On the Communication Efficiency of Statistically-Secure Asynchronous MPC with Optimal Resilience[*]

Ashish Choudhury[†] Arpita Patra[‡]

Abstract

Secure *multi-party computation* (MPC) is a fundamental problem in secure distributed computing. An MPC protocol allows a set of n mutually distrusting parties with private inputs to securely compute any publicly-known function of their inputs, by keeping their respective inputs as private as possible. While several works in the past have addressed the problem of designing communication-efficient MPC protocols in the *synchronous* communication setting, not much attention has been paid to the design of efficient MPC protocols in the *asynchronous* communication setting. In this work, we focus on the design of efficient *asynchronous* MPC (AMPC) protocol with *statistical* security, tolerating a *computationally unbounded* adversary, capable of corrupting up to t parties out of the n parties. The seminal work of Ben-Or, Kelmer and Rabin (PODC 1994) and later Abraham, Dolev and Stern (PODC 2020) showed that the *optimal resilience* for statistically-secure AMPC is $t < n/3$. Unfortunately, the communication complexity of the protocol presented by Ben-Or et al is significantly high, where the communication complexity per multiplication is $\Omega(n^{13}\kappa^2 \log n)$ bits (where κ is the statistical-security parameter). To the best of our knowledge, no work has addressed the problem of improving the communication complexity of the protocol of Ben-Or at al. In this work, our main contributions are the following.

- We present a new statistically-secure AMPC protocol with the *optimal resilience* $t < n/3$ and where the communication complexity is $\mathcal{O}(n^4\kappa)$ bits per multiplication. Apart from improving upon the communication complexity of the protocol of Ben-Or et al, our protocol is relatively simpler and based on very few sub-protocols, unlike the protocol of Ben-Or et al which involves several layers of sub-protocols. A central component of our AMPC protocol is a new and simple protocol for verifiable *asynchronous complete secret-sharing* (ACSS), which is of independent interest.
- As a side result, we give the security proof for our AMPC protocol in the standard *universal composability* (UC) framework of Canetti (FOCS 2001, JACM 2020), which is now the defacto standard for proving the security of cryptographic protocols. This is unlike the protocol of Ben-Or et al, which was missing the formal security proofs.

1 Introduction

A central concept in secure distributed computing introduced by Yao [51] is that of secure *multi-party computation* (MPC), which states that any distributed computation among mutually-distrusting participants which can be performed in the *presence* of a centralised trusted entity, can also be performed "securely" in the *absence* of any such trusted entity, by running a protocol among the participants themselves. Informally, an MPC protocol allows a set of n mutually-distrusting parties to perform a joint computation on their inputs, while keeping their inputs as private as possible. Due to its powerful abstraction, ever since its inception [35, 12, 21, 49], the MPC problem has been widely studied in various communication models and adversarial settings and several interesting results have been obtained regarding the theoretical possibility and feasibility of secure MPC (see for instance, [38, 33, 42, 40] and their references).

[*]This paper combines the results of [43] and [22]; this is the full and elaborate version with full security proofs.

[†]International Institute of Information Technology, Bangalore India. Email: ashish.choudhury@iiitb.ac.in.

[‡]Indian Institute of Science, Bangalore, India. Email: arpita@iisc.ac.in.

Synchronous vs Asynchronous Networks: Most of the works for MPC consider the *synchronous* communication setting, where the parties are synchronized by a *global* clock and where there exists a *publicly-known* upper bound on the message delays over the channels among the parties. That is, each party knows beforehand how long it has to wait for an expected message during the execution of a protocol, and if the message does not arrive within that time-bound, then it can be concluded that the sending party is *corrupt*. Unfortunately, it is impossible to ensure such strict time-outs in real-world networks like the Internet, where the communication channels may have arbitrary delays. Such networks are more appropriately captured by the *asynchronous* communication setting [11, 16], where there does not exist any global clock and the messages of the parties can be *arbitrarily* (though finitely) delayed. The only guarantee in this model is that any message sent by a party is *eventually* delivered and is not "stuck" in the network forever. Apart from a better modelling of real-world networks, asynchronous protocols have the advantage of running at the *actual* speed of the underlying network. More specifically, for a synchronous protocol, the participants have to pessimistically set the global delay Δ to a large value to ensure that the messages sent by every party at the beginning of each round reaches their destination within the Δ time frame. But if the *actual* network delay δ is such that $\delta << \Delta$, then the protocol fails to take advantage of the faster network and its running time will be proportional to Δ.

Even though the asynchronous communication model is practically more relevant, the biggest challenge in designing a *completely* asynchronous protocol is that during the protocol execution, a slow sender *cannot* be distinguished from a corrupt sender. That is, if a party does not receive an expected message, then unlike the synchronous setting, the receiving party cannot decide whether the sender is corrupt (and has not sent the message) or slow (it has sent the message, but it is delayed). Consequently, if t is the upper bound on the maximum number of corruptions in the system, then in a completely asynchronous protocol, a party can afford to receive messages from at most $n - t$ parties and has to proceed to the next step, to avoid endless waiting. However, in this process, the communication from up to t potentially honest parties may have to be ignored. Due to this phenomena, any synchronous protocol fails completely when executed in a completely asynchronous environment, as the security of the synchronous protocols depend upon the fact that the messages of *all* the honest parties are considered. In fact, there are sufficient evidences that asynchrony coupled with the possibility of arbitrary faults in a network, has devastating consequences on the computational capabilities of the network. For instance, the classic FLP impossibility result [32] completely rules out the possibility of deterministic protocols in the asynchronous communication setting, even for the basic task of distributed consensus, where unlike MPC, the privacy of the inputs of the participants need not be preserved.

Unconditionally-secure MPC: One of the earliest distinctions made in the domain of MPC is based on the computing power of corrupt participants. If the computing power of the corrupt parties is bounded (namely polynomial time), then the notion of security achieved is termed as *cryptographic* (also known as *computational* security) [51, 35]. On the other hand, *unconditionally-secure* MPC protocols [12, 21, 49] provide security even when the corrupt parties are *computationally unbounded*, provided the parties are connected by pair-wise private and authentic channels. The advantage of the latter class of protocols is two-fold. First, they provide ever-lasting security, as the underlying security is not based on any computational-hardness assumptions. Second, the primitives used in such protocols are computationally very simple and faster by several order of magnitude, compared to the cryptographic primitives used in cryptographically-secure MPC protocols. Unconditionally-secure protocols can be further divided into the class of *perfectly-secure* protocols where all the expected security properties are achieved in an error-free fashion and *statistically-secure* protocols where all the properties are achieved except with some negligible error probability of $2^{-\Omega(\kappa)}$, for a given statistical-security parameter κ.

The standard approach used in any generic unconditionally-secure MPC protocol is that of "shared circuit-evaluation" based on secret-sharing, first pioneered in [35, 12] and then subsequently followed in all generic unconditionally-secure MPC protocols. In this approach, it is assumed that the function f to be securely computed by the parties is expressed as a publicly known arithmetic circuit cir over some algebraic structure, which is typically a finite field $\mathbb{F}$. The circuit cir consists of addition (linear) gates and multiplication (non-linear) gates. The parties then "securely" evaluate each gate in cir in a shared/distributed fashion. More specifically, let t be the upper bound on the maximum number of corrupt parties, which are under the control of a centralized entity called *adversary*. Then each party secret-shares its inputs (for the function f) among all the parties using a *linear secret-sharing* scheme (LSS) [29] say Shamir's [50], with threshold t, where each party obtains a share of the underlying shared value. Such shared values are called *completely* t-shared (see Definition 2.3). Informally such a scheme ensures that the shared value remains information-theoretically secure even if up to t shares are compromised, while the shared value can be reconstructed uniquely if $t+1$ correct shares are available. The parties then maintain the following *invariant* for each gate in the circuit: *given completely t-shared inputs of the gate, the gate output is computed and made available among the parties in a completely t-shared fashion at the end of the gate evaluation*. Moreover, the shared gate evaluation does not reveal any additional information about the gate inputs and output. Once all the gates are evaluated as per the above invariant, the circuit output, which will be available in a completely t-shared fashion among the parties is publicly reconstructed, using the reconstruction algorithm of the underlying LSS. Intuitively, the privacy of the entire computation follows since each intermediate value during the shared circuit-evaluation remains secret-shared with threshold t.

Evaluating an addition gate as per the above invariant is easy and requires no interaction among the parties. Specifically, addition gates can be evaluated *locally* by the parties, due to the linearity property of the underlying LSS, which guarantees that given completely t-shared values a and b, and publicly known constants, say c_1 and c_2, then to compute a complete t-sharing of $c_1a + c_2b$, each party P_i just needs to locally compute $c_1a_i + c_2b_i$, where a_i and b_i are respectively the shares of a and b held by P_i, as part of complete t-sharing of a and b. However, evaluating the multiplication gates as per the above invariant requires interaction among the parties. The focus therefore is rightfully placed on measuring the total number of field elements communicated by the honest parties (called the *communication complexity*) during the evaluation of the multiplication gates in an MPC protocol. More specifically, the focus is on what is called the *amortized* communication complexity per multiplication gate, where the amortized communication complexity is derived under the assumption that the circuit is large enough so that the terms that are independent of the circuit size can be ignored.

1.1 Our Motivation and Results

In this work, we study the MPC problem in the *most powerful* adversarial model. Namely we consider a *computationally unbounded Byzantine* (malicious adversary), who can corrupt any t parties out of the n parties during the execution of a protocol and can force them to deviate from the protocol instructions in any arbitrary fashion. Moreover, we consider the more stronger *asynchronous* communication setting. The theoretical possibility and feasibility of unconditionally-secure *asynchronous* MPC (AMPC) protocols have been studied in the past. As discussed below, compared to synchronous counterparts, unconditionally-secure AMPC protocols perform badly in several aspects.

- *Resilience*: It is the number of faults t which can be tolerated by a given protocol. As shown in Table 1, the *optimal* resilience bounds (namely the maximum number of tolerable faults) are better for the synchronous MPC protocols, compared to AMPC protocols.
- *Communication Complexity Per Multiplication*: An enormous amount of research has been done in the literature to design *optimally-resilient* unconditionally-secure MPC protocols with lower (amortized) communication complexity per multiplication (see for instance [38, 8, 30, 14, 9, 10, 24] and their references). Specially, the focus has been to achieve a communication complexity of $\mathcal{O}(n)$ field elements

Type of Security	Communication Setting	Optimal Resilience	Reference
Perfect	Synchronous	$t < n/3$	[12]
	Asynchronous	$t < n/4$	[11]
Statistical	Synchronous	$t < n/2$	[49]
	Asynchronous	$t < n/3$	[13, 2]

Table 1: Optimal resilience bounds for unconditionally-secure MPC protocols.

per multiplication gate. Such protocols are considered "scalable" in the sense that for securely evaluating each multiplication gate, the communication done *per-party* is a *constant* and independent of the number of parties. As shown in Table 2, while scalable protocols with *optimal resilience* have been obtained in the synchronous communication setting, comparatively the communication complexity of the most efficient *optimally-resilient* AMPC protocol is high.

Security	Communication Setting	Resilience	Communication Complexity (in bits)	Reference
Perfect	Synchronous	$t < n/3$	$\mathcal{O}(n \log n)$	[36]
	Asynchronous	$t < n/4$	$\mathcal{O}(n^2 \log n)$	[45]
Statistical	Synchronous	$t < n/2$	$\mathcal{O}(n\kappa)$	[37]
	Asynchronous	$t < n/3$	$\Omega(n^{13}\kappa^2 \log n)$	[13]
	Asynchronous	$t < n/3$	$\mathcal{O}(n^4\kappa)$	This work

Table 2: Communication complexity per multiplication gate of the most efficient unconditionally-secure MPC protocols with optimal resilience. The perfectly-secure protocols operate over a field of size at least n where each field element is of size $\mathcal{O}(\log n)$ bits. Statistically-secure protocols operate over a field of size at least 2^κ where each field element is of size $\mathcal{O}(\kappa)$ bits and where κ is the statistical-security parameter.

As evident from Table 2, there is an enormous gap between the communication complexity of optimally-resilient statistically-secure synchronous and asynchronous MPC protocols. Quite a few works have addressed the problem of designing communication-efficient, optimally-resilient, statistically-secure MPC protocols in the *synchronous* setting [48, 28, 8, 14, 37]. Unfortunately, to the best of our knowledge, no attention has been paid to the design of communication efficient statistically-secure AMPC protocol with the optimal resilience of $t < n/3$, ever since the inception of the problem in [13]. Given the practical relevance of the asynchronous communication setting, the main motivation of our work is to improve this unfortunate state-of-affair by attempting to design a simple yet communication-efficient, optimally-resilient, statistically-secure AMPC protocol. Specifically, our main contributions in this paper are the following.

- In this work, we present a *new* statistically-secure AMPC protocol with the optimal resilience of $t < n/3$, which significantly improves upon the communication complexity of the protocol of [13]. Namely, the (amortized) communication complexity per multiplication gate of our protocol is $\mathcal{O}(n^4\kappa)$ bits, thus obtaining many-fold improvement over the protocol of [13]. Apart from better communication complexity, our protocol is conceptually very simple compared to the protocol of [13]. A detailed technical comparison of our protocol with the protocol of [13] appears in the sequel.
- The security of the underlying protocols of [13] is based on the "property-based" definition. Loosely speaking, in this definition the security of a cryptographic protocol is defined by enumerating a list of required security goals that the protocol should achieve.[1] However, as shown in [17, 18], the property-

[1]The typical security goals are that of **(a) Privacy:** which guarantees that the adversary does not learn anything "additional" about the inputs of the honest parties, beyond what can be inferred from the inputs and outputs of the corrupt parties; **(b) Cor-**

based security definition is not rigorous and there are several associated shortcomings. For instance, the definition *does not* guarantee that all the required security goals are indeed enumerated. More importantly, the definition does not say anything about what security guarantees are achieved, when multiple instances of a protocol (which is proved to be secure as per the property-based definition) are composed and executed in parallel. Motivated by these shortcomings, [17, 18] proposed the *Universal Composability* (UC)-security framework for defining and proving the security of MPC protocols (see Sec 2 for more details). Loosely speaking, in this definition, a *real-world* MPC protocol is considered to be secure if it "emulates" an *ideal-world* MPC protocol, where the parties privately provide their respective inputs to a centralized *trusted third-party* (TTP), who upon receiving the function inputs compute the function output and sends it back to the respective parties.

In the *synchronous* communication setting, the current defacto standard is to prove the security of an MPC protocol as per the more rigorous UC-security definition. However, to the best of our knowledge, the security of al most all unconditionally-secure AMPC protocols [9, 23, 45, 24, 46] has been proved using the less rigorous property-based definition, due to the additional technicalities introduced due to the UC proofs.[2] As a side result, in this work, we prove the security of our AMPC protocol according to the UC-security definition. The conceptual simplicity of our AMPC protocol, coupled with the involvement of fewer sub-protocols simplifies the overall process of giving a UC-security proof for our protocol. On contrary, the overall AMPC protocol of [13] is quite involved as it deploys several layers of sub-protocols. Even though it is not clear whether a UC-security proof for the protocol of [13] can be given, we believe that any such attempted proof will be highly involved.

On the Efficiency Gap Between Optimally-Resilient Perfectly-Secure and Statistically-Secure AMPC: From Table 2, one could see the efficiency gap that still exists between the communication complexity of *optimally-resilient* AMPC protocols with perfect and statistical security; while for perfect AMPC it is $\mathcal{O}(n^2 \log n)$ bits per multiplication, for statistical AMPC it is settled to $\mathcal{O}(n^4\kappa)$ due to this work. The main reason for this efficiency gap is the heavy communication complexity of the underlying (asynchronous) secret-sharing scheme deployed in the statistical AMPC protocol. In a more detail, for the case of perfect security, the optimal resilience bound is $t < n/4$, implying the presence of "more" number of honest parties, compared to the statistically-secure protocols with optimal resilience where $t < n/3$. Consequently, designing asynchronous secret-sharing scheme becomes relatively easier with $t < n/4$, compared to $t < n/3$, as discussed in details in the next section. The secret-sharing scheme deployed in [13] is highly *inefficient*, with an expensive communication complexity of $\Omega(n^{12}\kappa^2 \log n)$ bits. In a preliminary version of this paper [43], we significantly improved the secret-sharing scheme of [13] by presenting a completely new and relatively simpler secret-sharing scheme with $t < n/3$ and with communication complexity of $\mathcal{O}(n^4\kappa)$ bits. Then in a follow-up work [22], it was noticed that a slight modification of the secret-sharing scheme of [43] further leads to an improvement, resulting in a communication complexity of $\mathcal{O}(n^3\kappa)$ bits. Any further improvement in the secret-sharing scheme will automatically result in further improvements in the resultant AMPC protocol, thus narrowing the efficiency gap between the two classes of AMPC protocols.

rectness: which guarantees that the honest parties always learn the correct function output, irrespective of the behaviour of the corrupt parties; **(c) Termination**: which guarantees that the honest parties eventually complete the protocol and **(d) Independence of Inputs**: which guarantees that adversary should not be able to base the inputs of the corrupt parties for the protocol, based on the inputs provided by the honest parties in the protocol.

[2]The works of [26, 27] presented *cryptographically-secure* AMPC protocols with UC-security proofs. However, those proofs are not applicable for our protocol, since we aim for unconditional security. The work of [13] which was published as an extended abstract does refer to [16] for the real-world/ideal-world based security definition. However, the *exact* security proofs for the AMPC protocol of [13] and its underlying sub-protocols as per the UC style have *not* been worked out.

1.2 Technical Challenges, Detailed Technical Overview and Comparison with [13]

As discussed earlier, the main tool used in any unconditionally-secure MPC protocol is that of complete t-sharing. A complete t-sharing of a value can be generated by Shamir's secret-sharing protocol [50], which allows a designated party called *dealer* (denoted by D) to generate a complete t-sharing of its private input s.[3] However, the protocol is designed to deal with only *passive* corruptions (where all the parties including D follow the protocol instructions) and the *synchronous* communication setting. As observed in [13], there are inherent challenges in extending the protocol to the asynchronous communication setting against a *malicious* adversary (where even D can be potentially corrupt and may not distribute "consistent" shares) with $t < n/3$. To understand these challenges, let us consider the setting where $n = 3t + 1$, which is the *smallest* value of n, satisfying the condition $t < n/3$.

In any secret-sharing protocol tolerating a malicious adversary, the parties need to interact and verify whether a *potentially corrupt* D has distributed correct shares, without revealing anything additional about the shares in case D is *honest*. However, in the asynchronous communication setting, we can get the "positive confirmation" about the distribution of correct shares from at most $n-t$ parties, say $\mathcal{W}$, as up to t potentially corrupt share-holders may not respond during the verification process, even if D is *honest*. Hence, the resultant sharing will be *incomplete* in the sense that up to t *honest* parties outside $\mathcal{W}$ may not possess their respective shares. Later, if the parties want to reconstruct such incompletely t-shared value, then they can afford to wait for at most $|\mathcal{W}| - t = n - 2t = t + 1$ share-holders in $\mathcal{W}$ to produce their shares. This is because up to t share-holders in $\mathcal{W}$ may be *corrupt* who may not respond during the reconstruction phase. However, among these $t + 1$ share-holders, up to t could be potentially *corrupt* who may provide *incorrect* shares and hence the reconstructed value will be completely incorrect.

Another major problem with "incomplete" secret-sharing is that it is *not* suitable for performing shared circuit-evaluation, as the set $\mathcal{W}$ of $n - t$ share-holders may *not* be the same for all the shared values during the computation. For instance, let a and b be two values, owned by parties P_i and P_j respectively and suppose the goal is to securely compute $a + b$. Then P_i and P_j will independently act as dealers and secret-shares theirs inputs a and b respectively. If the underlying secret-sharing protocol is *incomplete*, then the set $\mathcal{W}_a$ of $n - t$ share-holders of a and the set $\mathcal{W}_b$ of $n - t$ share-holders of b need not be the same. This is because the sets $\mathcal{W}_a$ and $\mathcal{W}_b$ are determined asynchronously during independent instances of the underlying secret-sharing protocol. In the worst case, the set $\mathcal{W}_a \cap \mathcal{W}_b$ will have only $t + 1$ party, who can compute their respective shares of $a + b$. And at the time of reconstruction of the shared $a + b$, the parties could afford to wait for only $(t + 1) - t = 1$ party from the set $\mathcal{W}_a \cap \mathcal{W}_b$ to provide its share of $a + b$, which is clearly not sufficient to reconstruct $a + b$.

To deal with the above problems associated with incomplete secret-sharing, [13] introduced an asynchronous primitive called *asynchronous complete secret-sharing* (ACSS).[4] An ACSS protocol (see Section 5 for the formal definition) allows D to *verifiably* generate a complete t-sharing of its private input s. The verifiability here ensures that even if D is *corrupt*, the completion of the protocol guarantees the existence of some value $\overline{s}$, such that $\overline{s}$ is completely t-shared by D. From the description, it might look like an impossible task to give an instantiation of ACSS, as in a completely asynchronous setting, there is no way to prevent the adversary from delaying the participation of t *honest* parties and thus forcing the system to end the secret-sharing protocol without their participation. However, [13] presented a beautiful instantiation of the primitive.

Given the provision for generating complete t-sharing through ACSS, [13] could perform the shared

[3]In the protocol, to share a value s, the dealer D picks a polynomial of degree at most t, which is an otherwise a random polynomial except that its constant term is s. The share for every party then consists of a distinct point lying on the polynomial.

[4]In [13] they called this primitive as *ultimate secret-sharing* (USS). We prefer to call it complete secret-sharing, to signify that the resultant sharing is complete in the sense that *all* the (honest) share-holders will possess their respective shares at the end of the sharing.

circuit-evaluation by extending the techniques of the synchronous world [21, 48] to the asynchronous setting. We also follow the approach of deploying ACSS for shared circuit-evaluation as in [13]. However, the differences are in the exact instantiation of the ACSS and the way we deploy it for the shared circuit-evaluation. The details follow.

1.2.1 The ACSS of [13]

The instantiation of ACSS of [13] is quite complex and communication-intensive which is attributed to its dependency on several involved sub-protocols. Namely, they deploy a series of sub-protocols which are tied together to obtain their ACSS protocol. In a more detail, they followed the path AICP $\rightarrow$ ARS $\rightarrow$ AWSS $\rightarrow$ Two & Sum AWSS $\rightarrow$ AISS $\rightarrow$ ACSS to arrive at their ACSS protocol. Here $X \rightarrow Y$ denotes that protocol X is used as a sub-protocol in protocol Y. The AISS (asynchronous incomplete secret-sharing) protocol is taken from [19], which generates an *incomplete* sharing of the underlying value.[5] The AISS protocol is used as follows to get an ACSS protocol.

To generate a complete sharing of a secret s, the dealer D computes the shares $s_1, \ldots, s_n$ as per the Shamir's protocol. Then instead of directly handing over the respective shares to the corresponding share-holders, D further shares each share s_i by invoking an independent instance of AISS, which ensures that D has committed all its shares. Notice that each share is shared in an incomplete fashion as it is shared using AISS. Next D interacts with the parties and "proves" that the n committed shares are indeed "valid" Shamir-shares (namely all the n shares are distinct evaluations of a degree-t polynomial), without disclosing anything about the actual shares. This is done by deploying the cut-and-choose zero-knowledge protocol of [21]. Upon successful completion of the zero-knowledge protocol, the committed share s_i is reconstructed towards the designated share-holder P_i. The designated reconstruction will be successful, even if D is potentially corrupt, thus guaranteeing that each honest party eventually obtains its designated share, leading to a complete sharing of D's input. Hence the complexity of the ACSS protocol of [13] is equivalent to that of n instances of the AISS protocol of [19] plus the instances of zero-knowledge protocols. In [44], it is shown that one instance of the AISS protocol of [19] needs a communication of $\Omega(n^{11}\kappa^2 \log n)$ bits. So *excluding* the cost of zero-knowledge protocols, the ACSS protocol of [13] requires a communication of $\Omega(n^{12}\kappa^2 \log n)$ bits.

1.2.2 Our ACSS Protocol

Compared to [13], our instantiation of ACSS is relatively simpler and based on fewer primitives, thus leading to a significant gain in the communication complexity. Namely, we use a shorter path AICP $\rightarrow$ AISS $\rightarrow$ ACSS to arrive at our ACSS protocol. While our instantiation of AICP is somewhat similar to that of [19, 13], our instantiation of AISS and the way we use AISS to get ACSS is completely different from [13]. The details follow.

The *asynchronous information-checking protocol* (AICP) is used for generating *information-checking* (IC) signatures, which are information-theoretic analogue of digital signatures. We adapt the ICP of [8] designed for the synchronous setting, to the asynchronous communication setting. Using our AICP, we next design a very simple protocol for AISS, which generates what we call as *two-level secret-sharing with IC signatures* for a given secret. Loosely speaking, the protocol allows a designed dealer D to *verifiably* generate a Shamir secret-sharing of a secret s held by D with the guarantee that there exists a publicly known set of at least $n - t$ parties $\mathcal{W}$, who hold their respective shares of s. We call the parties in $\mathcal{W}$ as *primary share-holders* and the shares held by these parties as *primary-shares* of s. Apart from $\mathcal{W}$, the protocol also ensures that each primary-share s_j is further Shamir-shared among a publicly known set $\mathcal{W}_j$ of at least $n - t$

[5]In [19], the AISS protocol is called AVSS. But we prefer to call it AISS to signify that the sharing generated by the protocol is *incomplete*.

parties, with each party P_i in $\mathcal{W}_j$ possessing a *secondary-share* s_{ji} of the primary-share s_j. Furthermore, it is also ensured that the primary-share holder P_j of the share s_j holds each of the secondary-shares s_{ji}, IC-signed by the party P_i. The verifiability feature in the protocol ensures that even if D is *corrupt*, the completion of the protocol guarantees that there exists some value, say $\bar{s}$, which is secret-shared in the above fashion. We stress that the set of secondary-share holders might be different for each primary-share. Moreover, if D is *corrupt*, then the set $\mathcal{W}$ may not include all the parties. However, it will be guaranteed that if D is *honest* and if the parties keep running the protocol, then eventually all honest parties are included in the $\mathcal{W}$ set.

The AISS protocol of [19] generates a somewhat similar data structure as ours, but by involving additional primitives ARS, AWSS and Two & Sum AWSS and also by deploying interactive zero-knowledge protocols to verify that the parties are honestly following the protocol instructions. We completely avoid involving these primitives and also avoid the usage of any kind of zero-knowledge protocols. This is done by "tying" together the primary and secondary-shares in a bivariate polynomial of degree-t in two variables (see Fig 6). This idea of generating a two-level secret-sharing via bivariate polynomials has been used in many secret-sharing protocols in the *synchronous* communication setting (see for instance [20] and its references). The technique has been also used in some of the *asynchronous* secret-sharing protocols, but with the resilience $t < n/4$ (see for instance [45, 24]). The primary idea used here is that the value s which needs to be secret-shared is embedded in the constant term of a random bivariate polynomial $F(x, y)$ of degree t in x and y. The goal is then to ensure that a set $\mathcal{W}$ of at least $n - t$ parties P_j publicly confirm the receipt of univariate polynomials $f_j(x)$ of degree-t, where $f_j(x) \overset{\text{def}}{=} F(x, \alpha_j)$. And for every $P_j \in \mathcal{W}$, a set $\mathcal{W}_j$ of at least $n - t$ parties P_i publicly confirm the receipt of values $f_j(\alpha_i)$. Moreover, P_j should hold the IC-signed $f_j(\alpha_i)$ values. Here α_i is a publicly-known distinct *evaluation point*, associated with each party P_i. If the above goals are achieved, then the values $\{f_j(0)\}_{P_j \in \mathcal{W}}$ will constitute the primary-shares of s, as $F(0, y)$ constitutes the degree-t Shamir-sharing polynomial for sharing $s = F(0, 0)$. And for every $P_j \in \mathcal{W}$, the values $\{f_j(\alpha_i)\}_{P_i \in \mathcal{W}_j}$ will constitute the secondary-shares.

The standard mechanism which is used in the existing secret-sharing protocols to achieve the above goals is to let D hand over the univariate polynomials $f_i(x) \overset{\text{def}}{=} F(x, \alpha_i)$ and $g_i(y) \overset{\text{def}}{=} F(\alpha_i, y)$ to each party P_i. The polynomials $f_i(x)$ and $g_i(y)$ are also called as the *row* and *column* polynomials respectively, due to the nice two-dimensional "matrix-representation" of the values of these polynomials (see Fig 6). The parties then check whether D has distributed "consistent" row and column-polynomials to the parties, lying on a single bivariate polynomial of degree t, by performing what is called as the "pair-wise consistency checking". Namely each pair of parties P_i, P_j exchange the supposedly common values on their row and column-polynomials (if D, P_i and P_j are all *honest*, then $f_i(\alpha_j) = g_j(\alpha_i) = F(\alpha_j, \alpha_i)$ and $f_j(\alpha_i) = g_i(\alpha_j) = F(\alpha_i, \alpha_j)$ should hold) and publicly confirm the pair-wise consistency of their row and column-polynomials. The parties then check if there exists a set of at least $2t+1$ row-polynomial holders, say $\mathcal{R}$ and a set of at least $2t + 1$ column-polynomial holders, say $\mathcal{M}$ (where $\mathcal{R}$ could be different from $\mathcal{M}$), such that the pair-wise consistency between every $P_j \in \mathcal{R}$ and every $P_i \in \mathcal{M}$ has been confirmed. The existence of such an $\mathcal{R}$ and $\mathcal{M}$ set (which are bound to exist for an *honest* D) guarantees that D has distributed consistent row and column-polynomials lying on a single bivariate polynomial of degree t to the *honest* parties in $\mathcal{R}$ and $\mathcal{M}$ respectively. This follows from the property of bivariate polynomials of degree t (see Lemma 2.5) and the fact that there are at least $|\mathcal{R}| - t \geq t + 1$ *honest* parties in the set $\mathcal{R}$ and at least $|\mathcal{M}| - t \geq t + 1$ *honest* parties in the set $\mathcal{M}$. Confirming the existence of the sets $\mathcal{R}$ and $\mathcal{M}$ is very crucial, because only upon the confirmation of $\mathcal{R}$ and $\mathcal{M}$, the parties can proceed further and find out the set $\mathcal{W}$ and the sets $\mathcal{W}_j$ by performing some additional computation.

In the *synchronous* communication setting, confirming the existence of the sets $\mathcal{R}$ and $\mathcal{M}$ is relatively easier. This is because the public confirmations (both positive and negative) involving *all* the pairs of parties will be available and after resolving *all* the "negative confirmations" with the aid of D, one can easily verify the existence of the desired $\mathcal{R}$ and $\mathcal{M}$ sets as above. However, in the *asynchronous* communication setting, confirmations involving every pair of parties may not be available. Moreover, for two different parties P_j and P_k, the set of column-polynomial holders who confirm the pair-wise consistency with P_j's row-polynomial might be *different* from the set of column-polynomial holders who confirm the pair-wise consistency with P_k's row-polynomial. One could find a candidate $\mathcal{R}$ and $\mathcal{M}$ set by (asynchronously) constructing a "consistency graph" with n parties being the nodes and edges denoting the confirmation of pair-wise consistency among the corresponding parties, followed by checking the presence of a clique of size at least $n - t$ in the graph. However, this will require the parties to perform *exponential* amount of computation. The beautiful work of [11] shows how to instead look for an alternate structure in the consistency graph called (n, t)-*star*, which is a pair of subsets of nodes (C, D), with $C \subseteq D$, $|C| \geq n - 2t$ and $|D| \geq n - t$ and where there is an edge between every node in C and every node in D. In [11], it is shown how to check the presence of an (n, t)-*star* in polynomial amount of time.

The presence of a (n, t)-*star* (C, D) guarantees the existence of $\mathcal{R}$ and $\mathcal{M}$, but *only* when $t < n/4$. This is because, in this case, the cardinality of the sets C and D will be at least $2t + 1$ and $3t + 1$ respectively and hence one can consider the sets C and D as $\mathcal{R}$ and $\mathcal{M}$ respectively. On the other hand, for $t \leq n/3$, the maximum size of the set C could be only $2t$, which is not sufficient to consider it as a candidate $\mathcal{R}$ set. In fact, when $n = 3t + 1$, in the worst case, the set C could be only of size $t + 1$ and may have *only one* honest party. Hence we follow a *different* approach to check the existence of $\mathcal{R}$ and $\mathcal{M}$ sets.

