ and $F^{\eta_i}(k,j)$, as received from the honest dealer, and the flag $\delta^k = 0$.

- In Hybrid 2, honest parties act like in Hybrid 1, with the following modifications. In the offline round:
  - For every $p, q \in \{1,\dots,\ell\}$ the dealer samples random values a_p and sets $a_{pq} = a^p \cdot a^q$ and shares them via $\mathcal{F}_{\mathsf{vss}}$ and $\mathcal{F}_{\mathsf{tss}}$ calls. We denote the sharing polynomials by $F^p(x,y)$ and $F^{pq}(x,y)$.
  - For each $p, q \in \{1,\dots,\ell\}$ the dealer picks random field elements $\bar{a}^p, \bar{a}^q$ and $\bar{a}^{pq} := \bar{a}^p \cdot \bar{a}^q$. Then the dealer picks random polynomials $\bar{F}^p(x,y)$ and $\bar{F}^{pq}(x,y)$ conditioned on (a) $\bar{F}^p(x,i) = F^p(x,i)$ for $i \in \mathsf{C}$ and $F^p(0,0) = \bar{a}^p$, and (b) $\bar{F}^{pq}(x,i) = F^{pq}(x,i)$ for $i \in \mathsf{C}$ and $\bar{F}^{pq}(0,0) = \bar{a}^{pq}$.
  - For each $i \in \{1,\dots,m\}$ the dealer picks a random polynomial $\bar{F}^{\eta_i}(x,y)$ conditioned on $\bar{F}^{\eta_i}(x,k) = F^{\eta_i}(x,k)$ for every $k \in \mathsf{C}$, and $F^{\eta_i}(0,0) = 0$.
  - For any $k \in \mathsf{H}$ the dealer sends $\bar{F}^p(x,k)$, $\bar{F}^{pq}(x,k)$, and $\bar{F}^{\eta_i}(x,k)$ to P_k, for all $p, q \in \{1,\dots,\ell\}$ and $i \in \{1,\dots,m\}$.

  Also, in the online round:
  - The dealer sets $\Delta^p := z^p - \bar{F}^p(0,0)$.
  - For every $i \in \{1,\dots,m\}$ and $j \in \{1,\dots,n\}$ the functionality $\mathcal{F}^{ij}_{\mathsf{glinear}}$ has the following output: (1) a vector $\mathbf{a}$ consistent with Equation 5, and (2) the sum

$$\begin{aligned}\alpha^i_0 &+ \sum_{p\in\{1,\dots,\ell\}} \alpha^i_p(\bar{F}^p(0,j) + \Delta^p)\\ &+ \sum_{p,q\in\{1,\dots,\ell\}} \alpha^i_{pq}(\Delta^q \bar{F}^p(0,j) + \Delta^p \bar{F}^q(0,j) + \Delta^p \cdot \Delta^q + \bar{F}^{pq}(0,j)) + \bar{F}^{\eta_i}(0,j).\end{aligned}$$

    In addition, it has the following leakage: (1) the commitments $(\mathbf{C}^1_j, \dots, \mathbf{C}^\ell_j, \mathbf{C}^{1,1}_j, \dots, \mathbf{C}^{\ell,\ell}_j, \mathbf{C}^{\eta_i}_j)$, as well as the commitments corresponding to the default strong double t-sharing of 1, (2) the flags $(\delta^k = 0)_{k\in\mathsf{H}}$ and $\delta^G = 0$ of the honest parties, (3) the vector of coefficients that corresponds to the computation in Equation 5, (4) the values and openings $(\bar{F}^p(k,j), o^p_{kj})_{p\in\{1,\dots,\ell\},k\in\mathsf{C}}$, $(\bar{F}^{pq}(k,j), o^{pq}_{kj})_{p,q\in\{1,\dots,\ell\},k\in\mathsf{C}}$, and $\{\bar{F}^{\eta_i}(k,j), o^{\eta_i}_{kj}\}_{k\in\mathsf{C}}$, (5) the sum

$$\begin{aligned}\alpha^i_0 &+ \sum_{p\in\{1,\dots,\ell\}} \alpha^i_p(\bar{F}^p(k,j) + \Delta^p)\\ &+ \sum_{p,q\in\{1,\dots,\ell\}} \alpha^i_{pq}(\Delta^q \bar{F}^p(k,j) + \Delta^p \bar{F}^q(k,j) + \Delta^p \cdot \Delta^q + \bar{F}^{pq}(k,j)) + \bar{F}^{\eta_i}(k,j).\end{aligned}$$

    for $k \in \{1,\dots,n\}$.

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

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

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

and the Hybrid 2 random variables

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

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

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

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

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

Honest parties' outputs. We claim that both in the real-world and the ideal-world, the outputs of the honest parties in the output phase can be extracted from a good View by the following efficient deterministic process: Extract the inputs $z^1, \ldots, z^\ell$ from View and output $y(\mathbf{z})$. (Recall that the honest parties inputs are part of View.) Clearly, in the ideal-world the output of the honest parties is $y(\mathbf{z})$ with probability 1. We show that this also occurs in the real-world. Indeed, since D is honest, non of the $\mathcal{F}_{\mathsf{vss}}$ and $\mathcal{F}_{\mathsf{tss}}$ calls end with "D is corrupt". In addition, for every $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, n\}$ the output of $\mathcal{F}_{\mathsf{glinear}}^{ij}$ is $(\boldsymbol{\alpha}_{ij}, v_{ij})$, where $\boldsymbol{\alpha}_{ij}$ is consistent with Equation 5, and v_{ij} is equal to

$$\begin{aligned}\alpha_0^i + &\sum_{p\in\{1,\ldots,\ell\}} \alpha_p^i(F^p(0,j) + \Delta^p)\\ &+ \sum_{p,q\in\{1,\ldots,\ell\}} \alpha_{pq}^i(\Delta^q F^p(0,j) + \Delta^p F^q(0,j) + \Delta^p \cdot \Delta^q + F^{pq}(0,j)) + F^{\eta_i}(0,j).\end{aligned}$$

We conclude that the i-th output is

$$\begin{aligned}
y^i &= \alpha_0^i + \sum_{p\in\{1,\ldots,\ell\}} \alpha_p^i(F^p(0,0)+\Delta^p) \\
&\quad + \sum_{p,q\in\{1,\ldots,\ell\}} \alpha_{pq}^i(\Delta^q F^p(0,0) + \Delta^p F^q(0,0) + \Delta^p\cdot\Delta^q + F^{pq}(0,0)) + F^{\eta_i}(0,0) \\
&= \alpha_0^i + \sum_{p\in\{1,\ldots,\ell\}} \alpha_p^i(a^p+\Delta^p) + \sum_{p,q\in\{1,\ldots,\ell\}} \alpha_{pq}^i(\Delta^q a^p + \Delta^p a^q + \Delta^p\cdot\Delta^q + a^{pq}) + 0 \\
&= \alpha_0^i + \sum_{p\in\{1,\ldots,\ell\}} \alpha_p^i z^p + \sum_{p,q\in\{1,\ldots,\ell\}} \alpha_{pq}^i z^p\cdot z^q \\
&= y^i(\mathbf{z}).
\end{aligned}$$

Where we used the fact that D is honest so $\Delta^p = z^p - a^p$ and $a^{pq} = a^p\cdot a^q$ for all $p,q\in\{1,\ldots,\ell\}$ and $F^{\eta_i}(0,0)=0$, and the correctness of Beaver's trick (see Fact A.5). This concludes the analysis of an honest dealer.

