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

Benny Applebaum* Eliran Kachlon* Arpita Patra†

Abstract

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

In this paper, we use Minicrypt-type assumptions to realize 3-round MPC with full and active security. Our protocols come in two flavors: for a small (logarithmic) number of parties n, we achieve an optimal resiliency threshold of $t \leq \lfloor (n-1)/2 \rfloor$, and for a large (polynomial) number of parties we achieve an almost-optimal resiliency threshold of $t \leq 0.5n(1-\epsilon)$ for an arbitrarily small constant $\epsilon > 0$. Both protocols can be based on sub-exponentially hard injective one-way functions in the plain model.

If the parties have an access to a collision resistance hash function, we can derive *statistical everlasting security* for every NC1 functionality, i.e., the protocol is secure against adversaries that are computationally bounded during the execution of the protocol and become computationally unlimited after the protocol execution.

As a secondary contribution, we show that in the strong honest-majority setting ($t < n/3$), every NC1 functionality can be computed in 3 rounds with everlasting security and complexity polynomial in n based on one-way functions. Previously, such a result was only known based on collision-resistance hash function.

*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

Interaction is a valuable and expensive resource in cryptography and distributed computation. Consequently, a huge amount of research has been devoted towards characterizing the amount of interaction, typically measured via round complexity, that is needed for various distributed tasks (e.g., Byzantine agreement [LF82, DR85, FM85], coin flipping [Cle86, MNS16], and zero-knowledge proofs [GK96, CKPR01]) under different security models. In this paper, we focus on the problem of general secure-multiparty-computation (MPC) in the challenging setting of *full security* (including guaranteed output delivery) with *maximal resiliency*. That is, even an active (aka Byzantine or malicious) adversary that controls a minority (up to half) of the parties should not be able to violate privacy or to prevent the honest parties from receiving a valid output. In this setting, originally presented in the classical work of Rabin and Ben-Or [RB89], we assume that each pair of parties is connected by a secure and authenticated point-to-point channel and that all parties have access to a common broadcast channel, which allows each party to send a message to all parties and ensures that the received message is identical.

The round complexity of honest-majority fully-secure MPC protocols was extensively studied. The lower-bound of [GIKR02, GLS15] shows that two rounds are insufficient for this task even when the parties are given access to a common reference string (CRS). In [AJL+12], a 5-round protocol was constructed based on Threshold Fully-Homomorphic Encryption (TFHE) and Non-Interactive Zero-Knowledge proofs (NIZK). An optimal round complexity of three, was later obtained by [GLS15] in the CRS model by relying on a stronger variant of TFHE that can be based on the learning with errors (LWE) assumption. Later in [BJMS20] the CRS was removed, and in [ACGJ18] LWE was replaced by weaker public-key primitives like general public-key encryption (PKE) and two-round witness indistinguishable proofs (Zaps). (The latter can be based on primitives like trapdoor permutations [DN07] and indistinguishability obfuscation [BP15], or on intractability assumptions related to bilinear groups [GOS12] and LWE [BFJ+20, GJJM20].)

The above results may give the impression that public-key assumptions are essential for honest-majority fully-secure MPC. However, if one puts no restriction on the round complexity, then, as shown by Rabin and Ben-Or [RB89], one can obtain unconditional results and no assumptions are needed at all! Specifically, every efficiently computable function can be securely computed with statistical security against computationally-unbounded adversaries.[1] Constant-round versions of this protocol are known either with an exponential dependency in the circuit-depth (or space-complexity) of the underlying function [IK00], or with computational security under the weakest-known cryptographic assumption: the existence of one-way functions [BMR90, DI05]. Moreover, for the special case of 3 parties (and single corruption), 3-round protocols were constructed by [PR18] based on injective one-way functions.

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

> **Q1:** Are public-key assumptions inherently needed for 3-round fully-secure honest-majority MPC? Is it possible to replace these assumptions with symmetric-key assumptions?

[1] Interestingly, perfect security is impossible to achieve in this setting as it requires a strong honest-majority of $2n/3$ [BGW88].

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

We answer these questions to the affirmative. We show that 3-round MPC with full security at the presence of honest-majority can be realized based on Minicrypt-type assumptions without relying on PKE, and present variants of our protocol that achieve *statistical everlasting security*. To the best of our knowledge, this is the first construction of everlasting-secure protocol in this setting *regardless of the underlying assumptions*. We continue with a detailed description of our results.

1.1 Our Contribution

1.1.1 Round-Optimal MPC in Minicrypt

We present the first 3-round general MPC protocol under Minicrypt assumptions. In fact, our protocol consists of 1 offline (input-independent) round, and 2 online rounds. To obtain our main result, we reveal a strong connection between round-optimal MPC and round-optimal protocols for functionalities whose output depends on the input of a single party, aka *single input functionalities* (SIF). In particular, we prove the following theorem.

Theorem 1.1. *Assuming the existence of non-interactive commitment scheme, there exists a compiler that takes a protocol* sif *with 1 offline round and 1 online round for single input functionalities, and outputs a protocol with 1 offline round and 2 online rounds for general MPC, with the same resiliency as* sif*.*

In a recent result by the same authors [AKP22], a round-optimal SIF protocol was presented based on the existence of injective one-way functions with sub-exponential hardness. The protocol has optimal resiliency when the number of parties n is logarithmic in the security parameter, and almost-optimal resiliency when the number of parties is polynomial in the security parameter. Since injective one-way function implies the existence of perfectly-binding non-interactive commitment scheme [Nao91], we obtain the following theorem by plugging the protocol of [AKP22] in Theorem 1.1.

Theorem 1.2. *Assuming the existence of injective one-way functions with sub-exponential hardness, for every* $\epsilon > 0$*, every efficiently-computable functionality can be realized in 1 offline round and 2 online rounds in the plain model, with full security against an active rushing adversary, under one of the following conditions.*

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

As mentioned in [AKP22], we can actually push the parameter ϵ to be as small as $\epsilon = \Omega(\frac{1}{\sqrt{\log \kappa}})$ where κ is the security parameter. In addition, [AKP22] show that optimal-resiliency for polynomially many parties can be obtained if one is willing to make stronger assumptions (e.g., random oracle or correlation intractable functions), or if the adversary is non-rushing.

1.1.2 Round-Optimal MPC with Everlasting Security in Minicrypt

The notion of statistical everlasting security [MU10] can be viewed as a hybrid version of statistical and computational security. During the run-time, the adversary is assumed to be computationally-bounded (e.g., cannot find collisions in the hash function) but after the protocol terminates, the adversary hands its view to a computationally-unbounded analyst who can apply arbitrary computations in order to extract information on the inputs of the honest parties.[2] This feature is one of the main advantages of information-theoretic protocols: after-the-fact secrecy holds regardless of technological advances and regardless of the time invested by the adversary.

We show that Theorem 1.1 yields a round-optimal MPC protocol with everlasting security when it is instantiated with statistically-hiding commitments and everlasting secure round-optimal SIF protocol. Such a SIF protocol was also realized in [AKP22] based on collision-resistant hash functions. Since the latter are known to imply statistically-hiding commitments [DPP98, HM96], we derive the following theorem.

Theorem 1.3. *Given access to a collision resistant hash function, every NC^1 functionality can be realized in 1 offline round and 2 online rounds, with full* everlasting *security against an active rushing adversary, under the same conditions of Theorem 1.2.*

Remark 1.4 (On the use of hash function)**.** *Similarly to the everlasting SIF protocol from [AKP22], 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, this set-up mechanism can be realized distributively by a single round of public-coin messages by letting each party sample randomness r_i that specifies a hash function h_i and then taking h to be the concatenated hash function [Her09]. This simple set-up protocol remains secure even against an active rushing adversary that may corrupt all the participants except for a single one. Alternatively, the choice of the hash function can be abstracted by a CRS functionality, or even, using the multi-string model of [GO14] with a single honestly-generated string. 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. Finally, even if one counts this extra set-up step as an additional round, to the best of our knowledge, our protocol remains the only known solution that achieves everlasting security, regardless of the underlying assumptions.*

Remark 1.5 (On NC^1 functionalities)**.** *All our everlasting-security protocols are restricted to NC^1. More generally, the computational complexity of these protocols grows exponentially with the depth or space of the underlying function. This is expected since even for strictly-weaker notions of security (e.g., passive statistical security against a single corrupted party), it is unknown how to construct* efficient constant-round *protocols for functions beyond NC^1 and log-space. (In fact, this is a well-known open problem that goes back to [BFKR90].)*

The difference between everlasting and computational security is *fundamental* and is analogous to the difference between statistical commitments and computational commitments or statistical ZK arguments vs. computational ZK arguments (see, e.g., the discussions in [BCC88, NOVY98]). In both the former cases, we get computational security against "online cheating" and statistical security against after-the-fact attacks.

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

We note that all previous protocols inherently fail to achieve everlasting security. Indeed, for technical reasons (that will be discussed later in Section 2), previous constructions emulate private channels over a broadcast channel via the use of PKE. Furthermore, the (encrypted) information that is delivered over this channel fully determines the inputs. Thus, an analyst that collects the broadcast messages and later breaks the secrecy of the PKE (e.g., via brute-force) can learn all the private inputs of the parties.

1.1.3 Round-Optimal MPC for $t < n/3$ with Everlasting Security from OWF

For strong honest-majority, where $t < n/3$, we provide a 3-round protocol for general MPC with everlasting security *in the plain model*, from the minimal assumption of one-way functions. This protocol is round-optimal by the lower bound of [GIKR02].

Theorem 1.6. *Assuming the existence of one-way functions, every NC^1 functionality can be realized in the plain model by a 3-round protocol that provides* everlasting security *against an active rushing adversary corrupting $t < n/3$ of the parties. If we are willing to compromise to computational-security, we obtain a secure protocol for every efficiently computable functionality.*

Known round-optimal protocols in this regime, all appear in [AKP20a], either achieve (1) statistical-security but with running time exponential in n, or (2) everlasting-security from collision resistant hash-functions and a CRS as a trusted setup, or (3) computational-security from injective one-way function in the plain model. Therefore, our construction can be seen as the first round-optimal construction that efficiently achieves some form of security against unbounded adversaries in the plain model. Moreover, it does so only based on one-way functions. As a primary tool, we design a verifiable secret sharing (VSS) with everlasting security in 2 rounds from OWFs. Known VSS protocols in this regime either achieve (1) statistical-security but with running time exponential in n [AKP20a] with $t < n/3$, (2) everlasting-security from collision resistant hash-functions and a CRS as a trusted setup with $t < n/2$, or (3) computational-security from non-interactive commitments schemes with $t < n/2$.

1.1.4 Summary of the Results

We summarize our results in the honest-majority regime in Table 1 and compare them to the existing results. In Table 2 we summarize our results in the strong honest-majority regime, and compare them to the existing results.

Previous unpublished version and a sibling paper. A previous version of this paper contained a weak form of some of the current results together with 2-round SIF protocols based on the Fiat-Shamir heuristic. The SIF protocols were strengthened and were fully moved to [AKP22], and the derivation of the 3-round MPC protocols was significantly changed and modularized, leading to the new compiler (Theorem 1.1). Theorem 1.6 is also new and did not appear in previous versions. Overall, the current version of this writeup and [AKP22] contain a disjoint sets of results that together fully subsume the previous versions of this paper.

Ref.	Rounds	Threshold	Setup Plain / CRS	Security it / es / cs†	Cryptographic Assumptions
[RB89]	circuit-depth	$t < n/2$	Plain	it	–
[IK00]⋆	constant > 3	$t < n/2$	Plain	it	–
[BMR90, DI05]	constant > 3	$t < n/2$	Plain	cs	OWFs
[PR18]	3	$n = 3, t = 1$	Plain	cs	injective OWFs
[GLS15]	3	$t < n/2$	CRS	cs	threshold multi-key FHE
[BJMS20]	3	$t < n/2$	Plain	cs	LWE
[ACGJ18]	3	$t < n/2$	Plain	cs	PKE, Zaps
This	3	$t < n(\frac{1}{2} - \epsilon)^{\S}$	Plain	cs	sub-exponential injective OWFs
This⋆	3	$t < n(\frac{1}{2} - \epsilon)^{\S}$	CRS	es	collision resistant hash functions

† it: information-theoretic, es: everlasting security, cs: computational security.
⋆ For NC^1 circuits
§ We achieve $t < n/2$ when n is logarithmic in the security parameter.

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

Ref.	Rounds	Threshold	Setup Plain / CRS	Security it / es / cs†	Cryptographic Assumptions	Complexity in terms of n
[AKP20a]⋆	3	$t < n/3$	Plain	it	–	Exponential
[AKP20a]	3	$t < n/3$	Plain	cs	injective OWFs	polynomial
[AKP20a]⋆	3	$t < n/3$	CRS	es	collision-resistant hash-functions	polynomial
This⋆	3	$t < n/3$	Plain	es	OWFs	polynomial
This	3	$t < n/3$	Plain	cs	OWFs	polynomial

† it: information-theoretic, es: everlasting security, cs: computational security.
⋆ For NC^1 circuits

Table 2: Comparison of our work with the state-of-the-art relevant results for $t < n/3$

2 Technical Overview

In this section, we give a detailed overview of our constructions while emphasizing the main novelties. Section 2.1 is devoted to the proof of the main theorem (Theorem 1.1) and Section 2.2 is devoted to the strong honest-majority result (Theorem 1.6). Throughout, we assume that there are n parties, $P_1, \ldots, P_n$, of which at most t are corrupt, where we assume two settings: $t < n/2$ for Section 2.1 and $t < n/3$ for Section 2.2. We assume that the parties communicate over secure point-to-point channels and over a broadcast channel.

2.1 Main Theorem

Following previous works [GLS15, ACGJ18], we prove our main Theorem 1.1 by using the following outline: (1) We start with a 2 round protocol Π^{sm} with security against *semi-malicious* adversary that is allowed to choose its input and randomness, but other than that plays honestly; (2) We upgrade the security of the protocol to hold against a *first-round fail-stop* adversary that, in addition to choosing its input and randomness, is allowed to abort a corrupted party during the first round of the protocol; (3) We compile the protocol to a new protocol with an extra offline round that achieves security against a *fully fail-stop* adversary that is allowed to abort a corrupted party at any round; (4) We transform the protocol for fail-stop adversaries to a protocol for malicious adversaries. Jumping ahead, previous constructions employed Zaps/NIZK for the last step and PKE/threshold homomorphic encryption both for steps (3) and (4). We will show how to relax these assumptions.

The initial protocol Π^{sm}. Our starting point is a perfectly-secure 2-round protocol Π^{sm} for a rushing semi-malicious adversary that corrupts a minority of the parties. Such a protocol appears in [ABT18] and is fully described in Section 4. The first round of the protocol consists only of private messages, and the second round consists of broadcast messages. (In fact, using standard techniques we can transform any 2-round protocol to a protocol that satisfies this property, see e.g., [GIKR01].) We denote the first-round private message from P_i to P_j by a_{ij}, and the second-round broadcast of P_i by b_i.

2.1.1 Coping with First-Round Aborts

Roughly speaking, when an adversary aborts, we let the other parties emulate his role for the remaining rounds. The emulation is relatively simple when the abort happens in the first round of Π^{sm} since the parties have a chance to respond to the abort in the second round. Specifically, suppose that P_i aborts in the first round. Then the other parties face 2 problems: (1) P_i did not send her first round messages; and (2) the first-round messages that were *directed* to P_i were lost and will be missing later during the reconstruction of output. The first issue is solved by letting each party to locally generate the outgoing messages of P_i by running P_i on the all-zero input and the all-zero random tape.[3] To solve the second issue, we modify the protocol so that each first round message from P_j to P_i is also being shared among all other parties. That is, in the first round, every P_j shares each of its first-round outgoing messages $a_{j1}, \ldots, a_{jn}$ via Shamir's secret sharing, using degree-t polynomials. If P_i aborts during the first round then in the second round, the parties reconstruct all the 1st round incoming messages of P_i. After the second round, the parties have enough information to locally continue the emulation of P_i (with respect to the all-zero inputs) and generate her second round broadcast messages. We note that in previous works (e.g., [ACGJ18]) first-round aborts are handled differently by adding an additional "function-delayed" requirement on the initial protocol Π^{sm}.