In our AISS protocol, D first distributes *only* the column-polynomials to the respective parties and retains the row-polynomials. Then for *all* the n row-polynomials $f_j(x)$ held by D, it tries to get the corresponding common $f_j(\alpha_i)$ values IC-signed by the party P_i. Thus P_i gets the n values from D to be IC-signed, which are supposed to be lying on P_i's column-polynomial. Party P_i issues the signature only after checking the pair-wise consistency between the received values and its column-polynomial, and publicly announces the issuance of the signatures. The goal of D is then to get the common values on *all* the n row-polynomials signed as above by a set of at least $2t + 1$ column-polynomial holders. The set of these column-polynomial holders then constitute the $\mathcal{M}$ set. Once the values on the row-polynomials are signed, D then starts distributing the individual signed row-polynomials to respective parties. The presence of the signatures prevents a potentially corrupt D to distribute arbitrary polynomials as row-polynomials. Once a party receives a correctly signed row-polynomial, it publicly announces the same. The parties next look for a set of at least $2t + 1$ parties, who announce the receipt of their corresponding signed row-polynomials. The set of these row-polynomial holders then constitute the set $\mathcal{R}$. Once the sets $\mathcal{R}$ and $\mathcal{M}$ are confirmed, the parties then proceed to compute the set $\mathcal{W}$ and the sets $\mathcal{W}_j$, whose technical details are available in Section 4.

We associate a reconstruction protocol along with our AISS protocol, which allows any publicly-known designated party to reconstruct a value, shared by our AISS protocol. More specifically, a value s which is two-level secret-shared as above can be easily reconstructed by a designated party P_R as follows. Party P_R waits for at least $t + 1$ primary-share holders from $\mathcal{W}$ to correctly reveal their respective primary-shares. To reveal its primary-share s_j, the share-holder P_j actually reveals *all* the signed secondary-shares corresponding to the parties in $\mathcal{W}_j$. The presence of at least $t+1$ *honest* secondary-share holders in $\mathcal{W}_j$ guarantee that if P_j has revealed correctly IC-signed secondary-shares then they are correct (with overwhelming probability) and by interpolating these shares, the primary-share s_j can be correctly reconstructed. There are at least $t + 1$ honest primary-share holders in $\mathcal{W}$, who eventually complete the above revelation process and once completed, these primary-shares can be used to interpolate back the shared s.

From AISS to ACSS: Principal-wise, the road from our AISS to ACSS follows a similar path as followed in [13]. Namely, to share a value s, the dealer D generates the shares $s_1, \ldots, s_n$ as per the Shamir secret-sharing protocol. Then it further shares these shares by executing n independent instances of our AISS protocol. The parties next verify whether the shared values are valid Shamir-shares and once verified, the individual shares are then reconstructed towards the designated share-holders by using the designated reconstruction protocol associated with our AISS protocol. However, unlike [13], we *do not* deploy expensive zero-knowledge protocols to verify whether the values shared during the AISS instances are valid shares. Rather, we again use the properties of bivariate polynomials by embedding the shares $s_1, \ldots, s_n$ in a bivariate polynomial of degree t and we carefully "tie" together the various $\mathcal{W}$ sets generated during the different instances of our AISS protocol.

In a more detail, similar to our AISS protocol, D embeds the secret s in the constant term of a random bivariate polynomial of degree t and then computes the row and column-polynomials. The column-polynomials are distributed to respective parties, while the row-polynomials are treated as Shamir-sharing polynomials to share the constant term of these polynomials (which are actually the Shamir-shares of D's secret) via instances of our AISS protocol. Now as part of the AISS instances, the parties "implicitly" perform pair-wise consistency check. Namely every party P_i upon receiving the shares of the row-polynomials as part of the various AISS instances check if they lie on the column-polynomial of P_i and upon verification publicly announces the same. The goal of D is then to collect a set of at least $2t + 1$ column-polynomial holders, who have publicly confirmed the pair-wise verification *and* who also constitute a *common* $\mathcal{W}$ set for all the n instances of the AISS protocol invoked by D. We call these common set of parties as $\mathcal{V}$. Such a set $\mathcal{V}$ is always guaranteed to exist if D is *honest*, as the set of honest parties always constitute a candidate $\mathcal{V}$ set. And our AISS protocol guarantees that all honest parties are eventually included in the corresponding $\mathcal{W}$ set.

Once D finds and publicly announces a $\mathcal{V}$ set, it confirms that the row-polynomials used by D during the AISS instances and the column-polynomials held by the (honest) parties in $\mathcal{V}$ are consistent and lie on a single bivariate polynomial known to D. So the next goal is to get each row-polynomial reconstructed towards the designated share-holder, which will generate the complete secret-sharing of D secret. However, unlike our AISS protocol, we do not ask D to hand over the respective row-polynomials. Instead, they are reconstructed towards the designated share-holder by using the designated reconstruction protocol, associated with our AISS. This will ensure that the row-polynomials (and hence their constant terms) always get reconstructed by the designated party, even if D is *corrupt*. For the technical details, see Section 5.

1.2.3 Our AMPC Protocol vs AMPC Protocol of [13]

The AMPC protocol of [13] as well as ours follow the same BGW paradigm [12] of shared circuit-evaluation. However, the difference is in the way multiplication gates are evaluated. In both the AMPC protocols, each party first generates a complete-sharing of its input by executing an instance of ACSS. To avoid an indefinite wait, the parties run an instance of the *agreement on a common subset* (ACS) primitive [11, 13] to agree on a common subset $\mathcal{C}$ of $n - t$ parties, whose inputs are eventually completely-shared. The inputs of the remaining t parties are substituted as 0. The linear gates are evaluated non-interactively due to the linearity property of Shamir sharing. In [13], to evaluate a multiplication gate with shared gate inputs a and b, each party P_i secret-shares its share a_i and b_i of the secret a and b respectively and also secret-shares the product $c_i \stackrel{\text{def}}{=} a_i b_i$, all using instances of ACSS. Notice that the values $c_1, \ldots, c_n$ lie on a degree-$2t$ polynomial and hence constitute a complete-sharing of $c \stackrel{\text{def}}{=} ab$, but with degree-$2t$. Hence the goal is to securely "convert" these shares into another set of n random shares, which constitute a complete t-sharing of c. This process is termed as the "degree-reduction" process [12]. To do the degree-reduction, each party P_i interactively proves in a zero-knowledge fashion that it has followed the protocol honestly (namely it has shared the correct a_i, b_i and c_i). To avoid an indefinite wait, the parties then run an instance of ACS to agree on a common subset of $n - t$ share-holders P_i who correctly followed the protocol instructions and then apply the degree-reduction process of [12] on the c_i values shared by these $n - t$ parties.

As evident from the above description, for evaluating a *single* multiplication gate, a lot of computations and protocols are involved. Specifically, there are $\Theta(n)$ instances of ACSS and zero-knowledge protocols involved, apart from an instance of ACS, which in it self requires n invocations of an *asynchronous Byzantine agreement* (ABA) protocol [47]. Hence the number of ABA instances involved is *proportional* to the number of multiplication gates in the circuit. We follow a simpler and more efficient approach for evaluating the multiplication gates in the circuit using the Beaver's circuit-randomization method [7]. Namely, our protocol is divided into a *function-independent* pre-processing phase, followed by a function-dependent phase. In the pre-processing phase, the parties generate complete sharing of several random multiplication-triples in *parallel*, which are later used for efficient evaluation of each multiplication gate, at the cost of just publicly reconstructing two completely shared values per multiplication gate (see Section 2.5 for more details). To generate the shared multiplication-triples, we follow the efficient framework of [24]. The framework allows to efficiently generate several shared random multiplication-triples, using any given ACSS protocol as a black-box, such that the number of ABA instances involved is *independent* of the circuit size.

1.3 Other Related Works

A couple of approaches have been followed in the literature with the aim of designing efficient MPC protocols with *statistical* security in the asynchronous communication setting. The first approach is to design protocols with *non-optimal* resilience. Specifically, AMPC protocols with statistical security and resilience $t < n/4$ are proposed in [45, 24]. While the protocol of [45] requires a communication of $\mathcal{O}(n^2\kappa)$ bits per multiplication, the protocol of [24] brings down the communication complexity to $\mathcal{O}(n\kappa)$ bits per multiplication. The second approach is to design protocols in a *hybrid* communication setting, which is a mix of synchronous and asynchronous communication setting. Namely, in such a setting it is assumed that the network is completely *synchronous* during the first r rounds, after which the network is completely asynchronous. In [25], a statistically-secure MPC protocol in the hybrid communication setting is presented with $t < n/3$, where the first four rounds are assumed to be synchronous and where the communication complexity is $\mathcal{O}(n^2\kappa)$ bits per multiplication. Since our goal is to get an MPC protocol in a *completely* asynchronous setting and that too with the *optimal* resilience of $t < n/3$, our results are incomparable with the works of [45, 24].

2 Preliminaries, Definitions and Existing Tools

We follow the standard *secure-channel* model, where there is a set of n mutually-distrusting parties $\mathcal{P} = \{P_1, \ldots, P_n\}$, connected by pair-wise private and authentic channels. The distrust in the system is modelled by a centralized *computationally unbounded* adversary, who can corrupt any t out of the n parties. Moreover, $t < n/3$ which is a necessary condition for the existence of any statistically-secure AMPC protocol. The adversary is malicious (Byzantine) and can force the parties under its control to deviate from prescribed protocol instructions in any arbitrary manner.

2.1 The Security Model

We define the security of our protocols based on the standard *real-world/ideal-world* paradigm [17, 34], where the security of a protocol (for some computation) is argued by "comparing" the capabilities of the adversary in two separate worlds. In the real-world, the parties execute the protocol and exchange messages among themselves, computed as per the given protocol. In the ideal-world, the parties do not interact with each other, but rather with a *trusted third-party* (an ideal functionality), which assists the parties to get the result of the computation based on the inputs provided by the parties. Informally, a protocol is considered to be secure if whatever an adversary can do in the real protocol (where no trusted third-party is there), can be also done in the ideal-world. We now recall the high level description of the framework from [26] and refer to [17, 39, 27] for the complete formal details.

The Real World: An execution of a protocol Π in the real-world (also called as real model), consists of n *interactive Turing machines* (ITMs) representing the parties $P_1, \ldots, P_n$ respectively. Additionally, there is an ITM for representing the adversary Adv. The adversary Adv is *static* and decides the set of the corrupt parties $\mathcal{C}$ before the beginning of the protocol execution, where $|\mathcal{C}| \leq t$. The parties not under the control of Adv are called *honest*. The adversary controls the operations of the corrupted parties, as well as the delivery of the messages between the parties. The details follow.

Each ITM is initialized with the random coins and its possible inputs. Additionally, Adv may have some auxiliary input z. The protocol operates *asynchronously* by a sequence of *activations*, where at each point a single ITM is active. Once activated, a party can perform some local computation, write on its output tape or send messages to other parties. On the other hand, if the adversary is activated, it can send messages on behalf of the corrupted parties. To model the worst case scenario, the adversary is given the provision to schedule the delivery of the messages exchanged between the parties. Once Adv delivers a message to some party, this party gets activated. The adversary cannot omit, change or inject messages. However, the adversary can reorder the messages sent by the honest parties. That is, it can decide which message will be delivered and when. Moreover, even though the adversary can delay the delivery of the messages *arbitrarily*, it *cannot* delay them *indefinitely*. That is, every message sent by a party is *eventually* delivered. These requirements on adversarial scheduling are formalized using the *eventual-delivery secure message-transmission* ideal functionality in [39, 27]. The protocol execution is complete, once all honest parties obtain their respective outputs. We let $\text{REAL}_{\Pi,\mathsf{Adv}(z),\mathcal{C}}(\vec{x})$ denote the random variable, consisting of the output of the honest parties and the view of the adversary Adv, controlling the parties in $\mathcal{C}$ during the execution of a protocol Π, upon inputs $\vec{x} = (x^{(1)}, \ldots, x^{(n)})$ for the parties (where party P_i has input x_i) and auxiliary input z for Adv.

The Ideal World: A computation in the ideal-world (also called as ideal model) consists of n *dummy* parties $P_1, \ldots, P_n$, an ideal-world adversary $\mathcal{S}$ (also called as simulator) and an ideal functionality $\mathcal{F}$. We consider static corruptions, and so the set of corrupted parties $\mathcal{C}$ is fixed at the beginning of the computation and is known to $\mathcal{S}$. Moreover, following the notion of [4], we consider *corruption-aware* functionalities, where the identity of $\mathcal{C}$ is known to $\mathcal{F}$. The ideal functionality models the desired behaviour of the computation. Namely, $\mathcal{F}$ receives the inputs from the respective dummy parties, performs the desired computation on the received inputs and sends the outputs to the respective parties. The ideal-world adversary *does not* see and *cannot* delay the communication between the parties and $\mathcal{F}$, however it can communicate with $\mathcal{F}$ on the behalf of the parties in $\mathcal{C}$.

Since $\mathcal{F}$ models the desired behaviour of a real-world protocol which is *asynchronous*, ideal functionalities must consider some inherent limitations. For example, in a real-world protocol, the adversary can decide when each honest party learns the output, since adversary has full control over message scheduling. To model the notion of time in the ideal-world, [39] uses the concept of *number of activations*. Namely, once $\mathcal{F}$ has computed the output for some party, it does not ask "permission" from $\mathcal{S}$ to deliver it to the respective party. Instead, the corresponding party must "request" $\mathcal{F}$ for the output, which can be done only when the concerned party is active. Moreover, the adversary can "instruct" $\mathcal{F}$ to delay the output for each party by ignoring the corresponding requests, but only for a polynomial number of activations. If the party is activated sufficiently many times, the party will eventually receive the output from $\mathcal{F}$ and hence ideal computation eventually terminates. That is, each honest party eventually obtains its desired output. As in [26], we use the term *$\mathcal{F}$ sends a request-based delayed output to P_i*, to describe the above interaction between the $\mathcal{F}, \mathcal{S}$ and P_i.

Similar to the real-world, we let $\text{IDEAL}_{\mathcal{F},\mathcal{S}(z),\mathcal{C}}(\vec{x})$ denote the random variable, consisting of the output of the honest parties and the view of the adversary $\mathcal{S}$, controlling the parties in $\mathcal{C}$, upon inputs $\vec{x} =$

$(x^{(1)}, \ldots, x^{(n)})$ for the parties (where party P_i has input x_i) and auxiliary input z for $\mathcal{S}$.

Definition 2.1 (**Perfect and Statistical-Security** [39, 3])**.** Let $\mathcal{F}$ be a functionality and let Π be an n-party protocol involving $\mathcal{P}$. We say that Π perfectly-securely realizes $\mathcal{F}$ if and only if for every real-world adversary Adv, there exists an ideal-world adversary $\mathcal{S}$, whose running time is polynomial in the running time of Adv, such that for every $\mathcal{C} \subset \mathcal{P}$ where $|\mathcal{C}| \leq t$, every $\vec{x} \in (\{0,1\}^\star)^n$ where $|x_1| = \ldots = |x_n|$ and every $z \in \{0,1\}^\star$, it holds that the random variables

$$\left\{\mathrm{REAL}_{\Pi,\mathsf{Adv}(z),\mathcal{C}}(\vec{x})\right\} \qquad \text{and} \qquad \left\{\mathrm{IDEAL}_{\mathcal{F},\mathcal{S}(z),\mathcal{C}}(\vec{x})\right\}$$

are identically distributed. That is, the random variables should be perfectly indistinguishable.

For statistical-security, the parties and adversaries are parameterized with a statistical-security parameter κ, and the above random variables (which are viewed as ensembles, parameterized by κ) are required to be statistically-indistinguishable. That is, their statistical-distance should be a negligible function in κ.

We use the notations $\equiv$ and $\overset{s}{\equiv}$ to denote perfect and statistical-indistinguishability respectively.

The Universal-Composability (UC) Framework: While the real-world/ideal-world based security paradigm is used to define the security of a protocol in the "stand-alone" setting (namely when only instance of the protocol is running), the more powerful UC framework [17, 18] is used to define the security of a protocol when multiple instances of the protocol might be running in parallel, possibly along with other protocols. Informally, the security in the UC-framework is still argued by comparing the real-world and the ideal-world. However, now in both worlds, the computation takes place in the presence of an additional interactive process (modeled as an ITM), called the *environment* and denoted by $\mathcal{Z}$. Roughly speaking, $\mathcal{Z}$ models the "external environment" in which protocol execution takes place. The interaction between $\mathcal{Z}$ and the various entities takes place as follows in the two worlds.

In the real-world, the environment gives inputs to the honest parties, receives their outputs and can communicate with the adversary at any point during the execution. During the protocol execution, the environment gets activated first. Once activated, the environment can either activate one of the parties by providing some input or activate Adv by sending it a message. Once a party completes its operations upon getting activated, the control is returned to the environment. Once Adv gets activated, it can communicate with the environment (apart from sending the messages to the honest parties). The environment also fully controls the corrupt parties, that send to $\mathcal{Z}$ all the messages they receive, and follow the orders of $\mathcal{Z}$. The protocol execution is completed, once $\mathcal{Z}$ stops activating other parties and outputs a single bit.

In the ideal-model, the environment $\mathcal{Z}$ gives inputs to the (dummy) honest parties, receives their outputs and can communicate with $\mathcal{S}$ at any point during the execution. The dummy parties act as channels between $\mathcal{Z}$ and $\mathcal{F}$. That is, they send the inputs received from $\mathcal{Z}$ to $\mathcal{F}$ and transfer the output they receive from $\mathcal{F}$ to $\mathcal{Z}$. The activation sequence in this world is similar to the one in the real-world. The protocol execution is completed, once $\mathcal{Z}$ stops activating other parties and outputs a single bit.

A protocol is said to be UC-secure with *perfect-security*, if for every real-world adversary Adv there exists a simulator $\mathcal{S}$, such that for any environment $\mathcal{Z}$, the environment cannot distinguish the real-world from the ideal-world. On the other hand, the protocol is said to UC-secure with *statistical-security*, if the environment cannot distinguish the real-world from the ideal-world, except with a probability which is a negligible function in the security parameter κ.

The Hybrid Model: In a $\mathcal{G}$-hybrid model, a protocol execution proceeds as in the real-world. However, the parties have access to an ideal functionality $\mathcal{G}$ for some specific task. The parties during the protocol execution, communicate with $\mathcal{G}$ as in the ideal-world. The UC framework guarantees that an ideal functionality in a hybrid model can be replaced with a protocol that UC-securely realizes $\mathcal{G}$. This is specifically due to the following composition theorem from [17, 18].

Theorem 2.2 ([17, 18]). *Let Π be a protocol that UC-securely realizes a functionality $\mathcal{F}$ in the $\mathcal{G}$-hybrid model and let ρ be a protocol that UC-securely realizes $\mathcal{G}$. Moreover, let Π^ρ denote the protocol that is obtained from Π by replacing every ideal call to $\mathcal{G}$ with the protocol ρ. Then protocol Π^ρ UC-securely realizes $\mathcal{F}$ in the model where the parties do not have access to the ideal functionality $\mathcal{G}$.*

2.2 Computation Model

In our protocols, all computations are done over a Galois field $\mathbb{F} = \text{GF}(2^\kappa)$, with $|\mathbb{F}| > n$, such that each field element is represented by κ bits. Looking ahead, this will ensure that error-probability of our AMPC protocol is $2^{-\Omega(\kappa)}$.[6] We also assume that $n = \text{poly}(\kappa)$, where $\text{poly}(\kappa)$ denotes a polynomial function of κ. The parties want to compute a function f over $\mathbb{F}$, represented by a publicly known arithmetic circuit cir over $\mathbb{F}$. For simplicity and without loss of generality, we assume that each party $P_i \in \mathcal{P}$ has a single input $x^{(i)} \in \mathbb{F}$ for the function f and there is a single function output $y \overset{\text{def}}{=} f(x^{(1)}, \ldots, x^{(n)})$, which is supposed to be learnt by all the parties. Apart from the input and output gates, cir consists of linear (addition) gates and multiplication (non-linear) gates. We assume that cir consists of c_M number of multiplication gates, where c_M is bounded by $\text{poly}(\kappa)$.

We assume that $\alpha_1, \ldots, \alpha_n$ are distinct, non-zero elements from $\mathbb{F}$, where α_i is associated with P_i as the "evaluation point". By *communication complexity* of a protocol, we mean the total number of bits communicated by the honest parties in the protocol.

2.3 Definitions

A degree-d *univariate polynomial* over $\mathbb{F}$ is of the form $f(x) = a_0 + \ldots + a_d x^d$, where each $a_i \in \mathbb{F}$. A degree-(ℓ, m) *bivariate polynomial* over $\mathbb{F}$ is of the form $F(x, y) = \sum_{i,j=0}^{i=\ell, j=m} r_{ij} x^i y^j$, where each $r_{ij} \in \mathbb{F}$. Let $f_i(x) \overset{\text{def}}{=} F(x, \alpha_i), g_i(y) \overset{\text{def}}{=} F(\alpha_i, y)$. We call $f_i(x)$ and $g_i(y)$ as i^{th} *row* and *column-polynomial* respectively of $F(x, y)$. This is because the distinct evaluations of the polynomials $f_i(x)$ and $g_i(y)$ at $x = \alpha_1, \ldots, \alpha_n$ and at $y = \alpha_1, \ldots, \alpha_n$ respectively, constitute an $n \times n$ matrix of distinct points on $F(x, y)$ (see Fig 6).

We say a degree-t polynomial $\widetilde{f}_i(x)$, where $i \in \{1, \ldots, n\}$, *lie* on a degree-(t, t) polynomial $F(x, y)$, if $F(x, \alpha_i) = \widetilde{f}_i(x)$ holds. Similarly, we say a degree-t polynomial $\widetilde{g}_i(y)$, where $i \in \{1, \ldots, n\}$, *lie* on a degree-(t, t) polynomial $F(x, y)$, if $F(\alpha_i, y) = \widetilde{g}_i(y)$ holds.

We next give the definition of complete t-sharing, which is central to our AMPC protocol.

Definition 2.3 (**t-sharing and Complete t-sharing**). A value $s \in \mathbb{F}$ is said to be t-shared among a set of parties $\mathcal{W} \subseteq \mathcal{P}$ where $|\mathcal{W}| \geq 2t + 1$, if there exists a degree-t polynomial, say $f(x)$, with $f(0) = s$, such that each (honest) party $P_i \in \mathcal{W}$ holds its share $s_i \overset{\text{def}}{=} f(\alpha_i)$. The vector of shares of s corresponding to the (honest) parties in $\mathcal{W}$ is denoted as $[s]_t^{\mathcal{W}}$.[7] A set of values $S = (s^{(1)}, \ldots, s^{(L)}) \in \mathbb{F}^L$ is said to be t-shared among a set of parties $\mathcal{W}$, if each $s^{(i)} \in S$ is t-shared among $\mathcal{W}$.

A value $s \in \mathbb{F}$ is said to be *completely* t-shared, denoted as $[s]_t$, if s is t-shared among the entire set of parties $\mathcal{P}$; that is $\mathcal{W} = \mathcal{P}$ holds. Similarly, a set of values $S = (s^{(1)}, \ldots, s^{(L)}) \in \mathbb{F}^L$ is said to be completely t-shared, if each $s^{(i)} \in S$ is completely t-shared.

[6]The assumption of Galois field is just for the sake of simplicity. We confirm that our protocols can be easily modified to work over *any* finite field (without affecting the communication complexity) of size at least n (which is needed for instantiating Shamir secret-sharing scheme), while the error-probability can be upper bounded by $2^{-\Omega(\kappa)}$ by working over an extension field of size at least 2^κ, when performing random checks needed as part of the protocols.

[7]In the rest of the paper, we interchangeably use the term *shares of s* and *shares of the polynomial* $f(\cdot)$ to denote the values $f(\alpha_i)$.

Computing Linear Functions of Completely t-shared Values: Complete t-sharings are *linear*. That is, given $[a]_t, [b]_t$, then $[a+b]_t = [a]_t + [b]_t$ and $[c \cdot a]_t = c \cdot [a]_t$ hold, for any public $c \in \mathbb{F}$. In general, given ℓ complete t-sharings $[x^{(1)}]_t, \ldots, [x^{(\ell)}]_t$ and a publicly known linear function $g : \mathbb{F}^\ell \rightarrow \mathbb{F}^m$, where $g(x^{(1)}, \ldots, x^{(\ell)}) = (y^{(1)}, \ldots, y^{(m)})$, then $g([x^{(1)}]_t, \ldots, [x^{(\ell)}]_t) = ([y^{(1)}]_t, \ldots, [y^{(m)}]_t)$ holds. We say that *the parties locally compute* $([y^{(1)}]_t, \ldots, [y^{(m)}]_t) = g([x^{(1)}]_t, \ldots, [x^{(\ell)}]_t)$, to mean that every party $P_i \in \mathcal{P}$ (locally) computes $(y_i^{(1)}, \ldots, y_i^{(m)}) = g(x_i^{(1)}, \ldots, x_i^{(\ell)})$, where $y_i^{(l)}$ and $x_i^{(l)}$ denotes the i^{th} share of $y^{(l)}$ and $x^{(l)}$ respectively.

We will be using the standard Lagrange's polynomial-evaluation function $\mathsf{Lagrange}$ to compute a "new" point on a polynomial, in terms of "old" points lying on the polynomial. In a more detail, the function $\mathsf{Lagrange}$ takes as input a set $\mathcal{K}$, consisting of $|\mathcal{K}|$ pairs of values over $\mathbb{F}$, say $(u^{(i)}, v^{(1)}), \ldots, (u^{(|\mathcal{K}|)}, v^{(|\mathcal{K}|)})$, where $u^{(1)} \neq \ldots \neq u^{(|\mathcal{K}|)}$ holds. Let $f(\cdot)$ be the unique degree-$(|\mathcal{K}|-1)$ polynomial, such that $f(u^{(i)}) = v^{(i)}$ holds. Moreover let $u^{(\text{new})} \in \mathbb{F}$, which is different from every $u^{(1)}, \ldots, u^{(|\mathcal{K}|)}$. The output of the function $\mathsf{Lagrange}$ will be $v^{(\text{new})}$, where $v^{(\text{new})} \stackrel{\text{def}}{=} f(u^{(\text{new})})$. It is well known that $v^{(\text{new})}$ is a linear function of $v^{(1)}, \ldots, v^{(|\mathcal{K}|)}$. Namely, there exists linear combiners $c_1, \ldots, c_{|\mathcal{K}|}$, called *Lagrange's coefficients* [4] (which are *publicly-known* functions of $u^{(1)}, \ldots, u^{(|\mathcal{K}|)}, u^{(\text{new})}$), such that $v^{(\text{new})} = c_1 \cdot v^{(1)} + \ldots + c_{|\mathcal{K}|} \cdot v^{(|\mathcal{K}|)}$ holds. We denote this computation as:

$$v^{(\text{new})} = \mathsf{Lagrange}(|\mathcal{K}|, \{(u^{(i)}, v^{(1)}), \ldots, (u^{(|\mathcal{K}|)}, v^{(|\mathcal{K}|)})\}, u^{(\text{new})}).$$

It follows from the linearity of complete t-sharing that if the parties hold complete t-sharings of $v^{(1)}, \ldots, v^{(|\mathcal{K}|)}$ and if $u^{(1)}, \ldots, u^{(|\mathcal{K}|)}, u^{(\text{new})}$ are publicly known, then they can locally compute a complete t-sharing of $v^{(\text{new})}$.

Properties of Polynomials Over $\mathbb{F}$: We next state certain standard properties of degree-t univariate and degree-(t, t) bivariate polynomials over $\mathbb{F}$. In our protocols, to generate a random t-sharing of a given value s held by a designated *dealer*, we use Shamir secret-sharing scheme. Here to hide s, the dealer chooses a polynomial $f(x)$ randomly from the set $\mathcal{P}^{s,t}$, which denotes the set of all degree-t univariate polynomials over $\mathbb{F}$ whose constant term is s, where $|\mathcal{P}^{s,t}| = |\mathbb{F}|^t$. The dealer then give the share $f(\alpha_i)$ to each party P_i. Since the polynomial is randomly chosen by the dealer and adversary may learn at most t shares on the polynomial, it follows that the distribution of the shares as seen by the adversary is independent of the underlying secret, if the dealer is *honest*.[8] Formally:

Lemma 2.4 ([4]). *For any set of distinct non-zero elements $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$, any pair of values $s, s' \in \mathbb{F}$, any subset $\mathcal{C} \subset \{P_1, \ldots, P_n\}$ where $|\mathcal{C}| = \ell \leq t$, and every $\vec{y} \in \mathbb{F}^\ell$, it holds that:*

$$\Pr_{f(x) \in_R \mathcal{P}^{s,t}} \Big[\vec{y} = (\{f(\alpha_i)\}_{P_i \in \mathcal{C}})\Big] = \Pr_{g(x) \in_R \mathcal{P}^{s',t}} \Big[\vec{y} = (\{g(\alpha_i)\}_{P_i \in \mathcal{C}})\Big] = \frac{1}{|\mathbb{F}|^\ell},$$

where $f(x)$ and $g(x)$ are chosen uniformly and independently from the set of polynomials $\mathcal{P}^{s,t}$ and $\mathcal{P}^{s',t}$, respectively.

The following lemma states that if there are "sufficiently many" degree-t univariate polynomials which are "pair-wise consistent", then there exists a unique degree-(t, t) bivariate polynomial, passing through these univariate polynomials. Formally:

Lemma 2.5 (Pair-wise Consistency Lemma [16, 4]). *Let $f_{i_1}(x), \ldots, f_{i_\ell}(x), g_{j_1}(y), \ldots, g_{j_m}(y)$ be degree-t polynomials where $\ell, m \geq t+1$ and $i_1, \ldots, i_\ell, j_1, \ldots, j_m \in \{1, \ldots, n\}$. Moreover, let for every $i \in \{i_1, \ldots, i_\ell\}$ and every $j \in \{j_1, \ldots, j_m\}$, the condition $f_i(\alpha_j) = g_j(\alpha_i)$ holds. Then there exists a unique*

[8] We often use the term Shamir-sharing polynomial to denote the degree-t polynomial used by the dealer.

degree-(t,t) bivariate polynomial, say $\overline{F}(x,y)$, such that the row polynomials $f_{i_1}(x),\ldots,f_{i_\ell}(x)$ and the column polynomials $g_{j_1}(y),\ldots,g_{j_m}(y)$ lie on $\overline{F}(x,y)$.

In our AISS and ACSS protocol, a dealer on having a secret s does the following to share it: the dealer picks a random degree-t Shamir-sharing polynomial $q(\cdot)$ where $q(0)=s$. The sharing polynomial $q(\cdot)$ is further embedded into a random degree-(t,t) bivariate polynomial $F(x,y)$, where $F(0,y)=q(\cdot)$ holds. The dealer distributes the row-polynomial $f_i(x)=F(x,\alpha_i)$ and column-polynomial $g_i(y)=F(\alpha_i,y)$ to every party P_i. Similar to the case of Shamir secret-sharing, it holds that Adv learning at most t row and column-polynomials, does not learn any information about the underlying shared value s. In fact, it can be shown that for every two degree-t polynomials $q_1(\cdot)$ and $q_2(\cdot)$ such that $q_1(\alpha_i)=q_2(\alpha_i)=f_i(0)$ holds for every $P_i\in\mathcal{C}$, the distribution of the polynomials $\{f_i(x),g_i(y)\}_{P_i\in\mathcal{C}}$ received by the corrupted parties when $F(x,y)$ is chosen based on $q_1(\cdot)$, is identical to the distribution when $F(x,y)$ is chosen based on $q_2(\cdot)$. Formally:

Lemma 2.6 ([4]). *Let $\alpha_1,\ldots,\alpha_n$ be n distinct non-zero elements from $\mathbb{F}$, let $\mathcal{C}\subset\mathcal{P}$ where $|\mathcal{C}|\leq t$, and let $q_1(\cdot)$ and $q_2(\cdot)$ be two different degree-t polynomials over $\mathbb{F}$ such that $q_1(\alpha_i)=q_2(\alpha_i)$ holds for every $P_i\in\mathcal{C}$. Then,*

$$\Big\{\{F(x,\alpha_i),F(\alpha_i,y)\}_{P_i\in\mathcal{C}}\Big\}\equiv\Big\{\{F'(x,\alpha_i),F'(\alpha_i,y)\}_{P_i\in\mathcal{C}}\Big\}$$

holds, where $F(x,y)$ and $F'(x,y)$ are two different degree-(t,t) bivariate polynomials, chosen at random under the constraints that $F(0,y)=q_1(\cdot)$ and $F'(0,y)=q_2(\cdot)$ holds.

2.4 Some Ideal Functionalities

In this section, we present the ideal functionalities used in our protocols.

2.4.1 Asynchronous Reliable Broadcast (ACast)

Informally, an *asynchronous reliable broadcast* (ACast) protocol allows a designated *sender* $P_S\in\mathcal{P}$ with input $m\in\{0,1\}^\star$ to identically send m to all the parties, even in the presence of Adv. If P_S is *honest*, then all honest parties eventually complete the protocol with output m. On the other hand, if P_S is *corrupt* and some honest party outputs $m^\star$, then eventually every other honest party outputs the same $m^\star$. Notice that if P_S is *corrupt*, then it is not necessary that the honest parties complete the protocol, as P_S may *not* invoke the protocol at the first place.