I.1.2 Corrupt Dealer

Offline phase. The simulator takes the role of the honest parties, that have no inputs, and executes an instance of psiflog with $\mathcal{Z}$ and $\mathcal{A}$. In the offline phase this only includes receiving from $\mathcal{A}$ commitments and openings for the $\mathcal{F}_{\mathsf{vss}}$ and $\mathcal{F}_{\mathsf{tss}}$ calls, and giving the corresponding shares to the simulated honest parties.

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

At the end of the simulation, if D was disqualified then the simulator inputs $(0,\ldots,0)$ to $\mathcal{F}_{\mathsf{psif}}$ and terminates. Otherwise, D was not disqualified in any $\mathcal{F}_{\mathsf{vss}}$ call, and let $F^p(x,y)$ be the sharing polynomial of the weak double t-sharing of $\mathcal{F}^p_{\mathsf{vss}}$. Let $\Delta^1,\ldots,\Delta^\ell$ be the broadcasts of the dealer in the online round. The simulator sets $z^p := \Delta^p + F^p(0,0)$ for $p\in\{1,\ldots,\ell\}$ and inputs $(z^1,\ldots,z^\ell)$ to $\mathcal{F}_{\mathsf{psif}}$. This concludes the simulation.

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

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

Honest parties' outputs. We say that a view View is "good" if for any commitment C that appears in the view, and for any two openings o, and o' that appear in the view, it holds that either

$\mathsf{open}(C, o) = \bot$ or $\mathsf{open}(C, o') = \bot$ or $\mathsf{open}(C, o) = \mathsf{open}(C, o)$. Observe that, by the binding property of the commitment scheme, a view View is good with probability at least $1 - \epsilon$.

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

Since View is good, and there are $n - t \geq t + 1$ honest parties, by Fact A.2 it follows that that $F^p(x, y) = \tilde{F}^p(x, y)$ and $F^{pq}(x, y) = \tilde{F}^{pq}(x, y)$ for all $p, q \in \{1, \ldots, \ell\}$. Similarly, it holds that $F^{\gamma_i}(x, y) = \tilde{F}^{\gamma_i}(x, y)$ and $F^{\rho_i}(x, y) = \tilde{F}^{\rho_i}(x, y)$ for every $i \in \{1, \ldots, m\}$. Therefore, by the $\mathcal{F}_{\mathsf{tss}}$ functionality, we conclude that $F^{pq}(0, 0) = F^p(0, 0) \cdot F^q(0, 0)$ for all $p, q \in \{1, \ldots, \ell\}$. We also conclude that $\tilde{F}^{\eta_i}(0, 0) = \tilde{F}^{\gamma_i}(0, 0) \cdot \tilde{F}^{\rho_i}(0, 0) = 0$.

Let $z^p := \Delta^p + F^p(0, 0)$ for $p \in \{1, \ldots, \ell\}$. In the ideal world the output is $y(\mathbf{z})$ It remains to show that this also holds in the real world. Fix $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, n\}$ and consider the computation of the j-th row of y^i. Since D is not discarded, and View is good, we conclude that the j-th share of y^i is consistent with the shares of every honest P_k,

$$\begin{aligned}\alpha^i_0 &+ \sum_{p \in \{1, \ldots, \ell\}} \alpha^i_p(\tilde{F}^p(k, j) + \Delta^p) \\ &+ \sum_{p,q \in \{1, \ldots, \ell\}} \alpha^i_{pq}(\Delta^q \tilde{F}^p(k, j) + \Delta^p \tilde{F}^q(k, j) + \Delta^p \cdot \Delta^q + \tilde{F}^{pq}(k, j)) + \tilde{F}^{\eta_i}(k, j).\end{aligned}$$

and since there are $n - t \geq t + 1$ honest parties the j-th share of y^i must be

$$\begin{aligned}\alpha^i_0 &+ \sum_{p \in \{1, \ldots, \ell\}} \alpha^i_p(\tilde{F}^p(0, j) + \Delta^p) \\ &+ \sum_{p,q \in \{1, \ldots, \ell\}} \alpha^i_{pq}(\Delta^q \tilde{F}^p(0, j) + \Delta^p \tilde{F}^q(0, j) + \Delta^p \cdot \Delta^q + \tilde{F}^{pq}(0, j)) + \tilde{F}^{\eta_i}(0, j).\end{aligned}$$

Since there are $n - t \geq t + 1$ honest parties, we conclude that the parties recover the value

$$\begin{aligned}y^i = \alpha^i_0 &+ \sum_{p \in \{1, \ldots, \ell\}} \alpha^i_p(\tilde{F}^p(0, 0) + \Delta^p) \\ &+ \sum_{p,q \in \{1, \ldots, \ell\}} \alpha^i_{pq}(\Delta^q \tilde{F}^p(0, 0) + \Delta^p \tilde{F}^q(0, 0) + \Delta^p \cdot \Delta^q + \tilde{F}^{pq}(0, 0)) + \tilde{F}^{\eta_i}(0, 0) \\ = \alpha^i_0 &+ \sum_{p \in \{1, \ldots, \ell\}} \alpha^i_p z^p + \sum_{p,q \in \{1, \ldots, \ell\}} \alpha^i_{pq} z^p \cdot z^q + 0 \\ = y^i&(\mathbf{z}),\end{aligned}$$

where the second equality follows from Fact A.5 (or simply by using the definition of the z^p's). This concludes the analysis of a corrupt dealer, and the proof of security of $\mathsf{psiflog}$. □

I.1.3 Sufficient Conditions for Efficient Optimal-Resiliency SIF

The efficiency-bottleneck of the psiflog protocol is the TSS sub-protocol whose running time is *exponential* in the number of parties. In order to obtain a protocol with optimal resiliency and running time polynomial in n, we can slightly weaken the adversary by assuming that it is non-rushing. Indeed, in this case, instead of letting each set of $t+1$ parties to generate a challenge, we can simply let every party P_i broadcast a challenge α_i already in the first round, and let the parties recover $A(\alpha_i), B(\alpha_i)$ and $C(\alpha_i)$ in the second round. In fact, the same approach can be used even against a stronger adversary that is allowed to see all the messages of the honest parties except one, and only then send the messages of the corrupt parties. This adversary is even allowed to adaptively choose which messages to see first. We refer to such an adversary as *semi-rushing*.

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

Alternatively, an efficient protocol with optimal resiliency can be obtained by strengthening the underlying cryptographic assumptions. Following the Fiat-Shamir transform [33, 12], we can obtain an efficient protocol in the random-oracle model, by using the random-oracle to pick the challenge given the public commitments that the dealer generates in the first round.

As explained in the introduction, we do not take these routes and construct (in Section J) an efficient SIF protocol for a rushing adversary based on the same Minicrypt-type assumptions that were used so far, at the expense of slightly relaxing the resiliency threshold.