2.1.2 Coping with Second-Round Aborts

Second-round aborts are trickier to handle: When the honest parties send their second-round messages, they do not know which other parties are about to abort. Accordingly, one has to support "silent emulation", that is, any subset of $n - t$ second-round messages should suffice for emulating all other second-round messages. The implementation of this mechanism employs heavy tools (threshold homomorphic encryption in [GLS15] and PKE plus garbled circuits in [ACGJ18]) and

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

requires an additional offline round. We review these ideas and present an information-theoretic variant of them.

Ananth et al. [ACGJ18] (ACGJ) first use PKE to ensure that all the communication between the parties will be over the broadcast channel. That is, in a preprocessing round (denoted Round 0), every P_i generates keys $(\mathsf{pk}_i, \mathsf{sk}_i)$ for PKE, and broadcasts pk_i. In the following rounds, the private channel from P_j to P_i is emulated by letting P_j broadcast her message encrypted under the public key pk_i of P_i. After this modification, we can write the second-round message of party P_i as a function f_i that given

(1) the encrypted messages $(A_{ji})_{j\in\{1,\ldots,n\}}$ that P_i receives in Round 1,

(2) the input $\mathbf{x}(i)$ and randomness r_i of P_i in the simulation of Π^{sm}, and

(3) the secret key sk_i,

outputs the public broadcast message b_i that P_i sends in the second round. (That is, f_i decrypts the messages A_{ji} using sk_i in order to obtain a_{ji}, and then computes the second round broadcast b_i of P_i in Π^{sm} based on $(\mathbf{x}(i), r_i, (a_{ji})_{j\in\{1,\ldots,n\}})$.) Observe that f_i depends on private inputs (items 2, 3) and on some public values (item 1) that will be broadcasted during the first round. The key observation is that the private inputs are already known before the first round begins. This fact will be exploited to delegate the computation of f_i.

Specifically, at the beginning of the first round, we let every P_i generate a *garbled circuit* for a function f_i. During the first round, P_i broadcasts the garbled circuit together with the labels of $(\mathbf{x}(i), r_i)$ and sk_i. In addition, P_i secret-shares all the labels that correspond to *every potential* ciphertext value $(A_{ji})_{j\in[n]}$. The actual ciphertexts, $(A_{ji})_{j\in\{1,\ldots,n\}}$, are broadcasted concurrently during the first round by the corresponding parties, and so, in the second round, all the non-aborted parties publish the shares of the corresponding labels. Consequently, after this round, everyone can recover the correct labels via secret reconstruction of the secret sharing, and hence obtain the broadcast b_i of P_i. To make the proof go through, ACGJ assume that the garbled circuit is *adaptively* private [HR12] in the sense that privacy holds even if the adversary first gets to see the garbled circuit, and only then chooses the inputs to the circuit and receive the corresponding labels.

We note that the same approach can be applied without relying on any computational assumptions. First, instead of using PKE, we let the parties exchange one-time pads during the offline round. That is, in Round 0 we let every P_i sample random pads $\eta_i = (\eta_{i1}, \ldots, \eta_{in})$ and send the pad ("key") η_{ij} to P_j by using a *private channel*. Now a first-round message a_{ji} from P_j to P_i can be broadcasted in an encrypted form $A_{ji} := a_{ji} + \eta_{ij}$. (For technical reasons that will be explained later, we encrypt the message under the receiver's key.) The garbled circuits can also be instantiated with an information-theoretic garbled circuits, aka perfect randomized encodings. (The second-message function of Π^{sm} is "simple enough" to allow such a realization.) Furthermore, we avoid the need for adaptive garbled circuits, by sharing the garbled circuit together with the labels of $(\mathbf{x}(i), r_i)$ and η_i among all the other parties; these shares are later revealed during the second round.[4]

[4] In fact, in order to handle second-round aborts *together* with first-round aborts, we need to slightly modify the function f_i. See Section 4 for full details.

2.1.3 From Fail-Stop to Malicious Adversary

To obtain a protocol with security against a malicious adversary, we follow the GMW paradigm and ask each party to prove in zero-knowledge that she followed the protocol. Ignoring for now the exact details of the zero-knowledge proof, the basic idea is that a malicious deviation from the protocol will be caught due to the soundness properties of the proof, and will be treated as if the cheater aborted the computation. Crucially, here too one must assume that the underlying protocol works over a *broadcast* channel. As discussed in ACGJ, if the underlying semi-malicious protocol uses private channels, then a party may need to prove different statements to different parties in order to establish honest behavior, which may lead to inconsistent views regarding her "abort" status. Indeed, [GLS15, ACGJ18] make here another use of PKE in order to make sure that the protocol's messages are delivered over a broadcast channel. In fact, this usage of PKE dates back to the GMW compiler [GMW87].

Generating public committing transcript. We can use the previous maneuver to shift all private messages to Round 0 via one-time pads, however, the resulting protocol is still not ready for "zero-knowledge compilation". Indeed, even if we add a zero-knowledge layer, the adversary can cheat either by "claiming that she received different messages" (i.e., changing the keys that correspond to her incoming messages) or by "claiming that she sent different messages". Intuitively, the problem is that our information-theoretic solution is non-committing. We solve this problem via the use of non-interactive commitment (NICOM). Details follow.

In the preprocessing round (Round 0), we let each party P_i broadcast a vector of commitments, $(C_{i1}, \dots, C_{in})$ to all her private keys, $(\eta_{i1}, \dots, \eta_{in})$, for the one-time pads, and send o_{ij}, the opening of C_{ij}, to P_j over the private channel. In addition, we let all parties commit to their inputs and randomness for the fail-stop protocol in Round 1 just like in the standard GMW transform. (We emphasize that Round 0 is still input-independent.) Next, we employ some zero-knowledge primitive (to be discussed below) to prove that a party P_i computes a message properly with respect to the public commitments. Specifically, in the first round party P_i can prove that the garbled circuit for f_i was generated properly with respect to his committed randomness, committed input, and with respect to the one-time keys, $\eta_{1i}, \dots, \eta_{ni}$, that he received from all other parties in the preprocessing round. For the last part we exploit the fact that P_i also received a witness, o_{ji}, that connects the keys to their commitments.

This approach almost works. The only problem is that a party P_j may cheat in Round 0 by sending to P_i a "bad" pair of key/opening (η_{ji}, o_{ji}) that are inconsistent with the public commitment C_{ij}. Fortunately, there is a simple round-efficient solution: If the key is malformed, we simply send the messages from P_i to P_j in the clear un-encrypted. Formally, in Round 1, P_i broadcasts a list L_i of all parties that sent *invalid* openings in Round 0. If P_i needs to send a private message a_{ij} to a party P_j according to Π^{sm}, for $P_j \notin L_i$, then P_i simply sends the encrypted message $a_{ij} + \eta_{ji}$ over the broadcast channel. For a party $P_j \in L_i$, we simply let P_i send the message a_{ij} *unencrypted* over the broadcast channel. We also use the same mechanism for additional private messages that the parties have to exchange, that are not necessarily a part of the protocol Π^{sm} (e.g., sending private shares for the garbled circuit). As before, we only use encryption in Round 1, while Round 2 consists only of public unencrypted messages. This modification does not violate privacy since messages from P_i to P_j will be sent unencrypted only if one of these parties is corrupted, which means that the adversary is supposed to learn the message anyway.

Instantiating the zero-knowledge layer. Finally, we have to instantiate the zero-knowledge layer in a round-preserving way. Previous works either make use of NIZK at the expense of adding a CRS [AJL$^+$12, GLS15] or exploited the offline round to set-up some multi-party variant of ZK [GOS12, ACGJ18]. In terms of assumptions both approaches rely on NIZK/Zaps which are known to be equivalent assuming one-way functions [DN07]. We strongly exploit the existence of honest majority, and observe that these primitives can be replaced by a SIF protocol. Given a relation R, define the single input functionality that (1) takes the statement x and witness w from the prover, and (2) if $R(x, w) = 1$ it returns x to all parties, and if not, it returns a failure symbol $\perp$ to all parties. We can therefore realize a round-efficient variant of multi-verifier zero-knowledge proof (MVZK) based on SIF with 1 offline round and 1 online round. We emphasize that the security of SIF protocols is formulated via an MPC-based definition by relating the protocol to an *ideal SIF functionality*. This leads to security guarantees that are stronger than those achieved by standalone versions of the MVZK primitive (e.g., the SIF protocol provides *knowledge-extraction*).

Summary. Overall, the SIF is being employed as follows. In Round 0, the parties execute the offline round of the SIF protocol, exchange one-time pads and publish their commitments. In Round 1, we let every P_i commit to its input and randomness, and let P_i prove via SIF that (1) for every $P_j \notin L_i$, the public encrypted message from P_i to P_j is consistent with the committed input and randomness of P_i, and it is encrypted with the committed random pad η_{ji}; (2) for every $P_j \in L_i$, the public unencrypted message from P_i to P_j is consistent with the committed input and randomness of P_i. Similarly, in Round 2 every P_i proves via SIF that its public broadcast is consistent with (1) its committed input and randomness; (2) the unencrypted public incoming message from P_j, for every P_j for which $P_i \in L_j$; and (3) the decrypted incoming message from P_j, where the decryption used the committed random pad η_{ij}, for every P_j for which $P_i \notin L_j$.

Remark 2.1 (Everlasting security)**.** *All the components, except for the NICOM and SIF, are information-theoretic. As a result, we derive the everlasting security version of the protocol by plugging-in NICOM and SIF with everlasting security guarantees. The protocol remains the same and the proof of security is given in a unified way.*

Remark 2.2 (Reusing the preprocessing round)**.** *Recall that the preprocessing round consists of exchanging committed one-time pads, and initializing the SIF protocol. If one does not care about everlasting security, the one-time pads can be replaced with (committed) pairwise private-keys for a symmetric encryption scheme, and in this case the same keys can be used for many invocations of the protocol. Under this modification, we can reuse the preprocessing step (Round 0) or even treat it as a private-key infrastructure provided that the preprocessing step of the SIF is also reusable. While the construction from [AKP22] does not satisfy this property, other SIF constructions (e.g., based on NIZK) can be used to achieve this property. We remark that, even if one employs NIZK-based SIF, our approach is beneficial since it bypasses the need for PKE. Indeed, the Fiat-Shamir heuristic [FS86] suggests that NIZK can be based on strong symmetric-key assumptions like correlated robust hash functions [CGH04], and may not require PKE-based assumptions. (See [CCH$^+$19] for further discussion and references).*

2.2 Strong Honest-majority MPC with Everlasting Security from OWF

We continue with an overview of the 3-round MPC protocol that provides everlasting security in the plain model for strong honest-majority, $t < n/3$. In [AKP20a] it is shown that such a protocol follows from a 2-round protocol for *verifiable secret sharing* (VSS) that provides everlasting security. We design such a protocol based on digital signatures (that are equivalent to one-way functions).

The VSS functionality. We will need the following variant of VSS. The functionality receives a symmetric bivariate polynomial $F(x, y)$ of degree at most t in each variable from a distinguished party D, called the *dealer*, and delivers to each party P_i the univariate polynomial $f_i(x) := F(x, i)$. The use of symmetric bivariate polynomials can be seen as an extension of the standard Shamir's t-out-of-n secret sharing, that allow us to make a consistency-check between any pair of parties P_i and P_j, since $f_i(j) = F(j, i) = F(i, j) = f_j(i)$.

2-round VSS protocol. In the first round, we let D generate a signature-key and a verification-key for a digital signature scheme, and broadcast the verification-key. In addition, we let D send $f_i(x)$ to P_i, together with a signature on each point $f_i(1), \ldots, f_i(n)$.[5] At the end of the first round, a party is *happy* with D if all the signatures it received are valid, and it is *unhappy* with D otherwise. Observe that if D is honest then all honest parties are happy. The second round of the protocol consists of (1) consistency check for happy parties, and (2) public recovery of the shares of unhappy parties. We continue by discussing the consistency check.

2.2.1 Consistency Check

The goal of the consistency check is to ensure that (a) there are at least $t + 1$ happy honest parties, and that (b) all of them are consistent with each other, i.e., $f_i(j) = f_j(i)$ for every happy and honest P_i and P_j. Looking forward, this will imply that the shares of the happy honest parties fully determine a symmetric bivariate polynomial $F(x, y)$ of degree at most t in each variable, where for an honest D the polynomial $F(x, y)$ is the input polynomial of D.

It is not hard to achieve (a). In Round 2, each party declares, via broadcast, whether she is happy or not, and we discard the dealer if there are more than t unhappy parties. This guarantees that an honest dealer will never be discarded (since all honest parties are happy) and a corrupt dealer must gain the support of at least $(n - t) - t \geq t + 1$ happy honest parties in order to remain undiscarded.

2-wise consistency via Reveal-if-not-equal gadget. Pair-wise consistency (item b) is being handled via a special comparison gadget that takes from each pair of happy parties (P_i, P_j) the points $m_A = f_i(j), m_B = f_j(i)$ and their corresponding signatures s_A, s_B, and broadcasts an equality bit that indicates whether $m_A = m_B$ and in case of inequality releases the points and their signatures (m_A, s_A, m_B, s_B). When P_i and P_j are honest, a disagreement accompanied with valid signatures certifies that D is corrupted. Of course, when $m_A = m_B$, we do not want any information about m_A, m_B to be revealed to the other parties. If 3 rounds are allowed then we can easily realize the gadget by letting P_i and P_j compare their values privately on the second round (by exchanging messages over the private channel) and then announcing the result at the next round. We avoid this overhead by making an additional observation: When one of the parties, say P_i, is corrupt we do not care about the privacy nor the correctness of the gadget. Privacy does not matter since

[5] In fact, in order to put the signatures in context, we let D sign the *tuples* $(i, j, f_i(j))_{j \in \{1,\ldots,n\}}$, instead of just the field elements $f_i(1), \ldots, f_i(n)$.

the adversary already knows $m_B = f_j(i)$. As for correctness, even if the "gadget misbehaves", an honest dealer is protected against a disqualification by the security of the signatures.

We realize the gadget with the aid of garbled circuits (or perfect randomized encodings). Let g be a function that takes (m_A, m_B, s_A, s_B), returns 1 if $m_A = m_B$, and returns (m_A, m_B, s_A, s_B) otherwise. In the first round, we let Alice (P_i) generate a garbled circuit G for g, and send the randomness used to generate G to Bob (P_j). In the second round, Alice broadcasts G, together with the labels corresponding to her inputs in G, and Bob broadcasts the labels corresponding to his inputs in G. It is not hard to see that the properties of the protocol follow directly from the correctness and security of the garbled circuit. Based on this gadget, after the second round everyone learns whether Alice and Bob are in agreement, and, in case they disagree, whether the dealer should be discarded due to a conflicting pair of valid signatures. If the dealer was not discarded in any consistency check of a pair (P_i, P_j), we conclude that all happy honest parties are consistent.

2.2.2 Handling Unhappy Parties

It remains to explain how to help unhappy (honest) parties to recover a share that is consistent with all the happy honest parties. The main idea is to let every unhappy P_i ask from every other P_j to publicly reveal all the common information, i.e., the value $f_j(i)$ and the corresponding signature. Since we have only 1 additional round, we design an additional gadget with 1 offline round and 1 online round similarly to the reveal-if-not-equal gadget.[6] In this gadget, Alice inputs a bit flag_A, while Bob inputs some secret s_B. When Alice and Bob are honest, if $\mathsf{flag}_A = 0$ then the listeners learn no information about s_B, while if $\mathsf{flag}_A = 1$ they learn s_B. As before, when one of the parties is corrupt there are no security guarantees.