The ideal functionality $\mathcal{F}_{\mathsf{ACast}}$ capturing the requirements for asynchronous reliable broadcast is presented in Fig 1. The functionality upon receiving m from P_S, performs a request-based delayed delivery of m to all the parties. Notice that in case P_S is *corrupt*, it may not send m to the functionality for delivery, in which case parties obtain no output.

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

$\mathcal{F}_{\mathsf{ACast}}$ proceeds as follows, running with parties $\mathcal{P}=\{P_1,\ldots,P_n\}$ and an adversary $\mathcal{S}$. Let $\mathcal{C}$ denote the set of corrupt parties, where $|\mathcal{C}|\leq t$.

- Upon receiving $(\mathsf{sender},\mathsf{ACast},\mathsf{sid},m)$ from $P_S\in\mathcal{P}$, send $(P_S,\mathsf{ACast},\mathsf{sid},m)$ to $\mathcal{S}$ and send a request-based delayed output $(P_S,\mathsf{ACast},\mathsf{sid},m)$ to each $P_i\in\mathcal{P}\setminus\mathcal{C}$ (no need to send m to the parties in $\mathcal{C}$, as $\mathcal{S}$ gets m on their behalf).

Figure 1: The ideal functionality for asynchronous reliable broadcast for session id sid.

Bracha [15] presented an elegant protocol for asynchronous reliable broadcast. The protocol incurs a communication of $\mathcal{O}(n^2\cdot\ell)$ bits, if sender's message m consists of ℓ bits. To the best of our knowledge, no one has ever presented a UC-security proof for Bracha's ACast protocol. For the sake of completeness, in Appendix A we show that Bracha's Acast protocol UC-securely realizes the ideal functionality $\mathcal{F}_{\mathsf{ACast}}$ with *perfect* security, for any $t < n/3$.

2.4.2 Asynchronous Byzantine Agreement (ABA)

Byzantine agreement (BA) [47, 41, 5] is a fundamental primitive in secure distributed computing. In a *synchronous* BA protocol, each party has an input bit and an output bit and the protocol guarantees the following three properties.

– *Agreement*: The output bit of all honest parties is the same.
– *Validity*: If all honest parties have the same input bit, then this will be the common output bit.
– *Termination*: All honest parties eventually complete the protocol.

In an *asynchronous Byzantine agreement* (ABA) protocol, the above requirements are slightly weakened, since up to t (potentially honest) parties may not be able to provide their inputs to the protocol and the decision is taken based on the inputs of a set $\mathcal{CS}$ of $n - t$ parties. Moreover, since adversary can control the schedule of message delivery, it has full control in deciding the set $\mathcal{CS}$. Furthermore, the termination guarantees are weakened to take into consideration the FLP impossibility result [32] according to which any (deterministic) ABA protocol must have non-terminating runs, where the honest parties keep on running the protocol forever *without* obtaining any output. A common approach to circumvent this impossibility result is to go for randomized ABA protocols and the best we can hope from such protocols is that the honest parties eventually complete the protocol, asymptotically with probability 1. This property is often called as *almost-surely terminating* property [1, 6].

The formal specification of an ideal ABA functionality is presented in Fig 2, which is taken from [27]. Intuitively, it can be considered as a special case of ideal AMPC functionality (see Fig 21 in Section 7), which looks at the set of inputs provided by the set of parties in $\mathcal{CS}$, where $\mathcal{CS}$ is decided by the ideal-world adversary. Now if the input bits provided by all the honest parties in $\mathcal{CS}$ are the same, then it is set as the output bit. Else the output bit is set to be the input bit provided by some corrupt party in $\mathcal{CS}$ (for example, the first corrupt party in $\mathcal{CS}$ according to lexicographic ordering). In the functionality, the inputs bits provided by various parties are considered as the vote-input of the respective parties. We stress that the idea-world adversary *cannot* delay sending $\mathcal{CS}$ to the functionality infinitely. This is because in the real-world almost-surely terminating ABA protocols, which securely realize the functionality $\mathcal{F}_{\mathsf{ABA}}$ (see [1, 6] for such protocols), the set $\mathcal{CS}$ is eventually decided, irrespective of the message scheduling of the adversary.

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

$\mathcal{F}_{\mathsf{ABA}}$ proceeds as follows, running with parties $\mathcal{P} = \{P_1, \ldots, P_n\}$ and an adversary $\mathcal{S}$. Let $\mathcal{C}$ denote the set of corrupt parties, where $|\mathcal{C}| \leq t$ and let $\mathcal{H} = \mathcal{P} \setminus \mathcal{C}$. For each party P_i, initialize an input value $x^{(i)} = \bot$.

- Upon receiving a message $(\mathsf{vote}, \mathsf{sid}, b)$ from some $P_i \in \mathcal{P}$ where $b \in \{0, 1\}$, set $x^{(i)} = b$ if $\mathcal{CS}$ has not been recorded yet or if $P_i \in \mathcal{CS}$. Moreover, send $(\mathsf{vote}, \mathsf{sid}, P_i, b)$ to $\mathcal{S}$, if $P_i \notin \mathcal{C}$.
- Upon receiving a message $(\mathsf{coreset}, \mathsf{sid}, \mathcal{CS})$ from $\mathcal{S}$, verify that $\mathcal{CS}$ is a subset of $\mathcal{P}$ of size $n - t$, else ignore the message. If $\mathcal{CS}$ has not been recorded yet, then record $\mathcal{CS}$.
- If the set $\mathcal{CS}$ has been recorded and the value $x^{(i)}$ has been set to a value different from $\bot$ for every $P_i \in \mathcal{CS}$, then compute the output y as follows and generate a request-based delayed output $(\mathsf{decide}, \mathsf{sid}, (\mathcal{CS}, y))$ for every $P_i \in \mathcal{P}$.
  – If $x^{(i)} = b$ holds for all $P_i \in (\mathcal{H} \cap \mathcal{CS})$, then set $y = b$.
  – Else set $y = x^{(i)}$, where P_i is the party with the smallest index in $\mathcal{CS} \cap \mathcal{C}$.

Figure 2: The ideal functionality for asynchronous Byzantine agreement for session id sid.

2.4.3 Functionality for Generating Random t-sharing.

Functionality $\mathcal{F}_{\mathsf{Rand}}$ (see Fig 3) generates a random t-sharing and distributes the entire vector of shares to all the parties. For this, the functionality picks a random degree-t polynomial and distributes the polynomial to the parties. The functionality generates the output only upon being invoked by at least $t+1$ parties. Looking ahead, this guarantees that the output is generated only if at least one *honest* party asks the functionality to do so. The functionality allows the ideal-world adversary to specify the shares that it wants for the corrupt parties. The resultant t-sharing is generated, by "fixing" the shares of the corrupt parties in the t-sharing. Looking ahead, this provision is made to incorporate the fact that while realizing $\mathcal{F}_{\mathsf{Rand}}$ by a protocol, the adversary will have full control over the shares of the corrupt parties for the generated t-sharing. However, we stress that conditioned on the shares of the corrupt parties, the resultant t-sharing is uniformly distributed. This further implies that adversary has no "control" over the constant term of the resultant degree-t polynomial.

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

$\mathcal{F}_{\mathsf{Rand}}$ proceeds as follows, running with parties $\mathcal{P} = \{P_1, \ldots, P_n\}$ and an adversary $\mathcal{S}$. Let $\mathcal{C}$ denote the set of corrupt parties, where $|\mathcal{C}| \leq t$.

- Upon receiving $(\mathsf{Rand}, \mathsf{sid}, P_i)$ from P_i, send $(\mathsf{Rand}, \mathsf{sid}, P_i)$ to $\mathcal{S}$.
- Upon receiving $(\mathsf{shares}, \mathsf{sid}, \{r_i\}_{P_i \in \mathcal{C}})$ from $\mathcal{S}$, record these shares.
- If $(\mathsf{Rand}, \mathsf{sid}, P_i)$ has been received from $t+1$ parties P_i and if the shares of parties in $\mathcal{C}$ have been recorded, then do the following.
  - Select a random degree-t polynomial, say $R(\cdot)$, such that $R(\alpha_i) = r_i$ holds, for each $P_i \in \mathcal{C}$.
  - Send a request-based delayed output $(\mathsf{Rand}, \mathsf{sid}, R(\cdot))$ to each $P_i \in \mathcal{P}$.

Figure 3: The ideal functionality for generating a random t-sharing for the session id sid.

2.5 Existing Asynchronous Primitives

In our protocols, we use the following standard existing asynchronous primitives.

Private and Public Reconstruction of Completely t-shared Values: Let s be a value, which is completely t-shared. The well-know *online error-correction* (OEC) protocol [16] allows a designated party P_R to reconstruct s. In the protocol, every party sends its respective share of s to P_R, who (asynchronously) applies the Reed-Solomon (RS) error-correction procedure on the received shares. If $t < n/3$, then P_R will be able to eventually error-correct the received corrupt shares (if any) and reconstruct the entire degree-t polynomial through which s is t-shared. The protocol incurs a communication of $\mathcal{O}(n\kappa)$ bits.

If the value s needs to be publicly reconstructed, then the above protocol can be executed n times, once on the behalf of each party. We call the resultant protocol as OEC, whose communication complexity is $\mathcal{O}(n^2\kappa)$ bits.

Beaver's Circuit-Randomization: We use the well-known Beaver's circuit-randomization method [7] to evaluate multiplication gates in our AMPC protocol in a shared fashion. The method allows evaluation of a multiplication gate with secret-shared inputs, at the expense of publicly reconstructing two secret-shared values, using an auxiliary secret-shared multiplication triple. In a more detail, let the parties hold a complete t-sharing of x and y and the goal is to compute a complete t-sharing of $z \overset{\text{def}}{=} x \cdot y$. Moreover, let $([u]_t, [v]_t, [w]_t)$ be a completely t-shared multiplication triple available with the parties, such that $w = u \cdot v$ holds. We note that $z = (x - u + u) \cdot (y - v + v)$ holds and hence $z = (x - u) \cdot (y - v) + v \cdot (x - u) + u \cdot (y - v) + u \cdot v$. Based on this idea, to compute $[z]_t$, the parties first locally compute $[d]_t = [x - u]_t = [x]_t - [u]_t$

and $[e]_t = [y - v]_t = [y]_t - [v]_t$, followed by publicly reconstructing d and e by invoking two instances of OEC. The parties then locally compute $[z]_t = d \cdot e + d \cdot [v]_t + e \cdot [u]_t + [w]_t$.

It is easy to see that if u and v are random and private, then during the above process, the view of the adversary remains independent of x and y. Namely, even after learning d and e, the privacy of the inputs x and y (and hence the output z) is preserved. We also note that if the auxiliary triple (u, v, w) is *not* a multiplication-triple (namely if $w = u \cdot v + \Delta$ holds for some non-zero Δ), then in the above process, $z = x \cdot y + \Delta$ holds. The communication complexity of the protocol is $\mathcal{O}(n^2\kappa)$ bits.

3 The Asynchronous Information-Checking Protocol (AICP)

In this section, we present our AICP, which will be used in our AISS protocol. The notion of AICP extends the notion of ICP defined in the synchronous setting [49], to the asynchronous communication setting. An AICP involves three entities: a *signer* $\mathsf{S} \in \mathcal{P}$, an *intermediary* $\mathsf{I} \in \mathcal{P}$ and a *receiver* $\mathsf{R} \in \mathcal{P}$, along with the set of parties $\mathcal{P}$ acting as *verifiers*. Party S has a private input $\mathcal{S}$. An AICP can be considered as information-theoretically secure analogue of digital signatures, where S gives a "signature" on $\mathcal{S}$ to I, who eventually reveals it to R, claiming that it got the signature from S. If S and I are *honest*, then the signature is accepted by an *honest* R (correctness). Moreover, the signed values remain private (privacy) till they are revealed to R. If S and R are *honest*, then a *corrupt* I cannot forge S's signature (unforgeability). And finally, if S is *corrupt* and gave signed values to an *honest* I, then I can later reveal it to an *honest* R (non-repudiation).[9]

We stress that unlike digital signatures which are "transferable" (and hence offer *public* verifiability), the signatures generates by AICP are *not* transferrable. Namely, once the signatures are revealed to the designated R and verified, party R *cannot* further reveal the same signature (and get it verified) to any other party. However, this is not an issue as in our context, the use case of AICP requires revealing the signed values only once to a designated party. The protocol proceeds in the following three phases, each of which is implemented by a dedicated sub-protocol.

- **Distribution Phase**: Executed by a protocol Gen, where S sends $\mathcal{S}$ to I along with some *auxiliary information* and to each verifier, S gives some *verification information*.
- **Authentication Phase**: Executed by $\mathcal{P}$ through a protocol Ver, to verify whether S distributed "consistent" information to I and the verifiers. Upon successful verification I sets a Boolean variable $\mathsf{V}_{\mathsf{S},\mathsf{I}}$ to 1 and the information held by I is considered as the *information-checking signature* on $\mathcal{S}$, denoted as $\mathsf{ICSig}(\mathsf{S} \rightarrow \mathsf{I}, \mathcal{S})$. The notation $\mathsf{S} \rightarrow \mathsf{I}$ signifies that the signature is *given by* S *to* I.
- **Revelation Phase**: Executed by I, R and the verifiers by running a protocol $\mathsf{RevPriv}$, where I reveals $\mathsf{ICSig}(\mathsf{S} \rightarrow \mathsf{I}, \mathcal{S})$ to R, who either outputs $\mathcal{S}$ after verification or rejects it.

Definition 3.1 (AICP). A triplet of protocols $(\mathsf{Gen}, \mathsf{Ver}, \mathsf{RevPriv})$ where S has a private input $\mathcal{S} \in \mathbb{F}^L$ for Gen is called a $(1 - \epsilon_{\mathsf{AICP}})$-secure AICP, for a given error parameter ϵ_{AICP}, if all the following requirements hold for every possible adversary.

- **Correctness**: If S, I and R are *honest*, then I sets $\mathsf{V}_{\mathsf{S},\mathsf{I}}$ to 1 during Ver. Moreover, R outputs $\mathcal{S}$ at the end of $\mathsf{RevPriv}$.
- **Privacy**: If S, I and R are *honest*, then the view of adversary throughout is independent of $\mathcal{S}$.
- **Unforgeability**: If S and R are *honest*, I reveals $\mathsf{ICSig}(\mathsf{S} \rightarrow \mathsf{I}, \bar{\mathcal{S}})$ and if R outputs $\bar{\mathcal{S}}$ during $\mathsf{RevPriv}$, then except with probability at most ϵ_{AICP}, the condition $\bar{\mathcal{S}} = \mathcal{S}$ holds.

[9]We stress that the original notion of ICP as formulated in [49, 48] involves a *single* receiver, who also plays the role of verifier as well. In [44] this notion was extended to the case where all the parties play the role of receivers, as well as verifiers and where the signature is *publicly* revealed. We modify this notion where the signature is *not* publicly revealed, but rather to a designated receiver, while all the parties play the role of verifiers. The modification is done as it suits best for our AISS, where AICP is used as a building block. We also note that a similar modification has been proposed recently in the synchronous communication setting [3], where the primitive is referred as *interactive signatures*.

- **Non-repudiation**: If S is *corrupt* and if I, R are *honest* and if I sets $\mathsf{V}_{\mathsf{S},\mathsf{I}}$ to 1 holding $\mathsf{ICSig}(\mathsf{S} \rightarrow \mathsf{I}, \bar{\mathcal{S}})$ during Ver, then except with probability ϵ_{AICP}, R outputs $\bar{\mathcal{S}}$ during $\mathsf{RevPriv}$.

3.1 Our Instantiation of AICP

Our instantiation of AICP is an adaptation of the synchronous ICP of [8] to the asynchronous setting. Let $\mathcal{S} = (s^{(1)}, \ldots, s^{(L)}) \in \mathbb{F}^L$ be the input of S. The high level idea of the protocol is as follows: during the distribution phase, S gives $\mathcal{S}$ along with some *authentication tag* to I and a corresponding information-theoretic *verification tag* to each individual verifier. The tags with respect to a verifier P_i are computed by picking a random $y \in \mathbb{F}$ and fitting a degree-L polynomial $f(x)$ passing through $L+1$ distinct points $(0, y), (1, s^{(1)}), \ldots, (L, s^{(L)})$. The authentication tag is set to y, while the verification tag is set to (u, v), where u is randomly chosen from $\mathbb{F} \backslash \{0, \ldots, L\}$ and $v = f(u)$. Later, during the revelation phase, I provides $\mathcal{S}$ and the authentication tags to R and the verifiers provide the verification tags to R and if the revealed $\mathcal{S}$ and authentication tags are found to be consistent with "sufficiently many" verification tags, then $\mathcal{S}$ is accepted, else it is rejected.

The above distribution of information maintains the *privacy* of $\mathcal{S}$ for an *honest* S and I, from a corrupt verifier P_i. This is because the verifier P_i learns only a single point on $f(x)$, which is a degree-L polynomial. The above distribution of information also ensures that if S is *honest* and a *corrupt* I reveals an incorrect $\mathcal{S}$ to R, then with a high probability, it will fail the consistency-test with respect to the verification tag of an *honest* verifier, as the corresponding verification tag will *not* be known to I. This further guarantees the *unforgeability* property.

Unfortunately, the above distribution of information alone is not sufficient to ensure the *non-repudiation* property. This is because if S is *corrupt*, then it can distribute inconsistent data to I and the verifiers, which will later lead to the rejection of revealed $\mathcal{S}$. To get around this problem, I and the verifiers interact in a "zero-knowledge" fashion during the authentication phase, to verify the consistency of $\mathcal{S}$, authentication tags and verification tags, while maintaining the privacy of their respective information. The verification happens using the cut-and-choose technique, where instead of providing a *single* verification and authentication tag with respect to each verifier, S provides 2κ number of authentication tags to I and corresponding 2κ verification tags are given to each verifier P_i. Then during the authentication phase, P_i randomly reveals κ number of verification tags to I. We say that the cut-and-choose is "successful" with respect to P_i, if the revealed verification tags are found to be consistent with $\mathcal{S}$ and the corresponding authentication tags. If the cut-and-choose is successful, then with a high probability, it is ensured that *at least one* of the remaining undisclosed κ verification tags held by P_i is consistent with $\mathcal{S}$ and the corresponding authentication tag held by I.

In the protocol, the cut-and-choose step is executed independently between I and each individual verifier P_i. Party I sets $\mathsf{V}_{\mathsf{S},\mathsf{I}}$ to 1 as soon as it finds that the cut-and-choose test is successful with respect to a set of verifiers $\mathcal{R}$, where $|\mathcal{R}| = n - t$. Notice that due to asynchronous communication, I cannot wait for all the n instances of cut-and-choose to be successful, as the corrupt verifiers may not participate in their respective instances. Later, during the revelation phase, R accepts the $\mathcal{S}$ revealed by I, if there are at least $|\mathcal{R}| - t$ verifiers from the set $\mathcal{R}$, such that each of these verifiers produce at least one consistent verification tag from their list of undisclosed verification tags. Again, due to the asynchronous communication, R cannot wait for all the verifiers in $\mathcal{R}$ to disclose their undisclosed verification tags, as corrupt verifiers in $\mathcal{R}$ may not respond. However, there will be at least one *honest* verifier in the set of verifiers, with respect to whose verification tag, $\mathcal{S}$ is found to be consistent, thus guaranteeing that the revealed $\mathcal{S}$ is correct (with a high probability). The protocol is formally presented in Fig 5. The information exchanged among the various parties in the protocol is pictorially presented in Fig 4. In the formal steps of the AICP, sid denotes the session id.

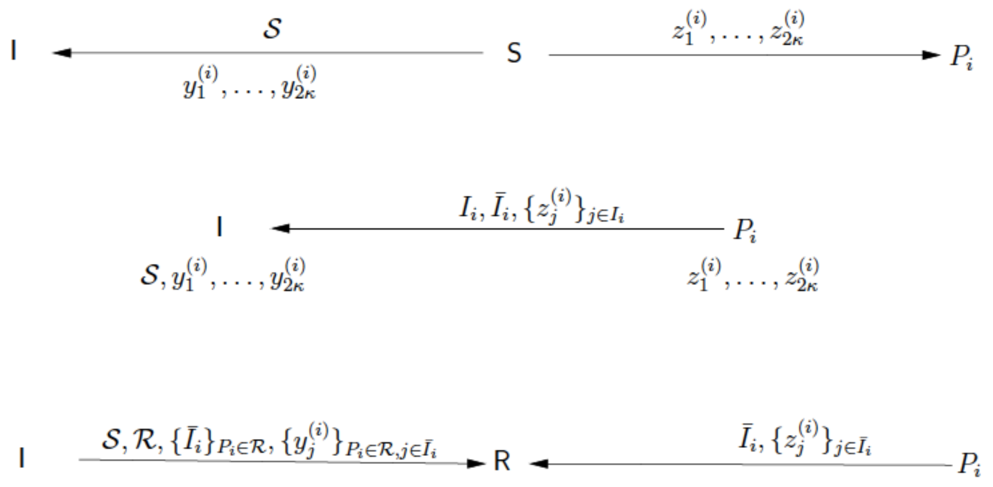

I
$\mathcal{S}$
S
$z_1^{(i)}, \ldots, z_{2\kappa}^{(i)}$
P_i
$y_1^{(i)}, \ldots, y_{2\kappa}^{(i)}$
I
$I_i, \bar{I}_i, \{z_j^{(i)}\}_{j \in I_i}$
P_i
$\mathcal{S}, y_1^{(i)}, \ldots, y_{2\kappa}^{(i)}$
$z_1^{(i)}, \ldots, z_{2\kappa}^{(i)}$
I
$\mathcal{S}, \mathcal{R}, \{\bar{I}_i\}_{P_i \in \mathcal{R}}, \{y_j^{(i)}\}_{P_i \in \mathcal{R}, j \in \bar{I}_i}$
R
$\bar{I}_i, \{z_j^{(i)}\}_{j \in \bar{I}_i}$
P_i

Figure 4: The information exchanged among various parties in the AICP. The first figure denotes the information distributed by S to verifier P_i during Gen and the corresponding information distributed to I. The second figure denotes the cut-and-choose interaction between verifier P_i and I during Ver, based on which I either includes or excludes P_i from $\mathcal{R}$. The last figure denotes the information sent by I and verifier P_i to R during RevPriv

Protocol AICP

Generation Phase: Protocol $\mathsf{Gen}(\mathsf{S}, \mathsf{I}, \mathcal{S})$**:** $\mathcal{S} = (s^{(1)}, \ldots, s^{(L)})$

– **Distribution by** S: With input $\mathcal{S}$, the following code is executed only by S.
  - Corresponding to each verifier $P_i \in \mathcal{P}$, pick 2κ random elements $y_1^{(i)}, \ldots, y_{2\kappa}^{(i)}$ and 2κ random *evaluation-points* $u_1^{(i)}, \ldots, u_{2\kappa}^{(i)}$ from $\mathbb{F} - \backslash\{0, \ldots, L\}$. Compute $v_1^{(i)}, \ldots, v_{2\kappa}^{(i)}$, such that for each $j = 1, \ldots, 2\kappa$, the $L+2$ points $(0, y_j^{(i)}), (1, s^{(1)}), \ldots, (L, s^{(L)}), (u_j^{(i)}, v_j^{(i)})$ lie on a degree-L polynomial.
  - Corresponding to each verifier $P_i \in \mathcal{P}$, set $y_1^{(i)}, \ldots, y_{2\kappa}^{(i)}$ as the authentication tags and set $z_1^{(i)} = (u_1^{(i)}, v_1^{(i)}), \ldots, z_{2\kappa}^{(i)} = (u_{2\kappa}^{(i)}, v_{2\kappa}^{(i)})$ as the verification tags.
  - Send $(\mathsf{sid}, \mathsf{data}, \mathcal{S})$ and $(\mathsf{sid}, \mathsf{authtag}, \{y_1^{(i)}, \ldots, y_{2\kappa}^{(i)}\}_{P_i \in \mathcal{P}})$ to I.
  - For $i = 1, \ldots, n$, send $(\mathsf{sid}, \mathsf{vertag}, \{z_1^{(i)}, \ldots, z_{2\kappa}^{(i)}\})$ to P_i.

Authentication Phase: Protocol $\mathsf{Ver}(\mathsf{S}, \mathsf{I}, \mathcal{S})$

– Each $P_i \in \mathcal{P}$ (including S, I) on receiving $(\mathsf{sid}, \mathsf{vertag}, \{z_1^{(i)}, \ldots, z_{2\kappa}^{(i)}\})$ from S does the following:
  - Randomly partition the index set $\{1, \ldots, 2\kappa\}$ into two equal halves I_i and $\bar{I}_i$ of size κ.
  - Send $(\mathsf{sid}, \mathsf{cut\&choose}, I_i, \bar{I}_i, \{z_j^{(i)}\}_{j \in I_i})$ to I.

– Party I does the following:
  - Wait to receive $(\mathsf{sid}, \mathsf{data}, \mathcal{S})$ and $(\mathsf{sid}, \mathsf{authtag}, \{y_1^{(i)}, \ldots, y_{2\kappa}^{(i)}\}_{P_i \in \mathcal{P}})$ from S. Upon receiving, initialize a set $\mathcal{R}$ to $\emptyset$.
  - Upon receiving $(\mathsf{sid}, \mathsf{cut\&choose}, I_i, \bar{I}_i, \{z_j^{(i)}\}_{j \in I_i})$ from P_i, verify if for *every* $j \in I_i$, the $L+2$ points $(0, y_j^{(i)}), (1, s^{(1)}), \ldots, (L, s^{(L)}), z_j^{(i)}$ lie on a degree-L polynomial. Include P_i to the set $\mathcal{R}$, if the verification is successful.

– Wait till $|\mathcal{R}| = n - t$. Then set $\mathsf{V}_{\mathsf{sid},\mathsf{S},\mathsf{I}} = 1$ and $\mathsf{ICSig}(\mathsf{sid}, \mathsf{S} \rightarrow \mathsf{I}, \mathcal{S}) = (\mathcal{S}, \{\bar{I}_i\}_{P_i \in \mathcal{R}}, \{y_j^{(i)}\}_{P_i \in \mathcal{R}, j \in \bar{I}_i})$.

Reveal Phase: Protocol $\mathsf{RevPriv}(\mathsf{S}, \mathsf{I}, \mathsf{R}, \mathcal{S})$

- **Revealing of the Signature by** I: I sends (sid, RevealSig, $\mathcal{R}$, $\mathsf{ICSig}(\mathsf{sid}, \mathsf{S} \rightarrow \mathsf{I}, \mathcal{S})$) to R, provided $\mathsf{V}_{\mathsf{sid},\mathsf{S},\mathsf{I}} = 1$.
- **Revealing of the Verification Tags by Verifiers**: Each verifier $P_i \in \mathcal{P}$ (including S and I) sends (sid, RevealTag, $\bar{I}_i, \{z_j^{(i)}\}_{j \in \bar{I}_i}$) to R.
- **Verifying the Signature and Verification Tags**: Upon receiving (sid, RevealSig, $\mathcal{R}$, $\mathsf{ICSig}(\mathsf{sid}, \mathsf{S} \rightarrow \mathsf{I}, \mathcal{S})$) from I, receiver R obtains $\mathcal{S} = (s^{(1)}, \ldots, s^{(L)})$, the set $\mathcal{R}$, the index sets $\{\bar{I}_i\}_{P_i \in \mathcal{R}}$ and the authentication tags $\{y_j^{(i)}\}_{P_i \in \mathcal{R}, j \in \bar{I}_i}$ from ICSig. The signature is then verified by R as follows:
  - Upon receiving (sid, RevealTag, $\bar{I}_i, \{z_j^{(i)}\}_{j \in \bar{I}_i}$) from verifier P_i, mark P_i as *consistent*, if all the following hold.
    - $P_i \in \mathcal{R}$.
    - The index set $\bar{I}_i$ received from P_i is the same as the index set corresponding to P_i as received from I as part of ICSig.
    - There exists some $j \in \bar{I}_i$, such that the $L + 2$ points $(0, y_j^{(i)}), (1, s^{(1)}), \ldots, (L, s^{(L)}), z_j^{(i)}$ lie on a degree-L polynomial.
  - If $t + 1$ verifiers are marked as consistent, then output $\mathcal{S}$.

Figure 5: Asynchronous Information-Checking Protocol for session id sid.

We now proceed to prove the properties of our AICP. In the proofs, for simplicity we skip the session id sid. We start with the correctness property.

Lemma 3.2 (Correctness). *If* S, I *and* R *are honest, then* I *eventually sets* $\mathsf{V}_{\mathsf{S},\mathsf{I}}$ *to* 1 *during* Ver. *Moreover,* R *eventually outputs* $\mathcal{S}$ *during* RevPriv.

Proof. Let S, I and R be *honest*. Then S distributes correct authentication tags to I and verification tags to individual verifiers during Gen. Consequently, the cut-and-choose interaction between I and each *honest* verifier will be successful and hence each honest verifier is included by I to the set $\mathcal{R}$. So I eventually finds a set $\mathcal{R}$ containing $n - t$ verifiers and sets $\mathsf{V}_{\mathsf{S},\mathsf{I}}$ to 1 during Ver. Since $\mathcal{R}$ contains at least $t + 1$ honest verifiers whose authentication tags are eventually delivered to R, it follows that R eventually finds $t + 1$ consistent verifiers from the set $\mathcal{R}$ and outputs $\mathcal{S}$ during RevPriv. □

We next prove the privacy property.

Lemma 3.3 (Privacy). *If* S, I *and* R *are honest, then the view of adversary during* Gen, Ver *and* RevPriv *is independent of* $\mathcal{S}$.

Proof. For simplicity and without loss of generality, let us assume that $P_1, P_2 \ldots P_t$ are under the control of Adv. During the protocol Gen, the adversary learns the verification tags corresponding to these t parties. However, these verification tags are independent of $\mathcal{S}$, as the corresponding authentication tags are not known to the adversary, as they are held by the *honest* S and I. More specifically, for any *corrupt* verifier P_i, each verification tag $z_j^{(i)} = (u_j^{(i)}, v_j^{(i)})$ is a distinct point on a degree-L polynomial passing through the $L+1$ points $(0, y_j^{(i)}), (0, s^{(1)}), \ldots, (0, s^{(L)})$, where $y_j^{(i)}$ is randomly chosen from $\mathbb{F}$ and not known to P_i. Hence, from the view-point of a corrupt P_i, the verification tag $z_j^{(i)}$ is consistent with every $\mathcal{S} \in \mathbb{F}^L$. Namely, for any candidate $\mathcal{S} = (s^{(1)}, \ldots, s^{(L)}) \in \mathbb{F}^L$, from the view-point of P_i, there exists a unique $y_j^{(i)}$, such that the $L + 1$ points $(0, y_j^{(i)}), (0, s^{(1)}), \ldots, (0, s^{(L)})$ along with the verification tag $z_j^{(i)}$ lie on some degree-L polynomial. Since $y_j^{(i)}$ is randomly chosen from $\mathbb{F}$ by D and not known to P_i, the verification tags learnt by the adversary are independent of the actual $\mathcal{S}$ held by D and I. Now during the protocols Ver and $\mathsf{RevPriv}$, no information is communicated by I to the corrupt verifiers (since R is assumed to be *honest*) and hence overall the view of the adversary remains independent of $\mathcal{S}$. $\square$

We next prove the unforgeability property.

Lemma 3.4 (Unforgeability). *If* S *and* R *are honest,* I *reveals* $\mathsf{ICSig}(\mathsf{S} \rightarrow \mathsf{I}, \bar{\mathcal{S}})$ *and if* R *outputs* $\bar{\mathcal{S}}$ *during* $\mathsf{RevPriv}$*, then except with probability at most* $\frac{n\kappa}{2^\kappa-(L+1)}$*, the condition* $\bar{\mathcal{S}} = \mathcal{S}$ *holds.*

Proof. Let S and R be *honest*. As per the protocol $\mathsf{RevPriv}$, party I invokes $\mathsf{RevPriv}$ only if $\mathsf{V}_{\mathsf{S},\mathsf{I}}$ is set to 1 during the protocol Ver. Now there are two possible cases. If I is *honest*, then the lemma statement is true with probability 1. This is because in this case, I holds $\mathsf{ICSig}(\mathsf{S} \rightarrow \mathsf{I}, \mathcal{S})$ during Ver. Moreover, there will be at least $t+1$ honest verifiers in $\mathcal{R}$, who sends verification tags consistent with $\mathsf{ICSig}(\mathsf{S} \rightarrow \mathsf{I}, \mathcal{S})$ to R.