J Public SIF for any Number of Parties

Here, we present a public SIF protocol, denoted psif, that efficiently implements $\mathcal{F}_{\mathsf{psif}}$ when $n = (2+\epsilon)t$ for some $\epsilon > 0$. For the high-level idea, we refer to Section 3.

J.1 Execution of psiflog with a Subset of Parties

We start by discussing the execution of psiflog with only some of the parties in $\{P_1,\dots,P_n\}$. Let $Q = (i_1,\dots,i_{n'}) \in \{1,\dots,n\}^{n'}$ be an ordered multiset of size n' and let $t' := \lfloor (n'-1)/2 \rfloor$. In an execution of psiflog with Q, we consider an execution of psiflog with n' parties, denoted $\{P'_1,\dots,P'_{n'}\}$ and resiliency threshold t', so that for every $j \in \{1,\dots,n'\}$, party P_{i_j} takes the role of P'_j in the protocol's execution. We note that a single party P_i may appear several times in Q, which means that

P_i can take the role of more than one party P'_{i_j} in an execution of psiflog with Q (this happens when $i = i_j = i_{j'}$ for some $j \neq j'$, so P_i takes both the role of P'_{i_j} and the role of $P'_{i'_j}$). When the parties in Q need to send a private message to each other they do so using the private channels among every two parties. However, when they want to send a broadcast message, they use the standard broadcast channel, so that *all* parties, inside and outside Q, can see the broadcast message. We denote such an execution by $\mathsf{psiflog}(Q)$.

We say that Q is good if it contains an honest majority (that is, if the number of corrupt parties among $(i_1, \ldots, i_{n'})$ is at most t', where we count with repetitions). Otherwise, we say that Q is bad. Our goal is to show that when the parties in Q execute psiflog among themselves, we obtain a secure implementation of the following functionality.

Functionality $\mathcal{F}_{\mathsf{psif}}(Q)$

$\mathcal{F}_{\mathsf{psif}}(Q)$ receives the set of corrupt parties C. The functionality depends on the set Q, and we split into cases.

Good Q**.** $\mathcal{F}_{\mathsf{psif}}(Q)$ receives inputs $\mathbf{z} = (z^1, \ldots, z^\ell)$ from D in Q. $\mathcal{F}_{\mathsf{psif}}(Q)$ returns $y(\mathbf{z}) = (y^1(\mathbf{z}), \ldots, y^m(\mathbf{z}))$ to all parties, inside and outside Q, where $y^i(z^1, \ldots, z^\ell) = \alpha_0^i + \sum_{p \in \{1,\ldots,\ell\}} \alpha_p^i z^p + \sum_{p,q \in \{1,\ldots,\ell\}} \alpha_{pq}^i z^p z^q$ is a degree-2 polynomial in the variables $z^1, \ldots, z^\ell$.

Bad Q**.** We split into cases.

- **Honest** D**.** D inputs $\mathbf{z}$ to $\mathcal{F}_{\mathsf{psif}}(Q)$, and $\mathcal{F}_{\mathsf{psif}}(Q)$ leaks $\mathbf{z}$ to the adversary. Then, $\mathcal{F}_{\mathsf{psif}}(Q)$ receives a vector $(y^1, \ldots, y^m)$ from the adversary and gives it to all parties, inside and outside Q.
- **Corrupt** D**.** $\mathcal{F}_{\mathsf{psif}}(Q)$ receives a vector $(y^1, \ldots, y^m)$ from the adversary and gives it to all parties, inside and outside Q.

Figure 11: Functionality $\mathcal{F}_{\mathsf{psif}}(Q)$

That is, when Q is good, $\mathcal{F}_{\mathsf{psif}}(Q)$ acts exactly like $\mathcal{F}_{\mathsf{psif}}$: it takes the input $\mathbf{z}$ of the dealer, and gives $y(\mathbf{z})$ to all the parties, inside and outside Q. On the other hand, when Q is bad, there are no security guarantees, other than that all the parties have the same output. That is, the input of an honest D is leaked to the adversary, and the adversary is allowed to choose the output of the functionality.

Note that $\mathsf{psiflog}(Q)$ is a UC-secure implementation of $\mathcal{F}_{\mathsf{psif}}(Q)$. Indeed, when Q is bad, simulation is trivial and all parties have the same output, by the public output property of the psiflog protocol, that guarantees that all parties, inside and outside Q, learn the output of $\mathsf{psiflog}(Q)$. When Q is good, the claim follows from Theorem 5.5. In addition, the complexity of the protocol is polynomial in n and exponential in $|Q|$, so the protocol is efficient if $|Q| = O(\log \kappa)$. A security statement appears in Theorem J.1.

Theorem J.1. *Let κ be a security parameter, let n be the number of parties, let $t < n/2$. Let n' be an integer, and let $Q \in \{1, \ldots, n\}^{n'}$ be a ordered multiset of parties. Assuming the existence of perfectly-binding sub-exponentially hiding NICOM, protocol $\mathsf{psiflog}(Q)$ is a UC-secure implementation of $\mathcal{F}_{\mathsf{psif}}(Q)$, against a static, active, rushing adversary corrupting up to t parties. The complexity of the protocol is $\mathrm{poly}(|\mathcal{C}|, n, 2^{|Q|}, \log|\mathbb{F}|, \kappa)$, where $\mathcal{C}$ is the circuit computing the functionality.*

Alternatively, if we replace the perfectly-binding NICOM with statistically-hiding NICOM, we also obtain everlasting security.

J.2 Public SIF

We continue with the description of psif, that efficiently implements $\mathcal{F}_{\mathsf{psif}}$ when $n = (2+\epsilon)t$ for some $\epsilon > 0$. We slightly deviate from the description in Section 3.2, and present a similar approach to that presented in Remark 3.1. We start by presenting Observation J.2 that shows that there exists some constant n' that depends only on ϵ, such that if we take the committees $A_1, \ldots, A_M$ to be all ordered multisets of size n', then at least $9/10$-fraction of the multisets will have an honest majority.

Observation J.2. *Let $\epsilon > 0$ and $n = (2+\epsilon)t$. Let $\mathsf{C} \subseteq \{1, \ldots, n\}$ be the set of corrupt parties, of size at most t. There exists a positive integer $n' = n'(\epsilon)$ that depends only on ϵ, so that the fraction of ordered multi-sets $(i_1, \ldots, i_{n'}) \in \{1, \ldots, n\}^{n'}$ that contain at most $(\frac{1}{2} + \frac{\epsilon}{20})n'$ honest parties is at most $1/10$.*

We note that the total number of ordered multi-sets of size n' is $n^{n'} = \mathrm{poly}(n)$, and we assume that n' is large enough with respect to ϵ so that $(\frac{1}{2} + \frac{\epsilon}{20})n' > \frac{n'}{2} + 1$.