We use this mechanism for every pair (P_i, P_j), where P_i takes the role of Alice and P_j takes the role of Bob. We let P_i input $\mathsf{flag}_A = 1$ if P_i is unhappy, and $\mathsf{flag}_i = 0$ otherwise; in addition, P_j sets s_B to be the share $f_j(i)$ together with the corresponding signature. Observe that if both P_i and P_j are honest and happy, then the adversary learns no information about their common point; however, if P_i is unhappy and P_j is happy, then *all the parties* learns the point $f_j(i)$ together with a valid signature.

An honest unhappy P_i will be able to reveal all evaluations $f_j(i)$ from happy honest parties P_j, together with valid signatures. We let *all* parties interpolate over all values whose corresponding signatures were valid, in order to obtain $f_i(x)$. Since there are at least $t+1$ happy honest parties, we are promised that $f_i(x)$ is either consistent with the polynomial $F(x, y)$ defined by the shares of the happy honest parties, or has degree more than t, in which case *all* the parties reject the dealer. Finally, for an honest D and a corrupt unhappy P_i, the values that are revealed with valid signatures must be consistent with $F(x, y)$, so the interpolated polynomial will have degree at most t, and D will not be discarded.

3 Preliminaries

We denote by κ the security parameter, by n the number of parties, and by t an upper bound on the number of corrupt parties. We consider two main settings: the optimal resiliency setting where $n \geq 2t+1$, and the almost optimal resiliency setting where $n \geq (2+\epsilon)t$ for an arbitrarily

[6] In fact, in our construction we merge the two gadgets.

small constant $\epsilon > 0$. We denote by C the set of corrupt parties, and by $\mathsf{H} = \{1, \dots, n\} \setminus \mathsf{C}$ the set of honest parties. We have $|\mathsf{C}| \leq t$. We let $\mathbb{F}$ be a finite field of size at least $n+1$, and, with some abuse of notation, let $1, \dots, n$ denote n distinct non-zero field elements. All our results are proved in the UC-framework. For more information, see Section A.1.

Our building blocks are non-interactive commitment scheme (NICOM), secret sharing scheme, and randomized encoding, all presented in Section 3.1. An additional ingredient of our construction is the protocol of [AKP22] for the computation of single-input functionalities, which we recall in Section 3.1.4.

3.1 Building Blocks

3.1.1 Non-Interactive Commitment Scheme (NICOM)

Definition 3.1 (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* $x' \in \{0,1\}^* \cup \{\perp\}$*. The symbol* $\perp$ *indicates a "failed openning".*

- *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*

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

  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.

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 [Blu81, Yao82, GL89] or even on standard one-way functions and worst-case derandomization assumptions [BOV03], and type (2) commitments can be based on collision resistance hash functions [DPP98, HM96]. In the former case, no public parameters are needed and we think of pp as an empty string. 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 yields computational protocols and type (2) instantiation yields protocols with everlasting security. The proofs typically treat both notions in a unified way with minor adaptations when needed.

Some variants of the SIF protocol from [AKP22] rely on perfectly-binding NICOM whose computational hiding property holds for $\epsilon \leq 2^{-\kappa}$, hereafter referred to as *sub-exponentially hiding* NICOM. The existence of such a (plain-model) NICOM follows from the existence of an injective OWF over m-bit inputs that cannot be inverted by a PPT adversary with probability better than $2^{-m^{\delta}}$ for some universal constant $\delta > 0$. Under worst-case derandomization assumptions [BOV03], such NICOMs can be based on general (not necessarily injective) sub-exponentially hard OWFs. These stronger variants are widely used, and are needed only for the SIF protocol and not for the reductions presented in this paper.

3.1.2 Secret Sharing

A central tool in our construction is Shamir's secret sharing scheme [Sha79]. We assume that the reader is familiar with Shamir's secret sharing scheme, and refer the interested reader to [AL17, Section 3]. We will use the following notation.

Notation 1. *We say that a party P_i shares a value s via degree-d polynomial if P_i samples a random degree-d polynomial $p(x)$ with $p(0) = s$, and gives party P_i the value $p(i)$. Sometimes we simply say that P_i shares a value s, which means that P_i shares s via degree-t polynomial. Similarly, we say that a party P_i computes the Shamir's shares of a values s, if P_i samples a degree-t polynomial $p(x)$ with $p(0) = s$, and computes the values $s_1, \ldots, s_n$, where $s_i := p(i)$.*

3.1.3 Randomized Encoding

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

Definition 3.2 (Perfect randomized encoding [IK00, AIK06])**.** *Let $f : X \to Y$ be a function. We say that a function $\hat{f} : X \times R \to Z$ is a* perfect randomized encoding *of f if there exists a pair of randomized algorithms, decoder* dec *and simulator $\mathcal{S}$, for which the following hold:*

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

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

Definition 3.3 (Perfect decomposable randomized encoding)**.** *Assume that the function f is an arithmetic function whose input $x = (x_1, \ldots, x_n)$ is a vector of elements of some ring X. We say that a randomized encoding $\hat{f}$ is a* decomposable *randomized encoding if each output of $\hat{f}$ depends on at most a single input x_i. Namely, $\hat{f}$ decomposes to $(\hat{f}_1(x_1; r), \ldots, \hat{f}_n(x_n; r))$, where $\hat{f}_i$ might output several ring elements.*

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

Definition 3.4 (2-decomposable randomized encoding)**.** *Let $f : X \to Y$ be a function where $X = X_1 \times X_2$. A randomized encoding $\hat{f}$ of f is* 2-decomposable *(also known as* 2-party private simultaneous messages*) if $\hat{f}(x_1, x_2; r)$ decomposes to $(\hat{f}_1(x_1; r), \hat{f}_2(x_2; r))$.*

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

Theorem 3.5. *There exists an efficiently constructible perfect decomposable randomized encoding for the class of polynomial-size arithmetic circuits with logarithmic depth over an arbitrary ring. In particular, there is a compiler that takes a size-S depth-D arithmetic circuit for f and in time $\mathrm{poly}(S, 2^D)$ outputs arithmetic circuits for $\hat{f}$, dec and $\mathcal{S}$.*

3.1.4 Single-Input Functionality

A *single-input functionality* $\mathcal{F}$ is a functionality that receives its input from a single party, called the *dealer*. In this work we always assume that the output is public, which means that all parties receive the same output. More formally, the functionality $\mathcal{F}$, which is parametrized by a function $f : \{0,1\}^* \to \{0,1\}^*$, receives from the dealer an input $\mathbf{x}$, computes $\mathbf{y} = f(\mathbf{x})$, and returns $\mathbf{y}$ to all the parties. In [AKP22] it was proved that general SIF can be realized in 1 offline round and 1 online round, with optimal resiliency when the number of parties is small (i.e., logarithmic in the security parameter), and with almost-optimal resiliency for a large number of parties. This is summarized in the following theorems.

Theorem 3.6 (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 3.7 (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.

4 MPC with (Almost) Honest-majority from Minicrypt

Our starting point is the following completeness theorem of [AKP20b].

Proposition 4.1 ([AKP20b]). *Let $\mathcal{G}$ be an n-party functionality that can be computed by a Boolean circuit of size S and depth D and let $\mathbb{F}$ be an arbitrary extension field of the binary field $\mathbb{F}_2$. Then, the task of securely-computing $\mathcal{G}$ non-interactively reduces to the task of securely-computing a degree-2 n-party functionality $\mathcal{F}$ over $\mathbb{F}$.*

The reduction preserves active perfect-security (resp., statistical-security) with resiliency threshold of $\lfloor \frac{n-1}{3} \rfloor$ (resp., $\lfloor \frac{n-1}{2} \rfloor$) and the complexity of the function $\mathcal{F}$ and the overhead of the reduction is $\mathrm{poly}(n, S, 2^D, \log |\mathbb{F}|)$. Furthermore, assuming one-way functions, one can get a similar reduction that preserves computational-security with resiliency threshold of $\lfloor \frac{n-1}{2} \rfloor$ and complexity/security-loss of $\mathrm{poly}(n, S, \log |\mathbb{F}|)$.

Therefore, our goal here is to provide a 3-round protocol for degree-2 computation $\mathcal{F}$ over some finite field $\mathbb{F}$ which, by default, is taken to be some binary extension field of size $n + 1 \leq |\mathbb{F}| \leq \mathrm{poly}(n)$.[7] We assume, without loss of generality, that $\mathcal{F}$ is a public-output functionality that delivers the same output to all the parties. Formally, the functionality $\mathcal{F}$ takes as an input m field elements $x^1, \ldots, x^m \in \mathbb{F}$ and delivers to all the parties L degree-2 polynomials over $(x^1, \ldots, x^m)$. For every party P_i, we denote by $I_i \subseteq \{1, \ldots, m\}$ the set of all indices j such that P_i holds the input x^j, and let $\mathbf{x}(i) := (x^j)_{j \in I_i}$ denote the inputs of P_i. As explained in Section 2, our starting point is a 2-round perfectly-secure protocol Π^{sm} against rushing semi-malicious adversaries. For concreteness, we take the protocol of [ABT18] as Π^{sm}. The protocol is presented in Figure 1 for the case where the output of $\mathcal{F}$ consists of a single output $f(x^1, \ldots, x^m)$. The extension to a multi-output function can be obtained in a straightforward way. (More generally, semi-malicious security is closed under parallel repetitions.)

Protocol Π^{sm}

Round 1: Each party shares each of its inputs via Shamir's secret sharing, using degree-t polynomials. In addition, each party shares the value 0 via Shamir's secret sharing, using degree-$2t$ polynomials. Denote the shares that P_i received by $x_i^1, \ldots, x_i^m, z_i^1, \ldots, z_i^n$, where z_i^j is the i-th share of the zero-sharing of P_j.

Round 2: Every P_i computes $w_i := f(x_i^1, \ldots, x_i^m) + z_i^1 + \ldots + z_i^n$ and broadcasts w_i. After the second round, the parties locally interpolate the shares $w_1, \ldots, w_n$ in order to obtain a degree-$2t$ polynomial $W(x)$, and output $W(0)$.

Figure 1: Protocol Π^{sm}

Building on the overview presented in Section 2, here we devote to the remaining finer details. We think of every set L_i as an n-bit string, whose j-th bit, denoted $L_i[j]$, is 1 if $P_j \in L_i$, and is 0 otherwise. We continue by presenting the function f_i that will be used to compute the Round 2 broadcast message of P_i, for every $i \in \{1, \ldots, n\}$.

The function f_i. The function f_i receives the following 4 inputs:

(1) the bits $L_1[i], \ldots, L_n[i]$, indicating whether P_i is in $L_1, \ldots, L_n$,

[7] More generally, our results apply to every finite field that supports pairwise-multiplication and n-wise addition in NC^1. This assumption holds for binary extension fields as well as for prime-order fields $\mathbb{F}_p$ that are defined over a prime p of polynomially-bounded bit length [BCH86]. See also [HV06] for a discussion on the complexity of field arithmetics.

(2) messages $A_i = (A_{ji})_{j \in \{1,\ldots,n\}}$ that P_i receives in Round 1 over broadcast channel,

(3) the input $\mathbf{x}(i)$ and randomness r_i of P_i in the simulation of Π^{sm}, and

(4) pads $\eta_i = (\eta_{i1}, \ldots, \eta_{in})$.

For every P_j with $L_j[i] = 0$ the function sets $a_{ji} := A_{ji} - \eta_{ij}$. Otherwise, it sets $a_{ji} := A_{ji}$. Then, the function computes Round 2 broadcast b_i of P_i in Π^{sm} given $(\mathbf{x}(i), r_i, (a_{ji})_{j \in \{1,\ldots,n\}})$. The output of the function f_i is b_i.

The four inputs of f_i will be viewed as strings of bit-lengths ℓ_1, ℓ_2, ℓ_3 and ℓ_4, respectively. (Observe that $\ell_1 = n$.) We let $\ell := \ell_1 + \ell_2 + \ell_3 + \ell_4$ denote the total bit-length of the inputs to f_i. For simplicity (and by possibly padding the inputs) we may assume that the input lengths are uniform across all the f_i's. It can be verified that the function f_i can be implemented by a Boolean NC^1 circuit[8], so, by Theorem 3.5, it has an efficient perfect decomposable randomized encoding, which we denote by $\hat{f}_i := (\hat{f}_{i1}, \ldots, \hat{f}_{i\ell})$.

Randomized encoding of f_i. In Round 1, we let P_i sample randomness for $\hat{f}_i$. The public inputs **(1)–(2)** are known to all the parties at the end of Round 1. Therefore, for every input-bit v that corresponds to the public inputs (that is, $1 \le v \le \ell_1 + \ell_2$), we let P_i compute the output of $\hat{f}_{iv}$ both on input 0 and input 1, and share the outputs among the parties. In this way, the parties will be able to recover the correct output of $\hat{f}_i$ in Round 2.

In addition, the private inputs **(3)–(4)** are known to P_i already at the beginning of Round 1. Therefore, we let P_i compute the output of $\hat{f}_{iv}$ for every input-bit v that corresponds to the private inputs (that is, $\ell_1 + \ell_2 + 1 \le v \le \ell$). Instead of broadcasting those inputs already in Round 1, we let P_i share those outputs among the parties, and in Round 2 we let the parties recover those values. (This step makes sure an adversary cannot make an adaptive choice for the public inputs **(1)–(2)** and hence our randomized encoding scheme need not be adaptive, unlike the garbling scheme of [ACGJ18].)

Tolerating fail-stop adversaries. Using the ideas developed so far, we present an intermediate protocol Π^{fs}, that can tolerate fail-stop adversaries and works as a stepping stone for our final protocol.[9] Our protocol Π^{fs}, presented in Figure 2, when augmented with the zero-knowledge proofs, give rise to the final protocol.

Protocol Π^{fs}

Round 0: In a preprocessing round, each P_i does the following

- Sample a random pad ρ_{ij} for every $j \in \{1, \ldots, n\}$.
- Compute the commitments and openings $(C_{ij}, o_{ij}) \leftarrow \mathsf{commit}_{\mathsf{crs}}(\rho_{ij})$ for $j \in \{1, \ldots, n\}$. We parse ρ_{ij} as $\rho_{ij} = (\rho_{ij}[1], \rho_{ij}[2], \rho_{ij}[3])$ and use each component to pad a different part of the message. We let $\eta_{ij} := \rho_{ij}[1]$ denote the pad that will be used for a_{ij}.

[8] For this, we have to assume the underlying field operations are in NC^1 as discussed in Footnote 7.

[9] The protocol slightly deviates from the fail-stop protocol described in Section 2.1, since it already uses *committed* random pads. This will simplify the presentation of the final protocol.

- Broadcast $(C_{ij})_{j\in\{1,\ldots,n\}}$ and send o_{ij} to P_j as a private message
- At the end of this round, compute a list of parties L_i, so that P_j is in L_i if P_i received from P_j an *invalid* opening for the commitment C_{ji}. For each party $P_j \notin L_i$, recover the value ρ_{ji} by opening C_{ji}. For each party $P_j \in L_i$ set ρ_{ji} to an all-zero vector $\mathbf{0}$.

Round 1: Each P_i receives its input $\mathbf{x}(i)$, and does as follows.