We next consider the case when I is *corrupt* and reveals $\mathsf{ICSig}(\mathsf{S} \rightarrow \mathsf{I}, \bar{\mathcal{S}})$ to R, such that $\bar{\mathcal{S}} \neq \mathcal{S}$. We claim that in this case, the probability that an *honest* verifier $P_i \in \mathcal{R}$ is marked as consistent by R, is at most $\frac{\kappa}{|\mathbb{F}|-(L+1)}$. Assuming that the claim is true, it follows via the union bound that the probability that *any* honest verifier from the $\mathcal{R}$ set is marked as consistent by R is upper bounded by $\frac{n\kappa}{|\mathbb{F}|-(L+1)}$, as there can be at most $(n-t) \leq n$ honest parties in $\mathcal{R}$. Since $|\mathbb{F}| = 2^\kappa$, it implies that except with probability at most $\frac{n\kappa}{2^\kappa-(L+1)}$, there can be at most t consistent verifiers from $\mathcal{R}$, corresponding to $\mathsf{ICSig}(\mathsf{S} \rightarrow \mathsf{I}, \bar{\mathcal{S}})$. This is because there can be at most t potentially corrupt verifiers in $\mathcal{R}$, who may send to R verification tags, consistent with $\mathsf{ICSig}(\mathsf{S} \rightarrow \mathsf{I}, \bar{\mathcal{S}})$. This implies that the honest R rejects the revealed $\mathsf{ICSig}(\mathsf{S} \rightarrow \mathsf{I}, \bar{\mathcal{S}})$ and does not output $\bar{\mathcal{S}}$.

We now prove our claim. So consider an arbitrary *honest* $P_i \in \mathcal{R}$. During $\mathsf{RevPriv}$, P_i sends the index set $\bar{I}_i$ and the verification tags $\{z_j^{(i)}\}_{j \in \bar{I}_i}$ to R. Since D and R are assumed to be honest, a corrupt I will not know these verification tags. Since I reveals $\mathsf{ICSig}(\mathsf{S} \rightarrow \mathsf{I}, \bar{\mathcal{S}})$ to R, such that $\bar{\mathcal{S}} \neq \mathcal{S}$, the verifier P_i will be marked as consistent by R, only if for one of the κ verification tags $\{z_j^{(i)}\}_{j \in \bar{I}_i}$, the points $(1, \overline{s}^{(1)}), \ldots, (L, \overline{s}^{(L)}), z_j^{(i)}$ lie on a degree-L polynomial, where $\bar{\mathcal{S}} = (\overline{s}^{(1)}, \ldots, \overline{s}^{(L)})$. Since each of the verification tags $\{z_j^{(i)}\}_{j \in \bar{I}_i}$ is randomly chosen by D, the probability of this event is the same as P_i correctly guessing at least one of the verification tags from the set $\{z_j^{(i)}\}_{j \in \bar{I}_i}$, which it can do with probability at most $\frac{\kappa}{|\mathbb{F}|-(L+1)}$. This is because each $z_j^{(i)} = (u_j^{(i)}, v_j^{(i)})$, where $u_j^{(i)}$ is randomly chosen from $\mathbb{F} - \{0, \ldots, L\}$. $\square$

We next prove the non-repudiation property.

Lemma 3.5 (Non-repudiation). *If* S *is corrupt,* I, R *are honest and if* I *sets* $\mathsf{V}_{\mathsf{S},\mathsf{I}}$ *to* 1 *holding* $\mathsf{ICSig}(\mathsf{S} \rightarrow \mathsf{I}, \mathcal{S})$ *during* Ver*, then except with probability at most* $\frac{n\kappa}{2^\kappa}$*,* R *outputs* $\mathcal{S}$ *during* $\mathsf{RevPriv}$.

Proof. Let S be *corrupt* and I, R be *honest*. Since I has set $\mathsf{V}_{\mathsf{S},\mathsf{I}}$ to 1 and holds $\mathsf{ICSig}(\mathsf{S} \rightarrow \mathsf{I}, \mathcal{S})$, it implies that I found a set $\mathcal{R}$ of size $n-t$ during Ver, such that the cut-and-choose verification test is successful for each verifier $P_i \in \mathcal{R}$. That is, for each $P_i \in \mathcal{R}$, party I receives the index sets $I_i, \bar{I}_i$ and the verification tags $z_j^{(i)}$ for each $j \in I_i$ from P_i, such that for *every* $j \in I_i$, the $L+2$ points $(0, y_j^{(i)}), (1, s^{(1)}), \ldots, (L, s^{(L)}), z_j^{(i)}$ lie on a degree-L polynomial. Here $\mathcal{S} = (s^{(1)}, \ldots, s^{(L)})$ is received by I from S during Gen. There are at least $t+1$ honest parties P_i in the set $\mathcal{R}$. We claim that for *any* such *honest* verifier $P_i \in \mathcal{R}$, at least one of the κ undisclosed verification tags is consistent with $\mathcal{S}$ and the corresponding authentication tag held by I, except with probability at most $\frac{\kappa}{2^\kappa}$. That is, there exists at least one $j \in \bar{I}_i$, such that except with probability at most $\frac{\kappa}{2^\kappa}$, the $L+2$ points $(0, y_j^{(i)}), (1, s^{(1)}), \ldots, (L, s^{(L)}), z_j^{(i)}$ lie on a degree-L univariate polynomial. This further implies that during the protocol $\mathsf{RevPriv}$, the verifier P_i will be marked as consistent by I. Now assuming that the claim is true, it follows that *all* the honest verifiers in $\mathcal{R}$ will be marked as consistent by I except with probability at most $\frac{n\kappa}{2^\kappa}$ and hence R outputs $\mathcal{S}$ during $\mathsf{RevPriv}$.

To prove the claim, consider an arbitrary *honest* $P_i \in \mathcal{R}$. If S is *honest*, then the claim is true with probability 1, as an honest S distributes correct and consistent verification and authentication tags. So we consider the case, when S is *corrupt*. In this case, in order that none of remaining κ verification tags corresponding to the indices in $\bar{I}_i$ are consistent with $\mathcal{S}$ and the corresponding authentication tags, S has to guess the index sets I_i and $\bar{I}_i$ in advance during the protocol Gen and distribute consistent verification and authentication tags for *all* the indices in I_i and inconsistent verification and authentication tags for *all* the indices in $\bar{I}_i$. This will ensure that P_i is included in the set $\mathcal{R}$ during Ver, but not marked as consistent during $\mathsf{RevPriv}$. Now the probability that S can guess I_i and $\bar{I}_i$ during Gen is $\frac{1}{\binom{2\kappa}{\kappa}}$ which is at most $\frac{\kappa}{2^\kappa}$. $\square$

We next prove the communication complexity.

Lemma 3.6. *The communication complexity of* $\mathsf{Gen}, \mathsf{Ver}$ *and* $\mathsf{RevPriv}$ *is* $\mathcal{O}(L\kappa + n\kappa^2)$, $\mathcal{O}(n\kappa^2)$ *and* $\mathcal{O}(L\kappa + n\kappa^2)$ *bits respectively.*

Proof. During the protocol Gen, S sends $\mathcal{S}$ and $2n\kappa$ authentication tags to I and total $2n\kappa$ verification tags to all the verifiers. During the protocol Ver, each verifier sends κ verification tags to I. Finally during $\mathsf{RevPriv}$, I sends $\mathcal{S}$ and total $|\mathcal{R}|\kappa$ authentication tags to R where $|\mathcal{R}| = \mathcal{O}(n)$, while each verifier sends κ verification tags to I. $\square$

Assuming that L is some polynomial function of κ (looking ahead, this will be finally the case in our AMPC protocol), Theorem 3.7 now follows from Lemma 3.2-3.6 and the fact that $n = \text{poly}(\kappa)$.

Theorem 3.7. *Protocols* $(\mathsf{Gen}, \mathsf{Ver}, \mathsf{RevPriv})$ *constitute a* $(1-\epsilon_{\mathsf{AICP}})$*-secure AICP against a static malicious adversary corrupting up to* $t < n/3$ *parties, where* $\epsilon_{\mathsf{AICP}} = 2^{-\Omega(\kappa)}$. *The communication complexity of* $\mathsf{Gen}, \mathsf{Ver}$ *and* $\mathsf{RevPriv}$ *is* $\mathcal{O}(L\kappa + n\kappa^2)$, $\mathcal{O}(n\kappa^2)$ *and* $\mathcal{O}(L\kappa + n\kappa^2)$ *bits respectively.*

Notice that in our AICP, any party can be designated to play the role of S, I and R. For convenience, in the rest of the paper, we use the following notations while using our AICP.

Notation 3.8 (Notation for Using AICP). While using our AICP, we say that:

- "*P_i gives* $\mathsf{ICSig}(\mathsf{sid}, P_i \rightarrow P_j, \mathcal{S})$ *to* P_j" to mean that P_i acts as a signer S and invokes an instance of the protocol $\mathsf{Gen}(\mathsf{S}, \mathsf{I}, \mathcal{S})$ with session id sid, where P_j plays the role of intermediary I.
- "*P_j receives* $\mathsf{ICSig}(\mathsf{sid}, P_i \rightarrow P_j, \mathcal{S})$ *from* P_i" to mean that P_j as an intermediary I holds $\mathsf{ICSig}(\mathsf{sid}, P_i \rightarrow P_j, \mathcal{S})$ and has set $\mathsf{V}_{\mathsf{sid}, P_i, P_j}$ to 1 during protocol Ver with session id sid, where P_i is the signer S.
- "*P_j reveals* $\mathsf{ICSig}(\mathsf{sid}, P_i \rightarrow P_j, \mathcal{S})$ *to* P_k" to mean P_j as an intermediary I invokes an instance of $\mathsf{RevPriv}$ with session id sid, with P_i and P_k playing the role of S and R respectively.
- "*P_k accepts* $\mathsf{ICSig}(\mathsf{sid}, P_i \rightarrow P_j, \mathcal{S})$" to mean that P_k as a receiver R outputs $\mathcal{S}$, during the instance of $\mathsf{RevPriv}$ with session id sid, invoked by P_j as I, with P_i playing the role of S.

Finally, we note that we separately do not put any termination condition for any party in our AICP. Since AICP will be used as a sub-protocol in our AMPC protocol, the termination condition of our AMPC protocol will automatically trigger the termination of all the instances of underlying AICP.

4 Asynchronous Incomplete Secret-Sharing (AISS)

We now present our AISS protocol Sh, which will be used as a building block in our ACSS protocol. Protocol Sh generates a *two-level* secret-sharing with IC-signatures. This sharing is an enhanced version of t-sharing, where each share of the secret is further t-shared. Moreover, for the purpose of authentication, each second-level share is IC-signed.

Definition 4.1 (Two-level t-Sharing with IC-signatures). A set of values $S = (s^{(1)}, \ldots, s^{(L)})$ is said to be two-level t-shared with IC-signatures if there exists a set $\mathcal{W} \subseteq \mathcal{P}$ with $|\mathcal{W}| \geq n - t$ and a set $\mathcal{W}_j \subseteq \mathcal{P}$ for each $P_j \in \mathcal{W}$ with $|\mathcal{W}_j| \geq n - t$, such that the following conditions hold.

- Each $s^{(k)} \in S$ is t-shared among $\mathcal{W}$, with each party $P_j \in \mathcal{W}$ holding its *primary-share* $s^{(k)}_j$.
- For each primary-share holder $P_j \in \mathcal{W}$, its primary-share $s^{(k)}_j$ is t-shared among $\mathcal{W}_j$, with each $P_i \in \mathcal{W}_j$ holding a *secondary-share* $s^{(k)}_{j,i}$ of the primary-share $s^{(k)}_j$.
- Each primary-share holder $P_j \in \mathcal{W}$ holds $\mathsf{ICSig}(P_i \rightarrow P_j, (s^{(1)}_{j,i}, \ldots, s^{(L)}_{j,i}))$, corresponding to each *honest* secondary-share holder $P_i \in \mathcal{W}_j$.

We stress that the $\mathcal{W}_j$ sets might be different for each $P_j \in \mathcal{W}$. Note that the primary-shares of the (honest) parties in $\mathcal{W}$ do not constitute a complete t-sharing of the underlying secrets, as the set $\mathcal{W}$ *may not* include all honest parties. However, even though the generated t-sharing may not be complete, it is *well-defined* since $\mathcal{W}$ will have at least $t + 1$ *honest* parties. Similarly, the t-sharings of the primary-shares *do not* constitute a complete t-sharing, as the corresponding $\mathcal{W}_j$ sets may not include all honest parties.

In the protocol Sh, there exists a designated *dealer* $\mathsf{D} \in \mathcal{P}$ with an input $S \in \mathbb{F}^L$ and the goal is to *verifiably* generate a two-level t-sharing with IC signatures of S. The verifiability here ensures that if the (honest) parties obtain an output in the protocol, then there exists some $\bar{S} \in \mathbb{F}^L$, where $\bar{S} = S$ for an *honest* D, such that $\bar{S}$ is two-level t-shared with IC-signatures. The protocol also ensures that if D is *honest*, then the view of the adversary is independent of S. For simplicity, we first present the protocol Sh assuming that D has a single value for sharing, that is $L = 1$. We then discuss the modifications needed to share L values simultaneously, which are straight-forward.

To share a value $s \in \mathbb{F}$, the dealer D hides s in the constant term of a *random* degree-(t, t) bivariate polynomial $F(x, y)$. The goal is then to let D distribute the row and column-polynomials of $F(x, y)$ to respective parties and then publicly verify if D has distributed consistent row and column-polynomials to sufficiently many parties, which lie on a single degree-(t, t) bivariate polynomial, say $\bar{F}(x, y)$, which is considered as D's *committed* bivariate polynomial (if D is honest then $\bar{F}(x, y) = F(x, y)$ holds). Once the existence of an $\bar{F}(x, y)$ is confirmed, the next goal is to let each P_j who holds its row-polynomial $\bar{F}(x, \alpha_j)$ lying on $\bar{F}(x, y)$, get signature on $\bar{F}(\alpha_i, \alpha_j)$ values from at least $n - t$ parties P_i. Finally, once $n - t$ parties P_j get their row-polynomials signed, it implies the generation of two-level t-sharing of $\overline{s} = \bar{F}(0, 0)$ with IC signatures. Namely, $\overline{s}$ will be t-shared through degree-t column-polynomial $\bar{F}(0, y)$. The set of signed row-polynomial holders P_j will constitute the set $\mathcal{W}$, where P_j holds the *primary-share* $\bar{F}(0, \alpha_j)$, which is the constant term of its row-polynomial $\bar{F}(x, \alpha_j)$. And the set of parties P_i who signed the values $\bar{F}(\alpha_i, \alpha_j)$ for P_j constitute the $\mathcal{W}_j$ set with P_i holding the *secondary-share* $\bar{F}(\alpha_i, \alpha_j)$, thus ensuring that the primary-share $\bar{F}(0, \alpha_j)$ is t-shared among $\mathcal{W}_j$ through degree-t row-polynomial $\bar{F}(x, \alpha_j)$. For a pictorial depiction of how the values on D's bivariate polynomial constitute a two-level t-sharing of its constant term, see Fig 6.

The above stated goals are achieved (asynchronously) in *four* stages in the protocol Sh. In the protocol steps of Sh (Fig 8), the purpose of the steps of each stage appears as a comment. We next explain the purpose of each stage.

Stage I: Distribution of Column-Polynomials and Collecting Signatures. To begin with, D distributes the column-polynomials to respective parties (the row-polynomials are retained by D) and tries to get the values on respective column-polynomials signed by a set $\mathcal{M}$ of $n - t$ column holders. That is, each P_i is given its column-polynomial $g_i(y) = F(\alpha_i, y)$ and is asked to sign the values $g_i(\alpha_1), \ldots, g_i(\alpha_n)$ for D. Let $f_{ji} \overset{\text{def}}{=} g_i(\alpha_j)$ for $j = 1, \ldots, n$. Party P_i signs the values $f_{1i}, \ldots, f_{ni}$ for D after verifying that all of them lie on a single degree-t polynomial and publicly announces the issuance of signatures to D by broadcasting a `SC` message (standing for "signed column"). Notice that by signing f_{ji}, party P_i implicitly

$$
\begin{array}{ccccccccccc}
 & & [s = F(0,0)]_t^{\mathcal{W}} & P_1 & \ldots & P_i & \ldots & P_{2t+1} & & \\
 & & \Downarrow & \Downarrow & & \Downarrow & & \Downarrow & & \\
P_1 & \Rightarrow & F(0,\alpha_1) & F(\alpha_1,\alpha_1) & \ldots & F(\alpha_i,\alpha_1) & \ldots & F(\alpha_{2t+1},\alpha_1) & \Leftarrow & [F(0,\alpha_1)]_t^{\mathcal{W}_1} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & \\
P_j & \Rightarrow & F(0,\alpha_j) & F(\alpha_1,\alpha_j) & \ldots & \underline{F(\alpha_i,\alpha_j)} & \ldots & F(\alpha_{2t+1},\alpha_j) & \Leftarrow & [F(0,\alpha_j)]_t^{\mathcal{W}_j} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & \\
P_{2t+1} & \Rightarrow & F(0,\alpha_{2t+1}) & F(\alpha_1,\alpha_{2t+1}) & \ldots & F(\alpha_i,\alpha_{2t+1}) & \ldots & F(\alpha_{2t+1},\alpha_{2t+1}) & \Leftarrow & [F(0,\alpha_{2t+1})]_t^{\mathcal{W}_{2t+1}}
\end{array}
$$

Figure 6: Two-level t-sharing with IC-signatures of $s = F(0,0)$. Here we assume that $n = 3t+1$, $\mathcal{W} = \{P_1,\ldots,P_{2t+1}\}$ and $\mathcal{W}_j = \{P_1,\ldots,P_{2t+1}\}$ for each $P_j \in \mathcal{W}$. Party P_j will possess all the values along the j^{th} row, which constitute the row-polynomial $f_j(x) = F(x,\alpha_j)$. Column-wise, P_i possesses the values in the column labelled with P_i, which lie on the column-polynomial $g_i(y) = F(\alpha_i, y)$. Party P_j will possess P_i's information-checking signature on the common value $f_j(\alpha_i) = F(\alpha_i,\alpha_j) = g_i(\alpha_j)$ between P_j's row-polynomial and P_i's column-polynomial, which is underlined. The value s will be t-shared among $\mathcal{W}$ through the column-polynomial $g_0(y) = F(0,y)$, where the primary-shares of the parties in $\mathcal{W}$ are shown in red color. The primary-share $f_j(0) = F(0,\alpha_j)$ of party P_j will be t-shared among the parties in $\mathcal{W}_j$ through $f_j(x)$, with the secondary-shares of $f_j(0)$ being shown in blue color.

signs the value $F(\alpha_i,\alpha_j)$, which is the same as the value of the j^{th} row-polynomial (held by D) at $x = \alpha_i$. Due to asynchronous communication, the parties cannot afford to wait for all the n parties to sign the values on their respective column-polynomials and hence they proceed to the next stage, as soon as $n-t$ parties $\mathcal{M}$ issue the signatures to D. To ensure that all parties agree on a common set $\mathcal{M}$, the dealer D is assigned the task of finding $\mathcal{M}$ and announcing the same, which gets verified in the next stage.

Once a set $\mathcal{M}$ of $n-t$ parties broadcast SC message, it confirms that the n row-polynomials held by D and the individual column-polynomials of the parties in $\mathcal{M}$, together lie on a single degree-(t,t) bivariate polynomial (due to the pair-wise consistency Lemma 2.5). This also confirms that D is committed to a single (yet unknown) degree-(t,t) bivariate polynomial. The next stage is to let D distribute the individual row-polynomials of this committed bivariate polynomial to respective parties. For a pictorial depiction of this stage of Sh, see Fig 7.

Stage II: Distribution of Signed Row-Polynomials. To prevent a potentially corrupt D from distributing arbitrary polynomials to the parties as row-polynomials, D actually sends the signed row-polynomials to the individual parties, where the values on the row-polynomials are signed by the parties in $\mathcal{M}$. Namely, to distribute the row-polynomial $f_j(x)$ to P_j, D reveals the $f_j(\alpha_i)$ values to P_j, signed by the parties $P_i \in \mathcal{M}$. The presence of the signatures ensure that the degree-t polynomial $f_j(x)$ revealed by D to P_j is indeed the j^{th} row-polynomial, lying on D's committed bivariate polynomial. This is because there are at least $t+1$ *honest* parties in $\mathcal{M}$, whose signed f_{ji} values (which are the same as $g_i(\alpha_j)$) uniquely define the j^{th} row-polynomial of D's committed bivariate polynomial.

Upon the receipt of correctly signed row-polynomial, P_j publicly announces it by broadcasting a RR message (standing for "row received"). The next stage is to let such parties P_j (who broadcasted RR message) obtain "fresh" signatures on $n-t$ values of $f_j(x)$ by at least $n-t$ parties $\mathcal{W}_j$. We stress that the signatures of the parties in $\mathcal{M}$ on the values of $f_j(x)$, which are revealed by D to P_j *cannot* be "re-used" and hence $\mathcal{M}$ cannot be considered as $\mathcal{W}_j$. This is because IC-signatures are *not* "transferable" and the signatures on the values of $f_j(x)$ were issued to D and *not* to P_j. We also stress that the parties in $\mathcal{M}$ *cannot* be enforced to re-issue "fresh" signatures on the common values of P_j's row-polynomial, as corrupt parties in $\mathcal{M}$ may now not participate honestly during this process. Hence, P_j has to ask for the fresh signatures on the common values of $f_j(x)$ from every potential party, which constitutes the third stage. For a pictorial depiction of this stage of Sh, see Fig 7.

Stage III: Recommitment of Row-Polynomials. The process of P_j getting values on $f_j(x)$ freshly signed can be viewed as P_j "recommitting" its received row-polynomial to a set of $n-t$ column-polynomial holders $\mathcal{W}_j$. However, extra care has to be taken to prevent a potentially corrupt P_j from getting *fresh* signatures on arbitrary values, which do not lie in $f_j(x)$, as $\mathcal{W}_j$ may be different from $\mathcal{M}$. This is done as follows. Any party P_i on receiving a "signature request" on f_{ji} from P_j signs it, only if it lies on P_i's column-polynomial which P_i received from D; that is $\mathsf{f}_{ji} = g_i(\alpha_j)$ holds. Then after receiving the signature from P_i, party P_j publicly announces the same. Now the condition for including P_i to $\mathcal{W}_j$ is that apart from P_j, there should exist at least $2t$ other parties P_k who has broadcasted $\mathtt{RR}$ messages and who also got their respective row-polynomials signed by P_i, as part of their respective recommitment process. This ensures that there are total $2t+1$ parties who broadcasted $\mathtt{RR}$ messages and whose respective row-polynomials are signed by P_i. Now among these $2t+1$ parties, at least $t+1$ parties P_k are honest, whose row-polynomials $f_k(x)$ lie on D's committed bivariate polynomial. Since these $t+1$ parties got signature on $f_k(\alpha_i)$ values from P_i, this further implies that $f_k(\alpha_i) = g_i(\alpha_k)$ holds for these $t+1$ honest parties P_k, further implying that P_i's column-polynomial $g_i(y)$ also lies on D's committed bivariate polynomial. Now since P_i ensures that $\mathsf{f}_{ji} = g_i(\alpha_j)$ holds for P_j as well, it implies that the value which P_j got signed by P_i is $g_i(\alpha_j)$, which is the same as $f_j(\alpha_i)$. For a pictorial depiction of this stage of Sh, see Figure (c) in Fig 7.

To ensure that all parties agree on a common $\mathcal{W}$ set and $\mathcal{W}_j$ subsets for each $P_j \in \mathcal{W}$, the dealer D is assigned the task of building the above sets and later publicly announce the same (during stage IV). Namely, for each P_j who has broadcasted a $\mathtt{RR}$ message, D keeps on populating the set $\mathcal{W}_j$ with new parties P_i, if the above conditions are satisfied. Once $\mathcal{W}_j$ achieves a size of $n-t$ (implying P_j has recommitted the correct $f_j(x)$ polynomial), party P_j is included in $\mathcal{W}$. For a pictorial depiction of P_i's inclusion in $\mathcal{W}_j$, see Fig 7.

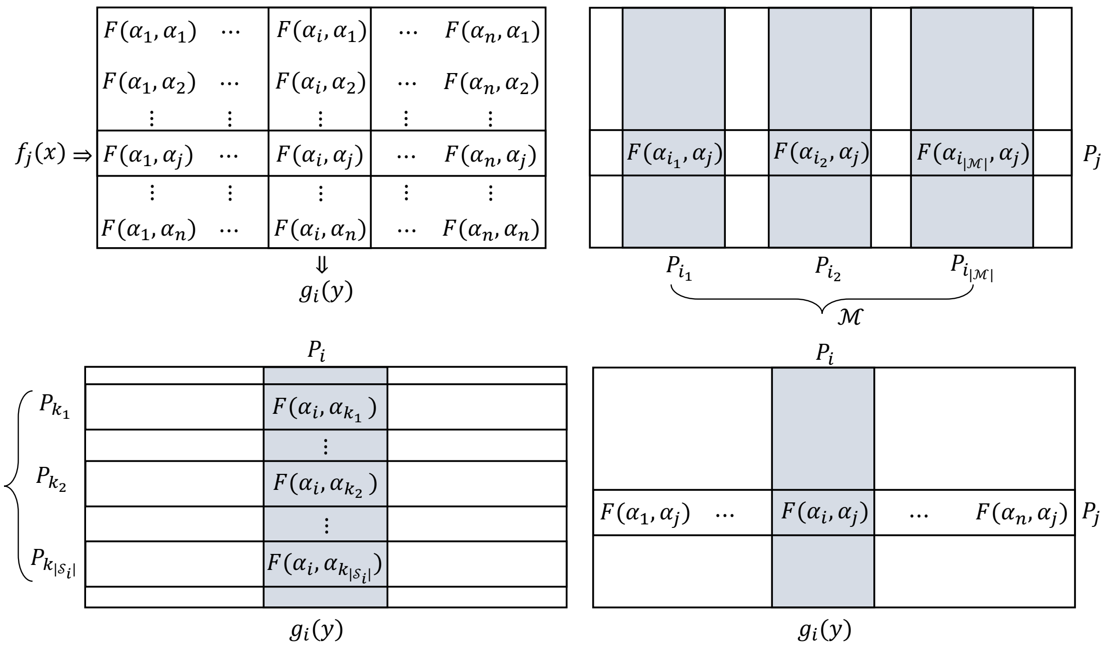
Stage IV: Public Announcement of $\mathcal{W}$ set and $\mathcal{W}_j$ Subsets. The last stage of Sh is the announcement of the $\mathcal{W}$ set and its public verification. We stress that this stage of the protocol Sh will be triggered in our ACSS protocol, where Sh will be used as a sub-protocol. Looking ahead, in our ACSS protocol, D will invoke several instances of Sh and a potential $\mathcal{W}$ set is built independently for each of these instances. Once all these individual $\mathcal{W}$ sets achieve the cardinality of at least $n-t$ and satisfy certain *additional* properties in the ACSS protocol, D will broadcast these individual $\mathcal{W}$ sets and parties will have to verify each $\mathcal{W}$ set individually. The verification of a publicly announced $\mathcal{W}$ set as part of an Sh instance is done by this last stage of the Sh protocol. To verify the $\mathcal{W}$ set, the parties check if its cardinality is at least $n-t$, each party P_j in $\mathcal{W}$ has broadcasted $\mathtt{RR}$ message and recommitted its row-polynomial correctly to the parties in $\mathcal{W}_j$.

We note that we separately *do not* put any termination condition for any party in protocol Sh. Since Sh will be used as a sub-protocol in our ACSS protocol, which is further used in our AMPC protocol, the termination condition of our AMPC protocol will automatically trigger the termination of all the underlying instances of Sh.

Protocol $\mathsf{Sh}(\mathsf{D}, s)$

%Stage I: Distribution of Column-Polynomials and Collecting Signatures.

- **Distribution of Column-Polynomials by** D: The following code is executed only by D.
  - On having the input s, select a random degree-(t,t) bivariate polynomial $F(x,y)$ over $\mathbb{F}$, such that $F(0,0) = s$.
  - Send $(\mathsf{sid}, \mathsf{columnpoly}, g_i(y))$ to each $P_i \in \mathcal{P}$, where $g_i(y) \overset{\text{def}}{=} F(\alpha_i, y)$.
- **Signing the Values on Column-Polynomials for** D: Each $P_i \in \mathcal{P}$ (including D) executes the following code.
  - Wait to receive $(\mathsf{sid}, \mathsf{columnpoly}, g_i(y))$ from D, where $g_i(y)$ is a degree-t polynomial.
  - Upon receiving, compute the values $f_{j1}, \ldots, f_{jn}$ where $f_{ji} \overset{\text{def}}{=} g_i(\alpha_j)$ for $j = 1, \ldots, n$.



$F(\alpha_1,\alpha_1)$ ⋯ $F(\alpha_i,\alpha_1)$ ⋯ $F(\alpha_n,\alpha_1)$
$F(\alpha_1,\alpha_2)$ ⋯ $F(\alpha_i,\alpha_2)$ ⋯ $F(\alpha_n,\alpha_2)$
$f_j(x) \Rightarrow$ $F(\alpha_1,\alpha_j)$ ⋯ $F(\alpha_i,\alpha_j)$ ⋯ $F(\alpha_n,\alpha_j)$
$F(\alpha_1,\alpha_n)$ ⋯ $F(\alpha_i,\alpha_n)$ ⋯ $F(\alpha_n,\alpha_n)$
$\Downarrow$ $g_i(y)$
$F(\alpha_{i_1},\alpha_j)$ $F(\alpha_{i_2},\alpha_j)$ $F(\alpha_{i_{|\mathcal{M}|}},\alpha_j)$ P_j
P_{i_1} P_{i_2} $P_{i_{|\mathcal{M}|}}$
$\mathcal{M}$
P_i
$\mathcal{S}_i$ P_{k_1} P_{k_2} $P_{k_{|\mathcal{S}_i|}}$
$F(\alpha_i,\alpha_{k_1})$ $F(\alpha_i,\alpha_{k_2})$ $F(\alpha_i,\alpha_{k_{|\mathcal{S}_i|}})$
$g_i(y)$
P_i
$F(\alpha_1,\alpha_j)$ ⋯ $F(\alpha_i,\alpha_j)$ ⋯ $F(\alpha_n,\alpha_j)$ P_j
$g_i(y)$

Figure 7: Pictorial representation of the various stages of the protocol Sh in the clock-wise direction. The first figure denotes the matrix of $n \times n$ values on the bivariate polynomial $F(x,y)$ computed by D. While the row-polynomials are held by D, the column-polynomials are issued to respective parties for getting their signatures. The second figure shows the delivery of signed row-polynomials by D to respective parties, where the values on row-polynomials are signed by the column-polynomial holders in the set $\mathcal{M}$, thus ensuring the pair-wise consistency of the column-polynomials of the honest parties in $\mathcal{M}$ and the row-polynomials of the respective parties who broadcast RR messages. The third figure shows the recommitment of row-polynomials by respective parties. Party P_i allows P_j to recommit its row-polynomial if the recommitted polynomial is pair-wise consistent with P_i's column-polynomial. The fourth figure shows the criteria for including party P_i to the set $\mathcal{W}_j$. Namely there should exist a set of $2t+1$ parties $\mathcal{S}_i$, including P_j, who recommitted their respective row-polynomials to P_i.

  - Give $\mathsf{ICSig}(\mathsf{sid}, P_i \rightarrow \mathsf{D}, f_{ji})$ to D for $j = 1, \ldots, n$ and send $(\mathsf{sender}, \mathsf{ACast}, \mathsf{sid}, \mathtt{SC}_i)$ to $\mathcal{F}_{\mathsf{ACast}}$.
- **Identifying Signed Column-Polynomials**: The following code is executed only by D:
  - Initialize a set $\mathcal{M}$ to $\emptyset$.
  - Include P_i to the set $\mathcal{M}$, if all the following hold.
    - $(P_i, \mathsf{ACast}, \mathsf{sid}, \mathtt{SC}_i)$ is received from $\mathcal{F}_{\mathsf{ACast}}$ where P_i is the sender.
    - $\mathsf{ICSig}(\mathsf{sid}, P_i \rightarrow \mathsf{D}, f_{ji})$ is received from P_i, for $j = 1, \ldots, n$, such that $f_{ji} = F(\alpha_i, \alpha_j)$ holds.
  - Wait till $|\mathcal{M}| = n - t$. Once $|\mathcal{M}| = n - t$, then send $(\mathsf{sender}, \mathsf{ACast}, \mathsf{sid}, \mathcal{M})$ to $\mathcal{F}_{\mathsf{ACast}}$.

% Stage II: Distribution of Signed Row-Polynomials by D and Verification by the Parties.