Proof. We have $n = (2+\epsilon)t$ parties, where t of them are corrupt and $(1+\epsilon)t$ are honest. Therefore, the fraction of honest parties is

$$\mu := \frac{(1+\epsilon)t}{(2+\epsilon)t} = \frac{1+\frac{1}{2}\epsilon}{2+\epsilon} + \frac{\frac{1}{2}\epsilon}{2+\epsilon} = \frac{1}{2} + \frac{\epsilon}{4+2\epsilon}.$$

Let $\gamma(\epsilon) := \epsilon/(4+2\epsilon)$, note that $\gamma > 0$ and that the fraction of honest parties is $\mu = \frac{1}{2} + \gamma$. Consider a random ordered multiset Q of size n', and observe that for every $i \in \{1, \ldots, n'\}$ the probability that the i-th element is honest is μ. Therefore, by an additive Chernoff bound, the probability that the fraction of honest parties in Q is less than $\mu - \gamma/2 = (\frac{1}{2} + \gamma) - \gamma/2$ is at most $\exp(-\gamma^2 n'/2)$. By taking $n' > 2\log(10)/\gamma^2$ we conclude that at most 1/10th of the ordered multisets of size n' have less than $\mu - \gamma/2 > \frac{1}{2} + \frac{\epsilon}{20}$ fraction of honest parties. This concludes the proof. □

In fact, it turns out that we can improve the resiliency by using the techniques of [28], as explained in the following remark.

Remark J.3 (Improving resiliency). *For every $0 < \epsilon, \delta < 1$, the techniques of [28, Lemma 5] allows the efficient generation of $M = n$ committees of size $n' = O(\frac{1}{\delta\epsilon^2})$, so that for every set of corrupt parties C of size at most $(\frac{1}{2} - \epsilon)n$, at most δ-fraction of the committees overlap with C in $n'/2$ or more committee-members. In our context $\delta = 1/10$, and since our running time will be exponential in n', we can push ϵ to be as small as $\Omega(1/\sqrt{\log \kappa})$. For simplicity, we continue by following the parameters set in Observation J.2.*

Protocol overview. As discussed in Section 3.2 and Remark 3.1, in order to compute a single input function f, we let D compute an $M/3$-out-of-M secret sharing of its input $\mathbf{z}$, denoted $(\mathbf{z}_1, \ldots, \mathbf{z}_M)$, and then compute "in-the-head" an interaction between the virtual parties $Q_1, \ldots, Q_M$, where Q_i holds as an input the share $\mathbf{z}_i$, and the virtual parties compute the BGW protocol for a functionality $\mathcal{G}_{\mathsf{bgw}}$ that first reconstructs $\mathbf{z}$ from the shares, and then returns $f(\mathbf{z})$ to all the parties. We let the dealer commit to the input and randomness of the virtual parties, as well as to every private and public message sent in the protocol. We identify the i-th virtual party Q_i with the corresponding committee A_i, and we let the dealer execute an instance $\mathsf{psiflog}(Q_i)$ for a functionality $\mathcal{G}_{\mathsf{zk}}$ that verifies that the committed view of Q_i is consistent with the BGW protocol, and if so, returns $f(\mathbf{z})$ to all the parties, inside and outside Q_i. The security of our protocol follows from the same argument presented in Section 3.2. We continue with a formal description of the protocol, and then provide a formal proof of security.

Protocol psif

Primitives: Public single input functionality for a subset of parties $\mathsf{psiflog}(Q)$.

Offline phase:

- *($\mathsf{psiflog}(Q)$ call)* Let $M := n^{n'}$ for a constant $n' = n'(\epsilon)$ as in Observation J.2. Let $Q_1, \ldots, Q_M$ be all ordered multi-sets of size n' of $\{1, \ldots, n\}$, and we add to each multi-set the dealer D as the last party. For every Q_k the parties engage in an execution of the offline phase of $\mathsf{psiflog}(Q_k)$ with D as the dealer of the execution, computing the functionality $\mathcal{G}_{\mathsf{zk}}$, defined in Figure 14.

Verification phase (R2):

- *(Inputs)* D holds inputs $z^1, \ldots, z^\ell$.
- *(MPC in the head)* For every $v \in \{1, \ldots, \ell\}$, D shares the input z^v via Shamir's secret sharing for M parties and threshold $T := \lfloor (M-1)/3 \rfloor$.[a] Let $s^v(k)$ be the k-th share of z^v.

  D executes *in the head* the BGW protocol with M virtual parties, which we identify with the multi-sets $Q_1, \ldots, Q_M$, and threshold T for the functionality $\mathcal{G}_{\mathsf{bgw}}$ defined in Figure 13, where the k-th party has inputs $(s^v(k))_{v \in \{1,\ldots,\ell\}}$ and randomness ρ_k. D samples commitments and openings to (1) the inputs and randomness of Q_k for all $k \in \{1, \ldots, M\}$, and (2) every message sent in the BGW execution (that is, for every round r and virtual party Q_k in the BGW execution, the dealer commits to the r-th round broadcast message m_k^r of Q_k, as well as to the private messages $(m_{k,k'}^r)_{k' \in \{1,\ldots,M\}}$ from Q_k to the rest of the parties $Q_{k'}$). D broadcasts all those commitments.
- *($\mathsf{psiflog}(Q)$ call)* For every $k \in \{1, \ldots, M\}$, the parties execute the online phase of $\mathsf{psiflog}(Q_k)$ computing $\mathcal{G}_{\mathsf{zk}}$ defined in Figure 14, where D inputs all the commitments and openings that correspond to the view of Q_k in the execution of BGW,[b], together with the index k.
- *(Local computation)* If there exists some vector y such that for at least $9M/10$ indices $k \in \{1, \ldots, M\}$ it holds that the output of $\mathsf{psiflog}(Q_k)$ is "valid" concatenated with (1) y, (2) the commitments that correspond to the view of Q_k and (3) the index k, then all parties output y. Otherwise, D is discarded and all parties output some default value $y(0, \ldots, 0)$.

[a]That is, T parties learn nothing about the secret, but $T+1$ parties can recover the secret.

[b]By the view of Q_k we mean the inputs $(s^v(k))_{v \in \{1,\ldots,\ell\}}$ and randomness ρ_k of Q_k, together with all the incoming messages $m_{k',k}^r$ and $m_{k'}^r$ and outgoing messages $m_{k,k'}^r$, m_k^r for every round r and party $Q_{k'}$

Figure 12: Protocol psif

Functionality $\mathcal{G}_{\mathsf{bgw}}$

Input. Each party P_k inputs shares $(s^v(k))_{v \in \{1,\ldots,\ell\}}$.

Output. The functionality recovers the secrets z^v from the shares $(s^v(k))_{k \in \{1,\ldots,M\}}$ for all $v \in \{1, \ldots, \ell\}$. That is, given $s^v(1), \ldots, s^v(M)$ the functionality interpolates the M points in order to obtain a polynomial $f^v(x)$ with $f^v(k) = s^v(k)$ for all $k \in \{1, \ldots, M\}$. If the degree of $f^v(x)$ is less than $M/3$ then the functionality sets $z^v := f^v(0)$[a]. Otherwise, if the degree $f^v(x)$ is at least $M/3$, the recovery fails.

If some recovery fails then the functionality returns $\perp$ to all parties. Otherwise, the functionality returns $y(\mathbf{z})$ to all parties.