- Broadcast the list L_i.
- Sample randomness r_i for Π^{sm} and compute its first round messages in Π^{sm} with input $\mathbf{x}(i)$ and randomness r_i, which we denote by $a_{i1},\ldots,a_{in}$, where a_{ij} is the private message from P_i to P_j. For every $j\in\{1,\ldots,n\}$, sample Shamir's shares $(a_{ij}[1],\ldots,a_{ij}[n])$ of a_{ij}.
- Sample a randomness r_i^{RE} for the randomized encoding $\hat{f}_i$ and:
  - *(For inputs* **(1)–(2)**.*)* Recall that inputs **(1)–(2)** correspond to indicators $(L_1[i],\ldots,L_n[i])$ and messages $A_i=(A_{ji})_{j\in\{1,\ldots,n\}}$. For the v-th input bit of f_i for $v\in[\ell_1+\ell_2]$, compute $\hat{f}_{iv}$ both for the input 0 and for the input 1, using randomness r_i^{RE} in both cases. Share the value $\hat{f}_{iv}(0;r_i^{\mathsf{RE}})$ via Shamir's sharing to $(s^0_{iv}[1],\ldots,s^0_{iv}[n])$ and similarly share $\hat{f}_{iv}(1;r_i^{\mathsf{RE}})$ to $(s^1_{iv}[1],\ldots,s^1_{iv}[n])$.
  - *(For inputs* **(3)–(4)**.*)* Recall that inputs **(3)–(4)** correspond to $\mathbf{x}(i), r_i$, and the pads $\eta_{i1},\ldots,\eta_{in}$. For the v-th input bit of f_i, for $v\in[\ell_1+\ell_2+1,\ell]$, compute the output of $\hat{f}_{iv}$ for this bit using randomness r_i^{RE}, and share the result to $(s_{iv}[1],\ldots,s_{iv}[n])$.
  - Denote by $\vec{s}_{ij}$ the vector of all shares that are directed to P_j. That is, $\vec{s}_{ij}$ contains $(s^b_{iv}[j])_{b\in\{0,1\}v\in[\ell_1+\ell_2]}$ and $(s_{iv}[j])_{v\in[\ell_1+\ell_2+1,\ell]}$.
- For every j, let $m_{ij}:=(a_{ij},(a_{ik}[j])_{k\in\{1,\ldots,n\}},\vec{s}_{ij})$ be the message directed to party P_j. Broadcast the encrypted message $M_{ij}:=m_{ij}+\rho_{ji}$. (Recall that ρ_{ji} is taken to be zero if $P_j\in L_i$.) We let $A_{ij}:=a_{ij}+\eta_{ji}$ denote the first part of M_{ij}.

Round 2: Let L be the set of parties that did not abort in Rounds 0 and 1. For every aborted party $P_j\notin L$, the parties set L_j to be the set that includes *all* the parties. The parties also set $\mathbf{x}(j)$ and r_j to be the all-zero string, and, based on these values, compute the outgoing messages of P_j in Π^{sm} as $(a_{jk})_{k\in\{1,\ldots,n\}}$. The parties also set $A_{jk}:=a_{jk}$.

In addition, for every i,j, party P_i broadcasts his shares for the party P_j as follows.

- *(If $P_j\notin L$).* For every $P_k\in L$, if $P_i\notin L_k$, then it broadcasts its first-round share $a_{kj}[i]$. (If $P_i\in L_k$ then $a_{kj}[i]$ is already public.)
- *(If $P_j\in L$).* Broadcast the i-th share of $\hat{f}_j((L_k[j])_{k\in[n]},A_j,\mathbf{x}(j),r_j,(\eta_{jk})_{k\in[n]};r_j^{\mathsf{RE}})$ by broadcasting the i-th share of the v-th part, $\hat{f}_{jv}$, as follows.
  - For $v\in[\ell_1]$, broadcast $s^b_{jv}[i]$ where $b=L_v[j]$.
  - For $v\in[\ell_1+1,\ell_1+\ell_2]$, broadcast $s^b_{jv}[i]$ where b is the (ℓ_1-v)-th bit of $A_j=(A_{kj})_{k\in[n]}$.
  - For $v\in[\ell_1+\ell_2+1,\ell]$, broadcast the share $s_{jv}[i]$.

(Local computation) Every P_i recovers b_j, the Round 2 broadcast of P_j for every P_j as follows:

- *(If $P_j\notin L$.)* Recover Round 1 messages of P_j as follows:
  - For every $P_k\notin L$, compute a_{kj} by setting $\mathbf{x}(k)$ and r_k to the all-zero string
  - For every $P_k\in L$, use the broadcasted shares of a_{kj} in order to recover a_{kj}

  On holding all the information about the first round of P_j in Π, compute Round 2 broadcast b_j.

- *(If $P_j \in L$.)* For every $P_j \in L$ and $k \in \{1, \ldots, \ell\}$, use broadcasted shares to recover the output of $\hat{f}_{jk}$. Decode $\hat{f}_j = (\hat{f}_{j1}, \ldots, \hat{f}_{j\ell})$ to obtain the output of f_j, which is set to b_j.

Compute the output of Π^{sm}, based on all broadcast values $(b_i)_{i \in \{1,\ldots,n\}}$, and output the result.

Figure 2: Protocol Π^{fs}

Tolerating malicious adversaries. Towards building our final construction, our first step is to identify the next-message functions of protocol Π^{fs} which are subsequently modeled as single input functionalities (SIFs). When we add these SIFs on top of Π^{fs}, it becomes maliciously-secure.

Let the next-message function of protocol Π^{fs}, be denoted as $\Pi^{\mathsf{fs}}_{i,r}$, for a party $i \in \{1, \ldots, n\}$ and round $r \in \{0, 1, 2, 3\}$ (where $r = 3$ is the function that returns the output of P_i at the end of the protocol). We denote the randomness that P_i used in Round 0 and Round 1 by $R_{i,0}$, and $R_{i,1}$ respectively. We continue with a formal description of the functions.

Function $\Pi^{\mathsf{fs}}_{i,0}$, $\Pi^{\mathsf{fs}}_{i,1}$ and $\Pi^{\mathsf{fs}}_{i,2}$

Function $\Pi^{\mathsf{fs}}_{i,0}$: It takes randomness $R_{i,0}$, and compute Round 0 of Π^{fs}. Specifically, given $R_{i,0}$, $\Pi^{\mathsf{fs}}_{i,0}$ samples random pads ρ_{ij}, and the corresponding commitments and openings (C_{ij}, o_{ij}), for $j \in \{1, \ldots, n\}$, and returns the commitments $C_{i1}, \ldots, C_{in}$ and the opening $o_{i1}, \ldots, o_{in}$.

Function $\Pi^{\mathsf{fs}}_{i,1}$: It takes two inputs

- the private view of P_i (i.e., the randomness $R_{i,0}$ and $R_{i,1}$, the input $\mathbf{x}(i)$, and all openings $o_{1i}, \ldots, o_{ni}$), denoted by $\mathsf{prView}_{i,1}$, and
- the public view of Round 0 (i.e., all the commitments $(C_{kj})_{k,j \in \{1,\ldots,n\}}$), denoted by $\mathsf{pubView}_{i,1}$

and performs the following computation

- verifies that $R_{i,0}$ is indeed the randomness used to generate $C_{i1}, \ldots, C_{in}$,
- returns a failure-symbol $\perp$ if the verification fails, and otherwise, returns the public broadcast of P_i in Round 1 according to Π^{fs}, by using $R_{i,1}$ as the randomness of P_i in Round 1.

Function $\Pi^{\mathsf{fs}}_{i,2}$: It takes two inputs

- the private view of P_i in Rounds 0 and 1 (i.e., the input $\mathbf{x}(i)$, randomness $R_{i,0}, R_{i,1}$ and all private messages that P_i received in Round 0), denoted $\mathsf{prView}_{i,2}$, and
- the public view of P_i (i.e., all broadcast messages in Rounds 0 and 1, including the broadcasts of P_i), denoted $\mathsf{pubView}_{i,2}$.

and performs the following computation

- verifies that the public messages of P_i are consistent with its view i.e., $R_{i,0}$ is indeed the randomness used to generate the commitments $C_{i1}, \ldots, C_{in}$, and that the broadcast of P_i in Round 1 is consistent with $\mathbf{x}(i), R_{i,0}, R_{i,1}$ and all messages that P_i received in Round 0.
- returns a failure-symbol $\perp$ if the verification fails, and otherwise returns the public broadcast of P_i in Round 2 according to Π^{fs}.

Figure 3: Function $\Pi^{\mathsf{fs}}_{i,0}$, $\Pi^{\mathsf{fs}}_{i,1}$ and $\Pi^{\mathsf{fs}}_{i,2}$

For a party P_i and a round $r \in \{1,2\}$, we denote by $\mathcal{F}_{i,r}$ the single input functionality that corresponds to $\Pi^{\mathsf{fs}}_{i,r}$ where P_i plays the role of the dealer.

Functionality $\mathcal{F}_{i,r}$

Inputs: $\mathcal{F}_{i,r}$ receives from the dealer the same inputs as $\Pi^{\mathsf{fs}}_{i,r}$ (i.e., $\mathsf{prView}_{i,r}$ and $\mathsf{pubView}_{i,r}$), as well as commitments and openings $(C_{\mathbf{x}(i)}, o_{\mathbf{x}(i)})$ and (C_{R_i}, o_{R_i}).

Computation: $\mathcal{F}_{i,r}$ verifies that $\mathsf{open}(C_{\mathbf{x}(i)}, o_{\mathbf{x}(i)})$ is equal to the input of P_i in $\mathsf{prView}_{i,r}$, and that $\mathsf{open}(C_{R_i}, o_{R_i})$ is equal to the randomness $(R_{i,0}, R_{i,1})$ in $\mathsf{prView}_{i,r}$. If the verification fails, or if $\Pi^{\mathsf{fs}}_{i,r}(\mathsf{prView}_{i,r}, \mathsf{pubView}_{i,r}) = \perp$ then it returns $\perp$ to all parties. Otherwise, the functionality returns $(\Pi^{\mathsf{fs}}_{i,r}(\mathsf{prView}_{i,r}, \mathsf{pubView}_{i,r}), \mathsf{pubView}_{i,r}, C_{\mathbf{x}(i)}, C_{R_i})$ (i.e., the output of $\Pi^{\mathsf{fs}}_{i,r}$ on the public view and the private view, together with the public view and the commitments) to all parties.

Figure 4: Functionality $\mathcal{F}_{i,r}$

The final protocol. We present the final protocol for degree-2 computation, which is secure against active adversaries. The protocol appears in Figure 5 and makes use of online/offline SIF protocol. A security statement is given in Theorem 4.2, and is proved in Section B.

Protocol dtc

Round 0: The parties do as follows.

- *(Randomness commitment)* Each P_i samples random strings $R_{i,0}$ and $R_{i,1}$, and samples $(C_{R_i}, o_{R_i}) \leftarrow \mathsf{commit}_{\mathsf{crs}}((R_{i,0}, R_{i,1}))$. P_i broadcasts C_{R_i}.
- *(Π^{fs} simulation)* The parties execute Round 0 of Π^{fs}, where P_i uses randomness $R_{i,0}$.
- *(sif offline round)* For every $i \in \{1, \ldots, n\}$ the parties execute the offline round of a sif instance with P_i as the dealer computing $\mathcal{F}_{i,1}$. We denote this instance of sif by $\mathsf{sif}_{i,1}$.

Round 1: Party P_i receives input $\mathbf{x}(i)$. The parties do as follows.

- *(Input commitment)* Each P_i samples $(C_{\mathbf{x}(i)}, o_{\mathbf{x}(i)}) \leftarrow \mathsf{commit}_{\mathsf{crs}}(\mathbf{x}(i))$, and broadcasts $C_{\mathbf{x}(i)}$.
- *(Π^{fs} simulation)* Each party P_i acts as a dealer in the instance of $\mathcal{F}_{i,1}$, and inputs $\mathsf{prView}_{i,1}$ and $\mathsf{pubView}_{i,1}$.
- *(sif offline round)* For every $i \in \{1, \ldots, n\}$ the parties execute the offline round of a sif instance with P_i as the dealer computing $\mathcal{F}_{i,2}$. We denote this instance of sif by $\mathsf{sif}_{i,2}$.
- *(Local computation)* For every P_i for which the output of $\mathcal{F}_{i,1}$ is $\perp$, the parties set the broadcast of P_i in the simulation of Π^{fs} to be $\perp$ (i.e., P_i aborted in Round 1 of Π^{fs}).

  Otherwise, let $(b_{i,1}, v_{i,1}, C_{i,1}, C'_{i,1})$ be the output of $\mathcal{F}_{i,1}$, where $b_{i,1}$ is the output of $\Pi^{\mathsf{fs}}_{i,1}$, $v_{i,1}$ is the public view of P_i, and $C_{i,1}$ and $C'_{i,1}$ are commitments. If $v_{i,1}$ is inconsistent with the public view in the simulation of Round 0, or $C_{i,1} \neq C_{\mathbf{x}(i)}$, or $C'_{i,1} \neq C_{R_i}$, then the parties set the broadcast of P_i in the simulation of Π^{fs} to be $\perp$ (i.e., P_i aborted in Round 1 of Π^{fs}).

  Otherwise, they set the broadcast to be $b_{i,1}$.

Round 2: The parties do as follows.

- *(Π^{fs} simulation)* Each party P_i acts as a dealer in the instance of $\mathcal{F}_{i,2}$, and inputs $\mathsf{prView}_{i,2}$ and $\mathsf{pubView}_{i,2}$.
- *(Local computation)* For every P_i for which the output of $\mathcal{F}_{i,2}$ is $\perp$, the parties set the broadcast of P_i in the simulation of Π^{fs} to be $\perp$ (i.e., P_i aborted in Round 2 of Π^{fs}).

  Otherwise, let $(b_{i,2}, v_{i,2}, C_{i,2}, C'_{i,2})$ be the output of $\mathcal{F}_{i,2}$. If $v_{i,2}$ is inconsistent with the public view in the simulation of Rounds 0 and 1, or $C_{i,2} \neq C_{\mathbf{x}(i)}$, or $C'_{i,2} \neq C_{R_i}$, then the parties set the broadcast of P_i in the simulation of Π^{fs} to be $\perp$ (i.e., P_i aborted in Round 2 of Π^{fs}). Otherwise, they set the broadcast to be $b_{i,2}$.

  Finally, each party locally executes the local computation step of Π^{fs}, in order to obtain its output in the simulation of Π^{fs}. This output is set to be the output of the protocol.

Figure 5: Protocol dtc

Theorem 4.2 (Security in hybrid model)**.** *Let κ be a security parameter, let n be the number of parties, $t < n/2$. Let $\mathbb{F}$ be a field of size at least $n+1$, and let $\mathcal{F}$ be a degree-2 n-party functionality over $\mathbb{F}$ with circuit size s. Assuming the existence of perfectly-binding computationally-hiding NICOM, protocol* dtc *is a UC-secure implementation of $\mathcal{F}$ in the $(\mathcal{F}_{i,r})_{i\in[n],r\in[2]}$-hybrid model, against a static, active, rushing adversary corrupting up to t parties. The complexity of the protocol is* $\mathrm{poly}(s, \log|\mathbb{F}|, n, \kappa)$.

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

By instantiating the protocol with the UC-secure SIF protocols of [AKP22] (as stated in Section 3.1.4), we immediately obtain the following corollaries.

Corollary 4.3 (Optimal-resiliency for a small number of parties)**.** *Let κ be a security parameter, let n be the number of parties and $t < n/2$. Let $\mathbb{F}$ be a field of size at least $n+1$, and let $\mathcal{F}$ be a degree-2 functionality over $\mathbb{F}$ with circuit size s. Assuming the existence of perfectly-binding sub-exponentially hiding NICOM, protocol* dtc *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* $\mathrm{poly}(s, \log|\mathbb{F}|, 2^n, \kappa)$.

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

Corollary 4.4 (Almost-optimal resiliency 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 be the number of corrupt parties such that $n = (2+\epsilon)t$. Let $\mathbb{F}$ be a field of size at least $n+1$, and let $\mathcal{F}$ be a degree-2 functionality over $\mathbb{F}$ with circuit size s. Assuming the existence of perfectly-binding sub-exponentially hiding NICOM, protocol* dtc *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* $\mathrm{poly}(s, \log|\mathbb{F}|, n, \kappa)$.

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

5 MPC with Strong Honest-Majority from OWF

We present a 3-round MPC protocol *with everlasting security* in the plain model, for degree-2 computation with strong honest-majority $t < n/3$, assuming the existence of one-way function. We first design a 2-round verifiable secret sharing protocol (VSS), and then simply plug in our VSS in the information-theoretic (statistical) framework of [AKP20a] to obtain degree-2 computation in 3 rounds.