- **Revealing Row-Polynomials to Respective Parties**: If D has sent $(\mathsf{sender}, \mathsf{ACast}, \mathsf{sid}, \mathcal{M})$ to $\mathcal{F}_{\mathsf{ACast}}$, then for $j = 1, \ldots, n$, dealer D reveals $\{\mathsf{ICSig}(\mathsf{sid}, P_i \rightarrow \mathsf{D}, f_{ji})\}_{P_i \in \mathcal{M}}$ to P_j.
- **Verifying the Consistency of Row-Polynomials Received from** D: Each $P_j \in \mathcal{P}$ (including D) sends $(\mathsf{sender}, \mathsf{ACast}, \mathsf{sid}, \mathtt{RR}_j)$ to $\mathcal{F}_{\mathsf{ACast}}$, if the following hold.
  - P_j receives $(\mathsf{D}, \mathsf{ACast}, \mathsf{sid}, \mathcal{M})$ from $\mathcal{F}_{\mathsf{ACast}}$ where D is the sender, such that $|\mathcal{M}| = n - t$.
  - P_j receives $(P_i, \mathsf{ACast}, \mathsf{sid}, \mathtt{SC}_i)$ from $\mathcal{F}_{\mathsf{ACast}}$ where P_i is the sender, for each $P_i \in \mathcal{M}$.
  - P_j accepted $\{\mathsf{ICSig}(\mathsf{sid}, P_i \rightarrow \mathsf{D}, f_{ji})\}_{P_i \in \mathcal{M}}$ and the points $\{(\alpha_i, f_{ji})\}_{P_i \in \mathcal{M}}$ lie on a degree-t polynomial $f_j(x)$.

%Stage III: Recommitment of Row-Polynomials.

- **Getting Signatures on Row-Polynomial**: Each $P_j \in \mathcal{P}$ (including D) executes the following.
  - If P_j has sent $(\mathsf{sender}, \mathsf{ACast}, \mathsf{sid}, \mathtt{RR}_j)$ to $\mathcal{F}_{\mathsf{ACast}}$, then for $i = 1, \ldots, n$, do the following.
    - Compute $\mathsf{f}_{ji} \overset{\text{def}}{=} f_j(\alpha_i)$. Send $(\mathsf{sid}, \mathsf{signrequest}, \mathsf{f}_{ji})$ to P_i.
    - If $\mathsf{ICSig}(\mathsf{sid}, P_i \rightarrow P_j, \mathsf{f}_{ji})$ is received from from P_i, send $(\mathsf{sender}, \mathsf{ACast}, \mathsf{sid}, (\mathtt{SR}_j, P_i))$ to $\mathcal{F}_{\mathsf{ACast}}$.
  - If $(\mathsf{sid}, \mathsf{signrequest}, \mathsf{f}_{ij})$ is received from P_i, then give $\mathsf{ICSig}(\mathsf{sid}, P_j \rightarrow P_i, \mathsf{f}_{ij})$ to P_i, provided all the following hold.
    - $(P_i, \mathsf{ACast}, \mathsf{sid}, \mathtt{RR}_i)$ has been received from $\mathcal{F}_{\mathsf{ACast}}$ where P_i is the sender.
    - $(\mathsf{sid}, \mathsf{columnpoly}, g_j(y))$ has been received from D, where $g_j(y)$ is a degree-t polynomial.
    - The condition $\mathsf{f}_{ij} = g_j(\alpha_i)$ holds.
- **Preparing the $\mathcal{W}_j$ Sets and $\mathcal{W}$ Set**: the following code is executed only by D.
  - Initialize the sets $\mathcal{S}_1, \ldots, \mathcal{S}_n, \mathcal{W}_1, \ldots, \mathcal{W}_n$ and $\mathcal{W}$ to $\emptyset$.
  - Include P_k to the set $\mathcal{S}_i$, if all the following hold.
    - $(P_k, \mathsf{ACast}, \mathsf{sid}, (\mathtt{SR}_k, P_i))$ is received from $\mathcal{F}_{\mathsf{ACast}}$ where P_k is the sender.
    - $(P_k, \mathsf{ACast}, \mathsf{sid}, \mathtt{RR}_k)$ is received from $\mathcal{F}_{\mathsf{ACast}}$, where P_k is the sender.
  - Include P_i in $\mathcal{W}_j$, if the following conditions hold.
    - The message $(P_j, \mathsf{ACast}, \mathsf{sid}, \mathtt{RR}_j)$ is received from $\mathcal{F}_{\mathsf{ACast}}$, where P_j is the sender.
    - $|\mathcal{S}_i| \geq 2t + 1$ holds.
    - $P_j \in \mathcal{S}_i$ holds.
  - Include $P_j \in \mathcal{W}$, if $|\mathcal{W}_j| \geq n - t$ holds. Keep on including new parties P_i in $\mathcal{W}_j$ even after including P_j to $\mathcal{W}$, if the above conditions for P_i's inclusion to $\mathcal{W}_j$ are satisfied.

% Stage IV: Public Announcement of $\mathcal{W}$ and Verification. This code will be triggered by our ACSS protocol.

- **Publicly Announcing the $\mathcal{W}$ Set**: D sends $(\mathsf{sender}, \mathsf{ACast}, \mathsf{sid}, \mathcal{W}, \{\mathcal{W}_j\}_{P_j \in \mathcal{W}})$ to $\mathcal{F}_{\mathsf{ACast}}$.
- **Verification of the $\mathcal{W}$ Set by the Parties**: Upon receiving $(\mathsf{D}, \mathsf{ACast}, \mathsf{sid}, \mathcal{W}, \{\mathcal{W}_j\}_{P_j \in \mathcal{W}})$ from $\mathcal{F}_{\mathsf{ACast}}$ where D is the sender, each party $P_m \in \mathcal{P}$ checks if $\mathcal{W}$ and $\{\mathcal{W}_j\}_{P_j \in \mathcal{W}}$ are *valid* by verifying if all the following conditions hold.
  - $|\mathcal{W}| \geq n - t$ and $(P_j, \mathsf{ACast}, \mathsf{sid}, \mathtt{RR}_j)$ is received from $\mathcal{F}_{\mathsf{ACast}}$ for every $P_j \in \mathcal{W}$, where P_j is the sender.
  - For each $P_j \in \mathcal{W}$, $|\mathcal{W}_j| \geq n - t$. Moreover, for each $P_i \in \mathcal{W}_j$, all the following conditions hold.
    - $(P_j, \mathsf{ACast}, \mathsf{sid}, (\mathtt{SR}_j, P_i))$ is received from $\mathcal{F}_{\mathsf{ACast}}$, where P_j is the sender.
    - There exist $2t$ parties $P_k \in \mathcal{P} \setminus \{P_j\}$, such that $(P_k, \mathsf{ACast}, \mathsf{sid}, \mathtt{RR}_k)$ is received from $\mathcal{F}_{\mathsf{ACast}}$ and also $(P_k, \mathsf{ACast}, \mathsf{sid}, (\mathtt{SR}_k, P_i))$ is received from $\mathcal{F}_{\mathsf{ACast}}$, where P_k is the sender.

Figure 8: Protocol for generating two-level secret-sharing with IC-signatures of a single secret for session id sid.

We next proceed to prove the properties of the protocol Sh. Before that, we would like to stress that we *do not* model the computation done in the protocol Sh by a corresponding ideal functionality and we do not provide a UC-security proof for the protocol. This is because as mentioned earlier, in our ACSS protocol, the dealer will invoke n instances of the protocol Sh, where the output of all the n instances of Sh are determined based on certain conditions, which should hold *jointly* for all the n instances of Sh. More specifically, the parties should keep executing all the n instances of Sh, till dealer finds a *common* $\mathcal{W}$ set of size $n - t$ for all the n instances and publicly gets it verified, which will mark the completion of all the n instances of Sh. Modelling these requirements in an ideal functionality will bring in additional technical challenges. Looking ahead, while giving a UC-security proof for our ACSS protocol, we will show how to simulate the steps executed as part of the n instances of ACSS. In this section, we only prove the relevant properties of Sh, which will be later utilized by us while giving the security proof for our ACSS protocol. Also during the proofs (and lemma statements) we skip the session id sid.

We first show that if D is *honest*, then it eventually finds a valid $\mathcal{W}$ set.

Lemma 4.2. *In protocol* Sh, *if* D *is honest, then except with probability* $n^2 \cdot \epsilon_{\mathsf{AICP}}$, *all honest parties are included in the* $\mathcal{W}$ *set. This further implies that* D *eventually finds a valid* $\mathcal{W}$ *set.*

Proof. Since D is *honest*, each *honest* P_i eventually receives the degree-t column-polynomial $g_i(y)$ from D. So eventually, P_i gives the signatures on the values of $g_i(y)$ to D and broadcasts $\texttt{SC}_i$. As there are at least $n-t$ honest parties who broadcast $\texttt{SC}_i$, it implies that D eventually finds a set $\mathcal{M}$ of size $n-t$ and broadcasts the same.

Next consider an arbitrary *honest* party P_j. Since D is honest, it follows that corresponding to *any* $P_i \in \mathcal{M}$, the signature $\mathsf{ICSig}(P_i \to D, f_{ji})$ revealed by D to P_j will be accepted by P_j: while this is always true for an *honest* P_i (follows the correctness property of AICP), for a *corrupt* $P_i \in \mathcal{M}$ it holds except with probability ϵ_{AICP} (follows from the non-repudiation property of AICP). Moreover, the revealed values $\{(\alpha_i, f_{ji})\}_{P_i \in \mathcal{M}}$ interpolate to a degree-t row-polynomial. As there can be at most $t \leq n$ corrupt parties P_i in $\mathcal{M}$, it follows that except with probability $n \cdot \epsilon_{\mathsf{AICP}}$, the conditions for P_j to broadcast $\texttt{RR}_j$ are satisfied and hence P_j eventually broadcasts $\texttt{RR}_j$. As there are at most n honest parties, it follows that except with probability $n^2 \cdot \epsilon_{\mathsf{AICP}}$, all honest parties eventually broadcast $\texttt{RR}$.

Finally, consider an arbitrary pair of *honest* parties P_i, P_j. Since D is *honest*, the condition $f_j(\alpha_i) = g_i(\alpha_j)$ holds. Now P_i eventually receives $\mathsf{f}_{ji} = f_j(\alpha_i)$ from P_j for signing and finds that $\mathsf{f}_{ji} = g_i(\alpha_j)$ holds and hence gives the signature $\mathsf{ICSig}(P_i \to P_j, \mathsf{f}_{ji})$ to P_j. Consequently, P_j eventually broadcasts $(\texttt{SR}_j, P_i)$. As there are at least $n-t$ honest parties P_k, who eventually broadcast $(\texttt{SR}_k, P_i)$, it follows that P_i is eventually included in the set $\mathcal{W}_j$. As there are at least $n-t$ honest parties, the set $\mathcal{W}_j$ eventually attains the size $n-t$ and hence P_j is eventually included in $\mathcal{W}$. ☐

We next show that if the honest parties receive any valid $\mathcal{W}$ set, then they receive the same $\mathcal{W}$ set.

Lemma 4.3. *In protocol* Sh*, if some honest party receives a valid* $\mathcal{W}$ *set from* D*, then every other honest party eventually receives the same valid* $\mathcal{W}$ *set from* D.

Proof. Since the $\mathcal{W}$ set is broadcasted, it follows from the properties of broadcast that all honest parties will receive the same $\mathcal{W}$ set, if at all D broadcasts any $\mathcal{W}$ set. Now it is easy to see that if a broadcasted $\mathcal{W}$ set is found to be valid by some *honest* party P_m, then it will be considered as valid by every other honest party. This is because in Sh the validity conditions for $\mathcal{W}$ which hold for P_m will eventually hold for every other honest party. Namely, P_m finds $\mathcal{W}$ to be valid, based on various broadcasted $\texttt{RR}$ and $\texttt{SR}$ messages, received by P_m. From the property of broadcast, the same messages are eventually received by every other honest party and hence they also consider $\mathcal{W}$ to be valid. ☐

We next show that if at least $n-t$ parties broadcast $\texttt{RR}$ messages, then it implies that D has distributed "consistent" row and column-polynomials lying on a unique degree-(t,t) bivariate polynomial to "sufficiently" many honest parties.

Lemma 4.4. *Let* $\mathcal{R}$ *be the set of parties* P_j*, who broadcast* $\texttt{RR}_j$ *messages during* Sh*. If* $|\mathcal{R}| \geq n-t$*, then except with probability* $n^2 \cdot \epsilon_{\mathsf{AICP}}$*, there exists a degree-*(t,t) *bivariate polynomial, say* $\overline{F}(x,y)$*, such that all the following hold.*

- *If* D *is honest then* $\overline{F}(x,y) = F(x,y)$.
- *Each honest* $P_j \in \mathcal{R}$ *holds a degree-*t *row-polynomial* $f_j(x)$ *that lies on* $F(x,y)$*. That is,* $f_j(x) = \overline{F}(x, \alpha_j)$ *holds.*
- *Each honest* $P_i \in \mathcal{M}$ *holds a degree-*t *column-polynomial* $g_i(y)$ *that lies on* $F(x,y)$*. That is,* $g_i(y) = \overline{F}(\alpha_i, y)$ *holds.*

Proof. Let l and m be the number of *honest* parties in the set $\mathcal{R}$ and $\mathcal{M}$ respectively. Since $|\mathcal{R}| \geq n-t$ and $|\mathcal{M}| = n-t$, it follows that $l, m \geq t+1$. For simplicity and without loss of generality, let $\{P_1, \ldots, P_l\}$ and $\{P_1, \ldots, P_m\}$ be the honest parties in $\mathcal{R}$ and $\mathcal{M}$ respectively. We claim that except with probability ϵ_{AICP}, the condition $f_j(\alpha_i) = g_i(\alpha_j)$ holds for each $j \in \{1, \ldots, l\}$ and $i \in \{1, \ldots, m\}$, where $f_j(x)$ and $g_i(y)$ are the degree-t row and column-polynomials held by P_j and P_i respectively. The lemma then follows from the properties of degree-(t,t) bivariate polynomials (Lemma 2.5) and the fact that there can be at most n^2 pairs of honest parties (P_i, P_j). We next proceed to prove our claim.

The claim is trivially true with probability 1, if D is *honest*, as in this case, the row and column-polynomials of each pair of honest parties P_i, P_j will be pair-wise consistent. So we consider the case when D is *corrupt*. Let P_j and P_i be arbitrary parties in the set $\{P_1, \ldots, P_l\}$ and $\{P_1, \ldots, P_m\}$ respectively. Since P_j broadcasts $\texttt{RR}_j$, it implies that P_j accepted the signature $\mathsf{ICSig}(P_i \rightarrow D, f_{ji})$, revealed by D to P_j. Moreover, the values $(\alpha_1, f_{j1}), \ldots, (\alpha_m, f_{jm})$ interpolated to a degree-t polynomial $f_j(x)$. Furthermore, P_j also receives $\texttt{SC}_i$ from the broadcast of P_i. From the unforgeability property of AICP, it follows that except with probability ϵ_{AICP}, the signature $\mathsf{ICSig}(P_i \rightarrow D, f_{ji})$ is indeed given by P_i to D. However, the value f_{ji} on which P_i issued the signature lies on the column-polynomial $g_i(y)$ of P_i. Namely, the value f_{ji} is the same as $g_i(\alpha_j)$, which further implies that $f_j(\alpha_i) = g_i(\alpha_j)$ holds, thus proving our claim.

Finally, it is easy to see that $\overline{F}(x,y) = F(x,y)$ holds for an honest D, as in this case, the row and column-polynomials of each party lie on $F(x,y)$. $\square$

The next lemma shows that if D makes public a valid $\mathcal{W}$ set, then it implies that D's input is eventually two-level t-shared with IC-signatures.

Lemma 4.5. *In the protocol* Sh*, if* D *broadcasts a valid* $\mathcal{W}$*, then except with probability* $n^2 \cdot \epsilon_{\mathsf{AICP}}$*, there exists some* $\overline{s} \in \mathbb{F}$*, where* $\overline{s} = s$ *for an honest* D*, such that* $\overline{s}$ *is eventually two-level* t*-shared with IC-signatures.*

Proof. Since the $\mathcal{W}$ set is valid, it implies that the honest parties receive $\mathcal{W}$ and $\mathcal{W}_j$ for each $P_j \in \mathcal{W}$ from the broadcast of D, where $|\mathcal{W}| \geq n - t = 2t+1$ and $|\mathcal{W}_j| \geq n - t = 2t+1$. Moreover, the parties receive $\texttt{RR}_j$ from the broadcast of each $P_j \in \mathcal{W}$. Since $|\mathcal{W}| \geq 2t+1$, it follows from Lemma 4.4, that except with probability $n^2 \cdot \epsilon_{\mathsf{AICP}}$, there exists a degree-$(t,t)$ bivariate polynomial, say $\overline{F}(x,y)$, where $\overline{F}(x,y) = F(x,y)$ for an honest D, such that the row-polynomial $f_j(x)$ held by each *honest* $P_j \in \mathcal{W}$ satisfies $f_j(x) = \overline{F}(x, \alpha_j)$ and the column-polynomial $g_i(y)$ held by each *honest* $P_i \in \mathcal{M}$ satisfies $g_i(y) = \overline{F}(\alpha_i, y)$. We define $\overline{s} \overset{\text{def}}{=} \overline{F}(0,0)$ and show that $\overline{s}$ is two-level t-shared with IC-signatures.

We first show the primary and secondary-shares corresponding to $\overline{s}$. Consider the degree-t polynomial $g_0(y) \overset{\text{def}}{=} \overline{F}(0,y)$. Since $\overline{s} = g_0(0)$, the value $\overline{s}$ is t-shared among $\mathcal{W}$ through $g_0(y)$, with each $P_j \in \mathcal{W}$ holding its primary-share $\overline{s}_j \overset{\text{def}}{=} g_0(\alpha_j) = f_j(0)$. Moreover, each primary-share $\overline{s}_j$ is further t-shared among $\mathcal{W}_j$ through the degree-t row-polynomial $f_j(x)$, with each $P_i \in \mathcal{W}_j$ holding its secondary-share $f_j(\alpha_i)$ in the form of $g_i(\alpha_j)$. If D is *honest*, then $\overline{s} = s$ as $\overline{F}(x,y) = F(x,y)$ for an honest D. We next show that each $P_j \in \mathcal{W}$ holds the IC-signatures of the *honest* parties from the $\mathcal{W}_j$ set on the secondary-shares of $\overline{s}_j$.

Consider an *arbitrary* $P_j \in \mathcal{W}$. We claim that corresponding to each *honest* $P_i \in \mathcal{W}_j$, party P_j holds the signature $\mathsf{ICSig}(P_i \rightarrow P_j, \mathsf{f}_{ji})$, where $\mathsf{f}_{ji} = \overline{F}(\alpha_i, \alpha_j)$. The claim is trivially true for an *honest* P_j. This is because $\mathsf{f}_{ji} = f_j(\alpha_i) = \overline{F}(\alpha_i, \alpha_j)$ and P_i is included in $\mathcal{W}_j$ because P_j broadcasts $(\texttt{SR}_j, P_i)$ message, further implying that P_j indeed received the signature $\mathsf{ICSig}(P_i \rightarrow P_j, \mathsf{f}_{ji})$ from P_i. We next show that the claim is true, even for a *corrupt* $P_j \in \mathcal{W}$. For this, we show that for each *honest* $P_i \in \mathcal{W}_j$, the column-polynomial $g_i(y)$ held by P_i satisfies the condition that $g_i(y) = \overline{F}(\alpha_i, y)$. The claim then follows from the fact that P_i gives the signature $\mathsf{ICSig}(P_i \rightarrow P_j, \mathsf{f}_{ji})$ to P_j, only after verifying that the condition $\mathsf{f}_{ji} = g_i(\alpha_j)$ holds.

So consider a *corrupt* $P_j \in \mathcal{W}$ and an *honest* $P_i \in \mathcal{W}_j$. We note that P_i is included to $\mathcal{W}_j$, only if there is a set $\mathcal{S}_i$ of at least $2t+1$ parties, such that $P_j \in \mathcal{S}_i$ and each party P_k in $\mathcal{S}_i$ has broadcasted $\texttt{RR}_k$ and $(\texttt{SR}_k, P_i)$ messages. Let $\mathcal{H}$ be the set of *honest* parties in $\mathcal{S}_i$. For each $P_k \in \mathcal{H}$, the row-polynomial $f_k(x)$ held by P_k satisfies the condition $f_k(x) = \overline{F}(x, \alpha_k)$ (follows from the proof of Lemma 4.4). Furthermore, for each $P_k \in \mathcal{H}$, the condition $f_k(\alpha_i) = g_i(\alpha_k)$ holds, where $g_i(y)$ is the degree-t column-polynomial held by the honest P_i. This is because P_k broadcasts $(\texttt{SR}_k, P_i)$, only after receiving the signature $\mathsf{ICSig}(P_i \rightarrow P_k, \mathsf{f}_{ki})$ from P_i, where $\mathsf{f}_{ki} = f_k(\alpha_i)$. And P_i gives the signature to P_k only after verifying that $\mathsf{f}_{ki} = g_i(\alpha_k)$ holds for P_i. Now since $|\mathcal{H}| \geq t+1$ and $g_i(\alpha_k) = f_k(\alpha_i) = \overline{F}(\alpha_i, \alpha_k)$ holds for each $P_k \in \mathcal{H}$, it follows that the column-polynomial $g_i(y)$ held by P_i satisfies the condition $g_i(y) = \overline{F}(\alpha_i, y)$. This is because both $g_i(y)$ and $\overline{F}(\alpha_i, y)$ are degree-t polynomials and two *different* degree-t polynomials can have at most t common values. □

We next show that in the protocol, if D is *honest*, then its input remains perfectly-secure.

Lemma 4.6. *If* D *is honest then in protocol* Sh*, the view of adversary is independent of* s*.*

Proof. Let $\mathcal{C}$ denote the set of corrupt parties, where $|\mathcal{C}| \leq t$. We claim that throughout the protocol Sh, the only information learnt by the adversary corresponding to the bivariate polynomial $F(x, y)$ selected by D are the degree-t row-polynomials $\{f_i(x)\}_{P_i \in \mathcal{C}}$ and the degree-t column-polynomials $\{g_i(y)\}_{P_i \in \mathcal{C}}$. Since $F(x, y)$ has degree-(t, t) and is selected uniformly at random by D where $F(0, 0)$ is the secret s (which is known only to D), the lemma easily follows from the properties of degree-(t, t) bivariate polynomials (see Lemma 2.6). We next proceed to prove the claim.

During the protocol Sh, the adversary gets the polynomials $\{g_i(y)\}_{P_i \in \mathcal{C}}$ and $\{f_i(x)\}_{P_i \in \mathcal{C}}$ from D during the stage I and stage II respectively. Next consider an arbitrary party $P_i \in \mathcal{C}$. Now corresponding to each *honest* party P_j, party P_i receives $\mathsf{f}_{ji} = f_j(\alpha_i)$ for signature from P_j during stage III. However the value f_{ji} is already known to P_i, since $\mathsf{f}_{ji} = g_i(\alpha_j)$ holds and hence this does not add any new information to the view of the adversary.

Next consider an arbitrary pair of *honest* parties P_i, P_j. These parties exchange f_{ji} and f_{ij} with each other over the pair-wise secure channel and hence nothing about these values is learnt by the adversary. Party P_i gives the signature $\mathsf{ICSig}(P_i \rightarrow P_j, \mathsf{f}_{ji})$ to P_j and from the privacy property of AICP, the view of the adversary remains independent of the signed value. Similarly, party P_j gives the signature $\mathsf{ICSig}(P_j \rightarrow P_i, \mathsf{f}_{ij})$ to P_i and from the privacy property of AICP, the view of the adversary remains independent of the signed value. □

Lemma 4.7. *The communication complexity of* Sh *is* $\mathcal{O}(n^3\kappa^2 + n^4)$ *bits.*

Proof. In the protocol D distributes n row and column-polynomials, which requires a communication of $\mathcal{O}(n^2\kappa)$ bits. There are $\Theta(n^2)$ instances of AICP, each dealing with $L = 1$ value, which from Theorem 3.7 costs a total communication of $\mathcal{O}(n^3\kappa^2)$ bits. In addition, D broadcasts a $\mathcal{W}$ set and at most n $\mathcal{W}_j$ sets, each of which can be represented by a n-bit vector. Using the Bracha's reliable broadcast protocol for realizing the instances of $\mathcal{F}_{\mathsf{ACast}}$, broadcasting $\mathcal{W}$ set and $\mathcal{W}_j$ sets will cost a communication of $\mathcal{O}(n^4)$ bits. Similarly, there are $\mathcal{O}(n^2)$ various messages involving pair of parties (P_i, P_j) which are broadcasted in the protocol, which will cost a communication of $\mathcal{O}(n^4)$ bits. □

4.1 Designated Reconstruction of Two-level t-shared Values

In our ACSS protocol, apart from generating two-level (incomplete) t-sharing of values with IC-signatures, we would also need a protocol for getting such shared values reconstructed *only* by some *designated* party. Protocol $\mathsf{RecPriv}$ is designed for the same purpose. In a more detail, let s be a value which has been two-level t-shared with IC-signatures by protocol Sh, with parties knowing a *valid* $\mathcal{W}$ set and respective $\mathcal{W}_j$ sets for each $P_j \in \mathcal{W}$. Then protocol $\mathsf{RecPriv}$ (see Fig 9) allows the reconstruction of s by a *designated* party P_R.[10] In the protocol, each party $P_j \in \mathcal{W}$ reveals its primary-share to P_R. Once $t+1$ "valid" primary-shares are

[10]Looking ahead, in our ACSS protocol, during the instances of $\mathsf{RecPriv}$, the identity of the corresponding P_R will be publicly known.

revealed to P_R, it uses them to reconstruct s, by interpolating a degree-t polynomial through these shares. For the validation of primary-shares, each party $P_j \in \mathcal{W}$ actually reveals the secondary-shares, signed by the parties in $\mathcal{W}_j$. The presence of at least $t+1$ *honest* parties in $\mathcal{W}_j$ ensures that a potentially *corrupt* P_j fails to reveal incorrect primary-share.

Protocol $\mathsf{RecPriv}(\mathsf{D}, s, P_R)$

The inputs to the parties are publicly known sets $\mathcal{W}$ and $\mathcal{W}_j$ for each $P_j \in \mathcal{W}$, where $|\mathcal{W}| \geq n-t$ and each $|\mathcal{W}_j| \geq n-t$. Additionally, each $P_j \in \mathcal{W}$ holds the signatures $\{\mathsf{ICSig}(\mathsf{sid}, P_i \rightarrow P_j, \mathsf{f}_{ji})\}_{P_i \in \mathcal{W}_j}$, where the values $\{\mathsf{f}_{ji}\}_{P_i \in \mathcal{W}_j}$ are the secondary-shares of P_j's primary-share of s.

- **Revealing the Signed Secondary-Shares**: Each $P_j \in \mathcal{W}$ executes the following code.
  - Corresponding to each $P_i \in \mathcal{W}_j$, reveal $\mathsf{ICSig}(\mathsf{sid}, P_i \rightarrow P_j, \mathsf{f}_{ji})$ to P_R.
- **Verifying the Signatures and Reconstruction**: The following code is executed only by P_R.
  - Include party $P_j \in \mathcal{W}$ to a set $\mathcal{K}$ (initialized to $\emptyset$), if all the following hold:
    - P_R accepted $\mathsf{ICSig}(\mathsf{sid}, P_i \rightarrow P_j, \mathsf{f}_{ji})$, corresponding to each $P_i \in \mathcal{W}_j$.
    - The values $\{(\alpha_i, \mathsf{f}_{ji})\}_{P_i \in \mathcal{W}_j}$ lie on a degree-t polynomial, say $f_j(x)$.
  - Wait till $|\mathcal{K}| = t+1$. Then interpolate a degree-t polynomial, say $g_0(y)$, using the values $\{\alpha_j, f_j(0)\}_{P_j \in \mathcal{K}}$. Output s, where $s = g_0(0)$.

Figure 9: Reconstruction of a two-level t-shared value by a designated party for session id sid.

The properties of RecPriv are stated in Lemma 4.8.

Lemma 4.8. *Let s be a value, which is two-level t-shared with IC-signatures through protocol* Sh. *Then in protocol* RecPriv*, the following hold for every possible adversary.*

- **Output Computation**: *If P_R is honest, then except with probability at most $n^2 \cdot \epsilon_{\mathsf{AICP}}$, it obtains an output.*
- **Correctness**: *Except with probability at most $n^2 \cdot \epsilon_{\mathsf{AICP}}$, an honest P_R outputs s.*
- **Privacy**: *If P_R is honest, then the view of the adversary is independent of s.*
- **Communication Complexity**: *The protocol needs a communication of $\mathcal{O}(n^3\kappa^2)$ bits.*

Proof. For output computation, we claim that an *honest* $P_j \in \mathcal{W}$ is eventually included in the set $\mathcal{K}$, except with probability at most $n \cdot \epsilon_{\mathsf{AICP}}$. Assuming that the claim is true, it follows that except with probability at most $n^2 \cdot \epsilon_{\mathsf{AICP}}$, all honest parties in $\mathcal{W}$ are eventually included in $\mathcal{K}$. Since $\mathcal{W}$ has at least $t+1$ honest parties, it follows that an honest P_R eventually obtains at least $t+1$ primary-shares of s, using which it can reconstruct s. For proving the claim, we observe that an honest P_j is included in $\mathcal{K}$, if all the signatures $\{\mathsf{ICSig}(P_i \rightarrow P_j, \mathsf{f}_{ji})\}_{P_i \in \mathcal{W}_j}$ revealed by P_j are accepted by P_R. While for every *honest* $P_i \in \mathcal{W}_j$, the signature $\mathsf{ICSig}(P_i \rightarrow P_j, \mathsf{f}_{ji})$ is accepted by P_R with probability 1 (follows from the correctness property of AICP), the signature $\mathsf{ICSig}(P_i \rightarrow P_j, \mathsf{f}_{ji})$ corresponding to any *corrupt* $P_i \in \mathcal{W}_j$ is accepted, except with probability ϵ_{AICP} (follows from the non-repudiation property of AICP). As there can be at most $t < n$ corrupt parties in $\mathcal{W}_j$, it follows that except with probability at most $n \cdot \epsilon_{\mathsf{AICP}}$, all the signatures $\{\mathsf{ICSig}(P_i \rightarrow P_j, \mathsf{f}_{ji})\}_{P_i \in \mathcal{W}_j}$ revealed by P_j are accepted by P_R.

For correctness, we observe that each *honest* $P_j \in \mathcal{K}$ reveals its secondary-shares correctly. We claim that for any *corrupt* $P_j \in \mathcal{K}$, the revealed secondary-shares are correct, except with probability at most $n \cdot \epsilon_{\mathsf{AICP}}$. Assuming that the claim is true, it follows that except with probability at most $n^2 \cdot \epsilon_{\mathsf{AICP}}$, the secondary-shares revealed by all the parties in $\mathcal{K}$ are correct, thus ensuring that an honest P_R reconstruct the correct s. For proving the claim, we note that there are at least $t+1$ *honest* parties P_i in the set $\mathcal{W}_j$. Since the signatures $\mathsf{ICSig}(P_i \rightarrow P_j, \mathsf{f}_{ji})$ corresponding to these honest parties P_i are accepted, it follows from the unforgeability property of AICP, that except with probability at most ϵ_{AICP}, party P_j revealed the correct secondary-share f_{ji} to P_R, corresponding to an *honest* $P_i \in \mathcal{W}_j$. The claim now follows from the fact that there can be at most n honest parties in $\mathcal{W}_j$.

The privacy follows from the privacy of AICP and the fact that all the instances of signature-revelation are towards P_R. Communication complexity follows from the communication complexity of AICP and the fact that there are $\Theta(n^2)$ instances of AICP involved. $\square$

4.2 Sharing and Reconstructing Degree-t Polynomial Using Sh and $\mathsf{RecPriv}$

Even though D's input in the protocol Sh is a value which it wants to share, we observe that D's computation in the protocol Sh can be recast as if D wants to share the degree-t Shamir-sharing polynomial $\bar{F}(0, y)$, where $\bar{F}(0, 0)$ is the value which D wants to Shamir-share. If D broadcasts a *valid* $\mathcal{W}$ set during stage IV, then it implies that each $P_j \in \mathcal{W}$ has received the share $\bar{F}(0, \alpha_j)$ from D, where P_j has received the degree-t signed row-polynomial $\bar{F}(x, \alpha_j)$ from the dealer. Here $\bar{F}(x, y)$ is the degree-(t, t) bivariate polynomial committed by D, which is the same as $F(x, y)$ for an *honest* D (see the pictorial representation in Fig 6 and the proof of Lemma 4.5). If D is *honest*, then adversary learns at most t shares of the polynomial $F(0, y)$, corresponding to the corrupt parties in $\mathcal{W}$ (see the proof of Lemma 4.6). In the protocol, apart from $P_j \in \mathcal{W}$, every other party P_j who broadcasts the message $\texttt{RR}_j$ also receives its share $\bar{F}(0, \alpha_j)$, lying on $\bar{F}(0, y)$, as the row-polynomial received by every such P_j also lies on $\bar{F}(x, y)$. Based on these observations, we propose the following alternate notation for using the protocol Sh, where the input for D is a degree-t polynomial, instead of a value. This notation will later simplify the presentation of our ACSS protocol.

Notation 4.9 (Sharing Polynomial Using Protocol Sh). In the rest of the paper, we use the following notations and interpretations, while using the protocol Sh.