[a]We have noted that, for an honest dealer, all virtual parties are honest but curious. This means that when D is honest all virtual parties input the correct shares, so there is no need in Reed-Solomon reconstruction, but just interpolation over the points.

Figure 13: Functionality $\mathcal{G}_{\mathsf{bgw}}$

Functionality $\mathcal{G}_{\mathsf{zk}}$

Input. The dealer inputs commitments and openings $((G_1, h_1), \ldots, (G_m, h_m))$ and an index $k \in \{1, \ldots, M\}$.

Output. If (1) every opening h_i is a valid opening for G_i, and (2) the opened values correspond to a view of a virtual party Q_k that follows the BGW protocol for $\mathcal{G}_{\mathsf{bgw}}$,[a], then $\mathcal{G}_{\mathsf{zk}}$ returns ("valid", $y, (G_1, \ldots, G_m), k$) to all parties, where y is the output according to the view of Q_k. Otherwise, $\mathcal{G}_{\mathsf{zk}}$ returns "invalid" to all parties.

[a]By the view of Q_k we mean the inputs and randomness of Q_k, together with all the incoming messages that Q_k received and outgoing messages that Q_k sent.

Figure 14: Functionality $\mathcal{G}_{\mathsf{zk}}$

We continue with a proof of Theorem 5.3.

J.3 Proof of security

Proof of Theorem 5.3. From the composition properties of UC-security, it is enough to prove security in the $\mathcal{F}_{\mathsf{psif}}(Q)$-hybrid model.

Let $\mathcal{A}$ be the dummy adversary against psif. We define the simulator $\mathcal{S}$ as follows. The simulator $\mathcal{S}$ uses $\mathcal{A}$ in a black-box manner, and forwards all messages between $\mathcal{Z}$ and $\mathcal{A}$. The simulator first receives the set of corrupt parties C. We split into cases.

J.3.1 Honest Dealer

Offline round. There is no communication in the offline round in the $\mathcal{F}_{\mathsf{psif}}(Q)$-hybrid model.

Online phase. The simulator receives the output y from $\mathcal{F}_{\mathsf{psif}}$. For every $v \in \{1, \ldots, \ell\}$, the simulator computes Shamir's secret sharing of 0 for M parties with resiliency T, denoted $\bar{s}^v(1), \ldots, \bar{s}^v(M)$, where $M = n^{n'} = \mathrm{poly}(n)$ and $T = \lfloor (M-1)/3 \rfloor$, just like in the protocol psif. Consider the BGW protocol for the computation of $\mathcal{G}_{\mathsf{bgw}}$ with M parties and resiliency T, let $\mathcal{A}'$ be the adversary against the protocol that always follows the protocol, and let $\mathcal{S}'$ be the corresponding simulator. Like in psif, we think of the virtual parties $Q_1, \ldots, Q_m$ of the BGW protocol as all ordered multisets of size n', and we add to each multiset the dealer D as the last party. Let C′ be the set of corrupt parties for the BGW protocol, which is defined to be all virtual parties Q_i that represent an ordered multiset with a dishonest majority (after the addition of D to the multiset).

The simulator $\mathcal{S}$ executes the simulator $\mathcal{S}'$ for the BGW protocol with C′ as the set of corrupt parties, and inputs $(\bar{s}^v(k))_{v \in \{1,\ldots,\ell\}}$ for every corrupt Q_k. When $\mathcal{S}'$ sends the inputs of the corrupt virtual parties to the ideal functionality $\mathcal{G}_{\mathsf{bgw}}$, then $\mathcal{S}$ returns y as the output of the ideal functionality. We emphasize that such a simulation is possible, because by Observation J.2, the fraction of corrupt parties Q_k is at most $1/10$. At this stage the dealer holds the simulated view of adversary $\mathcal{A}'$ in the execution BGW protocol. Those include (1) the inputs and randomness of the parties in C', (2) every r-th round private message $\bar{m}^r_{k,k'}$ where either Q_k is corrupt or $Q_{k'}$ is corrupt (or both), and (3) all broadcast messages $\bar{m}^r_k$ for every round r and every party Q_k (either honest or dishonest). For the missing values in the view of honest parties, i.e., inputs and randomness of honest parties, and messages $\bar{m}^r_{k,k'}$ between two honest parties Q_k and Q'_k, the simulator $\mathcal{S}$ sets the value 0. The simulator computes the commitments of those values and broadcasts the commitments. We emphasize that for every corrupt Q_k the simulator holds openings to the commitments that reveal the view of Q_k as generated by $\mathcal{S}'$.

For every $\mathcal{F}_{\mathsf{psif}}(Q_k)$ call for an honest Q_k, the simulator returns the output: "valid" concatenated to (1) the vector y, (2) the commitments that correspond to the view of Q_k and (3) the index k. For every $\mathcal{F}_{\mathsf{psif}}(Q_k)$ call for a corrupt Q_k, the simulator leaks all commitments and openings that correspond to the view of the corrupt Q_k, together with the index k. This concludes the communication from the honest parties to the corrupt parties. At this stage, the adversary $\mathcal{A}$ sends the output out_k of $\mathcal{F}_{\mathsf{psif}}(Q_k)$ for every corrupt Q_k, and the simulator returns out_k as the output of $\mathcal{F}_{\mathsf{psif}}(Q_k)$. This concludes the simulation.

Fix a polynomial-time environment $\mathcal{Z}$ with input ζ, and assume without loss of generality that it is deterministic.

Analysis. We observe that there is no communication in the offline round, so in both worlds the environment $\mathcal{Z}$ picks the inputs $\mathbf{z}$ of D in the same way. Fix those inputs. Consider the real-world shares $(s^v(k))_{v\in\{1,\ldots,\ell\},k\in\mathsf{C}'}$ and note that they have the same distribution as the ideal-world shares $(\bar{s}^v(k))_{v\in\{1,\ldots,\ell\},k\in\mathsf{C}'}$. Fix those shares as well.

Consider the simulation of $\mathcal{S}'$ by $\mathcal{S}$. We claim that it has the same distribution as an execution of $\mathcal{S}'$ in the virtual ideal-world of $\mathcal{G}_{\mathsf{bgw}}$, where the virtual party Q_k receives $(s^v(k))_{v\in\{1,\ldots,\ell\}}$. In order to prove it, it is enough to show that in the virtual ideal-world the functionality $\mathcal{G}_{\mathsf{bgw}}$ returns $y(\mathbf{z})$ with probability 1. Indeed, by the perfect correctness of the BGW protocol, the output of the honest virtual parties in the virtual real-world is $y(\mathbf{z})$ with probability 1. Therefore, in the virtual ideal-world of $\mathcal{G}_{\mathsf{bgw}}$ the output of $\mathcal{G}_{\mathsf{bgw}}$ is $y(\mathbf{z})$ with probability 1 as well. Note that this means that for every corrupt Q_k, $\mathcal{S}'$ always inputs the correct shares $(s^v(k))_{v\in\{1,\ldots,\ell\}}$ to $\mathcal{G}_{\mathsf{bgw}}$ (or otherwise the output of $\mathcal{G}_{\mathsf{bgw}}$ will not be y).