5.1 Verifiable Secret Sharing

In this section our goal is to implement the following functionality.[10]

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

Inputs.

- An honest D inputs a symmetric bivariate polynomial $F(x, y)$ of degree t in each variable.
- A corrupt D inputs a polynomial $F(x, y)$.

Outputs.

- For an honest D, the functionality returns the univariate polynomial $f_i(x) := F(x, i)$ to every party P_i.
- For a corrupt D, if the input $F(x, y)$ is not a symmetric bivariate polynomial $F(x, y)$ of degree t in each variable, the functionality resets $F(x, y)$ to be the zero-polynomial. The functionality $\mathcal{F}_{\mathsf{vss}}$ returns the univariate polynomial $f_i(x) := F(x, i)$ to every P_i.

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

As discussed in Section 2.2, we follow the footsteps of typical symmetric bivariate polynomial based approach (see, e.g., [KKK09, AKP20b, AKP20a], see also Section C.1 for useful facts about symmetric bivariate polynomials), and in addition we use a digital signature scheme and randomized encoding (see Sections C.1 and 3 for formal definitions). We continue with the complete description of function g that will be used as the basis for our (combined) gadget for 2-wise consistency and share publication for unhappy parties.

The function g. The function g is defined as follows.

- **Inputs.** The function receives two triples $(\mathsf{flag}_A, w_A, \sigma_A)$ and $(\mathsf{flag}_B, w_B, \sigma_B)$, where $\mathsf{flag}_A, \mathsf{flag}_B \in \{0, 1\}$ indicate whether the parties are unhappy, w_A, w_B are "polynomial evaluation" tuples $w_A = (i_A, j_A, f_A)$ and $w_B = (i_B, j_B, f_B)$ for $i_A, j_A, i_B, j_B \in \{1, \ldots, n\}$ and $f_A, f_B \in \mathbb{F}$, and σ_A and σ_B are signatures.
- **Outputs.** If $\mathsf{flag}_A = 1$ or $\mathsf{flag}_B = 1$ then g outputs $(\mathsf{flag}_A, w_A, \sigma_A)$ and $(\mathsf{flag}_B, w_B, \sigma_B)$. Otherwise $\mathsf{flag}_A = \mathsf{flag}_B = 0$. In this case, if $f_A = f_B$ then g outputs "equal". Otherwise, g outputs "not equal" and the pairs (w_A, σ_A) and (w_B, σ_B).

[10] In the following functionality, we assume that the input of an honest D is well formed (i.e., it is a symmetric bivariate polynomial $F(x, y)$ of degree t in each variable). However, in the UC-security model, the inputs of the honest parties are arbitrarily chosen. Therefore, we define that whenever the inputs of the honest parties are not well formed, a *complete break-down* occurs, which means that the inputs of the honest parties are leaked to the adversary, and the adversary can also determine the outputs of the functionality. This makes simulation trivial, so we ignore this case from now on.

Observe that g can be implemented by a binary NC^1 circuit, and so, by Theorem 3.5 it has a 2-decomposable randomized encoding $\hat{g} = (\hat{g}_A, \hat{g}_B)$, where $\hat{g}_A$ takes $(\mathsf{flag}_A, w_A, \sigma_A)$ and randomness r, and $\hat{g}_B$ takes $(\mathsf{flag}_B, w_B, \sigma_b)$ and (the same) randomness r. We sometimes refer to the first triple as the inputs of Alice, and to the second triple as the inputs of Bob.

The protocol. The protocol is presented in Figure 7. A security statement appears in Theorem 5.1, and a proof of security appears in Section C.

Protocol vss

Primitives: A digital signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ (see Section C.1)

Inputs: D holds a symmetric bivariate polynomial $F(x, y)$ of degree-t in both variable.

Round 1:

- *(Key setup for signature schemes).* D samples a signature-key and a verification-key $(sk, vk) \leftarrow \mathsf{Gen}(1^\kappa)$. D broadcasts the verification-key vk.
- *(Polynomials distribution).* For every $i, j \in \{1, \ldots, n\}$, D computes $w_{ij} := (i, j, F(i, j))$ and $\sigma_{ij} := \mathsf{Sign}_{sk}(w_{ij})$. For every $i \in \{1, \ldots, n\}$, D sends $F(x, i)$ to P_i, together with the signatures $\sigma_{i1}, \ldots, \sigma_{in}$.
- *(Randomness sampling for randomized encoding).* For every $i < j$, P_i samples randomness r_{ij} for an instance of the randomized encoding $\hat{g}$, which we denote by $\hat{g}^{ij}$. P_i sends r_{ij} to P_j.
- *(Setting flags).* Every party P_i does as follows. Let $\bar{f}_i(x)$ be the degree-t polynomial that P_i received from D, let $\bar{\sigma}_{i1}, \ldots, \bar{\sigma}_{in}$ be the signatures that P_i received from D, and let $\bar{w}_{ij} := (i, j, \bar{f}_i(j))$. If there exists $j \in \{1, \ldots, n\}$ such that $\mathsf{Vrfy}_{vk}(\bar{w}_{ij}, \bar{\sigma}_{ij}) = 0$ then P_i sets $\mathsf{flag}_i = 1$. Otherwise, P_i sets $\mathsf{flag}_i = 0$.

Round 2:

- *(Broadcasting flags).* Every P_i broadcasts its flag flag_i.
- *(Pairwise consistency checking via randomized encoding).* For every $i < j$, P_i and P_j do as follows. P_i holds $(\mathsf{flag}_i, \bar{w}_{ij}, \bar{\sigma}_{ij})$ which we think of as the inputs of Alice to g, and P_j holds $(\mathsf{flag}_j, \bar{w}_{ji}, \bar{\sigma}_{ji})$, which we think of as the inputs of Bob to g. P_i broadcasts $\hat{g}_A^{ij}(\mathsf{flag}_i, \bar{w}_{ij}, \bar{\sigma}_{ij}; r_{ij})$, and P_j broadcasts $\hat{g}_B^{ij}(\mathsf{flag}_j, \bar{w}_{ji}, \bar{\sigma}_{ji}; r_{ij})$.

Local computation:

- *(Decoding output)* For every $i < j$ the parties decode the output of $\hat{g}^{ij} = (\hat{g}_A^{ij}, \hat{g}_B^{ij})$ using the broadcasts of P_i and P_j. Denote the output by Out_{ij}
- *(Inconsistency check)* If there exist $i < j$ so that (1) Out_{ij} is "not equal" together with $(\bar{w}_{ij}, \bar{\sigma}_{ij})$ and $(\bar{w}_{ji}, \bar{\sigma}_{ji})$, where $\bar{w}_{ij} = (i, j, f_{ij})$, $\bar{w}_{ji} = (j, i, f_{ji})$, and $f_{ij}, f_{ji} \in \mathbb{F}$, (2) $f_{ij} \neq f_{ji}$, and (3) $\mathsf{Vrfy}_{vk}((i, j, f_{ij}), \bar{\sigma}_{ij}) = 1$ and $\mathsf{Vrfy}_{vk}((j, i, f_{ji}), \bar{\sigma}_{ji}) = 1$, then D is discarded.[a]
- *(Share recovery for unhappy parties)* Otherwise, let L be the set of all parties P_i that broadcasted $\mathsf{flag}_i = 0$. If the size of L is less than $n - t$ then D is discarded.

  Otherwise, the size of L is at least $n - t$. For every $P_i \notin L$, let L_i be the set of all $P_j \in L$ such that (1) Out_{ij} is $(\mathsf{flag}_i, \bar{w}_{ij}, \sigma_{ij})$ and $(\mathsf{flag}_j, \bar{w}_{ji}, \bar{\sigma}_{ji})$, (2) $\mathsf{flag}_i = 1$ and $\mathsf{flag}_j = 0$, (3) $\bar{w}_{ji} = (j, i, f_{ji})$ for some $f_{ji} \in \mathbb{F}$, and (4) $\mathsf{Vrfy}_{vk}(\bar{w}_{ji}, \bar{\sigma}_{ji}) = 1$.

  If the set L_i is of size at least $n - 2t$, reset the polynomial $\bar{f}_i(x)$ to be the polynomial obtained by interpolating $(f_{ji})_{P_j \in L_i}$. If $\bar{f}_i(x)$ has degree more than t then D is discarded.

- *(Output)* If D was not discarded then every P_i outputs $\bar{f}_i(x)$.

[a] When D is discarded we simply assume that every P_i resets $\bar{f}_i(x)$ to be the zero-polynomial, and outputs $\bar{f}_i(x)$.

Figure 7: Protocol vss

Theorem 5.1. *Let κ be a security parameter, let n be the number of parties and $t < n/3$. Let $\mathbb{F}$ be a field of size at least $n+1$. Assuming the existence of one-way functions, protocol* vss *is a UC-secure implementation of $\mathcal{F}_{\mathsf{vss}}$* with everlasting security*, against a static, active, rushing adversary corrupting up to t parties. The complexity of the protocol is* $\mathrm{poly}(\log|\mathbb{F}|, n, \kappa)$.

5.2 From VSS to Degree-2 Computation

Applebaum et al. [AKP20a] presented a reduction from degree-2 computation to secure implementation of the $\mathcal{F}_{\mathsf{vss}}$ functionality. Specifically, they demonstrate a 2-round protocol where the first round has access to an ideal VSS channel. This leads to a compiler that turns every r-round secure realization of the functionality $\mathcal{F}_{\mathsf{vss}}$ into an $(r+1)$-round secure realization of degree-2 computation, and where the compiler preserves even statistical security. The following theorem follows.

Theorem 5.2. *Let κ be a security parameter, let n be the number of parties and $t < n/3$. Let $\mathbb{F}$ be a field of size at least $n+1$. Assuming the existence of one-way functions, there exists a UC-secure implementation of $\mathcal{F}_{\mathsf{dtc}}$* with everlasting security*, against a static, active, rushing adversary corrupting up to t parties. The complexity of the protocol is* $\mathrm{poly}(\log|\mathbb{F}|, n, \kappa)$.

References

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

[ACGJ18] Prabhanjan Ananth, Arka Rai Choudhuri, Aarushi Goel, and Abhishek Jain. Round-optimal secure multiparty computation with honest majority. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, pages 395–424, 2018.

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

[AJL+12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, pages 483–501, 2012.

[AKP20a] Benny Applebaum, Eliran Kachlon, and Arpita Patra. The resiliency of MPC with low interaction: The benefit of making errors (extended abstract). In Rafael Pass and

Krzysztof Pietrzak, editors, *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part II*, volume 12551 of *Lecture Notes in Computer Science*, pages 562–594. Springer, 2020.

[AKP20b] Benny Applebaum, Eliran Kachlon, and Arpita Patra. The round complexity of perfect MPC with active security and optimal resiliency. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1277–1284. IEEE, 2020.

[AKP22] Benny Applebaum, Eliran Kachlon, and Arpita Patra. Verifiable relation sharing and multi-verifier zero-knowledge in two rounds: Trading nizks with honest majority. Cryptology ePrint Archive, Report 2022/167, 2022. https://ia.cr/2022/167.

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

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

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.

[BCH86] Paul Beame, Stephen A. Cook, and H. James Hoover. Log depth circuits for division and related problems. *SIAM J. Comput.*, 15(4):994–1003, 1986.

[BCH+20] Christian Badertscher, Ran Canetti, Julia Hesse, Björn Tackmann, and Vassilis Zikas. Universal composition with global subroutines: Capturing global setup within plain uc. In *Theory of Cryptography Conference*, pages 1–30. Springer, 2020.

[BFJ+20] Saikrishna Badrinarayanan, Rex Fernando, Aayush Jain, Dakshita Khurana, and Amit Sahai. Statistical ZAP arguments. In *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part III*, pages 642–667, 2020.

[BFKR90] Donald Beaver, Joan Feigenbaum, Joe Kilian, and Phillip Rogaway. Security with low communication overhead. In *Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings*, pages 62–76, 1990.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 1–10, 1988.

[BJMS20] Saikrishna Badrinarayanan, Aayush Jain, Nathan Manohar, and Amit Sahai. Secure MPC: laziness leads to GOD. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part III*, volume 12493 of *Lecture Notes in Computer Science*, pages 120–150. Springer, 2020.

[Blu81] Manuel Blum. Coin flipping by telephone. In *Advances in Cryptology: A Report on CRYPTO 81, CRYPTO 81, IEEE Workshop on Communications Security, Santa Barbara, California, USA, August 24-26, 1981.*, pages 11–15, 1981.

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 503–513, 1990.

[BOV03] Boaz Barak, Shien Jin Ong, and Salil P. Vadhan. Derandomization in cryptography. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 299–315. Springer, 2003.

[BP15] Nir Bitansky and Omer Paneth. Zaps and non-interactive witness indistinguishability from indistinguishability obfuscation. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 401–427. Springer, 2015.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 136–145, 2001.

[CCH+19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-shamir: from practice to theory. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1082–1090. ACM, 2019.

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

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.

[CKPR01] Ran Canetti, Joe Kilian, Erez Petrank, and Alon Rosen. Black-box concurrent zero-knowledge requires omega˜(log n) rounds. In *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 570–579, 2001.

[Cle86] Richard Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 364–369, 1986.

[DI05] Ivan Damgård and Yuval Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, pages 378–394, 2005.

[DN07] Cynthia Dwork and Moni Naor. Zaps and their applications. *SIAM J. Comput.*, 36(6):1513–1543, 2007.

[DPP98] Ivan Damgård, Torben P. Pedersen, and Birgit Pfitzmann. Statistical secrecy and multi-bit commitments. *IEEE Trans. Inf. Theory*, 44(3):1143–1151, 1998.

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

[FM85] Paul Feldman and Silvio Micali. Byzantine agreement in constant expected time (and trusting no one). In *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*, pages 267–276, 1985.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[GIKR01] 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.

[GIKR02] Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. On 2-round secure multiparty computation. In *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, pages 178–193, 2002.

[GJJM20] Vipul Goyal, Abhishek Jain, Zhengzhong Jin, and Giulio Malavolta. Statistical zaps and new oblivious transfer protocols. In *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part III*, pages 668–699, 2020.

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

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, pages 25–32, 1989.

[GLS15] S. Dov Gordon, Feng-Hao Liu, and Elaine Shi. Constant-round MPC with fairness and guarantee of output delivery. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 63–82, 2015.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to solve any protocol problem. In *Proc. of STOC*, 1987.

[GO14] Jens Groth and Rafail Ostrovsky. Cryptography in the multi-string model. *Journal of cryptology*, 27(3):506–543, 2014.

[Gol04] Oded Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.

[GOS12] Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for noninteractive zero-knowledge. *J. ACM*, 59(3):11:1–11:35, 2012.

[Her09] Amir Herzberg. Folklore, practice and theory of robust combiners. *Journal of Computer Security*, 17(2):159–189, 2009.

[HM96] Shai Halevi and Silvio Micali. Practical and provably-secure commitment schemes from collision-free hashing. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 201–215. Springer, 1996.

[HR12] Mihir Bellareand Viet Tung Hoang and Phillip Rogaway. Adaptively secure garbling with applications to one-time programs and secure outsourcing. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, pages 134–153. Springer, 2012.

[HV06] Alexander Healy and Emanuele Viola. Constant-depth circuits for arithmetic in finite fields of characteristic two. In Bruno Durand and Wolfgang Thomas, editors, *STACS 2006, 23rd Annual Symposium on Theoretical Aspects of Computer Science, Marseille, France, February 23-25, 2006, Proceedings*, volume 3884 of *Lecture Notes in Computer Science*, pages 672–683. Springer, 2006.

[IK00] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 294–304, 2000.

[IK02] Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In *Automata, Languages and Programming, 29th International Colloquium, ICALP 2002, Malaga, Spain, July 8-13, 2002, Proceedings*, pages 244–256, 2002.

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

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

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

[MU10] Jörn Müller-Quade and Dominique Unruh. Long-term security and universal composability. *J. Cryptol.*, 23(4):594–671, 2010.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991.

[NOVY98] 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.