- We say that D *invokes* $\mathsf{Sh}(\mathsf{D}, r(\cdot))$, where $r(\cdot)$ is some degree-t polynomial possessed by D, to denote that D invokes the protocol Sh by picking a degree-(t, t) bivariate polynomial $F(x, y)$ during stage I, which is otherwise a random polynomial, except that $F(0, y) = r(\cdot)$ holds.
- If D broadcasts a valid $\mathcal{W}$ set and corresponding $\mathcal{W}_j$ sets during stage IV of the instance $\mathsf{Sh}(\mathsf{D}, r(\cdot))$, then it implies that there exists some degree-t polynomial, say $\bar{r}(\cdot)$, where $\bar{r}(\cdot) = r(\cdot)$ for an honest D, such that each $P_j \in \mathcal{W}$ holds a primary-share $\bar{r}(\alpha_j)$. Moreover, each primary-share $\bar{r}(\alpha_j)$ is further t-shared among the parties in $\mathcal{W}_j$.
- We say that a party P_j *receives* a share r_j during $\mathsf{Sh}(\mathsf{D}, r(\cdot))$ from D to denote that P_j receives a degree-t signed row-polynomial from D during stage II, with r_j as its constant term and has broadcasted $\texttt{RR}_j$ message.

The computations done by the parties in $\mathsf{RecPriv}$ can be similarly recast as if parties enable a designated P_R to reconstruct a degree-t polynomial $r(\cdot)$, which has been shared by D during an instance $\mathsf{Sh}(\mathsf{D}, r(\cdot))$ of Sh (see Notation 4.9). This is because in $\mathsf{RecPriv}$, party P_R recovers the entire column-polynomial $g_0(y)$, which is the same as $F(0, y)$. And as discussed in Notation 4.9, to share $r(\cdot)$, the dealer D executes Sh by setting $F(0, y)$ to $r(\cdot)$. Based on this discussion, we propose the following alternate notation for reconstructing a shared polynomial by P_R using $\mathsf{RecPriv}$, which will later simplify the presentation of our ACSS protocol.

Notation 4.10 (Reconstructing a Shared Polynomial Using $\mathsf{RecPriv}$). Let $r(\cdot)$ be a degree-t polynomial which has been shared by D by executing an instance $\mathsf{Sh}(\mathsf{D}, r(\cdot))$ of Sh. Then $\mathsf{RecPriv}(\mathsf{D}, r(\cdot), P_R)$ denotes that the parties execute the steps of the protocol $\mathsf{RecPriv}$ to enable P_R reconstruct $r(0)$, which implicitly allows P_R to reconstruct the entire polynomial $r(\cdot)$.

4.3 Protocols Sh and $\mathsf{RecPriv}$ for L Polynomials

To share L number of degree-t polynomials $r^{(1)}(\cdot), \ldots, r^{(L)}(\cdot)$, D can execute L independent instances of Sh (as per Notation 4.9), where the ℓ^{th} instance is used for sharing $r^{(\ell)}(\cdot)$. This will require a broadcast of L instances of $\mathcal{W}$ sets and the corresponding $\mathcal{W}_j$ sets for each $\mathcal{W}$. Each instance of broadcast needs an execution of the Bracha's Acast protocol, which has a communication overhead of $\mathcal{O}(n^2)$ bits. Instead, by making slight modifications, we ensure that the broadcast complexity is *independent* of the number of shared polynomials L.

In the modified protocol, each P_i while issuing signatures to any party, issues a *single* signature on all the required values, on the behalf of all the L instances. For instance, as part of the recommitment of row-polynomials during stage III, party P_j will have L row-polynomials (one from each Sh instance) and there will be L common values on these polynomials between P_i and P_j, so P_i needs to sign L values for P_j. Party P_i issues signature on the common values on all these L polynomials simultaneously and for this *only one* instance of AICP is executed, instead of L instances. Similarly, during the first stage, each party P_i will receive L degree-t column-polynomials for signing and it has to sign total $L \cdot n$ values on these polynomials. Now instead of executing $L \cdot n$ independent instances of AICP, party P_i will execute only n instances of AICP, where in the j^{th} instance, P_i issues signature on the j^{th} value of all the L column-polynomials (namely the value of all the column-polynomials at $y = \alpha_j$).

Thus all instances of AICP now deal with L values. To make the broadcast complexity independent of L, each P_j broadcasts a *single* $\texttt{RR}_j, \texttt{SC}_j$ and $(\texttt{SR}_j, P_i)$ message, if the conditions for broadcasting these messages are satisfied with respect to *all* the L instances of Sh. For instance, P_j broadcasts a *single* $\texttt{RR}_j$ message after receiving all the L signed row-polynomials from D on the behalf of all the L instances of Sh. Finally, as part of recommitment of P_j's row-polynomials, a single $\mathcal{W}_j$ set is constructed on the behalf of all the L instances of Sh. And similarly D constructs a single $\mathcal{W}$ set with respect to all the L instances of Sh. We call the resultant modified protocol as $\mathsf{MSh}(\mathsf{D}, (r^{(1)}(\cdot), \ldots, r^{(L)}(\cdot)))$. As most of the steps of MSh are similar to L instances of Sh being executed in parallel (with the above modifications), to avoid repetition, we do not give the complete formal details of MSh. The communication complexity for MSh will be $\mathcal{O}(L \cdot n^2\kappa + n^3\kappa^2 + n^4)$ bits.

To enable P_R reconstruct the polynomials $r^{(1)}(\cdot), \ldots, r^{(L)}(\cdot)$ shared using MSh, the parties execute L instances of $\mathsf{RecPriv}$. But each instance of signature revelation now deals with L values. The communication complexity of the protocol will be $\mathcal{O}(L \cdot n^2\kappa + n^3\kappa^2)$ bits.

5 Asynchronous Complete Secret Sharing (ACSS)

In this section, we present a protocol which allows a designated *dealer* $\mathsf{D} \in \mathcal{P}$ to verifiably generate complete t-sharing of a set of values $S = (s^{(1)}, \ldots, s^{(L)}) \in \mathbb{F}^L$ held by D. The *verifiability* here ensures that if the (honest) parties output shares, then they correspond to complete t-sharing of some $\bar{S} \in \mathbb{F}^L$ held by the dealer, where $\bar{S} = S$ holds for an *honest* dealer.

The ideal functionality $\mathcal{F}_{\mathsf{ACSS}}$ modelling the requirements of our ACSS protocol is present in Fig 10. The functionality upon receiving a set of L polynomials (which we call as the *sharing-polynomials*) from the dealer, distributes distinct points on the polynomials to the respective parties, provided the input polynomials are degree-t univariate polynomials. This implicitly allows the constant term of the input polynomials to be completely t-shared. If any of the received polynomials is not of degree-t, then the functionality sends the output $\perp$ to each party, indicating that the sharing-polynomials of the dealer are "invalid". This ensures that the underlying sharing-polynomials are *verified* by the functionality. Notice that the functionality generates an output for the parties, *only* upon receiving the sharing-polynomials from the dealer. If the dealer does not provide any polynomial (which is possible only if the dealer is corrupt), then the functionality does not generate any output for the parties.

Looking ahead, the way this functionality will be used in our AMPC protocol is as follows: if P_i has a set of L values $S = (s^{(1)}, \ldots, s^{(L)})$, which it wants to completely t-share among the parties, then P_i acts as a dealer and picks L random degree-t univariate polynomials, whose constant terms are $s^{(1)}, \ldots, s^{(L)}$ and calls the functionality $\mathcal{F}_{\mathsf{ACSS}}$ with these polynomials. This will ensure that the values $s^{(1)}, \ldots, s^{(L)}$ remain information-theoretically secure for an *honest* Dealer.

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

$\mathcal{F}_{\mathsf{ACSS}}$ proceeds as follows, running with parties $\mathcal{P} = \{P_1, \ldots, P_n\}$ and an adversary $\mathcal{S}$.

- Upon receiving $(\mathsf{Dealer}, \mathsf{ACSS}, L, \mathsf{sid}, \{q^{(1)}(\cdot), \ldots, q^{(L)}(\cdot)\})$ from $\mathsf{D} \in \mathcal{P}$, generate a request-based delayed output for the parties as follows:
  - If all the polynomials $q^{(1)}(\cdot), \ldots, q^{(L)}(\cdot)$ are degree-t univariate polynomials, then send a request-based delayed output $(\mathsf{D}, \mathsf{Dealer}, L, \mathsf{sid}, \{q^{(1)}(\alpha_i), \ldots, q^{(L)}(\alpha_i)\})$ to each $P_i \in \mathcal{P}$.
  - If any of the polynomials $q^{(1)}(\cdot), \ldots, q^{(L)}(\cdot)$ is not a degree-t polynomial, then send a request-based delayed output $(\mathsf{D}, \mathsf{Dealer}, \mathsf{sid}, \perp)$ to each $P_i \in \mathcal{P}$.

Figure 10: The ideal functionality for asynchronous complete secret-sharing.

5.1 ACSS Protocol for Realizing $\mathcal{F}_{\mathsf{ACSS}}$ with $L = 1$

We now present our ACSS protocol Π_{ACSS} for securely realizing the functionality $\mathcal{F}_{\mathsf{ACSS}}$. We first explain the protocol assuming D has a single polynomial (and hence a single value) for sharing, namely when $L = 1$. The modifications for sharing L polynomials are straight forward.

Before explaining the idea of Π_{ACSS}, we first pause for a moment and try to understand that why protocol Sh fails to generate a complete t-sharing of D's input. The reason is that in Sh, if D is *corrupt*, then the set $\mathcal{W}$ *may not* include *all* the honest parties, as D *may not* reveal the respective signed row-polynomials to all the honest parties during stage II. Consequently, up to t honest parties may lie outside $\mathcal{W}$ and hence will not have (primary) shares corresponding to D's committed secret. Moreover, even if D is *honest*, the parties cannot confirm whether all the honest parties have been included in $\mathcal{W}$. This is because to avoid an indefinite wait, the parties cannot wait beyond till $\mathcal{W}$ achieves a size of $n - t$, leaving up to t potential honest parties outside $\mathcal{W}$. In protocol Π_{ACSS} we get rid of this problem and ensure that if some honest party obtains its share of D's committed secret, then eventually every other honest party obtains its share of the same secret, even if D is *corrupt*.

The idea of Π_{ACSS} is similar to that of Sh, where D embeds its sharing-polynomial $q(\cdot)$ in a random degree-(t, t) bivariate polynomial $H(x, y)$ at $x = 0$, that is $H(0, y) = q(\cdot)$ holds. The row-polynomials of $H(x, y)$ are *retained* by D, while the column-polynomials are given to respective parties. Next it is publicly verified whether there exists a set $\mathcal{V}$ of $n - t$ column holders, whose column-polynomials are pair-wise consistent with *all* the n row-polynomials held by D. This will confirm that the row-polynomials of D and the column-polynomials of the *honest* parties in $\mathcal{V}$ lie on a single degree-(t, t) bivariate polynomial, say $\bar{H}(x, y)$, which will be considered as D's committed bivariate polynomial, and $\bar{H}(0, y)$ is considered as D's committed sharing-polynomial. Moreover, if D is *honest*, then $\bar{H}(x, y)$ will be same as $H(x, y)$.

Once D's commitment is publicly confirmed, we ensure that every party P_i gets the i^{th} row-polynomial $\bar{H}(x, \alpha_i)$ held by D, which leads to the complete t-sharing of $\bar{H}(0, 0)$ through $\bar{H}(0, y)$, as $\bar{H}(0, \alpha_i)$ will be considered as P_i's share. We stress that the delivery of i^{th} row-polynomial to P_i will be successful, even if D is potentially *corrupt* and does not participate in the delivery process. It is this delivery process, which distinguishes protocol Π_{ACSS} from Sh. In protocol Sh, after the confirmation of D's committed bivariate polynomial, the delivery of individual row-polynomials compulsorily requires D's honest participation. And a potentially *corrupt* D may purposely decide *not* to distribute the row-polynomials to up to t honest parties, thus barring them from their shares of D's committed polynomial. We next discuss the construction of the set $\mathcal{V}$ and the delivery process of row-polynomials.

After D distributes the column-polynomials to respective parties, the parties proceed to verify if D delivered consistent column-polynomials, lying on a single degree-(t, t) bivariate polynomial. For this, D is asked to share its row-polynomials by invoking instances of Sh (this is where we use our interpretation of sharing degree-t univariate polynomial using Sh as discussed in Notation 4.9). Namely, D invokes n instances of Sh, where j^{th} instance is used to share the j^{th} row-polynomial. The goal is then to let D publicly identify a set $\mathcal{V}$ of $n-t$ column holders, whose column-polynomials are pair-wise consistent with *all* the n row-polynomials shared by D.

The set $\mathcal{V}$ is constructed as follows. Every party P_i participates in *all* the Sh instances invoked by D and checks if the shares received by P_i from D in these instances (see Notation 4.9 for the meaning of P_i receiving a share from D during an Sh instance) lie on the column-polynomial received by P_i (which should be the case if D is honest). If the check is successful, then party P_i publicly notifies this by broadcasting an OK message. The set $\mathcal{V}$ is then assigned to be a set of $n-t$ parties who broadcast OK messages and who also constitute a $\mathcal{W}$ set for *all* the n instances of Sh invoked by D. Notice that if D is *honest*, then such a common $\mathcal{V}$ set is eventually obtained, as there are at least $n-t$ honest parties, who constitute a potential $\mathcal{V}$ set. This is because if D keeps on running the Sh instances and keeps expanding the $\mathcal{W}$ sets of individual Sh instances, then eventually every honest party is included in the $\mathcal{W}$ sets of all the Sh instances (see Lemma 4.2). Once D identifies a common $\mathcal{W}$ set for all the n instances of Sh, it sets this common $\mathcal{W}$ as $\mathcal{V}$ and publicly announces the same. The parties then verify the publicly announced $\mathcal{V}$ set and check if it constitutes a valid $\mathcal{W}$ set for all the n instances of Sh invoked by D (this is where the the last stage of protocol Sh, shown in red color in Fig 8, is invoked for checking the validity of a publicly announced $\mathcal{W}$ set).

Once the set $\mathcal{V}$ is publicly announced and verified, to obtain a complete t-sharing of D's secret, we need to ensure that each party obtains the respective row-polynomial held by D. However, unlike the protocol Sh, this needs to be ensured even if a potentially *corrupt* D does not "help" in the process of delivering the row-polynomials to respective parties. Hence, the parties invoke instances of RecPriv, where the i^{th} instance is used to let *only* party P_i reconstruct the i^{th} row-polynomial (this is where we use our interpretation of using RecPriv to enable designated reconstruction of a shared degree-t polynomial, as discussed in Notation 4.10). We stress that once the common set $\mathcal{V}$ is publicly identified, *each* P_i obtains the desired row-polynomial, even if D is *corrupt*, as the corresponding RecPriv instance is eventually completed for P_i even for a corrupt D. Once the parties obtain their respective row-polynomials, the constant term of these polynomials constitute a complete t-sharing of D's committed value. For the formal details of Π_{ACSS}, see Fig 12 and for a pictorial depiction of the protocol, see Fig 11. In the protocol, we assume that the parties have access to the broadcast functionality $\mathcal{F}_{\mathsf{ACast}}$.

Protocol Π_{ACSS})

– **Distribution of Column-Polynomials and Sharing of Row-Polynomials by the dealer**: If $P_i = \mathsf{D}$, then execute the following steps.
  - On having the input $q(\cdot)$ where $q(\cdot)$ is a degree-t univariate polynomial, select a random degree-(t,t) bivariate polynomial $H(x,y)$ over $\mathbb{F}$, such that $H(0,y) = q(\cdot)$ holds.[a]
  - For $i = 1, \ldots, n$, send $(\mathsf{sid}, \mathsf{column}, c_i(y))$ to P_i, where $c_i(y) \overset{\text{def}}{=} H(\alpha_i, y)$.
  - For $j = 1, \ldots, n$, execute an instance $\mathsf{Sh}(\mathsf{sid}, \mathsf{D}, r_j(x))$, where $r_j(x) \overset{\text{def}}{=} H(x, \alpha_j)$. Let this instance of Sh be denoted as $\mathsf{Sh}^{(j)}_{\mathsf{sid}}$.[b]

– **Pair-wise Consistency Check**: Execute the following steps to receive data from D and verify it.
  - If $(\mathsf{sid}, \mathsf{column}, c_i(y))$ is received from D, then participate in the instances $\mathsf{Sh}^{(1)}_{\mathsf{sid}}, \ldots, \mathsf{Sh}^{(n)}_{\mathsf{sid}}$, provided $c_i(y)$ is a degree-t polynomial.
  - Send $(\mathsf{sender}, \mathsf{ACast}, \mathsf{sid}, \mathtt{OK}_i)$ to $\mathcal{F}_{\mathsf{ACast}}$ if all the following hold.
    – The shares $r_{1i}, \ldots, r_{ni}$ are received from D during the instances $\mathsf{Sh}^{(1)}_{\mathsf{sid}}, \ldots, \mathsf{Sh}^{(n)}_{\mathsf{sid}}$ respectively.[c]
    – The condition $r_{ji} = c_i(\alpha_j)$ holds for each $j = 1, \ldots, n$.

– **Construction of $\mathcal{V}$ and Public Announcement**: If $P_i = \mathsf{D}$, then execute the following steps.
  - Let $\mathcal{W}^{(\ell)}$ denote the instance of $\mathcal{W}$ set constructed during the instance $\mathsf{Sh}^{(\ell)}_{\mathsf{sid}}$ and for each $P_k \in \mathcal{W}^{(\ell)}$, let

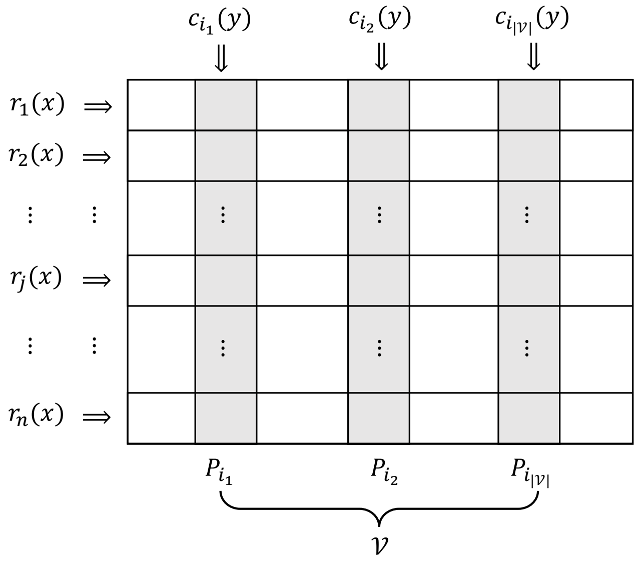


Figure 11: Pictorial depiction of the protocol Π_{ACSS}. The row-polynomials $r_1(x), \ldots, r_n(x)$ are shared by D by invoking n instances $\mathsf{Sh}_1, \ldots, \mathsf{Sh}_n$ of Sh, while the column-polynomials are distributed to respective parties. The parties in $\mathcal{V} = \{P_{i_1}, \ldots, P_{i_{|\mathcal{V}|}}\}$ constitute a common $\mathcal{W}$ set for all the n instances of Sh. Each party in $\mathcal{V}$ publicly confirms that the shares received by it during $\mathsf{Sh}_1, \ldots, \mathsf{Sh}_n$ lie on its column-polynomial, thus confirming the pair-wise consistency (highlighted by the shaded column polynomial) of its column polynomial and the row-polynomials shared by D. Once the set $\mathcal{V}$ is identified, the respective row-polynomials are reconstructed towards the designated parties by invoking instances of RecPriv

$\mathcal{W}_k^{(\ell)}$ denote the corresponding instance of $\mathcal{W}_k$ set during $\mathsf{Sh}_{\mathsf{sid}}^{(\ell)}$. Moreover, let $\mathcal{W} \overset{\text{def}}{=} \mathcal{W}^{(1)} \cap \ldots \cap \mathcal{W}^{(n)}$.

- For $\ell = 1, \ldots, n$, keep updating the set $\mathcal{W}^{(\ell)}$ and the corresponding sets $\mathcal{W}_k^{(\ell)}$ during the instance $\mathsf{Sh}_{\mathsf{sid}}^{(\ell)}$, till a set $\mathcal{V} \subseteq \mathcal{W}$ satisfying all the following conditions is obtained.
  - $|\mathcal{V}| \geq n - t$ holds.
  - For every $P_j \in \mathcal{V}$, the message $(P_j, \mathsf{ACast}, \mathsf{sid}, \mathtt{OK}_j)$ is received from $\mathcal{F}_{\mathsf{ACast}}$ where P_j is the sender.
- Send $(\mathsf{sender}, \mathsf{ACast}, \mathsf{sid}, \mathcal{V}, \{\mathcal{W}_j^{(1)}, \ldots, \mathcal{W}_j^{(n)}\}_{P_j \in \mathcal{V}})$ to $\mathcal{F}_{\mathsf{ACast}}$, once a set $\mathcal{V}$ satisfying the above conditions is obtained.

– **Verification of** $\mathcal{V}$: Execute the following steps.

- If $(\mathsf{D}, \mathsf{ACast}, \mathsf{sid}, \mathcal{V}, \{\mathcal{W}_j^{(1)}, \ldots, \mathcal{W}_j^{(n)}\}_{P_j \in \mathcal{V}})$ is received from $\mathcal{F}_{\mathsf{ACast}}$ where D is the sender, then check whether the set $\mathcal{V}$ and the sets $\{\mathcal{W}_j^{(1)}, \ldots, \mathcal{W}_j^{(n)}\}_{P_j \in \mathcal{V}}$ are *valid*, by verifying the following.
  - For every $P_j \in \mathcal{V}$, the message $(P_j, \mathsf{ACast}, \mathsf{sid}, \mathtt{OK}_j)$ is received from $\mathcal{F}_{\mathsf{ACast}}$ where P_j is the sender.
  - For $\ell = 1, \ldots, n$, the set $\mathcal{V}$ along with the sets $\{\mathcal{W}_j^{(\ell)}\}_{P_j \in \mathcal{V}}$ are valid during the instance $\mathsf{Sh}_{\mathsf{sid}}^{(\ell)}$.[d]
- If the set $\mathcal{V}$ and the sets $\{\mathcal{W}_j^{(1)}, \ldots, \mathcal{W}_j^{(n)}\}_{P_j \in \mathcal{V}}$ are found to be invalid, then output $(\mathsf{D}, \mathsf{Dealer}, \mathsf{sid}, \bot)$. Else invoke an instance $\mathsf{RecPriv}_{\mathsf{sid}}^{(i)} \overset{\text{def}}{=} \mathsf{RecPriv}(\mathsf{D}, r_i(x))$ to reconstruct the polynomial $r_i(x)$. Moreover, participate in the instances $\mathsf{RecPriv}_{\mathsf{sid}}^{(1)}, \ldots, \mathsf{RecPriv}_{\mathsf{sid}}^{(n)}$.[e]

– **Share Computation**: Execute the following steps.

- Wait to complete the instance $\mathsf{RecPriv}_{\mathsf{sid}}^{(i)}$ and obtain a degree-t row-polynomial $r_i(x)$.
- Upon reconstructing $r_i(x)$, output $(\mathsf{D}, \mathsf{Dealer}, 1, \mathsf{sid}, s_i)$, where $s_i \overset{\text{def}}{=} r_i(0)$.

[a] If $q(\cdot)$ is a polynomial in a variable different from y, then the variable can be renamed as y to ensure this condition holds.
[b] See Notation 4.9 for the interpretation of sharing degree-t univariate polynomial using Sh.
[c] See Notation 4.9 for the interpretation of a party receiving a share from D during an instance of Sh.

[d]For this, the party executes the code shown in red color for stage IV in protocol Sh.
[e]See Notation 4.10 for the interpretation of using $\mathsf{RecPriv}$ to enable reconstruction of a shared degree-t polynomial.

Figure 12: Protocol for securely realizing the functionality $\mathcal{F}_{\mathsf{ACSS}}$ for a single polynomial with session id sid in the $\mathcal{F}_{\mathsf{ACast}}$-hybrid model. The above code is executed by every $P_i \in \mathcal{P}$, including the dealer D.

We note that we separately *do not* put any termination condition for any party in protocol Π_{ACSS}. Since Π_{ACSS} will be used as a sub-protocol in our AMPC protocol, the termination condition of our AMPC protocol will automatically trigger the termination of all the underlying instances of Π_{ACSS}.

We next prove the security of the protocol Π_{ACSS}.

Theorem 5.1. *Protocol Π_{ACSS} UC-securely realizes the functionality $\mathcal{F}_{\mathsf{ACSS}}$ for $L = 1$ with statistical security in the $\mathcal{F}_{\mathsf{ACast}}$-hybrid model, in the presence of a static malicious adversary, corrupting at most $t < \frac{n}{3}$ parties. The protocol needs a communication of $\mathcal{O}(n^4\kappa^2 + n^5)$ bits.*

Proof. In the protocol, there are n instances of Sh involved, which from Lemma 4.7, incurs a communication of $\mathcal{O}(n^4\kappa^2 + n^5)$ bits. In the protocol, D needs to broadcast a $\mathcal{V}$ set of size $\mathcal{O}(n)$ and for each party $P_j \in \mathcal{V}$, it also needs to broadcast n number of $\mathcal{W}_j$ sets, each of size $\mathcal{O}(n)$. So overall D needs to broadcast $\mathcal{O}(n^3)$ bits, as each set can be represented by $\mathcal{O}(n)$ bits. By realizing $\mathcal{F}_{\mathsf{ACast}}$ with Bracha's reliable broadcast protocol, this requires a communication of $\mathcal{O}(n^5)$ bits.

We next prove the security of the protocol. Let Adv be an arbitrary real-world adversary, attacking protocol Π_{ACSS} and let $\mathcal{Z}$ be an arbitrary environment. We show the existence of a simulator $\mathcal{S}_{\mathsf{ACSS}}$, such that for any set of corrupted parties $\mathcal{C}$ with $|\mathcal{C}| \leq t$ and for all inputs, the output of all parties and the adversary in an execution of the real protocol Π_{ACSS} with Adv is statistically-indistinguishable from the outputs in an execution with $\mathcal{S}_{\mathsf{ACSS}}$ involving $\mathcal{F}_{\mathsf{ACSS}}$ in the ideal model. This further implies that from the view-point of $\mathcal{Z}$, the executions in the real-world and the ideal-world are statistically indistinguishable. The steps of the simulator (see Fig 13) will be different, depending upon whether the dealer is honest or corrupt.

The high level idea of the simulator is as follows. If the dealer is *honest*, then the simulator interacts with the functionality $\mathcal{F}_{\mathsf{ACSS}}$ and receives the shares of the corrupt parties, corresponding to the dealer's sharing-polynomial. The simulator then picks a random degree-t sharing polynomial on the behalf of the dealer, consistent with the shares of the corrupt parties received from $\mathcal{F}_{\mathsf{ACSS}}$ and with this polynomial being dealer's input, the simulator plays the role of the honest parties (including the dealer) as per the protocol Π_{ACSS} and interacts with Adv. Moreover, the role of $\mathcal{F}_{\mathsf{ACast}}$ is played by the simulator itself.

For the case when dealer is *corrupt*, the simulator first plays the role of the honest parties and participates in the execution of Π_{ACSS} with Adv. If during the execution simulator finds Adv broadcasting a *valid* $\mathcal{V}$ set and the corresponding $\mathcal{W}_j$ sets, then based on the shares of the *honest* parties in $\mathcal{V}$ set, the simulator "extracts" the sharing-polynomial of the dealer, which it sends to $\mathcal{F}_{\mathsf{ACSS}}$ in the ideal-world.

Simulator $\mathcal{S}_{\mathsf{ACSS}}$

$\mathcal{S}_{\mathsf{ACSS}}$ constructs virtual real-world honest parties and invokes the real-world adversary Adv. The simulator simulates the view of Adv, namely its communication with $\mathcal{Z}$, the messages sent by the honest parties and the interaction with $\mathcal{F}_{\mathsf{ACast}}$. In order to simulate $\mathcal{Z}$, the simulator $\mathcal{S}_{\mathsf{ACSS}}$ forwards every message it receives from $\mathcal{Z}$ to Adv and vice-versa. The simulator then simulates the various phases of the protocol as follows, depending upon whether the dealer is honest or corrupt.

Simulation when D is Honest

Interaction with $\mathcal{F}_{\mathsf{ACSS}}$: the simulator interacts with the functionality $\mathcal{F}_{\mathsf{ACSS}}$ and receives a request based delayed output $(\mathsf{D}, \mathsf{Dealer}, 1, \mathsf{sid}, \{s_i\}_{P_i \in \mathcal{C}})$ on the behalf of the parties in $\mathcal{C}$. The simulator then plays the role of the honest parties as per the protocol Π_{ACSS} and simulates the interaction between Adv and the honest parties during the various steps of Π_{ACSS} as follows.

Distribution of Column-Polynomials and Sharing of Row-Polynomials by the dealer: The simulator picks a random degree-t univariate polynomial $\widetilde{q}(\cdot)$ on the behalf of D, subject to the condition that $\widetilde{q}(\alpha_i) = s_i$ holds, for every $P_i \in \mathcal{C}$. The simulator then selects a random degree-(t,t) bivariate polynomial $\widetilde{H}(x,y)$ over $\mathbb{F}$, such that $\widetilde{H}(0,y) = \widetilde{q}(\cdot)$ holds. On the behalf of the dealer, the simulator sends $(\mathsf{sid}, \mathsf{column}, \widetilde{c}_i(y))$ to Adv, for every $P_i \in \mathcal{C}$, where $\widetilde{c}_i(y) \overset{\text{def}}{=} \widetilde{H}(\alpha_i, y)$. For $j = 1, \ldots, n$, the simulator invokes an instance $\widetilde{\mathsf{Sh}}_{\mathsf{sid}}^{(j)} \overset{\text{def}}{=} \mathsf{Sh}(\mathsf{sid}, \mathsf{D}, \widetilde{r}_j(x))$, where $\widetilde{r}_j(x) \overset{\text{def}}{=} \widetilde{H}(x, \alpha_j)$ and plays the roles of the dealer in these instances and interacts with Adv as per the steps of the protocol Sh, on the behalf of the dealer.

Pair-wise Consistency Check: The simulator plays the role of the honest parties as per the steps of the protocol Sh in the instances $\widetilde{\mathsf{Sh}}_{\mathsf{sid}}^{(1)}, \ldots, \widetilde{\mathsf{Sh}}_{\mathsf{sid}}^{(n)}$ and interact with Adv in these instances, on the behalf of the honest parties. Whenever Adv requests output from $\mathcal{F}_{\mathsf{ACast}}$ corresponding to any sender $P_i \notin \mathcal{C}$, the simulator sends the output $(P_i, \mathsf{ACast}, \mathsf{sid}, \mathtt{OK}_i)$ to Adv on the behalf of $\mathcal{F}_{\mathsf{ACast}}$.

Construction of $\mathcal{V}$ and Public Announcement: The simulator keeps constructing the set $\mathcal{W}^{(\ell)}$ and the sets $\mathcal{W}_k^{(\ell)}$ for each $P_k \in \mathcal{W}^{(\ell)}$ during the instances $\widetilde{\mathsf{Sh}}_{\mathsf{sid}}^{(\ell)}$, by playing the role of the dealer in these instances as per the steps of the protocol Sh. Once a valid $\mathcal{V}$ set and sets $\{\mathcal{W}_j^{(1)}, \ldots, \mathcal{W}_j^{(n)}\}_{P_j \in \mathcal{V}}$ are ready, then whenever Adv requests output from $\mathcal{F}_{\mathsf{ACast}}$ corresponding to the sender D, the simulator sends the output $(\mathsf{D}, \mathsf{ACast}, \mathsf{sid}, \mathcal{V}, \{\mathcal{W}_j^{(1)}, \ldots, \mathcal{W}_j^{(n)}\}_{P_j \in \mathcal{V}})$ to Adv, on the behalf of $\mathcal{F}_{\mathsf{ACast}}$.

Verification of $\mathcal{V}$: The simulator plays the role of the honest parties, as per the steps of Sh during the instances $\widetilde{\mathsf{Sh}}_{\mathsf{sid}}^{(1)}, \ldots, \widetilde{\mathsf{Sh}}_{\mathsf{sid}}^{(n)}$. Moreover, for $i = 1, \ldots, n$, the simulator starts participating in the instances $\widetilde{\mathsf{RecPriv}}_{\mathsf{sid}}^{(i)} \overset{\text{def}}{=} \mathsf{RecPriv}(\mathsf{D}, \widetilde{r}_i(x))$, by playing the role of the honest parties in these instances, as per the steps of the protocol $\mathsf{RecPriv}$.

Share Computation: For each $P_i \in \mathcal{C}$, the simulator plays the role of the honest parties as per the steps of the protocol $\mathsf{RecPriv}$ during the instance $\widetilde{\mathsf{RecPriv}}_{\mathsf{sid}}^{(i)}$.

<u>Simulation when D is Corrupt</u>

In this case, the simulator $\mathcal{S}_{\mathsf{AMTSS}}$ interacts with Adv during the various steps of Π_{ACSS} as follows.