By the perfect privacy of the BGW protocol, we conclude that the joint view of all corrupt Q_k's in the BGW execution has the same distribution in both worlds. Fix those values. Conditioned on those values it is not hard to see that the corresponding commitments and openings have the same distributions in both worlds, and we fix them as well. Finally, the rest of the commitments, that correspond to inputs and randomness of honest parties, as well as to private messages between honest parties, are $\mathrm{poly}(n,\ell,m,\log|\mathbb{F}|)\cdot\epsilon$-close in both worlds; indeed, note that the total number of commitments is bounded by the complexity of the BGW protocol for $\mathcal{G}_{\mathsf{zk}}$, which is $\mathrm{poly}(n,\ell,m,\log|\mathbb{F}|)$. Given those values, it is not hard to see that the rest of the view can be generated by the same efficient process in both worlds.

We continue by an analysis of the honest parties' output. We claim that in both worlds, given the view of $\mathcal{Z}$, the output of the honest parties can be computed by the same deterministic efficient process that simply outputs $y(\mathbf{z})$. Indeed, in the ideal-world all parties output $y(\mathbf{z})$ with probability 1. In addition, in the real-world, for every honest Q_k the output of $\mathcal{F}_{\mathsf{psif}}(Q_k)$ is "valid" concatenated to (1) $y(\mathbf{z})$ (2) the commitments that correspond to the view of Q_k and (3) the index k. Since there are at least $9M/10$ honest virtual parties, all honest parties output $y(\mathbf{z})$ as well. This concludes the analysis of an honest dealer.

J.3.2 Corrupt Dealer

Offline round. There is no communication in the offline round in the $\mathcal{F}_{\mathsf{psif}}(Q)$-hybrid model.

Online round. The simulator takes the role of the honest parties, that have no inputs, in order to simulate an execution of the protocol. The simulator receives the broadcasts of D, that consist of the commitments to the view of the virtual parties in the BGW execution, as well as the inputs of D to $\mathcal{F}_{\mathsf{psif}}(Q_k)$ for a multiset Q_k with an honest majority (which we think of as honest virtual parties), and the output of $\mathcal{F}_{\mathsf{psif}}(Q_k)$ for a multiset Q_k with a dishonest majority (which we think of as corrupt virtual parties). For every honest Q_k the dealer computes the output of $\mathcal{F}_{\mathsf{psif}}(Q_k)$ and gives it to the adversary and to the simulated honest parties. For every corrupt Q_k the dealer received from $\mathcal{A}$ the output of $\mathcal{F}_{\mathsf{psif}}(Q_k)$, and the simulator gives it to the adversary and to the simulated honest parties.

At the end of the simulation, if D is discarded, then $\mathcal{S}$ inputs $(0,\ldots,0)$ to $\mathcal{F}_{\mathsf{psif}}$. Otherwise, D is not discarded by the honest parties, and let y be the outpu of the honest parties. Let V be the set of all honest virtual parties Q_k so that the output of $\mathcal{F}_{\mathsf{psif}}(Q_k)$ is "valid" concatenated to (1) y, (2) the commitments that correspond to the view of Q_k, and (3) the index k. Since D is not discarded, and, by Observation J.2, the number of corrupt Q_k is at most $M/10$, we conclude that the size of V is at least $\frac{9M}{10} - \frac{M}{10} = \frac{8M}{10}$. For every Q_k in V, since the output of $\mathcal{F}_{\mathsf{psif}}(Q_k)$ is not "invliad", the input of D to $\mathcal{F}_{\mathsf{psif}}(Q_k)$ has to include correct openings to the committed view of Q_k. For each Q_k in V, the simulator uses those openings to extract the shares $(s^v(k))_{v\in\{1,\ldots,\ell\}}$, that correspond to Q_k's input in the BGW execution. Then, the simulator interpolates over the shares of all Q_k's in V in order to obtain the values $z^1,\ldots,z^\ell$ (if the interpolation fails then the simulator aborts), and send $z^1,\ldots,z^\ell$ to $\mathcal{F}_{\mathsf{psif}}$ as the inputs of D. This concludes the simulation.

Fix a polynomial-time environment $\mathcal{Z}$ with input ζ, and assume without loss of generality that $\mathcal{Z}$ is deterministic.

Analysis. It is not hard to verify that the online-round simulation is perfect. Therefore, it only remains to analyse the honest parties' outputs. We say that a view View is "good" if for any commitment C that appears in the view, and for any two openings o, and o' that appear in the view, it holds that either $\mathsf{open}(C,o) = \perp$ or $\mathsf{open}(C,o') = \perp$ or $\mathsf{open}(C,o) = \mathsf{open}(C,o)$. Observe that, by the binding property of the commitment scheme, a view View is good with probability at least $1-\epsilon$.

Fix any good view View. If D is discarded according to View, then in both worlds the output is the default value $y(0,\ldots,0)$. Otherwise, D is not discarded. Consider a virtual party Q_k in V, and note that D has provided valid openings to the committed view of Q_k, which we denote by View_k. Note that, since View is good, the views of any two parties in V, View_k and $\mathsf{View}_{k'}$ are consistent with each other (that is, they agree on all broadcast messages, and on all private messages between Q_k and $Q_{k'}$). Therefore, the set of views $\{\mathsf{View}_k\}_{k\in\mathsf{V}}$ is consistent with some execution of BGW, where the adversary corrupts the set $\mathsf{C}' := \{1,\ldots,m\} \setminus \mathsf{V}$. In the real-world, by the perfect correctness of the BGW protocol, the output y of the BGW protocol is consistent with the values $z^1,\ldots,z^\ell$ which are shared via the shares $(s^v(k))_{v\in\{1,\ldots,\ell\},k\in\mathsf{V}}$.[28] In the ideal world,

[28]Observe that *perfect* correctness is crucial for this argument, since the dealer chooses the randomness of the honest virtual parties. Indeed, if we allowed a positive error probability then the dealer could pick bad randomness for the virtual honest parties, so that the correctness property would be violated.

the simulator inputs $z^1, \ldots, z^\ell$ to $\mathcal{F}_{\mathsf{psif}}$, so the output is y as well. This concludes the analysis of a corrupt dealer, and the proof of security of psif. $\square$

K General Single Input Functionality

In this section, we present a protocol for general SIF, when $n = (2+\epsilon)t$ for some constant $\epsilon > 0$. This protocol is a reduction from the general SIF to public SIF as discussed in Section 3.

Let $\mathcal{F}$ be some SIF, that takes as an input a vector $\mathbf{z}$ from D, returns $f_i(\mathbf{z})$ to P_i, for some functions $f_1(\mathbf{z}), \ldots, f_n(\mathbf{z})$. We will reduce the computation of $\mathcal{F}$ to the computation of a SIF $\mathcal{G}$ with public output.