[PR18] Arpita Patra and Divya Ravi. On the exact round complexity of secure three-party computation. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, pages 425–458, 2018.

[RB89] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, pages 73–85, 1989.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[Yao82] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 80–91, 1982.

A Appendix: Security Model, Useful Facts and Standard Primitives

A.1 Security model

In this section we give a high-level description of the UC-framework, due to [Can01]. For more details, the reader is referred to [Can01]. We begin with a short description of the standard model, and then explain how the UC-framework augments it. At a high level, in the standard model, security of a protocol is argued by comparing the real-world execution to an ideal-world execution. In an ideal-world execution, the inputs of the parties are transferred to a trusted party $\mathcal{F}$ (called the *ideal functionality*) over a perfectly secure channel, the trusted party computes the function based on these inputs and sends to each party its respective output. Informally, a protocol π securely implements $\mathcal{F}$ if for any real-world adversary $\mathcal{A}$, there exists an ideal-world adversary $\mathcal{S}$ (called the *simulator*), that controls the same parties as $\mathcal{A}$, so that the global output of an execution of π with $\mathcal{A}$ (consisting of the honest parties' outputs and the output of $\mathcal{A}$), is indistinguishable from the global output of the ideal-world execution with $\mathcal{F}$ and $\mathcal{S}$ (consisting of the honest parties' outputs and the output of $\mathcal{S}$).

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

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

The dummy-adversary. Since the above definition quantifies over all environments, we can merge the adversary $\mathcal{A}$ with the environment $\mathcal{Z}$. That is, it is enough to require that the simulator $\mathcal{S}$ will be able to simulate, for any environment $\mathcal{Z}$, the *dummy adversary* that simply delivers messages from $\mathcal{Z}$ to the protocol machines. For more information, see [Can01].

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.

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 [MU10]. In a nutshell, instead of considering environments that are unbounded after the execution, it is enough to consider only environments that are always polynomially-bounded, but are not limited to a single bit output. In particular, such environments can output their whole view. Using the same notation as before, $\text{REAL}_{\pi,\mathcal{Z}(\zeta),\mathcal{A}}(\kappa)$ and $\text{IDEAL}_{\mathcal{F},\mathcal{Z}(\zeta),\mathcal{S}}(\kappa)$, to denote the output distribution of $\mathcal{Z}$ in the real-world and in the ideal-world (where now the output may contain more than one bit), we say that a protocol π UC-emulates an ideal functionality $\mathcal{F}$ with everlasting security, if for every polynomial-time real-world adversary $\mathcal{A}$ there exists an ideal-world polynomial-time simulator $\mathcal{S}$ such that for any polynomial-time environment $\mathcal{Z}$ and any input ζ to $\mathcal{Z}$, the random variables $\{\text{REAL}_{\pi,\mathcal{Z}(\zeta),\mathcal{A}}(\kappa)\}_\kappa$, and $\{\text{IDEAL}_{\mathcal{F},\mathcal{Z}(\zeta),\mathcal{S}}(\kappa)\}_\kappa$ are *statistically* indistinguishable. Therefore, in general, in order to prove security it is enough to show that the view of the environment in the real-world is statistically-close to the view of the environment in the ideal-world.

We mention that the composition theorems of UC-security hold for protocols with everlasting security (i.e., the composition of two protocols with everlasting security results in a protocol with everlasting security). For a formal definition and statement of the composition theorem, the reader is referred to [MU10].

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, whenever we consider everlasting security, 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., [BCH+20, Section 1]).

B Proof of Theorem 4.2

In the following section we provide formal security proof for protocol dtc. The same proof that shows that protocol dtc securely implements degree-2 computation when the underlying commitment scheme is computationally-hiding, also shows that dtc securely implements degree-2 computation *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 by polynomial-size circuits 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.

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).

To simplify the notation, we assume that $\mathcal{F}$ computes a degree-2 function with a single output

$$f(x^1, \ldots, x^m) = \alpha_0 + \sum_{i \in \{1,\ldots,m\}} \alpha_i \cdot x^i + \sum_{i,j \in \{1,\ldots,m\}} \alpha_{ij} \cdot x^i \cdot x^j,$$

for some field elements $(\alpha_i)_{i \in \{0,\ldots,m\}}$, $(\alpha_{ij})_{i,j \in \{1,\ldots,m\}}$. A generalization to the case of multi-output function is straightforward, but requires cumbersome notation. We follow the notation presented in Section 4, and for every party P_i, we denote by $I_i \subseteq \{1, \ldots, m\}$ the set of all indices j such that P_i holds the input x^j. We denote the inputs of P_i by $\mathbf{x}(i) := (x^j)_{j \in I_i}$.

B.1 The Simulator

We continue with the proof that protocol dtc UC-emulates the degree-2 functionality $\mathcal{F}$ (with everlasting security when the underlying commitment scheme is statistically-hiding). By the composition properties of UC-security, it is enough to prove security in the $\mathcal{F}_{\mathsf{sif}}$-hybrid model. 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, and acts as follows.

Round 0. For every honest P_i the simulator acts exactly like in the protocol. That is, the simulator samples randomness $(R_{i,0}, R_{i,1})$ on behalf of P_i, samples $(C_{R_i}, o_{R_i}) \leftarrow \mathsf{commit}_{\mathsf{crs}}((R_{i,0}, R_{i,1}))$, and broadcasts C_{R_i} on behalf of P_i. In addition, the simulator computes the commitments and openings $(C_{i1}, o_{i1}), \ldots, (C_{in}, o_{in})$ according to $\Pi^{\mathsf{fs}}_{i,0}(R_{i,0})$, broadcasts $C_{i1}, \ldots, C_{in}$ and sends o_{ij} to any corrupt P_j. This concludes the communication from honest parties to corrupt parties.

At the end of the round, the simulator receives from $\mathcal{A}$ the messages from the corrupt parties to the honest parties. That is, for every corrupt P_i the simulator receives the commitments $C_{i1}, \ldots, C_{in}$, as well as the private message o_{ij} to every honest P_j. We note that both the broadcast messages and the private messages might possibly be $\perp$. This concludes the simulation of Round 0.

Round 1. We denote by $\mathsf{pubView}_1$ the public view of the parties at the beginning of Round 1 (i.e., all broadcast message in Round 0). For every honest P_i the simulator does as follows.

- *(Input commitment)* The simulator samples $(C_{\mathbf{x}(i)}, o_{\mathbf{x}(i)}) \leftarrow \mathsf{commit}_{\mathsf{crs}}(0)$ and broadcasts $C_{\mathbf{x}(i)}$ on behalf of P_i.
- *(Computation of L_i)* The simulator computes the set L_i that contains all corrupt parties P_j that sent invalid openings to P_i in Round 0.

- *(Simulation of* Π^{sm}*)* For every $j \in I_i$ the simulator samples random field elements $(x_k^j)_{k \in \mathsf{C}}$ as the shares of the corrupt parties. In addition, the simulator samples random field elements $(z_k^i)_{k \in \mathsf{C}}$ as the shares of the corrupt parties in the zero-sharing. The simulator sets $a_{ik} := (z_k^i, (x_k^j)_{j \in I_i})$ for every $k \in \mathsf{C}$.
- *(Sharing computation)* For every $j \in \{1, \ldots, n\}$, the simulator samples random elements $(a_{ij}[k])_{k \in \mathsf{C}}$ as the shares of the corrupt parties in the sharing of a_{ij}.

  The simulator also samples random shares $\vec{s}_{ij}$ for every corrupt P_j as the shares of the randomized encoding outputs.
- *(Messages of* Π^{fs}*)* For every corrupt $P_j \notin L_i$, let $\rho_{ji} = \mathsf{open}(C_{ji}, o_{ji})$, where C_{ji} is the i-th commitment that P_j broadcasted in Round 0, and o_{ji} is the opening that P_j sent to P_i in Round 0. The simulator sets $M_{ij} := a_{ij} + \rho_{ji}$ as the message to P_j.

  For every corrupt $P_j \in L_i$, the simulator sets $M_{ij} := a_{ij}$ as the message to P_j.

  For every honest P_j, the simulator samples a random message M_{ij} as the message to P_j.
- *(*sif *simulation)* For every honest P_i the simulator returns $((L_i, M_{i1}, \ldots, M_{in}), \mathsf{pubView}_1, C_{\mathbf{x}(i)}, C_{R_i})$ as the output of $\mathcal{F}_{i,1}$.

This concludes the communication from honest parties to corrupt parties. At the end of the round, the simulator receives from $\mathcal{A}$ the inputs of every corrupt P_i to the functionality $\mathcal{F}_{i,1}$. The simulator computes the output of the functionality (that depends only on the inputs of P_i), and returns it to all the corrupt parties. This concludes the simulation of Round 1.

Communication with $\mathcal{F}$. Let L be the set of all parties. For every corrupt P_i the simulator does as follows. If the output of $\mathcal{F}_{i,1}$ is $\perp$, or if the output is $(b_{i,1}, v_{i,1}, C_{i,1}, C'_{i,1})$ where $v_{i,1} \neq \mathsf{pubView}_1$, or $C_{i,1} \neq C_{\mathbf{x}(i)}$ or $C'_{i,1} \neq C_{R_i}$, then the simulator (1) removes P_i from L, and (2) sets $\mathbf{x}(i)$ to be the all-zero vector, and sends $\mathbf{x}(i)$ to $\mathcal{F}$. Otherwise, the output of $\mathcal{F}_{i,1}$ is $(b_{i,1}, \mathsf{pubView}_1, C_{\mathbf{x}(i)}, C_{R_i})$. In this case, consider the inputs $(C_{\mathbf{x}(i)}, o_{\mathbf{x}(i)})$ of P_i to $\mathcal{F}_{i,1}$, and let $\mathbf{x}(i) := \mathsf{open}(C_{\mathbf{x}(i)}, o_{\mathbf{x}(i)})$. The simulator inputs $\mathbf{x}(i)$ to $\mathcal{F}$ as the inputs of P_i. Finally, the simulator receives the output y of $\mathcal{F}$.

Round 2. We denote by $\mathsf{pubView}_2$ the public view of the parties at the beginning of Round 2 (i.e., all broadcast message in Rounds 0 and 1). The simulator samples a random degree-$2t$ polynomial $W(x)$ conditioned on $W(0) = y$ and

$$W(i) = f(x_i^1, \ldots, x_i^m) + z_i^1 + \ldots + z_i^n,$$

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

For every honest P_i the simulator does as follows.

- *(Randomized encoding simulation)* The simulator samples $(X_{iv})_{v \in \{1,\ldots,\ell\}} \leftarrow \mathcal{S}_i^{\mathsf{RE}}(W(i))$, where $\mathcal{S}_i^{\mathsf{RE}}$ is the simulator of the randomized encoding $\hat{f}_i$, and X_{iv} is the simulated output of f_{iv}.

  For every input-bit $\ell_1 + \ell_2 + 1 \leq v \leq \ell$, that belongs to inputs **(3)–(4)** of f_i, the simulator generates the honest parties' shares of the output of $\hat{f}_{iv}$ as follows. Let $(X_{iv}[k])_{k \in \mathsf{C}}$ be the shares that the simulator picked in the simulation of Round 1. The simulator picks a random degree-t polynomial $h_{iv}(x)$ conditioned on $h_{iv}(0) = X_{iv}$ and $h_{iv}(k) = X_{iv}[k]$ for all $k \in \mathsf{C}$. The simulator sets $X_{iv}[k] := h_{iv}(k)$ as the share of an honest P_k.

For every input-bit $1 \le v \le \ell_1 + \ell_2$, that belongs to inputs **(1)–(2)** of f_i, the simulator holds the value of the v-th input bit, which we denote by β_v, and it generates the honest parties' shares of the output of $\hat{f}_{iv}$ as follows. Let $(X_{iv}^{\beta}[k])_{k \in \mathsf{C}, \beta \in \{0,1\}}$ be the shares that the simulator picked in the simulation of Round 1, where $X_{iv}^{\beta}[k]$ is the k-th share of the output of $\hat{f}_{iv}$ on input β. The simulator picks a random degree-t polynomial $h_{iv}(x)$ conditioned on $h_{iv}(0) = X_{iv}$ and $h_{iv}(k) = X_{iv}^{\beta_v}[k]$ for all $k \in \mathsf{C}$. The simulator sets $X_{iv}[k] := h_{iv}(k)$ as the share of an honest P_k.

- *(Shares of messages)* For every corrupt P_j not in L, pick a random degree-t polynomial $h_{ij}(x)$ conditioned on $h_{ij}(0) = a_{ij}$ and $h_{ij}(k) = a_{ij}[k]$ for all $k \in \mathsf{C}$. The simulator sets $a_{ij}[k] := h_{ij}(k)$ for all $k \in \mathsf{H}$.

For every honest P_i the simulator computes the output of $\mathcal{F}_{i,2}$ as follows.

- For every honest P_k the simulator sets $B_{ik} := (X_{kv}[i])_{v \in \{1,\dots,\ell\}}$.
- For every corrupt P_k in L, the simulator holds all the messages that P_k sent to P_i in the simulation of Π^{fs}. Given those messages, the simulator sets B_{ik} to be the shares that P_i would broadcast in Π^{fs} in order to recover the output of $\hat{f}_k$.
- For every corrupt P_k not in L, the simulator sets $B_{ik} := (\bar{a}_{jk}[i])_{P_j \in L}$, where (1) $\bar{a}_{jk}[i] := a_{jk}[i]$ when P_j is honest, (2) $\bar{a}_{jk}[i]$ is the share of a_{jk} that P_j sent to P_i in Round 1 for corrupt P_j with $P_i \notin L_j$, and (3) $\bar{a}_{jk}[i] := \perp$ when P_j is corrupt and $P_i \in L_j$.

Finally, the simulator sets $B_i := (B_{i1}, \dots, B_{in})$, and sets the output of $\mathcal{F}_{i,2}$ to be $(B_i, \mathsf{pubView}_2, C_{\mathbf{x}(i)}, C_{R_i})$. This concludes the communication from honest parties to corrupt parties. At the end of the round, the simulator receives from $\mathcal{A}$ the inputs of every corrupt P_i to the functionality $\mathcal{F}_{i,2}$, computes the output, and returns it to all corrupt parties. This concludes the simulation.

B.2 Analysis

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.

B.2.1 Hybrids

First define some hybrid experiments in which the honest parties know the identity of each other.

Hybrid 1. In Round 0 the parties play according to protocol dtc. In Round 1, for every $i \in \mathsf{H}$ we change the ideal functionality $\mathcal{F}_{i,1}$ as follows. The functionality receives from P_i (1) the public view in Round 0 and the commitments $C_{\mathbf{x}(i)}, C_{R_i}$, (2) the pads $\rho_{i1}, \dots, \rho_{in}$, (3) the set L_i, as well as the pads ρ_{ji} for every $P_j \notin L_i$, and (4) the input $\mathbf{x}(i)$ and randomness $R_{i,1}$ of P_i. The functionality computes the broadcast of P_i in Round 1 of Π^{fs} according to inputs (2)–(4), and returns the broadcast, the public view that P_i provided, and the commitments $C_{\mathbf{x}(i)}, C_{R_i}$ that P_i provided. Note that the functionality does not verify the consistency of the private view with the public view. We instruct P_i to provide $\mathcal{F}_{i,1}$ with the correct inputs, and other than that act according to dtc.

In Round 2, for every $i \in \mathsf{H}$ we change the ideal functionality $\mathcal{F}_{i,2}$ as follows. The functionality receives from P_i (1) the public view in Round 0 (2) the public view in Round 1, (3) the pads $\rho_{i1}, \ldots, \rho_{in}$, and (4) the set L_i, as well as the pads ρ_{ji} for every $P_j \notin L_i$. The functionality computes the broadcast of P_i in Round 2 of Π^{fs} according to inputs (2)–(4), and returns the broadcast and the public view that P_i provided. We instruct an honest P_i to provide $\mathcal{F}_{i,2}$ with the correct inputs and other than that act according to dtc.