Distribution of Column-Polynomials and Sharing of Row-Polynomials by the dealer: The simulator plays the role of the honest parties as per the protocol Π_{ACSS} and interacts with Adv on the behalf of the honest parties. If Adv sends $(\mathsf{sid}, \mathsf{column}, c_i(y))$ on the behalf of D for any $P_i \notin \mathcal{C}$, then the simulator records it, provided $c_i(y)$ is a degree-t polynomial.

Pair-wise Consistency Check: If for any $P_i \notin \mathcal{C}$ the simulator has recorded a degree-t column-polynomial on the behalf of P_i, then the simulator plays the role of P_i as per the steps of the protocol Sh in the instances $\mathsf{Sh}_{\mathsf{sid}}^{(1)}, \ldots, \mathsf{Sh}_{\mathsf{sid}}^{(n)}$ and interact with Adv in these instances, on the behalf of P_i. Whenever Adv requests output from $\mathcal{F}_{\mathsf{ACast}}$ corresponding to any sender $P_i \notin \mathcal{C}$, the simulator sends the output $(P_i, \mathsf{ACast}, \mathsf{sid}, \mathtt{OK}_i)$ to Adv on the behalf of $\mathcal{F}_{\mathsf{ACast}}$, provided $r_{ji} = c_i(\alpha_j)$ holds for each $j = 1, \ldots, n$. Here r_{ji} denotes the share which simulator received from Adv on the behalf of the dealer, corresponding to P_i during the instance $\mathsf{Sh}_{\mathsf{sid}}^{(j)}$.

Construction of $\mathcal{V}$ and Public Announcement: The simulator plays the role of the honest parties as per the steps of Sh during the instances $\mathsf{Sh}_{\mathsf{sid}}^{(1)}, \ldots, \mathsf{Sh}_{\mathsf{sid}}^{(n)}$.

Verification of $\mathcal{V}$: If Adv sends $(\mathsf{sender}, \mathsf{ACast}, \mathsf{sid}, \mathcal{V}, \{\mathcal{W}_j^{(1)}, \ldots, \mathcal{W}_j^{(n)}\}_{P_j \in \mathcal{V}})$ to $\mathcal{F}_{\mathsf{ACast}}$, then the simulator plays the roles of the honest parties as per the steps of Sh during the instances $\mathsf{Sh}_{\mathsf{sid}}^{(1)}, \ldots, \mathsf{Sh}_{\mathsf{sid}}^{(n)}$ and verifies if these sets are valid. Accordingly, the simulator interacts with $\mathcal{F}_{\mathsf{ACSS}}$ as follows.

- If the verification fails, then the simulator sends $(\mathsf{Dealer}, \mathsf{ACSS}, 1, \mathsf{sid}, q(\cdot))$ to $\mathcal{F}_{\mathsf{ACSS}}$, where $q(\cdot)$ is a polynomial of degree more than t.
- Else the simulator sends $(\mathsf{Dealer}, \mathsf{ACSS}, 1, \mathsf{sid}, q(\cdot))$ to $\mathcal{F}_{\mathsf{ACSS}}$, where $q(\cdot)$ is the dealer's sharing-polynomial, computed as follows.
  - For $j = 1, \ldots, n$, compute the polynomial $r_j(\cdot)$ by interpolating the points $\{(\alpha_i, r_{ji})\}_{P_i \in \mathcal{H} \cap \mathcal{V}}$. Here $\mathcal{H} \overset{\text{def}}{=} \mathcal{P} \setminus \mathcal{C}$.

– Compute the polynomial $q(\cdot)$ by interpolating the points $\{(\alpha_j, r_j(0))\}_{j \in \{1,\ldots,n\}}$

Share Computation: If the simulator has sent a sharing-polynomial on the behalf of the dealer to $\mathcal{F}_{\mathsf{ACSS}}$, then for every $P_i \in \mathcal{C}$, the simulator plays the role of the honest parties as per the steps of the protocol RecPriv during the instance $\mathsf{RecPriv}^{(i)}_{\mathsf{sid}}$, which is the instance of RecPriv, corresponding to P_i.

Figure 13: Simulator for the protocol Π_{ACSS} where Adv corrupts the parties in set $\mathcal{C}$, where $|\mathcal{C}| \leq t$

We now prove a series of claims, which will help us to finally prove the theorem. In these claims, we skip the session id sid. We first consider the case when the dealer is *honest* and show that the view of Adv is *identically* distributed, both in the real execution of Π_{ACSS} involving real honest parties, as well as in the simulated execution of Π_{ACSS}, where the role of the honest parties is played by the simulator.

Claim 5.2. If the dealer is honest, then the view of Adv in the simulated execution of Π_{ACSS} with $\mathcal{S}_{\mathsf{ACSS}}$ is identically distributed as the view of Adv in the real execution of Π_{ACSS} involving honest parties.

Proof. The view of Adv during the real execution of Π_{ACSS} consists of the following:

– The degree-t row-polynomials $\{H(x, \alpha_i)\}_{P_i \in \mathcal{C}}$ and degree-t column-polynomials $\{H(\alpha_i, y)\}_{P_i \in \mathcal{C}}$.
– The view generated during the instances $\mathsf{Sh}^{(1)}, \ldots, \mathsf{Sh}^{(n)}$.
– The view generated during the instances $\mathsf{RecPriv}^{(1)}, \ldots, \mathsf{RecPriv}^{(n)}$.
– The broadcasted sets $\mathcal{V}$ and $\{\mathcal{W}^{(1)}_j, \ldots, \mathcal{W}^{(n)}_j\}_{P_j \in \mathcal{V}}$.

By Lemma 2.6, the distribution of $\{H(x, \alpha_i), H(\alpha_i, y)\}_{P_i \in \mathcal{C}}$ is independent of the underlying polynomial $H(0, \alpha_y)$, which is the input $q(\cdot)$ of the dealer. Since in both the real as well as simulated execution, the bivariate polynomial is selected uniformly at random, it follows that the distribution of the row and column-polynomials of the parties in $\mathcal{C}$ are identically distributed in both the executions and so we fix these polynomials. Now conditioned on the polynomials $\{H(x, \alpha_i), H(\alpha_i, y)\}_{P_i \in \mathcal{C}}$, the multiple instances of Sh in the simulated execution are executed exactly as in the real execution, on random inputs for *honest* dealer. Moreover, from Lemma 4.6, conditioned on $\{H(x, \alpha_i), H(\alpha_i, y)\}_{P_i \in \mathcal{C}}$, the view of Adv during the instances $\{\mathsf{Sh}^{(i)}\}_{P_i \notin \mathcal{C}}$ remains independent of the underlying row-polynomials $\{H(x, \alpha_i)\}_{P_i \notin \mathcal{C}}$. Hence the view of Adv will have the same distribution in both the executions during the instances $\mathsf{Sh}^{(1)}, \ldots, \mathsf{Sh}^{(n)}$ and so we fix the view of Adv during these instances as well. This automatically fixes the sets $\mathcal{V}$ and $\{\mathcal{W}^{(1)}_j, \ldots, \mathcal{W}^{(n)}_j\}_{P_j \in \mathcal{V}}$. This is because the set $\mathcal{W}^{(\ell)}$ and the sets $\{\mathcal{W}^{(\ell)}\}_{P_j \in \mathcal{P}}$ generated during the instance $\mathsf{Sh}^{(\ell)}$ is part of the view of Adv for the instance $\mathsf{Sh}^{(\ell)}$. Finally, from Lemma 4.8, conditioned on $\{H(x, \alpha_i), H(\alpha_i, y)\}_{P_i \in \mathcal{C}}$, the view generated during the instances $\mathsf{RecPriv}^{(1)}, \ldots, \mathsf{RecPriv}^{(n)}$ are identically distributed in both the executions. $\square$

We next claim that if the dealer is *honest*, then conditioned on the view of the adversary (which is identical in both the executions as per Claim 5.2), the outputs of the honest parties are identically distributed in the real-world as well as the ideal-world, except with a negligible probability.

Claim 5.3. If D is honest, then conditioned on the view of Adv, the output of the honest parties during the execution of Π_{ACSS} involving Adv has the same distribution as the output of the honest parties in the ideal-world involving $\mathcal{S}_{\mathsf{ACSS}}$ and $\mathcal{F}_{\mathsf{ACSS}}$, except with probability $n^3 \cdot \epsilon_{\mathsf{AICP}}$.

Proof. Let D be *honest* and let View be an arbitrary view of Adv. Moreover, let $\{r_i(x), c_i(y)\}_{P_i \in \mathcal{C}}$ be the degree-t row and column-polynomials as per View. Notice that the degree-(t,t) bivariate polynomial $H(x, y)$ selected by D is uniformly distributed, conditioned on the polynomials $\{r_i(x), c_i(y)\}_{P_i \in \mathcal{C}}$ and $q(\cdot) \overset{\text{def}}{=} H(0, y)$ (in fact, if $|\mathcal{C}| = t$, then $H(x, y)$ gets fixed based on $\{c_i(y)\}_{P_i \in \mathcal{C}}$ and $q(\cdot)$). Let us fix any such polynomial $H(x, y)$ and hence the corresponding $q(\cdot)$ polynomial. In the ideal-world, each honest P_i obtain the request-based delayed output $q(\alpha_i)$ from $\mathcal{F}_{\mathsf{ACSS}}$. We show that except with probability $n^3 \cdot \epsilon_{\mathsf{AICP}}$, each honest party P_i eventually outputs $q(\alpha_i)$ in the real-world as well. For this, we define an event E which is the event that during the real execution of Π_{ACSS}, for $\ell = 1, \ldots, n$, all honest parties are eventually included in the set $\mathcal{W}^{(\ell)}$ during the instance $\mathsf{Sh}^{(\ell)}$; moreover for every $P_i \notin \mathcal{C}$, party P_i eventually reconstructs the degree-t row-polynomial $r_i(x)$, during the instance $\mathsf{RecPriv}^{(i)}$. From Lemma 4.2 and Lemma 4.8 (along with the Notation 4.9 and Notation 4.10), event E occurs, except with probability $n^3 \cdot \epsilon_{\mathsf{AICP}}$. We now show that conditioned on the event E, each honest party P_i eventually outputs $q(\alpha_i)$.

Let the event E occur. Since D is *honest*, for *every* party P_i, the condition $c_i(\alpha_j) = r_j(\alpha_i)$ holds, for $j = 1, \ldots, n$. As there are at least $n - t$ *honest* parties P_i, this implies that eventually D finds a common set $\mathcal{V}$ of size $n - t$, such that $\mathcal{V}$ constitutes a valid $\mathcal{W}$ set for all the instances $\mathsf{Sh}^{(1)}, \ldots, \mathsf{Sh}^{(n)}$ and each party P_i in $\mathcal{V}$ has broadcast $\mathtt{OK}_i$ message. Upon the broadcast of $\mathcal{V}$, each honest party eventually validates it and invokes the instances $\mathsf{RecPriv}_1, \ldots, \mathsf{RecPriv}_n$. Consequently, each honest P_i eventually reconstructs the row-polynomial $r_i(x)$ and outputs its share s_i, which is the same as $q(\alpha_i)$. This is because $r_i(x) = H(x, \alpha_i)$ holds and hence s_i which is the same as $r_i(0)$ will be $H(0, \alpha_i)$. Now $H(0, \alpha_i)$ is same as $q(\alpha_i)$ because the bivariate polynomial $H(x, y)$, satisfies the condition that $H(0, y) = q(\cdot)$ holds. □

We next prove certain claims with respect to a *corrupt* dealer. The first obvious claim is that the view of Adv in this case is also identically distributed in both the real execution as well as simulated execution of Π_{ACSS}. This is simply because in this case, the honest parties have no inputs (in the protocol, only dealer has the input and we are analysing the case when dealer is under the control of Adv) and the simulator $\mathcal{S}_{\mathsf{ACSS}}$ plays the role of the honest parties exactly as per the steps of Π_{ACSS} in the ideal-world execution.

Claim 5.4. If the dealer is corrupt, then the the view of Adv in the simulated execution of Π_{ACSS} with $\mathcal{S}_{\mathsf{ACSS}}$ is identically distributed as the view of Adv in the real execution of Π_{ACSS} involving honest parties.

Proof. The proof follows from the fact that if D is *corrupt*, then $\mathcal{S}_{\mathsf{ACSS}}$ participates in a full execution of the protocol Π_{ACSS} by playing the role of the honest parties. Hence, there is a one-to-one correspondence between simulated executions and real executions. □

We next claim that if the dealer is *corrupt*, then conditioned on the view of the adversary (which is identical in both the executions as per Claim 5.4), the outputs of the honest parties are identically distributed in the real-world as well as the ideal-world, except with a negligible probability.

Claim 5.5. If D is corrupt, then conditioned on the view of Adv, the output of the honest parties during the execution of Π_{ACSS} involving Adv has the same distribution as the output of the honest parties in the ideal-world involving $\mathcal{S}_{\mathsf{ACSS}}$ and $\mathcal{F}_{\mathsf{ACSS}}$, except with probability $n^3 \cdot \epsilon_{\mathsf{AICP}}$.

Proof. Let D be corrupt and let View be an arbitrary view of Adv. We note that it can be find out from View whether a valid $\mathcal{V}$ set (and corresponding $\mathcal{W}_j^{(\ell)}$ subsets) have been generated during the corresponding execution of Π_{ACSS}. We now consider the following different cases.

– *No $\mathcal{V}$ set is generated as per* View: It is easy to see that in this case, the outputs of the honest parties are *identically* distributed in both the worlds. This is because in the real-world, the honest parties do not participate in any of the $\mathsf{RecPriv}$ instances and hence honest parties do not obtain any output. Correspondingly, the simulator $\mathcal{S}_{\mathsf{ACSS}}$ does not provide any input to $\mathcal{F}_{\mathsf{ACSS}}$ in the ideal-world on the behalf of the dealer and hence $\mathcal{F}_{\mathsf{ACSS}}$ does not produce any output for the honest parties as well.

– *Sets $\mathcal{V}, \{\mathcal{W}_j^{(1)}, \ldots, \mathcal{W}_j^{(n)}\}_{P_j \in \mathcal{V}}$ are generated as per* View *but are invalid*: It is easy to see that in this case also, the outputs of the honest parties are *identically* distributed in both the worlds. This is because in this case, the honest parties do not participate in any of the $\mathsf{RecPriv}$ instances and instead output $\perp$ as their shares after finding that the sets $\mathcal{V}, \{\mathcal{W}_j^{(1)}, \ldots, \mathcal{W}_j^{(n)}\}_{P_j \in \mathcal{V}}$ are *invalid*. Since $\mathcal{S}_{\mathsf{ACSS}}$ will know these sets as well and finds them to be invalid, in the ideal-world, the simulator $\mathcal{S}_{\mathsf{ACSS}}$ provides a polynomial of degree more than t as the input on the behalf of D to $\mathcal{F}_{\mathsf{ACSS}}$, ensuring that the honest parties also output $\perp$ in the ideal-world.

– *Sets $\mathcal{V}, \{\mathcal{W}_j^{(1)}, \ldots, \mathcal{W}_j^{(n)}\}_{P_j \in \mathcal{V}}$ are generated as per* View *and are valid*: Let $\mathcal{H}$ be the set of *honest* parties in $\mathcal{V}$, that is, $\mathcal{H} \overset{\text{def}}{=} \mathcal{V} \setminus \mathcal{C}$. For $\ell = 1, \ldots, n$, let $r_\ell(x)$ be the polynomial, shared by D among the parties in $\mathcal{H}$ in the instance $\mathsf{Sh}^{(\ell)}$, during the execution of Π_{ACSS}. That is, $r_\ell(x)$ is the polynomial determined by the shares received by the parties in $\mathcal{H}$ during $\mathsf{Sh}^{(\ell)}$. Notice that the polynomials $r_1(x), \ldots, r_n(x)$ are completely determined by View. This is because these polynomials are completely determined by the view of Adv during the instances of Sh executed inside Π_{ACSS}, as these polynomials are shared by the dealer which is under the control of Adv and $\mathcal{S}_{\mathsf{ACSS}}$ plays the role of honest parties during all the Sh instances. We say that View is *good*, if the polynomials $r_1(x), \ldots, r_n(x)$ are degree-t polynomials. From Lemma 4.5 (along with Notation 4.9), it follows that except with probability $n^3 \cdot \epsilon_{\mathsf{AICP}}$, the view View is indeed *good*. We now show that conditioned on the event that View is good, the output of the honest parties are identical in both the worlds.
  Let View be good. It then follows that the polynomials $r_1(x), \ldots, r_n(x)$ lie on a single single degree-(t,t) bivariate polynomial $H(x,y)$. This is because since $\mathcal{V}$ is *valid*, it implies that each $P_i \in \mathcal{H}$ has broadcasted an $\texttt{OK}_i$ message, which further implies that $r_{ji} = c_i(\alpha_j)$ holds for all $j = 1, \ldots, n$. Here $c_i(y)$ is the degree-t column-polynomial received by P_i from D and r_{ji} denotes the share $r_j(\alpha_i)$ received by P_i from D during the instance $\mathsf{Sh}^{(j)}$. Since $|\mathcal{H}| \geq t+1$, it implies that the degree-t row-polynomials $r_1(x), \ldots, r_n(x)$ are pair-wise consistent with $t+1$ degree-t column-polynomials, implying that the polynomials $r_1(x), \ldots, r_n(x)$ lie on a single degree-(t,t) bivariate polynomial, say $H(x,y)$ (follows from Lemma 2.5). From the steps of $\mathcal{S}_{\mathsf{ACSS}}$, it follows that in the ideal-world, the output of the honest parties will the shares $q(\alpha_i)$, where $q(\cdot) \overset{\text{def}}{=} H(0,y)$ holds. In the real-world, the output of each *honest* P_i will be the constant term of the polynomial reconstructed by P_i during the instance $\mathsf{RecPriv}^{(i)}$. From Lemma 4.8 (along with Notation 4.9 and Notation 4.10), it follows that except with probability $n^2 \cdot \epsilon_{\mathsf{AICP}}$, party P_i reconstructs $r_i(x)$ during the instance $\mathsf{RecPriv}^{(i)}$ and hence outputs $r_i(0)$, which is the same as $q(\alpha_i)$.

□

Since $\epsilon_{\mathsf{AICP}} = 2^{-\Omega(\kappa)}$ and $n = \text{poly}(\kappa)$, from Claims 5.2-5.5, we conclude that

$$\left\{\text{HYBRID}^{\mathcal{F}_{\mathsf{ACast}}}_{\Pi_{\mathsf{ACSS}},\mathsf{Adv}(z),\mathcal{C}}(q(\cdot))\right\}_{z\in\{0,1\}^\star} \overset{s}{\equiv} \left\{\text{IDEAL}_{\mathcal{F}_{\mathsf{ACSS}},\mathcal{S}_{\mathsf{ACSS}}(z),\mathcal{C}}(q(\cdot))\right\}_{z\in\{0,1\}^\star}$$

holds, thus proving the theorem. □

5.2 ACSS Protocol for Realizing $\mathcal{F}_{\mathsf{ACSS}}$ with $L > 1$

To generate a complete t-sharing of $S = (s^{(1)}, \ldots, s^{(L)})$, the parties execute the steps of the protocol Π_{ACSS} independently L times with the following modifications: corresponding to each party P_j, the dealer D will now have L number of degree-t row-polynomials to share. Instead of executing L instances of Sh to share these polynomials, D shares all of them *simultaneously* by executing a *single* instance MSh_j of MSh. Similarly, each party P_i broadcasts a single $\texttt{OK}_i$ message, if the conditions for broadcasting the $\texttt{OK}_i$ message is satisfied for P_i in all the L instances. Since most of steps of the protocol are similar to executing L parallel instances of Π_{ACSS} (with above modifications), we do not give the formal details to avoid repetition. The resultant protocol is still called as Π_{ACSS}. The proof of the following theorem follows from the fact that there are n instances of MSh and RecPriv involved, each dealing with L polynomials. The security proof will be similar as in Theorem 5.1 and to avoid repetition, we do not give the formal details.

Theorem 5.6. *Protocol Π_{ACSS} UC-securely realizes the functionality $\mathcal{F}_{\mathsf{ACSS}}$ for any $L \geq 1$ with statistical security in the $\mathcal{F}_{\mathsf{ACast}}$-hybrid model, in the presence of a static malicious adversary, corrupting at most $t < \frac{n}{3}$ parties. The protocol needs a communication of $\mathcal{O}(L \cdot n^3\kappa + n^4\kappa^2 + n^5)$ bits*

6 Protocol for the Pre-Processing Phase

In this section, we present our protocol for the pre-processing phase, which will be later used in our AMPC protocol. The protocol allows the parties to generate complete t-sharing of c_M random multiplication-triples, which are used for securely evaluating the multiplication gates in our AMPC protocol.[11] The protocol realizes the ideal functionality $\mathcal{F}_{\mathsf{APrep}}$ (see Fig 14). The functionality gets activated on being invoked by at least $t+1$ parties (thus guaranteeing that at least one *honest* party invokes it) and generates complete t-sharing of c_M random multiplication-triples. The functionality allows the ideal-world adversary to specify all the "data" that the corrupted parties would like to hold as part of the various sharings generated by the functionality. Namely it specifies the shares for each of the generated t-sharing on the behalf of corrupt parties. The functionality then "completes" the sharings randomly, while keeping them "consistent" with the shares specified by the adversary. Namely, for each shared value, the sharing polynomial is selected randomly, such that when evaluated at the evaluation-points corresponding to the corrupt parties, it produces the shares as specified by the adversary.[12] Once the t-sharings are computed, the functionality performs a request-based delayed delivery of the respective shares of the honest parties (the shares of the corrupt parties need not be delivered, as they are already available with the adversary).

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

$\mathcal{F}_{\mathsf{APrep}}$ proceeds as follows, running with parties $\mathcal{P} = \{P_1, \ldots, P_n\}$ and an adversary $\mathcal{S}$. Let $\mathcal{C}$ denote the set of corrupt parties, where $|\mathcal{C}| \leq t$. If at least $t+1$ parties P_i have sent $(\mathsf{prep}, \mathsf{sid}, P_i)$ messages, then prepare the output as follows.

- Generate a complete t-sharing of c_M number of random multiplication triples. To generate one such sharing, randomly select u, v and compute $w = u \cdot v$ and execute the steps for "Single $[\cdot]_t$-sharing Generation" (see below) for u, v and w.
- Let $\{(u^{(\ell)}, v^{(\ell)}, w^{(\ell)})\}_{\ell \in \{1,\ldots,c_M\}}$ be the random multiplication triples, whose complete t-sharing are generated. Moreover, let $[u^{(\ell)}]_t = (u_1^{(\ell)}, \ldots, u_n^{(\ell)})$, $[v^{(\ell)}]_t = (v_1^{(\ell)}, \ldots, v_n^{(\ell)})$ and $[w^{(\ell)}]_t = (w_1^{(\ell)}, \ldots, w_n^{(\ell)})$ respectively. Send a request-based delayed output $(\mathsf{triple\text{-}shares}, \mathsf{sid}, \{(u_i^{(\ell)}, v_i^{(\ell)}, w_i^{(\ell)})\}_{\ell \in \{1,\ldots,c_M\}})$ to each $P_i \in \mathcal{P} \setminus \mathcal{C}$ (no need to send the shares to the parties in $\mathcal{C}$, as $\mathcal{S}$ already has these shares).

Single $[\cdot]_t$-sharing Generation: The functionality does the following to generate a complete t-sharing of a given value s.

- Upon receiving $(\mathsf{shares}, \mathsf{sid}, \{s_i\}_{P_i \in \mathcal{C}})$ from $\mathcal{S}$, select a degree-t univariate polynomial $S(x)$, which is otherwise a random polynomial except that $S(0) = s$ and $S(\alpha_i) = s_i$ holds for each $P_i \in \mathcal{C}$.
- Compute $s_i \overset{\text{def}}{=} S(\alpha_i)$, for each $P_i \in \mathcal{P} \setminus \mathcal{C}$.

Figure 14: The ideal functionality for asynchronous pre-processing phase.

Before going into the realization of $\mathcal{F}_{\mathsf{APrep}}$, we first discuss a sub-protocol which will be used in our protocol.

6.1 Asynchronous Verifiable Multiplication-Triple Sharing (AMTSS)

We present a protocol Π_{AMTSS}, which allows a designated dealer to *verifiably* generate complete t-sharing of ℓ multiplication-triples available with the dealer. While the shared triples are always multiplication-triples for an *honest* dealer, the protocol allows the parties to publicly verify whether the shared triples are multiplication-triples even for a *corrupt* dealer. The privacy of the triples are maintained during the public verification, if the dealer is *honest*.

[11]Recall that c_M is the number of multiplication gates in the circuit cir, representing the function f to be securely computed.

[12]Looking ahead, in our realization of $\mathcal{F}_{\mathsf{APrep}}$, the real-world adversary will have full control over the shares of the corrupt parties, corresponding to the resultant random multiplication-triples generated in the protocol. To capture it, we make the provision for the adversary to specify the shares of the corrupt parties in $\mathcal{F}_{\mathsf{APrep}}$.

The ideal functionality realized by the protocol Π_{AMTSS} is presented in Fig 15. The functionality is similar to $\mathcal{F}_{\mathsf{ACSS}}$ and generates a complete t-sharing of the inputs provided by a designated *dealer*. The difference is that the shared values constitute multiplication-triples. In a more detail, the functionality waits to receive a bunch of triplets of polynomials from the dealer. Upon receiving, the functionality verifies whether each triplet of polynomial consists of degree-t polynomials and whether their constant terms constitute a multiplication-triple. If the verification is successful, the functionality delivers a request-based delayed output consisting of distinct points on all the polynomials to the respective parties. On the other hand, if the verification fails then the functionality sends an output $\perp$ to each party, indicating that the inputs of the dealer are "invalid". Notice that similar to $\mathcal{F}_{\mathsf{ACSS}}$, the functionality $\mathcal{F}_{\mathsf{AMTSS}}$ generates an output for the parties, *only* upon receiving valid inputs from the dealer.

Looking ahead, in our pre-processing phase protocol, the functionality $\mathcal{F}_{\mathsf{AMTSS}}$ will be used as follows: if a party wants to completely t-share some multiplication-triples, then it acts as a dealer and randomly picks degree-t polynomials whose constant terms constitute the multiplication-triples and then invokes $\mathcal{F}_{\mathsf{AMTSS}}$ with these polynomials. Notice that the inputs of the dealer, namely the constant terms of the polynomial-triplets are well defined. Moreover, the triples will remain random from the point of view of the adversary, if the dealer is *honest*.

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

$\mathcal{F}_{\mathsf{AMTSS}}$ proceeds as follows, running with the parties $\mathcal{P} = \{P_1, \ldots, P_n\}$ and an adversary $\mathcal{S}$.

- Upon receiving $(\mathsf{Dealer}, \mathsf{AMTSS}, \mathsf{sid}, \{(A^{(l)}(\cdot), B^{(l)}(\cdot), C^{(l)}(\cdot))\}_{l \in \{1,\ldots,\ell\}})$ from the dealer $\mathsf{D} \in \mathcal{P}$, verify the following:
  - For $l = 1, \ldots, \ell$, the polynomials $A^{(l)}(\cdot), B^{(l)}(\cdot)$ and $C^{(l)}(\cdot)$ are degree-t polynomials.
  - For $l = 1, \ldots, \ell$, the condition $C^{(l)}(0) \stackrel{?}{=} A^{(l)}(0) \cdot B^{(l)}(0)$ holds.
- If the verification is successful, send a request-based delayed output $(\mathsf{D}, \mathsf{Dealer}, \mathsf{AMTSS}, \mathsf{sid}, \{(A^{(l)}(\alpha_i), B^{(l)}(\alpha_i), C^{(l)}(\alpha_i))\}_{l \in \{1,\ldots,\ell\}})$ to each $P_i \in \mathcal{P}$. Else, send a request-based delayed output $(\mathsf{D}, \mathsf{Dealer}, \mathsf{AMTSS}, \mathsf{sid}, \perp)$ to each $P_i \in \mathcal{P}$.

Figure 15: The ideal functionality for asynchronous verifiable multiplication-triple sharing.

We next present a statistically-secure protocol from [24], which securely realizes the functionality $\mathcal{F}_{\mathsf{AMTSS}}$. We stress that the security of this protocol was not proved as per the UC model. Hence, we first recast it in the UC framework and then prove that the protocol securely realizes the functionality $\mathcal{F}_{\mathsf{AMTSS}}$ in the $\mathcal{F}_{\mathsf{ACSS}}$-hybrid model.

6.1.1 Statistically-Secure Protocol for Realizing $\mathcal{F}_{\mathsf{AMTSS}}$

Protocol Π_{AMTSS} for realizing the functionality $\mathcal{F}_{\mathsf{AMTSS}}$ is presented in Fig 16. The protocol is based on the following idea: the dealer on having the triplets of input polynomials $(A^{(l)}(\cdot), B^{(l)}(\cdot), C^{(l)}(\cdot))$ sends them to $\mathcal{F}_{\mathsf{ACSS}}$ for complete t-sharing. To enable public verification of whether the constant terms $(a^{(l)}, b^{(l)}, c^{(l)})$ of these polynomials constitute multiplication-triples, the dealer additionally picks ℓ random triplets of *auxiliary polynomials* $(X^{(l)}(\cdot), Y^{(l)}(\cdot), Z^{(l)}(\cdot))$, whose constant terms $(x^{(l)}, y^{(l)}, z^{(l)})$ are multiplication-triples and sends the auxiliary polynomials to $\mathcal{F}_{\mathsf{ACSS}}$ for complete t-sharing.

Once the multiplication-triples of the dealer are shared, the parties then call the functionality $\mathcal{F}_{\mathsf{Rand}}$ to generate a random value, say r, and then uses the standard "sacrificing trick" (see, for example [31]), to verify whether the triples $(a^{(l)}, b^{(l)}, c^{(l)})$ are multiplication-triples, by "sacrificing" the triples $(x^{(l)}, y^{(l)}, z^{(l)})$. Namely, the parties publicly verify if $r \cdot [c^{(l)} - a^{(l)} \cdot b^{(l)}] \stackrel{?}{=} [z^{(l)} - x^{(l)} \cdot y^{(l)}]$ holds. The idea here is that if D is *honest*, then the verification will be always successful for *any* r, as the triples shared by an honest D are indeed multiplication-triples. On the other hand, if D is *corrupt* and the shared triples are *not* multiplication-triples, the verification fails with a high probability. This is because in this case, the verification will be successful, only for some specific values of r and a corrupt D will have no information apriori about the random r generated by $\mathcal{F}_{\mathsf{Rand}}$, when D provides its triples for sharing to $\mathcal{F}_{\mathsf{ACSS}}$. If D is *honest*, then throughout the protocol, adversary's view remains independent of the multiplication-triples $(a^{(l)}, b^{(l)}, c^{(l)})$ and hence the parties output the complete t-sharing of these triples, based on the polynomials $(A^{(l)}(\cdot), B^{(l)}(\cdot), C^{(l)}(\cdot))$.

Protocol Π_{AMTSS}

Sharing Multiplication-Triples: If P_i is the dealer, then on input $\{(A^{(l)}(\cdot), B^{(l)}(\cdot), C^{(l)}(\cdot))\}_{l \in \{1,\ldots,\ell\}}$, do the following.

1. Randomly pick degree-t polynomials $\{(X^{(l)}(\cdot), Y^{(l)}(\cdot), Z^{(l)}(\cdot))\}_{l=1,\ldots,\ell}$, such that $Z^{(l)}(0) = X^{(l)}(0) \cdot Y^{(l)}(0)$ holds.
2. Send $(\mathsf{Dealer}, \mathsf{ACSS}, 6\ell, \mathsf{sid}, \{(A^{(l)}(\cdot), B^{(l)}(\cdot), C^{(l)}(\cdot)), (X^{(l)}(\cdot), Y^{(l)}(\cdot), Z^{(l)}(\cdot))\}_{l=1,\ldots,\ell})$ to $\mathcal{F}_{\mathsf{ACSS}}$.