Let $\mathcal{G}$ be a SIF that takes as an input a vector $\mathbf{z}$ and a list of commitments and openings $(C_1, o_1), \ldots, (C_n, o_n)$. For $i \in \{1, \ldots, n\}$, let $r_i := \mathsf{open}(C_i, o_i)$. If some r_i is $\perp$ the functionality returns a special abort symbol $\perp$ to all parties. Otherwise, the functionality returns $(C_1, \ldots, C_n)$ and $(f_1(\mathbf{z}) + r_1, \ldots, f_n(\mathbf{z}) + r_n)$ to all parties. We note that $\mathcal{G}$ has public output, and that the circuit size of $\mathcal{G}$ is at most polynomial in the circuit size of $\mathcal{F}$, in κ and in $\log|\mathbb{F}|$.

We continue by describing a reduction that "almost works", and then we explain how to fix it. In the offline round the dealer samples random pads $r_1, \ldots, r_n$, computes a commitment and opening (C_i, o_i) for every r_i, broadcasts $C_1, \ldots, C_n$ and sends o_i to P_i. In addition, the parties execute the offline phase for psif for functionality $\mathcal{G}$. In the online phase the dealer receives the input $\mathbf{z}$ and the parties compute the online phase for psif, where D inputs $\mathbf{z}$ and $(C_1, o_1), \ldots, (C_n, o_n)$. If the output of psif is $\perp$ then D is discarded and all parties output some default output. Otherwise, let the output be $(C'_1, \ldots, C'_n)$ and $(y_1, \ldots, y_n)$. If $(C'_1, \ldots, C'_n) \neq (C_1, \ldots, C_n)$, D is discarded. Otherwise, P_i outputs y_0 and $y_i - r_i$.

We note that the reduction works, as long as D sends each P_i its opening o_i in the offline round. However, a corrupt D might refuse to send the opening in the offline round to an honest P_i, in which case P_i will be unable to decrypt its output. In order to solve this problem, we use a mechanism, called *undeniable transmission*, that allows an unhappy P_i to force the dealer to reveal the opening to all the parties in the online round. In this way, all the parties will be able to verify that D reveals a valid opening in the online round, and if not, D will be discarded. We mention that undeniable transmission appears implicitly in [7]. Details follow.

Undeniable transmission Consider the case where in the first round a dealer D should send a message o to, say, P_1. We want a mechanism that allows P_1 to force D to reveal o in the second round to all parties. In order to do so, in the first round we let P_1 sample a random pad g, compute a commitment and opening (G, h) of g, broadcast the commitment G and send the opening h to D. In the second round, if D is happy with the values received from P_1 (that is, if h is a valid opening of G), then D broadcasts $\alpha := o + g$. Otherwise D concludes that P_1 is corrupt, and simply reveals the opening o to all parties. (This reveals no information that the adversary did not already know, as P_1 must be corrupt.) At this stage, if P_1 wants to force D to reveal o, then she can broadcast h in the second round. Indeed, if D is unhappy with P_1 then D broadcasts o anyway. If D is happy with P_1, then the only value that P_1 can open from G is g, so all the parties can compute $\alpha - g$ in order to obtain o. Finally, when both D and P_1 are honest then they are both happy with each other, no information about o is revealed to the adversary, as g is a random string that is used as a one-time pad.

Protocol sif appears in Figure 15.

Protocol sif

Primitives: Public SIF psif.

offline phase:

- The dealer D samples random pads $r_1, \ldots, r_n$, where the length of r_i is as the length of $f_i(\mathbf{z})$. The dealer samples $(C_i, o_i) \leftarrow \mathsf{commit}(r_i)$ for every $i \in \{1, \ldots, n\}$, broadcasts the commitments $(C_1, \ldots, C_n)$ and sends o_i to P_i for every $i \in \{1, \ldots, n\}$.
- Every P_i samples a random pad g_i of the same length as o_i. P_i samples $(G_i, h_i) \leftarrow \mathsf{commit}(g_i)$, broadcasts G_i and sends (g_i, h_i) to D.
- *(*psif *offline call)* All parties execute the offline round of psif for the functionality $\mathcal{G}$.

online phase:

- *(Inputs)* The dealer receives the input $\mathbf{z}$.
- If D received an opening h_i from P_i such that $\mathsf{open}(G_i, h_i) \neq g_i$, then D broadcasts o_i. Otherwise, D broadcasts $\alpha_i := o_i + g_i$.
- If P_i received an invalid opening o_i from D, then P_i broadcasts h_i.
- *(*psif *online call)* All parties execute the online round of psif, where the dealer inputs $\mathbf{z}$ and $(C_1, o_1), \ldots, (C_n, o_n)$.
- *(Local computation)* All parties do as follows.
  - If the output of psif is $\perp$ then D is discarded and the parties output $\mathcal{F}(0, \ldots, 0)$.
  - Otherwise, let the output be $(C'_1, \ldots, C'_n)$ and $(y_1, \ldots, y_n)$. If $(C'_1, \ldots, C'_n) \neq (C_1, \ldots, C_n)$ then D is discarded.
  - A party P_i is conflicted with D if P_i broadcasted h_i such that the value $g_i := \mathsf{open}(G_i, h_i)$ is not $\perp$. For every P_i conflicting with D, let $\bar{o}_i$ be (1) o_i if D broadcasted o_i in the online phase, or (2) $\alpha_i - g_i$ if D broadcasted α_i in the online phase. If $\mathsf{open}(C_i, \bar{o}_i) = \perp$ for some P_i conflicting with D, then D is discarded.
  - Otherwise, for every P_i not conflicting with D let $\bar{o}_i$ be the opening received from D in the offline round. Every P_i (either conflicting or not conflicting with D) sets $\bar{r}_i := \mathsf{open}(C_i, \bar{o}_i)$, and outputs $y_i - \bar{r}_i$.

Figure 15: Protocol sif

We continue with a proof of Theorem 5.4 and Theorem 5.5.

K.1 Proof of Security

Proof of Theorem 5.4 and Theorem 5.5. In this section we prove that protocol sif UC-emulates $\mathcal{F}$ (with everlasting security if the underlying commitment-scheme is statistically-hiding). From the composition properties of UC-security, it is enough to prove security in the $\mathcal{G}$-hybrid model. Let $\mathcal{A}$ be the dummy adversary against sif. We define the simulator $\mathcal{S}$ as follows. The simulator $\mathcal{S}$ uses $\mathcal{A}$ in a black-box manner, and forwards all messages between $\mathcal{Z}$ and $\mathcal{A}$. The simulator first receives the set of corrupt parties C. We split into cases.

K.1.1 Honest Dealer

Offline phase. For every $i \in \{1, \ldots, n\}$ the simulator samples a random string r_i of the same length as $f_i(\mathbf{z})$ and computes $(C_i, o_i) \leftarrow \mathsf{commit}(r_i)$. The simulator broadcasts $(C_1, \ldots, C_n)$ on behalf of the dealer, and sends o_i to P_i for every $i \in \mathsf{C}$.

On behalf of every honest P_i, the simulator samples a random pad g_i of the same length as o_i. The simulator samples $(G_i, h_i) \leftarrow \mathsf{commit}(g_i)$ and broadcasts G_i on behalf of P_i. At this stage, for every corrupt P_i the simulator receives from $\mathcal{A}$ the broadcasts G_i and the private message (g_i, h_i) to the dealer.