Hybrid 2. For every $i \in \mathsf{H}$, we modify the ideal functionalities $\mathcal{F}_{i,1}$ and $\mathcal{F}_{i,2}$ like in Hybrid 1. In addition, we make the following changes in the protocol. In Round 0, the honest parties play exactly like in Round 0 of Hybrid 1. In addition, for every ordered pair of honest parties (P_i, P_j), P_i picks an additional random pad ρ'_{ij}, and sends it to P_j. For every honest P_i, We set $\bar{\rho}_{ij} := \rho_{ij}$ if $j \in \mathsf{C}$, and $\bar{\rho}_{ij} := \rho'_{ij}$ if $j \in \mathsf{H}$. For a corrupt P_i we let $\bar{\rho}_{ij} := \rho_{ij}$ for all $j \in \{1, \ldots, n\}$.

In Round 1, every honest P_i samples fresh randomness $\bar{R}_{i,1}$ and inputs to the modified ideal functionality $\mathcal{F}_{i,1}$ the following inputs: (1) the public view in Round 0 and the commitments $C_{\mathbf{x}(i)}, C_{R_i}$, (2) the pads $\bar{\rho}_{i1}, \ldots, \bar{\rho}_{in}$, (3) the set L_i, as well as the pads $\bar{\rho}_{ji}$ for every $P_j \notin L_i$, and (4) the input $\mathbf{x}(i)$ and randomness $\bar{R}_{i,1}$. Other than that P_i acts exactly like in Hybrid 1.

In Round 2, every honest P_i inputs to the modified ideal functionality $\mathcal{F}_{i,1}$ the following inputs: (1) the public view in Round 0, (2) the public view in Round 1, (3) the pads $\bar{\rho}_{i1}, \ldots, \bar{\rho}_{in}$, (4) the set L_i, as well as the pads $\bar{\rho}_{ji}$ for every $P_j \notin L_i$. Other than that P_i acts exactly like in Hybrid 1.

Hybrid 3. The honest parties act like in Hybrid 2, except that in the input-commitment in Round 1, each honest P_i commits to the all zero string instead of its input. We denote the commitment by $\bar{C}_{\mathbf{x}(i)}$.

We continue by proving that the real-world is $O(n^2\epsilon)$-close to the ideal-world.

B.2.2 Real-world vs. Hybrid 1

It is not hard to see that the real-world view has the same distribution as the view in Hybrid 1. This follows because, given the correct pads $\rho_{i1}, \ldots, \rho_{in}$, set L_i and pads ρ_{ji} for P_j in L_i, the modified functionalities have the same output as the original functionalities.

B.2.3 Hybrid 1 vs. Hybrid 2

We show that Hybrid 1 is $O(n^2\epsilon)$-close to Hybrid 2. Consider the Hybrid 1 random variables

$$((C_{ij})_{i\in\mathsf{H},j\in\{1,\ldots,n\}}, (\rho_{ij})_{i\in\mathsf{H},j\in\mathsf{H}}, (o_{ij})_{i\in\mathsf{H},j\in\mathsf{C}}, (C_{R_i}, R_{i,1})_{i\in\mathsf{H}}), \tag{1}$$

and the Hybrid 2 random variables

$$((C_{ij})_{i\in\mathsf{H},j\in\{1,\ldots,n\}}, (\bar{\rho}_{ij})_{i\in\mathsf{H},j\in\mathsf{H}}, (o_{ij})_{i\in\mathsf{H},j\in\mathsf{C}}, (C_{R_i}, \bar{R}_{i,1})_{i\in\mathsf{H}}), \tag{2}$$

as well as the following hybrids,

$$((C_{ij})_{i\in\mathsf{H},j\in\{1,\dots,n\}},(\rho_{ij})_{i\in\mathsf{H},j\in\mathsf{H}},(o_{ij})_{i\in\mathsf{H},j\in\mathsf{C}},(\bar{C}_{R_i},R_{i,1})_{i\in\mathsf{H}}), \tag{3}$$

and

$$((C_{ij})_{i\in\mathsf{H},j\in\{1,\dots,n\}},(\bar{\rho}_{ij})_{i\in\mathsf{H},j\in\mathsf{H}},(o_{ij})_{i\in\mathsf{H},j\in\mathsf{C}},(\bar{C}_{R_i},\bar{R}_{i,1})_{i\in\mathsf{H}}), \tag{4}$$

where $\bar{C}_{R_i}$ is a commitment of the all-zero string (that is, it is independent of $R_{i,0},R_{i,1}$).

First, observe that the random variables in Equation 1 are $O(n\epsilon)$-close to the random variables in Equation 3. Indeed, by the hiding property of the commitment scheme the random variables $(R_{i,0},R_{i,1},C_{R_i})_{i\in\mathsf{H}}$ are $O(n\epsilon)$-close to the random variables $(R_{i,0},R_{i,1},\bar{C}_{R_i})_{i\in\mathsf{H}}$, and there exists an efficient process that given a sample from $(R_{i,0},R_{i,1},C_{R_i})_{i\in\mathsf{H}}$ outputs a sample from Equation 1, and given a sample from $(R_{i,0},R_{i,1},\bar{C}_{R_i})_{i\in\mathsf{H}}$ outputs a sample from Equation 3. A similar argument shows that the random variables in Equation 2 are $O(n\epsilon)$-close to those in Equation 4.

In addition, the random variables in Equation 3 are $O(n^2\epsilon)$-close to those in Equation 4. Indeed, the random variables $(\bar{C}_{R_i},R_{i,1})_{i\in\mathsf{H}}$ have the same distribution as $(\bar{C}_{R_i},\bar{R}_{i,1})_{i\in\mathsf{H}}$. Conditioned on those values, the random variables $(\rho_{ij})_{i\in\mathsf{H},j\in\mathsf{H}}$ and $(\bar{\rho}_{ij})_{i\in\mathsf{H},j\in\mathsf{H}}$ have the same distribution. Conditioned on those values, the random variables $(C_{ij},o_{ij})_{i\in\mathsf{H},j\in\mathsf{C}}$ have the same distribution. Conditioned on those values, the commitments $(C_{ij})_{i\in\mathsf{H},j\in\mathsf{H}}$ are $O(n^2\epsilon)$-close.

We conclude that the Hybrid 1 random variables in Equation 1 are $O(n^2\epsilon)$-close to the Hybrid 2 random variables in Equation 2. Finally, one can verify that in both hybrids the rest of the view can be obtained from those random variables by the same efficient process. We conclude that he view in Hybrid 1 is $O(n^2\epsilon)$-close to the view in Hybrid 2.

B.2.4 Hybrid 2 vs. Hybrid 3

We show that Hybrid 2 is $O(n\epsilon)$-close to Hybrid 3. It is not hard to see that the Hybrid 2 random variables

$$((C_{ij})_{i\in\mathsf{H},j\in\{1,\dots,n\}},(\bar{\rho}_{ij})_{i\in\mathsf{H},j\in\mathsf{H}},(o_{ij})_{i\in\mathsf{H},j\in\mathsf{C}},(C_{R_i})_{i\in\mathsf{H}}),$$

have the same distribution as the corresponding random variables in Hybrid 3. Conditioned on those value, the Round 0 messages of the corrupt parties have the same distribution. Fix those as well. Then the inputs of the honest parties are picked by $\mathcal{Z}$ in the same way in both hybrids, so they have the same distribution. Conditioned on the inputs, the Hybrid 2 random variables $(C_{\mathbf{x}(i)})_{i\in\mathsf{H}}$ are $O(n\epsilon)$-close to the Hybrid 3 random variables $(\bar{C}_{\mathbf{x}(i)})_{i\in\mathsf{H}}$. Finally, one can verify that in both hybrids the rest of the view can be obtained from those random variables by the same efficient process. We conclude that the view in Hybrid 2 is $O(n\epsilon)$-close to the view in Hybrid 3.

B.2.5 Hybrid 3 vs. Ideal-world.

We show that view in Hybrid 3 is ϵ-close *in statistical distance* to the view in the ideal-world.

Round 0. It is not hard to see that the view in Round 0 is the same in both worlds. Conditioned on those values, the inputs of the honest parties are picked in the same way in both worlds, and we fix those inputs as well.

Round 1. In Round 1, the commitment $\bar{C}_{\mathbf{x}(i)}$ has the same distribution in both worlds. In addition, in both worlds, for every honest P_i the Round 1 broadcasts M_{ij} for an honest P_j are uniformly distributed. It remains to analyse the broadcasts M_{ij} from an honest P_i to a corrupt P_j. Since in both worlds P_i uses the pads in the same way, it is enough to consider the distribution of the messages in Π^{fs}, that is, the messages $(m_{ij})_{j\in\mathsf{C}}$. Those messages consist of (1) the messages $(a_{ij})_{j\in\mathsf{C}}$, that consist of the the corrupt parties' shares in the input-sharing and zero-sharing of P_i in Π^{sm}, (2) the corrupt parties shares' of the messages $(a_{ij})_{j\in\{1,\ldots,n\}}$, and (3) corrupt parties shares' of the randomized encoding outputs. By the perfect privacy of Shamir's secret sharing, and the fact that in Hybrid 3 the honest parties use fresh randomness $\bar{R}_{i,1}$ in Round 1, we conclude that in both worlds those messages have the same distribution (that is, all the shares are uniformly distributed). We conclude that the view in Round 1 has the same distribution in both worlds.

Process. Consider the efficient process, that receives the (partial) view of $\mathcal{Z}$ in Rounds 0 and 1 and does as follows. First, it sets L to be the set of all parties. For every corrupt P_i, if the output of $\mathcal{F}_{i,1}$ is $\perp$, or if the output is $(b_{i,1}, v_{i,1})$ where $v_{i,1}$ is not equal to the public view, then the process (1) removes P_i from L, and (2) sets $\mathbf{x}(i)$ to be the all-zero vector. Otherwise, the process takes the inputs $(C_{\mathbf{x}(i)}, o_{\mathbf{x}(i)})$ of P_i to $\mathcal{F}_{i,1}$, and sets $\mathbf{x}(i) := \mathsf{open}(C_{\mathbf{x}(i)}, o_{\mathbf{x}(i)})$. In addition, for every honest P_i, the process holds the inputs $\mathbf{x}(i)$ that $\mathcal{Z}$ gave to P_i. The process computes the value $\bar{y} := f(x^1, \ldots, x^m)$. Given L and $\bar{y}$ the process generates the Round 2 messages from honest parties to corrupt parties exactly like the simulator, and outputs the view of $\mathcal{Z}$ in Rounds 0, 1 and 2.

Round 2. The output of the process when receiving a partial view from the ideal-world has the same as the view of $\mathcal{Z}$ in the ideal-world, so we continue with the analysis of Hybrid 3. Consider all commitments and openings that are generated by the honest parties in Rounds 0 and 1, as well as the commitments broadcasted by the corrupt parties, the openings that are sent from corrupt parties to honest parties, and the commitments and openings defined by the randomness $(R_{i,0})_{i\in\mathsf{C}}$ that the corrupt parties send to $\mathcal{F}_{i,1}$ in Round 1 (i.e., the commitments and openings generated by $\Pi^{\mathsf{fs}}_{i,0}(R_{i,0})$). We say that the view of the environment in Rounds 0 and 1 is *good* if for every pair of commitments C and C' and any opening o the following holds: 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 of the commitment scheme it follows that the view is good with probability at least $1 - \epsilon$.

Fix any good view of Rounds 0 and 1. For every P_i in L, and every $j \in I_i$, denote by $f^{x^j}(x)$ the degree-t polynomial that P_i used for the sharing of x^j. Similarly, denote by $f^{z^i}(x)$ the degree-$2t$ polynomial that P_i used for the zero-sharing. Note that those polynomials are computed by $\mathcal{F}_{i,1}$ given $\mathbf{x}(i)$ and $R_{i,1}$. For every P_i not in L, the inputs $\mathbf{x}(i)$ and randomness r_i are set to the all-zero string, and we let $f^{x^j}(x)$ and $f^{z^i}(x)$ be the sharing polynomials defined by $\mathbf{x}(i)$ and r_i. Consider the degree-$2t$ polynomial

$$f^{\mathsf{out}}(x) = \alpha_0 + \sum_{i\in\{1,\ldots,m\}} \alpha_i \cdot f^{x^i}(x) + \sum_{i,j\in\{1,\ldots,m\}} \alpha_{ij} \cdot f^{x^i}(x) \cdot f^{x^j}(x) + f^{z^1}(x) + \ldots + f^{z^n}(x),$$

and observe that $f^{\mathsf{out}}(x)$ is a random degree-$2t$ polynomial conditioned on $f^{\mathsf{out}}(0) = \bar{y}$ and $f^{\mathsf{out}}(i) = f(x_i^1, \ldots, x_i^m) + z_i^1 + \ldots + z_i^n$, for every $i \in \mathsf{C}$. We conclude that $f^{\mathsf{out}}(x)$ has the same distribution as the polynomial $W(x)$ defined by the process. Fix those polynomials. We continue by analysing the recovery of the parties broadcast in Round 2. We split into cases.

- Let P_i be a corrupt party in L. Since the process computes the broadcast of every honest P_k according to the protocol, we conclude that the broadcasts of the honest parties that correspond to the reconstruction of P_i's broadcast in the second round of Π^{sm} are fixed and have the same distribution in both worlds.

- Let P_i be a corrupt party outside L. For every corrupt P_j, and every message a_{ji}, the shares of the honest parties that are generated by the process are the same as the shares in Hybrid 3. For every honest P_j, by the perfect privacy of the secret sharing scheme, the shares of the honest parties that are generated by the process are the same as the shares in Hybrid 3.

- Fix an honest P_i, and observe that P_i is in L. In Hybrid 3, let $\beta_1, \ldots, \beta_{\ell_1+\ell_2}$ be the binary representation of the public inputs **(1)–(2)** of f_i (which are known to all the parties), and let $\beta_{\ell_1+\ell_2+1}, \ldots, \beta_\ell$ be the binary representation of the private inputs **(3)–(4)** of f_i, that is, of $\mathbf{x}(i)$, $\bar{r}_i$ and $\bar{\eta}_{i1}, \ldots, \bar{\eta}_{in}$, where $\bar{r}_i$ is the randomness defined by $\bar{R}_{i,1}$, and $\bar{\eta}_{ij}$ is the first entry of $\bar{\rho}_{ij}$. Observe that since the view is good, then $f_i(\beta_{i,1}, \ldots, \beta_{i,\ell}) = f^{\mathsf{out}}(i)$.

  Let $\bar{r}_i^{\mathsf{RE}}$ be the randomness of the randomized encoding defined by $\bar{R}_{i,1}$, and observe that it is uniformly distributed. By the perfect privacy of the randomized encoding, the distribution of $\hat{f}_i(\beta_{i,1}, \ldots, \beta_{i,\ell}; \bar{r}_i^{\mathsf{RE}})$ is the same as the distribution of $\mathcal{S}_i^{\mathsf{RE}}(f^{\mathsf{out}}(i))$. Therefore, by the perfect privacy of the secret sharing scheme, the shares of the honest parties corresponding to the reconstruction of the output of $\hat{f}_i$ in Hybrid 3, have the same distribution as the shares generated by the process.

We conclude that the view in Hybrid 3 is $(1-\epsilon)$-statistically-close to the view in the ideal-world.

B.2.6 Honest parties' outputs

We say that the view of the environment is *good* if for every pair of commitments C and C' and any opening o the following holds: 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 of the commitment scheme it follows that the view is good with probability at least $1 - \epsilon$.

When the view is good then in both worlds the output can be extracted from the view by the following efficient process. The process extract the values $x^1, \ldots, x^m$ as follows: (1) for an honest P_i let $\mathbf{x}(i)$ the inputs of P_i according to the view, (2) for a corrupt P_i in L let $(C_{\mathbf{x}(i)}, o_{\mathbf{x}(i)})$ be the inputs of P_i to $\mathcal{F}_{i,1}$, and let $\mathbf{x}(i) := \mathsf{open}(C_{\mathbf{x}(i)}, o_{\mathbf{x}(i)})$, and (3) for a corrupt P_i not in L, let $\mathbf{x}(i)$ be the all-zero string. The process outputs $f(x^1, \ldots, x^m)$. It is not hard to see that when the view is taken from the ideal-world, the process outputs the output of the honest parties in the ideal-world.

We continue with the analysis of the real-world. In Round 1, every honest party follows the protocol. For every corrupt party P_i that abort, the parties set $\mathbf{x}(i)$ and r_i to be the all-zero string, and so they hold all of its outgoing messages in Π^{sm}. Consider a corrupt party P_i that did not abort in the simulation of Π^{fs} in Round 1, let $\mathbf{x}(i) := \mathsf{open}(C_{\mathbf{x}(i)}, o_{\mathbf{x}(i)})$, and $(R_{i,0}, R_{i,1}) := \mathsf{open}(C_{R_i}, o_{R_i})$, where $(C_{\mathbf{x}(i)}, o_{\mathbf{x}(i)})$ and (C_{R_i}, o_{R_i}) are provided to $\mathcal{F}_{i,1}$ by P_i. Let $(a_{ij})_{j \in \{1,\ldots,n\}}$ be the messages of P_i in Π^{sm} with input $\mathbf{x}(i)$ and randomness r_i, let $a_{ij}[k]$ be the k-th share of a_{ij}, when sampled using $R_{i,1}$, and let $\vec{s}_{ij}$ be the vector of j-th shares of the randomized encoding, which is sampled according to Π^{fs} using $R_{i,1}$. Then, for every P_j the following holds.

- If $P_j \notin L_i$, the message that correspond to P_j is $(a_{ij}, (a_{ik}[j])_{k \in \{1,\ldots,n\}}, \vec{s}_{ij}) + \rho_{ji}$, where $\rho_{ij} := \mathsf{open}(C_{ji}, o_{ji})$, where o_{ji} is provided to $\mathcal{F}_{i,1}$ as the Round 0 message from P_j. Since the view is good, for every honest P_j, the pad ρ_{ji} is the same pad that P_j sampled in Round 0.

- In addition, for every $P_j \in L_i$, the message that correspond to P_j is $(a_{ij}, (a_{ik}[j])_{k\in\{1,\dots,n\}}, \vec{s}_{ij})$.

In Round 2, observe that all honest P_i's are in L. In addition, for every P_i in L the following holds: (1) Every honest P_j provides the correct shares of the output of the randomized encoding, and (2) since the view is good, every corrupt P_j either provides the correct shares of the randomized encoding, or is considered as an aborting party. Since there are at least $t+1$ honest parties, there are at least $t+1$ shares, and the honest parties can recover the output of the randomized encoding. By the perfect correctness of the randomized encoding, we conclude that the output is the output of f_i on inputs (1) the bits $L_1[i], \dots, L_n[i]$, so that $L_j[i] = 1$ if P_j aborted in Round 1 (in which case the parties set L_j to be the set of all parties) or if P_j did not abort and P_i is in L_j, (2) the messages $(A_{ji})_{j\in\{1,\dots,n\}}$, so that $A_{ji} = a_{ji}$ if P_j aborted in Round 1 or if P_j did not abort and $P_i \in L_j$, and $A_{ji} = a_{ji} + \rho_{ij}$ if P_j did not abort and $P_i \notin L_j$, (3) the input $\mathbf{x}(i)$ and randomness r_i of P_i in the simulation of Π^{sm}, and (4) pads $\eta_{i1}, \dots, \eta_{in}$. Therefore, the output is exactly the broadcast message b_i of P_i in Round 2 of Π^{sm}, when holding $(\mathbf{x}(i), r_i, (a_{ji})_{j\in\{1,\dots,n\}})$.

In addition, since the view is good, for every corrupt P_i which is not in L, the messages $(a_{ji})_{j\in\{1,\dots,n\}}$ are being recovered in Round 2, so the parties recover the broadcast message b_i of P_i in Round 2 of Π^{sm}, when holding $(\mathbf{x}(i), r_i, (a_{ji})_{j\in\{1,\dots,n\}})$. We conclude that the output in protocol dtc is the same output as in an execution of Π^{sm} where P_i holds input $\mathbf{x}(i)$ and randomness r_i. By the perfect correctness of Π^{sm} against a semi-malicious adversary, it follows that the output is $f(x^1, \dots, x^m)$, which is exactly the output of the process. This concludes the proof of security.

C Proof of Theorem 5.1

In the following section we provide formal security proof for protocol vss. 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 always assume that the adversary is the dummy adversary (see Section A.1).

C.1 Additional Preliminaries

C.1.1 Digital Signature Scheme

We begin with the definition of a digital signature scheme. The following definition is adopted from [Gol04].

Definition C.1 (Digital signature scheme.). *A digital signature scheme is a triple* (Gen, Sign, Vrfy) *of probabilistic polynomial-time algorithms satisfying the following properties.*

- *On input 1^κ, algorithm* Gen *(called the* key generator *outputs a secret signature key sk and a public verification key vk.*
- (Correctness) *For every pair (sk, vk) in the image of* $\mathsf{Gen}(1^\kappa)$*, and every $w \in \{0,1\}^*$, algorithm* Sign *and* Vrfy *satisfy*
$$\Pr[\mathsf{Vrfy}_{vk}(w, \mathsf{Sign}_{sk}(m)) = 1] = 1,$$
where the probability is taken over the internal coin tosses of algorithms Sign *and* Vrfy*.*

- (Unforgeability) *For every non-uniform probabilistic polynomial time oracle machine M, there exists a negligible function $\mu(\cdot)$, such that for all sufficiently large n it holds that*

$$\Pr\left[\begin{array}{l}\mathsf{Vrfy}_{vk}(m,\sigma)=1\ \&\ m\notin Q_M^{\mathsf{Sign}_{sk}(\cdot)}\\ \textit{where } (sk,vk)\leftarrow\mathsf{Gen}(1^\kappa) \textit{ and } (m,\sigma)\leftarrow M^{\mathsf{Sign}_{sk}(\cdot)}(vk)\end{array}\right]<\mu(\kappa),$$

where the probability is taken over the randomness of $\mathsf{Gen}, Vrfy$ *and* Sign*, as well as the random coins of M, and $Q_M^{\mathsf{Sign}_{sk}(\cdot)}$ is the set of queries that M makes to* $\mathsf{Sign}_{sk}(\cdot)$.

C.1.2 Bivariate Polynomials

We continue with a useful fact regarding bivariate polynomials (see., e.g., [AL17]).

Fact C.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 $k\in K$.*

C.2 The Simulator

We continue with the proof that protocol vss UC-emulates $\mathcal{F}_{\mathsf{vss}}$. 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, and acts as follows. We split into cases.

C.2.1 Honest D

Round 1. The simulator receives the polynomials $f_i(x):=F(x,i)$, for every $i\in\mathsf{C}$, from $\mathcal{F}_{\mathsf{vss}}$. The simulator samples $(sk,vk)\leftarrow\mathsf{Gen}(1^\kappa)$, and broadcasts vk on behalf of D. For every corrupt P_i, the simulator does as follows.

- For every $j\in\{1,\ldots,n\}$, the simulator computes $w_{ij}:=(i,j,f_i(j))$ and $\sigma_{ij}\leftarrow\mathsf{Sign}_{sk}(w_{ij})$, and sends $(f_i(x),\sigma_{i1},\ldots,\sigma_{in})$ to P_i on behalf of D.
- For every $j<i$ such that P_j is honest, the simulator samples randomness r_{ji} for the randomized encoding $\hat{g}^{ji}$ and sends r_{ji} to P_i on behalf of P_j.

This concludes the communication from honest parties to corrupt parties. At this stage the simulator receives from $\mathcal{A}$ the messages from the corrupt parties to the honest parties. For a corrupt P_i and an honest P_j with $i<j$, we denote by r_{ij} the randomness that P_i sends to P_j.

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

- The simulator sets $\mathsf{flag}_i=0$ and broadcasts flag_i on behalf of P_i.
- For every honest P_j with $i<j$, the simulator samples $(z_A^{ij},z_B^{ij})\leftarrow\mathcal{S}^{\mathsf{RE}}(\text{“equal”})$, where $\mathcal{S}^{\mathsf{RE}}$ is the simulator of the randomized encoding $\hat{g}$. The simulator broadcasts z_A^{ij} on behalf of P_i, and , z_B^{ij} on behalf of P_j.

- For every corrupt P_j with $i < j$, the simulator sets $w_{ij} := (i, j, f_j(i))$ and $\sigma_{ij} \leftarrow \mathsf{Sign}_{sk}(w_{ij})$. The simulator computes $z_A^{ij} := \hat{g}_A(\mathsf{flag}_i, w_{ij}, \sigma_{ij}; r_{ij})$ and broadcasts z_A^{ij} on behalf of P_i.
- For every corrupt P_j with $j < i$, the simulator sets $w_{ij} := (i, j, f_j(i))$ and $\sigma_{ij} \leftarrow \mathsf{Sign}_{sk}(w_{ij})$. The simulator computes $z_B^{ji} := \hat{g}(\mathsf{flag}_i, w_{ij}, \sigma_{ij}; r_{ij})$ and broadcasts z_B^{ji} on behalf of P_i.

This concludes the communication from honest parties to corrupt parties. Finally, the simulator receives from $\mathcal{A}$ the messages from the corrupt parties to the honest parties. 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. We begin by showing that the view of $\mathcal{Z}$ in the real-world has the same distribution as the view of $\mathcal{Z}$ in the ideal world.

$\mathcal{Z}$'s view. is not hard to see that the view of $\mathcal{Z}$ in the first round is identical in both worlds. Fix this view. In Round 2, every honest P_i broadcasts $\mathsf{flag}_i = 0$ in both worlds. In addition, in the real-world for every pair of honest parties P_i and P_j the value Out_{ij} is "equal". Hence, the perfect privacy of the randomized encoding implies that the output of $\hat{g}_{ij}$ which is broadcasted by P_i and P_j is has the same distribution in both worlds. Finally, for every honest P_i and corrupt P_j, the broadcast of P_i corresponding to the inconsistency check of P_i and P_j is fixed, and equal in both worlds. This concludes the analysis of $\mathcal{Z}$'s view.

Honest parties' outputs. It remains to analyse the outputs of the honest parties. In the ideal-world every honest party P_i output $F(x, i)$.

We continue with the analysis of the real-world. We say that the view of $\mathcal{Z}$ is *good* if one of the following holds for every pair of parties P_i and P_j:

- $\mathsf{Out}_{ij} =$ "equal".
- $\mathsf{Out}_{ij} =$ ("not equal", $(w'_{ij}, \sigma'_{ij}), (w'_{ji}, \sigma'_{ji})$) so that the following two conditions hold.
  - Either (1) the first entry of w'_{ij} is not i, or (2) the second entry of w'_{ij} is not j, or (3) $\mathsf{Vrfy}_{vk}(w'_{ij}, \sigma'_{ij}) = 0$, or (4) $w'_{ij} = (i, j, F(i, j))$.
  - Either (1) the first entry of w'_{ji} is not j, or (2) the second entry of w'_{ji} is not i, or (3) $\mathsf{Vrfy}_{vk}(w'_{ji}, \sigma'_{ji}) = 0$, or (4) $w'_{ji} = (j, i, F(j, i))$.
- $\mathsf{Out}_{ij} = ((\mathsf{flag}_i, w'_{ij}, \sigma'_{ij}), (\mathsf{flag}_j, w'_{ji}, \sigma'_{ji}))$, so that the following two conditions hold.
  - Either (1) the first entry of w'_{ij} is not i, or (2) the second entry of w'_{ij} is not j, or (3) $\mathsf{Vrfy}_{vk}(w'_{ij}, \sigma'_{ij}) = 0$, or (4) $w'_{ij} = (i, j, F(i, j))$.
  - Either (1) the first entry of w'_{ji} is not j, or (2) the second entry of w'_{ji} is not i, or (3) $\mathsf{Vrfy}_{vk}(w'_{ji}, \sigma'_{ji}) = 0$, or (4) $w'_{ji} = (j, i, F(j, i))$.

Observe that, by the unforgeability property of the signature scheme, the probability that the view is good is at least $1 - \mathsf{negl}(\kappa)$. Fix any good view, and observe that (1) D is not discarded in the inconsistency check, (2) all honest parties are happy, so $|L| \geq n - t$, and (3) for every corrupt unhappy P_i, and every $P_j \in L_i$ it holds that Out_{ij} is $(\mathsf{flag}_i, \bar{w}_{ij}, \bar{\sigma}_{ij})$ and $(\mathsf{flag}_j, \bar{w}_{ji}, \bar{\sigma}_{ji})$, with

$w_{ji} = (j, i, F(i,j))$, so the values $\{f_{ji}\}_{P_j \in L_i}$ are consistent with the degree-t polynomial $F(x,i)$. We conclude that D is not discarded, and that every honest P_i outputs $F(x,i)$, just like in the ideal-world. This concludes the analysis of an honest dealer.

C.2.2 Corrupt D

Since the honest parties hold no inputs, the simulator can take their role in the execution of the protocol, and perfectly simulate their behaviour. This is done as follows.

Round 1. The simulator takes the role of every honest P_i, and sends its messages to the corrupt parties and the other simulated honest parties. At the end of the first round, the simulator receives from $\mathcal{A}$ the messages from corrupt parties to honest parties, and transfers them to the simulated honest parties.

Round 2. The simulator continues the simulation of the honest parties, and computes the messages that every P_i sends. At the end of the first round, the simulator receives from $\mathcal{A}$ the messages from corrupt parties to honest parties, and transfers them to the simulated honest parties.

Communication with $\mathcal{F}_{\mathsf{vss}}$. The simulator computes the output of the honest parties in the execution. Let $f_i(x)$ be the output of an honest P_i. It will follow from the analysis that the polynomials $\{f_i(x)\}_{i \in \mathsf{H}}$ define a symmetric bivariate polynomial $F(x,y)$ of degree at most t in each variable. The simulator sends $F(x,y)$ to $\mathcal{F}_{\mathsf{vss}}$. This concludes the simulation.

Fix a polynomial-time environment $\mathcal{Z}$ with input ζ. It is not hard to see that the view of $\mathcal{Z}$ is perfectly simulated. Therefore, it is enough to show that in every execution of the protocol, the outputs of the honest parties are consistent with some symmetric bivariate polynomial $F(x,y)$ of degree at most t in each variable.

Observe that the honest parties always agree on whether D was discarded, and that whenever D is discarded the outputs of the honest parties are consistent with the zero-polynomial $F(x,y) = 0$. Therefore, we consider an execution where D was not discarded. Since D is not discarded, then $|L| \geq n - t$, which means that L contains at least $(n-t) - t \geq t+1$ honest parties, whose output polynomials are consistent with each other, i.e., $f_i(j) = f_j(i)$ for every honest $P_i, P_j \in L$. (Indeed, if there exist a pair of honest parties P_i and P_j that are happy, and whose polynomials are not consistent, then D would be discarded in the inconsistency check of P_i and P_j.) Since L contains at least $t+1$ honest parties that are consistent with each other, then, by Fact C.2, their polynomials define a symmetric bivariate polynomial $F(x,y)$ of degree at most t in each variable, such that $F(x,i) = f_i(x)$ for every honest P_i in L.

It remains to show that for every honest P_i outside L (i.e., P_i is unhappy), it holds that the output polynomial of P_i is $F(x,i)$. Observe that P_i recovers the polynomial $f_i(x)$ from the values $\{f_{ji}\}_{P_j \in L_i}$, and that all happy honest parties are in L_i. Therefore, L_i contains at least $t+1$ honest parties P_j for which $f_{ji} = F(i,j)$. Therefore, it must hold that $f_i(x) = F(i,x) = F(x,i)$ or otherwise $f_i(x)$ has degree more then t, which means that D is discarded, in contradiction. This completes the proof.