Online phase. The simulator receives y_0 and $(y_i)_{i \in \mathsf{C}}$ from $\mathcal{F}$ as the output of the corrupt parties. For every honest P_i the simulator broadcasts a random string α_i on behalf of the dealer. For every corrupt P_i such that $g_i \neq \mathsf{open}(G_i, h_i)$ the simulator broadcasts o_i on behalf of the dealer. For every corrupt P_i such that $g_i = \mathsf{open}(G_i, h_i)$ the simulator broadcasts $\alpha_i := o_i + g_i$ on behalf of D. The simulator sends no messages on behalf of the other honest parties.

For every $i \in \mathsf{H}$ let y'_i be a random string, and for $i \in \mathsf{C}$ let $y'_i := y_i + r_i$. The simulator gives the values $(C_1, \ldots, C_n)$ and $(y'_1, \ldots, y'_n)$ to $\mathcal{A}$ as the output of the functionality $\mathcal{G}$. At this stage the simulator receives from $\mathcal{A}$ the broadcasts of the corrupt parties. This concludes the simulation of an honest dealer.

Fix a polynomial-time environment $\mathcal{Z}$ with input ζ. We begin by showing that the view of $\mathcal{Z}$ in the real-world is close to the view of $\mathcal{Z}$ in the ideal world.

$\mathcal{Z}$'s view. The adversary's view consists of (1) the crs, (2) the offline-round broadcasts of the honest parties, (3) the offline-round messages from the dealer to the corrupt parties, (4) the inputs to the honest dealer, (5) the online-round broadcasts of the honest parties, and (6) the output of $\mathcal{G}$. In order to prove that the real-world view is close to the ideal-world view, we define the following hybrid-worlds, where we assume that the honest parties know the set H.

- In Hybrid 1, the honest parties act like in the real-world, except that in the online-phase, (a) an honest party P_i never sends a broadcast message with the opening h_i, (b) for every $i \in \mathsf{H}$, D does not verify that $\mathsf{open}(G_i, h_i) \stackrel{?}{=} g_i$, but D always broadcasts α_i, (c) for every $i \in \mathsf{H}$, the computation of α_i by D is done using the value g_i that P_i sends to D (and not the values extracted from the commitments), and (d) we let the dealer provides $\mathcal{G}$ with the random pads $r_1, \ldots, r_n$ as well, and we change the functionality $\mathcal{G}$ so that it always returns the commitments $(C_1, \ldots, C_n)$ and $(y_1 + r_1, \ldots, y_n + r_n)$, where $(C_1, \ldots, C_n)$ are the commitments received from D and $(y_1, \ldots, y_n)$ are the outputs of $\mathcal{F}(\mathbf{z})$ (that is, the functionality does not open the commitment C_i; instead, it uses the random pad r_i provided by the dealer).

- In Hybrid 2, the honest parties act like in Hybrid 1, with the following modification: (a) in the offline-round, an honest P_i samples random elements g_i and g'_i, samples $(G_i, h_i) \leftarrow \mathsf{commit}(g'_i)$, broadcasts G_i, and sends (g_i, h_i) to D (so that value g_i will be used in the computation of α_i), (b) in the offline-round, for every honest P_i the dealer samples random elements r_i and r'_i, samples $(C_i, o_i) \leftarrow \mathsf{commit}(r'_i)$, and broadcasts C_i; for a corrupt P_i the dealer samples r_i, computes $(C_i, o_i) \leftarrow \mathsf{commit}(r_i)$, and broadcasts C_i; in the online-round the dealer sends $(r_1, \ldots, r_n)$ to $\mathcal{G}$ as the random pads.

Real-world vs. Hybrid 1. We claim that the real-world view has the same distribution as in Hybrid 1. This follows by noting that in the real-world (a) an honest P_i never sends a broadcast message with h_i in the online-round when D is honest, (b) in the online round D always sends α_i for an honest P_i, (c) for every honest P_i it holds that $g_i = \mathsf{open}(G_i, h_i)$, so α_i is computed in the same way in both worlds, and (d) the dealer always provides $\mathcal{G}$ with the correct openings $(o_1, \ldots, o_n)$ that correspond to the random pads $r_1, \ldots, r_n$.

Hybrid 1 vs. Hybrid 2. We claim that the view in Hybrid 1 is $O(n\epsilon)$-close to the view in Hybrid 2. Indeed, the random variables $(\mathsf{crs}, (r_i, g_i)_{i\in\mathsf{H}})$ have the same distribution in both hybrids, where g_i's are the values sent from P_i to D in the offline phase, and r_i's are sampled by D in the offline phase. Fix those values, and note that the Hybrid 1 random variables $(C_i, G_i)_{i\in\mathsf{H}}$ are $O(n\epsilon)$-close to the corresponding Hybrid 2 random variables. Finally, one can verify that in both hybrids the rest of view can be obtained from $(\mathsf{crs}, (C_i, G_i)_{i\in\mathsf{H}}, (r_i, g_i)_{i\in\mathsf{H}})$ by the same efficient process. We conclude that the view in Hybrid 1 is $O(n\epsilon)$-close to the view in Hybrid 2.

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

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

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

First, note that whenever View is good then D is not discarded in the real-world. In addition, when View is good it is not hard to see that both in the real-world and the ideal-world, the outputs of the honest parties can be extracted from View by the following efficient deterministic process: each honest P_i outputs y_i, where $(y_1, \ldots, y_n) = \mathcal{F}(\mathbf{z})$.

K.1.2 Corrupt Dealer

The simulator. The simulator takes the role of the honest parties, that have no inputs, and executes an instance of sif with $\mathcal{A}$ and $\mathcal{Z}$. At the end of the execution, if D was discarded then $\mathcal{S}$ inputs $(0, \ldots, 0)$ to $\mathcal{F}$. Otherwise, let $((C_1, o_1), \ldots, (C_n, o_n))$ and $\mathbf{z}$ be the inputs of the corrupt dealer to $\mathcal{G}$. Then D inputs $\mathbf{z}$ to $\mathcal{F}$.

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

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

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

Fix any good view View. If D is discarded according to View, then the output in both worlds is $\mathcal{F}(0, \ldots, 0)$. Therefore, we focus on the case where D is not discarded according to View. In the ideal-world, each honest P_i outputs y_i where $(y_1, \ldots, y_n) = \mathcal{F}(\mathbf{z})$. We show that this is the case in the real-world as well. Note that every honest P_i holds a valid opening o_i to C_i, or otherwise D would be discarded. Let $r_i := \mathsf{open}(C_i, o_i)$ for an honest P_i. In addition, since D is not discarded, then D must input the broadcasted commitments $(C_1, \ldots, C_n)$ to $\mathcal{G}$, together with some openings $(o'_1, \ldots, o'_n)$ and a vector $\mathbf{z}$. Since View is good it must hold that $\mathsf{open}(C_i, o_i) = r_i$ for every honest P_i. Therefore, the output of $\mathcal{G}$ is $(C_1, \ldots, C_n)$ together with a vector $(y_1 + r_1, \ldots, y_n + r_n)$, where $(y_1, \ldots, y_n) := \mathcal{F}(\mathbf{z})$. We conclude that every P_i outputs y_i. This concludes the analysis of a corrupt dealer, and completes the proof of security of sif. $\square$