Verifying the Shared Multiplication-Triples:

1. Request output from $\mathcal{F}_{\mathsf{ACSS}}$.
2. If a request-based delayed output $(\mathsf{D}, \mathsf{Dealer}, \mathsf{sid}, \perp)$ is received from $\mathcal{F}_{\mathsf{ACSS}}$, then output $(\mathsf{D}, \mathsf{Dealer}, \mathsf{AMTSS}, \mathsf{sid}, \perp)$. Else, if a request-based delayed output $(\mathsf{D}, \mathsf{Dealer}, \mathsf{sid}, 6\ell, \{(a_i^{(l)}, b_i^{(l)}, c_i^{(l)}), (x_i^{(l)}, y_i^{(l)}, z_i^{(l)})\}_{l=1,\ldots,\ell})$ is received from $\mathcal{F}_{\mathsf{ACSS}}$, send $(\mathsf{Rand}, \mathsf{sid}, P_i)$ to $\mathcal{F}_{\mathsf{Rand}}$.
3. Request output from $\mathcal{F}_{\mathsf{Rand}}$. Upon receiving a request-based delayed output $(\mathsf{Rand}, \mathsf{sid}, R(\cdot))$, compute $\rho_i^{(l)} \overset{\text{def}}{=} r \cdot a_i^{(l)} - x_i^{(l)}$ and $\sigma_i^{(l)} \overset{\text{def}}{=} b_i^{(l)} - y_i^{(l)}$ for $l = 1, \ldots, \ell$, where $r \overset{\text{def}}{=} R(0)$.
4. Send $(\mathsf{sid}, \{\rho_i^{(l)}, \sigma_i^{(l)}\}_{l=1,\ldots,\ell})$ to every $P_j \in \mathcal{P}$. Keep receiving $(\mathsf{sid}, \{\rho_j^{(l)}, \sigma_j^{(l)}\}_{l=1,\ldots,\ell})$ from parties $P_j \in \mathcal{P}$ and executing the steps of OEC, till $\rho^{(l)}$ and $\sigma^{(l)}$ is reconstructed, for $l = 1, \ldots, \ell$.
5. Upon reconstructing $\rho^{(l)}$ and $\sigma^{(l)}$, compute $\tau_i^{(l)} \overset{\text{def}}{=} r \cdot c_i^{(l)} - z_i^{(l)} - \sigma^{(l)} \cdot x_i^{(l)} - \rho^{(l)} \cdot y_i^{(l)} - \sigma^{(l)} \cdot \rho^{(l)}$, for $l = 1, \ldots, \ell$.
6. Send $(\mathsf{sid}, \{\tau_i^{(l)}\}_{l=1,\ldots,\ell})$ to every $P_j \in \mathcal{P}$. Keep receiving $(\mathsf{sid}, \{\tau_j^{(l)}\}_{l=1,\ldots,\ell})$ from parties $P_j \in \mathcal{P}$ and executing the steps of OEC, till $\tau^{(l)}$ is reconstructed, for $l = 1, \ldots, \ell$.
7. If $\tau^{(l)} = 0$ holds for $l = 1, \ldots, \ell$, output $(\mathsf{D}, \mathsf{Dealer}, \mathsf{AMTSS}, \mathsf{sid}, \{(a_i^{(l)}, b_i^{(l)}, c_i^{(l)})\}_{l \in \{1,\ldots,\ell\}})$. Else output $(\mathsf{D}, \mathsf{Dealer}, \mathsf{AMTSS}, \mathsf{sid}, \perp)$.

Figure 16: Statistically-Secure Protocol for realizing $\mathcal{F}_{\mathsf{AMTSS}}$ in the $(\mathcal{F}_{\mathsf{ACSS}}, \mathcal{F}_{\mathsf{Rand}})$-hybrid model. The above code is executed by every $P_i \in \mathcal{P}$. In the protocol, $\mathsf{D} \in \mathcal{P}$ is the designated dealer.

We next prove the security of the protocol Π_{AMTSS} in the $(\mathcal{F}_{\mathsf{ACSS}}, \mathcal{F}_{\mathsf{Rand}})$-hybrid model.

Theorem 6.1. *Protocol Π_{AMTSS} UC-securely realizes the functionality $\mathcal{F}_{\mathsf{AMTSS}}$ with statistical security in the $(\mathcal{F}_{\mathsf{ACSS}}, \mathcal{F}_{\mathsf{Rand}})$-hybrid model, in the presence of a static malicious adversary, corrupting at most $t < \frac{n}{3}$ parties. The protocol needs a communication of $\mathcal{O}(\ell \cdot n^2 \kappa)$ bits over the point-to-point channels.*

Proof. The communication complexity simply follows from the fact that in the protocol, the parties publicly reconstruct 3ℓ completely t-shared values and reconstructing one such value needs a communication of $\mathcal{O}(n^2)$ field elements.

For security, let Adv be an arbitrary real-world adversary, attacking protocol Π_{AMTSS} and let $\mathcal{Z}$ be an arbitrary environment. We show the existence of a simulator $\mathcal{S}_{\mathsf{AMTSS}}$, such that for any set of corrupted parties $\mathcal{C}$ with $|\mathcal{C}| \leq t$ and for all inputs, the output of all parties and the adversary in an execution of the real protocol Π_{AMTSS} with Adv is identically distributed as the outputs in an execution with $\mathcal{S}_{\mathsf{AMTSS}}$ involving $\mathcal{F}_{\mathsf{AMTSS}}$ in the ideal model, except with a negligible probability. This further implies that from the viewpoint of $\mathcal{Z}$, the executions in the real-world and the ideal-world are statistically indistinguishable. The steps of the simulator (formally given in Fig 17) will be different, depending upon whether the dealer D is honest or corrupt.

The high level idea of the simulator is as follows. If the dealer is *honest*, then the simulator interacts with the functionality $\mathcal{F}_{\mathsf{AMTSS}}$ and receives the shares of the corrupt parties, corresponding to the dealer's polynomials. In this case, the constant terms of dealer's polynomials (which are degree-t polynomials) will constitute multiplication-triples. The simulator then picks arbitrary degree-t polynomials whose constant terms are multiplication-triples, consistent with the shares of the corrupt parties, received from $\mathcal{F}_{\mathsf{AMTSS}}$. Additionally, the simulator picks random degree-t auxiliary polynomials, whose constant terms are also multiplication-triples. After picking the polynomials, the simulator then plays the role of the dealer with these polynomials and interacts with Adv on the behalf of the honest parties. The simulator also plays the role of $\mathcal{F}_{\mathsf{Rand}}$ and generates the random value, used for verifying the dealer's polynomials.

For the case when dealer is *corrupt*, the simulator first plays the role of $\mathcal{F}_{\mathsf{ACSS}}$ and "extracts" the polynomials used by the dealer. If the dealer provides no polynomials for $\mathcal{F}_{\mathsf{ACSS}}$, then the simulator provides no inputs to $\mathcal{F}_{\mathsf{AMTSS}}$ as well. Else the simulator learns the polynomials provided by the dealer to $\mathcal{F}_{\mathsf{ACSS}}$. If the polynomials are of degree-t with constant terms being multiplication-triples, then the simulator forwards the input polynomials of the dealer to $\mathcal{F}_{\mathsf{AMTSS}}$. Otherwise, the simulator forwards "invalid" polynomials as input polynomials of the dealer to $\mathcal{F}_{\mathsf{AMTSS}}$, causing the honest parties to output $\perp$ in the ideal world.

Simulator $\mathcal{S}_{\mathsf{AMTSS}}$

$\mathcal{S}_{\mathsf{AMTSS}}$ constructs virtual real-world honest parties and invokes the real-world adversary Adv. The simulator simulates the view of Adv, namely its communication with $\mathcal{Z}$, the messages sent by the honest parties and the interaction with various functionalities. In order to simulate $\mathcal{Z}$, the simulator $\mathcal{S}_{\mathsf{AMTSS}}$ forwards every message it receives from $\mathcal{Z}$ to Adv and vice-versa. The simulator then simulates the various phases of the protocol as follows, depending upon whether the dealer D is honest or corrupt.

Steps of the simulator when D is Honest

- **Interaction with $\mathcal{F}_{\mathsf{AMTSS}}$**: The simulator interacts with the functionality $\mathcal{F}_{\mathsf{AMTSS}}$ and receives the request based delayed output $(\mathsf{D}, \mathsf{Dealer}, \mathsf{AMTSS}, \mathsf{sid}, \{(a_i^{(l)}, b_i^{(l)}, c_i^{(l)})\}_{l \in \{1,\ldots,\ell\}, P_i \in \mathcal{C}})$ on the behalf of the parties in $\mathcal{C}$. The simulator then plays the role of the honest parties as per the protocol Π_{AMTSS} and simulates the interaction between Adv and the honest parties during the various steps of Π_{AMTSS} as follows.
- **Sharing Multiplication-Triples**: $\mathcal{S}_{\mathsf{AMTSS}}$ randomly picks polynomials $\{(\widetilde{A}^{(l)}(\cdot), \widetilde{B}^{(l)}(\cdot), \widetilde{C}^{(l)}(\cdot))\}_{l \in \{1,\ldots,\ell\}}$ and $\{(\widetilde{X}^{(l)}(\cdot), \widetilde{Y}^{(l)}(\cdot), \widetilde{Z}^{(l)}(\cdot))\}_{l \in \{1,\ldots,\ell\}}$, each of degree-$t$, satisfying the following requirements.
  - $\widetilde{A}^{(l)}(0) = \widetilde{a}^{(l)}$, $\widetilde{B}^{(l)}(0) = \widetilde{b}^{(l)}$, $\widetilde{C}^{(l)}(0) = \widetilde{c}^{(l)}$, $\widetilde{X}^{(l)}(0) = \widetilde{x}^{(l)}$, $\widetilde{Y}^{(l)}(0) = \widetilde{y}^{(l)}$ and $\widetilde{Z}^{(l)}(0) = \widetilde{z}^{(l)}$ holds, for $l = 1, \ldots, \ell$, where $(\widetilde{a}^{(l)}, \widetilde{b}^{(l)}, \widetilde{c}^{(l)})$ and $(\widetilde{x}^{(l)}, \widetilde{y}^{(l)}, \widetilde{z}^{(l)})$ are random multiplication-triples.
  - $\widetilde{A}^{(l)}(\alpha_i) = a_i^{(l)}$, $\widetilde{B}^{(l)}(\alpha_i) = b_i^{(l)}$, $\widetilde{C}^{(l)}(\alpha_i) = c_i^{(l)}$, $\widetilde{X}^{(l)}(\alpha_i) = x_i^{(l)}$, $\widetilde{Y}^{(l)}(\alpha_i) = y_i^{(l)}$ and $\widetilde{Z}^{(l)}(\alpha_i) = z_i^{(l)}$ holds, for each $P_i \in \mathcal{C}$.

  The simulator plays the role of the honest parties as per Π_{AMTSS}, assuming the polynomials $\{(\widetilde{A}^{(l)}, \widetilde{B}^{(l)}(\cdot), \widetilde{C}^{(l)}(\cdot)), (\widetilde{X}^{(l)}, \widetilde{Y}^{(l)}(\cdot), \widetilde{Z}^{(l)}(\cdot))\}_{l \in \{1,\ldots,\ell\}}$ as the input-polynomials on the behalf of D. In response to the call to $\mathcal{F}_{\mathsf{ACSS}}$, the simulator sends $(\mathsf{D}, \mathsf{Dealer}, 6\ell, \mathsf{sid}, \{(a_i^{(l)}, b_i^{(l)}, c_i^{(l)}), (x_i^{(l)}, y_i^{(l)}, z_i^{(l)})\}_{l \in \{1,\ldots,\ell\}, P_i \in \mathcal{C}})$ to Adv, as the output for the parties in $\mathcal{C}$ from $\mathcal{F}_{\mathsf{ACSS}}$.
- **Verifying the Shared Multiplication-Triples**:
  - The simulator plays the role of $\mathcal{F}_{\mathsf{Rand}}$, as per $\mathcal{F}_{\mathsf{Rand}}$. Namely, upon receiving $(\mathsf{Rand}, \mathsf{sid}, \{r_i\}_{P_i \in \mathcal{C}})$ from Adv on the behalf of the parties in $\mathcal{C}$, the simulator randomly picks a degree-t polynomial $\widetilde{R}(\cdot)$, such that $\widetilde{R}(\alpha_i) = r_i$ holds for each $P_i \in \mathcal{C}$. The simulator then sends the output $(\mathsf{Rand}, \mathsf{sid}, \widetilde{R}(\cdot))$ on the behalf of $\mathcal{F}_{\mathsf{Rand}}$ to Adv.
  - Let $\widetilde{r} \overset{\text{def}}{=} \widetilde{R}(0)$. On the behalf of every $P_i \notin \mathcal{C}$, the simulator computes $\widetilde{\rho}_i^{(l)} \overset{\text{def}}{=} \widetilde{r} \cdot \widetilde{A}^{(l)}(\alpha_i) - \widetilde{X}^{(l)}(\alpha_i)$ and $\widetilde{\sigma}_i^{(l)} \overset{\text{def}}{=} \widetilde{B}^{(l)}(\alpha_i) - \widetilde{Y}^{(l)}(\alpha_i)$ for $l = 1, \ldots, \ell$. It then sends $(\mathsf{sid}, \{\widetilde{\rho}_i^{(l)}, \widetilde{\sigma}_i^{(l)}\}_{l \in \{1,\ldots,\ell\}})$ to Adv, on the behalf of every $P_i \notin \mathcal{C}$.
  - The simulator computes $\widetilde{\rho}^{(l)} \overset{\text{def}}{=} \widetilde{r} \cdot \widetilde{A}^{(l)}(0) - \widetilde{X}^{(l)}(0)$ and $\widetilde{\sigma}_i^{(l)} \overset{\text{def}}{=} \widetilde{B}^{(l)}(0) - \widetilde{Y}^{(l)}(0)$ for $l = 1, \ldots, \ell$. On the behalf of each $P_i \notin \mathcal{C}$, the simulator computes $\widetilde{\tau}_i^{(l)} \overset{\text{def}}{=} \widetilde{r} \cdot \widetilde{C}^{(l)}(\alpha_i) - \widetilde{Z}^{(l)}(\alpha_i) - \widetilde{\sigma}^{(l)} \cdot \widetilde{X}^{(l)}(\alpha_i) - \widetilde{\rho}^{(l)} \cdot \widetilde{Y}^{(l)}(\alpha_i) - \widetilde{\sigma}^{(l)} \cdot \widetilde{\rho}^{(l)}$, for $l = 1, \ldots, \ell$. It then sends $(\mathsf{sid}, \{\widetilde{\tau}_i^{(l)}\}_{l \in \{1,\ldots,\ell\}})$ to Adv, on the behalf of

every $P_i \notin \mathcal{C}$.

Steps of the simulator when D is Corrupt

In this case, the simulator $\mathcal{S}_{\mathsf{AMTSS}}$ interacts with Adv as follows.

- **Sharing Multiplication-Triples**: The simulator plays the role of $\mathcal{F}_{\mathsf{ACSS}}$. Namely, upon receiving $(\mathsf{Dealer}, \mathsf{ACSS}, 6\ell, \mathsf{sid}, \{(A^{(l)}(\cdot), B^{(l)}(\cdot), C^{(l)}(\cdot)), (X^{(l)}(\cdot), Y^{(l)}(\cdot), Z^{(l)}(\cdot))\}_{l \in \{1,\ldots,\ell\}})$ as the message for $\mathcal{F}_{\mathsf{ACSS}}$ from Adv on the behalf of the dealer D, the simulator does the following.
  - If any of the polynomials $\{(A^{(l)}(\cdot), B^{(l)}(\cdot), C^{(l)}(\cdot)), (X^{(l)}(\cdot), Y^{(l)}(\cdot), Z^{(l)}(\cdot))\}_{\{l \in 1,\ldots,\ell\}}$ is not a degree-t polynomial, then it directly goes to the step "**Interaction with $\mathcal{F}_{\mathsf{AMTSS}}$**".
  - Else, the simulator learns the complete t-sharing of the triples $\{(A^{(l)}(0), B^{(l)}(0), C^{(l)}(0)), (X^{(l)}(0), Y^{(l)}(0), Z^{(l)}(0))\}_{\{l \in 1,\ldots,\ell\}}$.
- **Verifying the Shared Multiplication-Triples**: If the simulator has learnt the dealer's polynomials for sharing dealer's triples, then the simulator interacts with Adv on the behalf of the honest parties as per the steps of the protocol Π_{AMTSS}, using the shares of the honest parties, corresponding to the triples as follows.
  - The simulator plays the role of $\mathcal{F}_{\mathsf{Rand}}$ and simulates the interaction between Adv and $\mathcal{F}_{\mathsf{Rand}}$. Namely, upon receiving $(\mathsf{Rand}, \mathsf{sid}, \{r_i\}_{P_i \in \mathcal{C}})$ from Adv on the behalf of the parties in $\mathcal{C}$, the simulator randomly picks a degree-t polynomial $\widetilde{R}(\cdot)$, such that $\widetilde{R}(\alpha_i) = r_i$ holds for each $P_i \in \mathcal{C}$. The simulator then sends the output $(\mathsf{Rand}, \mathsf{sid}, \widetilde{R}(\cdot))$ on the behalf of $\mathcal{F}_{\mathsf{Rand}}$ to Adv.
  - The simulator computes the shares of $\rho^{(l)}, \sigma^{(l)}$ and $\tau^{(l)}$ with respect to $\widetilde{r} \overset{\text{def}}{=} \widetilde{R}(0)$, corresponding to the honest parties and sends them to Adv on the behalf of the honest parties.
- **Interaction with $\mathcal{F}_{\mathsf{AMTSS}}$**: If simulator has learnt the dealer's polynomials $\{(A^{(l)}(\cdot), B^{(l)}(\cdot), C^{(l)}(\cdot)), (X^{(l)}(\cdot), Y^{(l)}(\cdot), Z^{(l)}(\cdot))\}_{\{l \in 1,\ldots,\ell\}}$, then it interacts with $\mathcal{F}_{\mathsf{AMTSS}}$ on the behalf of the dealer D as follows.
  - If for $l = 1, \ldots, \ell$, the polynomials $A^{(l)}(\cdot), B^{(l)}(\cdot), C^{(l)}(\cdot), X^{(l)}(\cdot), Y^{(l)}(\cdot)$ and $Z^{(l)}(\cdot)$ are degree-t polynomials and if $(C^{(l)}(0) = A^{(l)}(0) \cdot B^{(l)}(0))$ as well as $(Z^{(l)}(0) = X^{(l)}(0) \cdot Y^{(l)}(0))$ holds, then the simulator sends $(\mathsf{Dealer}, \mathsf{AMTSS}, \mathsf{sid}, \{(A^{(l)}(\cdot), B^{(l)}(\cdot), C^{(l)}(\cdot))\}_{l \in \{1,\ldots,\ell\}})$ to $\mathcal{F}_{\mathsf{AMTSS}}$ on the behalf of the dealer D.
  - Else, the simulator assigns some arbitrary polynomials of degree more than t to $A^{(l)}(\cdot), B^{(l)}(\cdot), C^{(l)}(\cdot)$ and sends $(\mathsf{Dealer}, \mathsf{AMTSS}, \mathsf{sid}, \{(A^{(l)}(\cdot), B^{(l)}(\cdot), C^{(l)}(\cdot))\}_{l \in \{1,\ldots,\ell\}})$ to $\mathcal{F}_{\mathsf{AMTSS}}$ on the behalf of the dealer D.

Figure 17: Simulator for the protocol Π_{AMTSS} where Adv corrupts the parties in set $\mathcal{C}$, where $|\mathcal{C}| \leq t$

We now prove a series of claims, which will help us to finally prove the theorem. In these claims, we skip the session id sid. We first consider the case when the dealer is *honest* and show that the view of Adv is *identically* distributed, both in the real execution of Π_{ACSS} involving real honest parties, as well as in the simulated execution of Π_{ACSS}, where the role of the honest parties is played by the simulator. Intuitively, this is because in the protocol, the polynomials used by the dealer are degree-t polynomials and adversary learns at most t shares of the polynomials.

Claim 6.2. If the dealer is honest, then the view of Adv in the simulated execution with $\mathcal{S}_{\mathsf{AMTSS}}$ is identically distributed as the view of Adv in the real execution of Π_{AMTSS}.

Proof. Let the dealer be *honest*. From the steps of $\mathcal{S}_{\mathsf{AMTSS}}$, the difference between the real execution and simulated execution is the following: in the real execution, the shares $\{(a_i^{(l)}, b_i^{(l)}, c_i^{(l)})\}_{l \in \{1,\ldots,\ell\}, P_i \in \mathcal{C}}$ and $\{(x_i^{(l)}, y_i^{(l)}, z_i^{(l)})\}_{l \in \{1,\ldots,\ell\}, P_i \in \mathcal{C}}$ correspond to degree-t polynomials $\{(A^{(l)}(\cdot), B^{(l)}(\cdot), C^{(l)}(\cdot))\}_{l \in \{1,\ldots,\ell\}}$ and $\{(X^{(l)}(\cdot), Y^{(l)}(\cdot), Z^{(l)}(\cdot))\}_{l \in \{1,\ldots,\ell\}}$ respectively. On the other hand, in the simulated execution, the shares correspond to degree-t polynomials $\{(\widetilde{A}^{(l)}(\cdot), \widetilde{B}^{(l)}(\cdot), \widetilde{C}^{(l)}(\cdot))\}_{l \in \{1,\ldots,\ell\}}$ and $\{(\widetilde{X}^{(l)}(\cdot), \widetilde{Y}^{(l)}(\cdot), \widetilde{Z}^{(l)}(\cdot))\}_{l \in \{1,\ldots,\ell\}}$ respectively. Apart from being degree-t polynomials, the constant terms of the triplet of polynomials $(A^{(l)}(\cdot), B^{(l)}(\cdot), C^{(l)}(\cdot))$, $(X^{(l)}(\cdot), Y^{(l)}(\cdot), Z^{(l)}(\cdot))$, $(\widetilde{A}^{(l)}(\cdot), \widetilde{B}^{(l)}(\cdot), \widetilde{C}^{(l)}(\cdot))$ and $(\widetilde{X}^{(l)}(\cdot), \widetilde{Y}^{(l)}(\cdot), \widetilde{Z}^{(l)}(\cdot))$ constitute multiplication-triples. Since $|\mathcal{C}| \leq t$, it follows from the properties of degree-t univariate polynomials (see Lemma 2.4) that the shares $\{(a_i^{(l)}, b_i^{(l)}, c_i^{(l)})\}_{l \in \{1,\ldots,\ell\}, P_i \in \mathcal{C}}$ and $\{(x_i^{(l)}, y_i^{(l)}, z_i^{(l)})\}_{l \in \{1,\ldots,\ell\}, P_i \in \mathcal{C}}$

of the multiplication-triples seen by Adv in the real execution and in the simulated execution are identically distributed. Let us fix these shares.

Now conditioned on the shares $\{(a_i^{(l)}, b_i^{(l)}, c_i^{(l)})\}_{l \in \{1,\ldots,\ell\}, P_i \in \mathcal{C}}$ and $\{(x_i^{(l)}, y_i^{(l)}, z_i^{(l)})\}_{l \in \{1,\ldots,\ell\}, P_i \in \mathcal{C}}$, it is easy to see that the view of Adv is identically distributed in both the executions for the rest of the steps of the protocol Π_{AMTSS}. This is because the simulator plays the role of the honest parties as per the protocol and also the role of $\mathcal{F}_{\mathsf{Rand}}$. Namely, in both the executions, Adv learns a complete t-sharing of a random value as the output from $\mathcal{F}_{\mathsf{Rand}}$. In the real execution, Adv sees the shares of $\rho^{(l)}$ and $\sigma^{(l)}$ values from the honest parties, while in the simulated execution, it sees the shares of corresponding $\widetilde{\rho}^{(l)}$ and $\widetilde{\sigma}^{(l)}$ values. Finally, in the real execution, Adv sees shares of $\tau^{(l)}$ values from the honest parties where $\tau^{(l)} = 0$, while in the simulated execution it sees shares of corresponding $\widetilde{\tau}^{(l)}$ values where $\widetilde{\tau}^{(l)} = 0$. □

We next claim that if the dealer is *honest*, then conditioned on the view of the adversary, the outputs of the honest parties are identically distributed in the real-world as well as the ideal-world. Intuitively this is because the input polynomials of the dealer in both the worlds are "valid" (namely degree-t with constant terms being multiplication-triples). And in the protocol, the auxiliary polynomials selected by the dealer also have the same properties. So the verification of the dealer's polynomials will always be successful.

Claim 6.3. If D is honest, then conditioned on the view of Adv, the output of the honest parties during the execution of Π_{AMTSS} involving Adv has the same distribution as the output of the honest parties in the ideal-world involving $\mathcal{S}_{\mathsf{AMTSS}}$ and $\mathcal{F}_{\mathsf{AMTSS}}$.

Proof. Consider an arbitrary view View of Adv, and according to View, let $\{(a_i^{(l)}, b_i^{(l)}, c_i^{(l)})\}_{l \in \{1,\ldots,\ell\}, P_i \in \mathcal{C}}$ and $\{(x_i^{(l)}, y_i^{(l)}, z_i^{(l)})\}_{l \in \{1,\ldots,\ell\}, P_i \in \mathcal{C}}$ be the shares received by Adv from $\mathcal{F}_{\mathsf{ACSS}}$. The degree-$t$ triple-sharing polynomials $(A^{(l)}(\cdot), B^{(l)}(\cdot), C^{(l)}(\cdot))$ of the *honest* dealer are identically distributed (both in the real-world as well as ideal-world) conditioned on the shares $\{(a_i^{(l)}, b_i^{(l)}, c_i^{(l)})\}_{l \in \{1,\ldots,\ell\}, P_i \in \mathcal{C}}$ and the multiplication-triple $(A^{(l)}(0), B^{(l)}(0), C^{(l)}(0))$. Let us fix any such triplet of polynomials $(A^{(l)}(\cdot), B^{(l)}(\cdot), C^{(l)}(\cdot))$. It is easy to see that in the ideal-world, each *honest* P_i outputs $(A^{(l)}(\alpha_i), B^{(l)}(\alpha_i), C^{(l)}(\alpha_i))$. We next show that the same holds even in the real-world as well. For this, it is enough to show that every honest party obtains $\tau^{(l)}$, where $\tau^{(l)} = 0$ holds. This will ensure that the honest parties will output the shares corresponding to the polynomials $A^{(l)}(\cdot), B^{(l)}(\cdot)$ and $C^{(l)}(\cdot)$ as provided by $\mathcal{F}_{\mathsf{ACSS}}$.

The fact that $\tau^{(l)} = 0$ holds for all $l \in \{1, \ldots, \ell\}$ simply follows from the fact that for every l, the triples $(A^{(l)}(0), B^{(l)}(0), C^{(l)}(0))$, as well as the auxiliary triples $(X^{(l)}(0), Y^{(l)}(0), Z^{(l)}(0))$ are multiplication-triples. Hence irrespective of the value of $r \overset{\text{def}}{=} R(0)$ obtained from $\mathcal{F}_{\mathsf{Rand}}$, the value $r \cdot [C^{(l)}(0) - A^{(l)}(0) \cdot B^{(l)}(0)] - [Z^{(l)}(0) - X^{(l)}(0) \cdot Y^{(l)}(0)]$, which is the same as $\tau^{(l)}$, will be 0. In the protocol, the value $\tau^{(l)}$ will be completely t-shared and since it is reconstructed by executing OEC, it follows that Adv cannot enforce any honest party to obtain an incorrect $\tau^{(l)}$ by providing incorrect shares. □

We next prove certain claims with respect to a *corrupt* dealer. The first obvious claim is that the view of Adv in this case also is identically distributed in both the real execution as well as simulated execution of Π_{AMTSS}. This is simply because in this case, the honest parties have no inputs (in the protocol, only dealer has the input and we are analysing the case when dealer is under the control of Adv) and the simulator $\mathcal{S}_{\mathsf{AMTSS}}$ plays the role of the honest parties exactly as per the steps of Π_{AMTSS} in the ideal-world execution.

Claim 6.4. If the dealer is corrupt, then the the view of Adv in the simulated execution of Π_{AMTSS} with $\mathcal{S}_{\mathsf{AMTSS}}$ is identically distributed as the view of Adv in the real execution of Π_{AMTSS} involving honest parties.

Proof. The proof follows from the fact that if D is *corrupt*, then $\mathcal{S}_{\mathsf{AMTSS}}$ participates in a full execution of the protocol Π_{AMTSS} by playing the role of the honest parties. Moreover, $\mathcal{S}_{\mathsf{AMTSS}}$ plays the role of $\mathcal{F}_{\mathsf{Rand}}$.

Hence, there is a one-to-one correspondence between simulated executions and real executions of Π_{AMTSS}, with respect to the view of Adv. □

We next claim that if the dealer is *corrupt*, then conditioned on the view of the adversary, the outputs of the honest parties are identically distributed in the real world as well as the ideal world, except with a negligible probability.

Claim 6.5. If D is corrupt, then conditioned on the view of Adv, the output of the honest parties during the execution of Π_{AMTSS} involving Adv has the same distribution as the output of the honest parties in the ideal-world involving $\mathcal{S}_{\mathsf{AMTSS}}$ and $\mathcal{F}_{\mathsf{AMTSS}}$, except with a negligible probability.

Proof. Let View be an arbitrary view of Adv generated during an execution of Π_{AMTSS}. Since D is under the control of Adv, View completely determines whether Adv provided any polynomials to $\mathcal{F}_{\mathsf{ACSS}}$ for sharing. Based on this, we consider the following possible cases.

– D *does not provide any polynomial for sharing to* $\mathcal{F}_{\mathsf{ACSS}}$: In this case, the honest parties do not output anything in the real-world. It is easy to see that even in the ideal-world, the honest parties do not output anything, as $\mathcal{S}_{\mathsf{AMTSS}}$ does not provide any input to $\mathcal{F}_{\mathsf{AMTSS}}$ on the behalf of D. Hence, the outputs of the honest parties are identically distributed.

– *At least one of the polynomials provided by* D *for sharing to* $\mathcal{F}_{\mathsf{AMTSS}}$ *has degree more than* t: In this case, the honest parties obtain an output $\perp$ in the real-world. It is easy to see that even in the ideal-world, the honest parties output $\perp$, as $\mathcal{S}_{\mathsf{AMTSS}}$ provides polynomials of degree more than t as inputs to $\mathcal{F}_{\mathsf{AMTSS}}$ on the behalf of D. Hence, the outputs of the honest parties are identically distributed.

– D *provides triplets of degree-t polynomials for sharing to* $\mathcal{F}_{\mathsf{ACSS}}$*, whose constant terms are multiplication-triples*: This case is identical to the case when D is *honest* and in this case, the outputs of the honest parties are identically distributed in both the worlds. Namely, the honest parties output shares corresponding to the multiplication-triple sharing polynomials of D.

– D *provides triplets of degree-t polynomials for sharing to* $\mathcal{F}_{\mathsf{ACSS}}$ *and there exists some* $l \in \{1, \ldots, \ell\}$*, where either* $(A^{(l)}(0), B^{(l)}(0), C^{(l)}(0))$ *or* $(X^{(l)}(0), Y^{(l)}(0), Z^{(l)}(0))$ *is not a multiplication-triple*: Let for some $l \in \{1, \ldots, \ell\}$, either $(A^{(l)}(0), B^{(l)}(0), C^{(l)}(0))$ or $(X^{(l)}(0), Y^{(l)}(0), Z^{(l)}(0))$ is not a multiplication-triple. Then there are two further sub-cases.

  – If *both* the triples $(A^{(l)}(0), B^{(l)}(0), C^{(l)}(0))$ and $(X^{(l)}(0), Y^{(l)}(0), Z^{(l)}(0))$ are not multiplication-triples, then in the ideal-world, the honest parties output $\perp$. This is because $\mathcal{S}_{\mathsf{AMTSS}}$ provides polynomials of degree more than t as inputs to $\mathcal{F}_{\mathsf{AMTSS}}$ on the behalf of D. We now show that with a high probability, the honest parties output $\perp$ even in the real-world. Let r be the random value, used in the protocol for verifying the triples of the dealer. Notice that r will not be known to the dealer when it selects its polynomials for $\mathcal{F}_{\mathsf{ACSS}}$. This is because the value of r is decided by $\mathcal{F}_{\mathsf{Rand}}$, which is invoked only after honest parties receive valid shares as outputs from $\mathcal{F}_{\mathsf{ACSS}}$. Now consider the value $r \cdot [C^{(l)}(0) - A^{(l)}(0) \cdot B^{(l)}(0)] - [Z^{(l)}(0) - X^{(l)}(0) \cdot Y^{(l)}(0)]$, which is the same as $\tau^{(l)}$. Moreover, since $\tau^{(l)}$ is completely t-shared and reconstructed using OEC, it follows that irrespective of the behaviour of the corrupt parties, the honest parties eventually reconstruct the correct $\tau^{(l)}$. Since $C^{(l)}(0) \neq A^{(l)}(0) \cdot B^{(l)}(0)$ as well as $Z^{(l)}(0) \neq X^{(l)}(0) \cdot Y^{(l)}(0)$, it follows that $\tau^{(l)}$ will be 0, provided if $r = [Z^{(l)}(0) - X^{(l)}(0) \cdot Y^{(l)}(0)] \cdot [C^{(l)}(0) - A^{(l)}(0) \cdot B^{(l)}(0)]^{-1}$ holds, which can happen only with probability $\frac{1}{|\mathbb{F}|} = 2^{-\Omega(\kappa)}$. Since there are ℓ possible values for l, it follows from the union bound that except with probability at most $\frac{\ell}{|\mathbb{F}|} \equiv 2^{-\Omega(\kappa)}$, the honest parties would reconstruct $\tau^{(l)} \neq 0$ and output $\perp$ in the real-world.

  – If *exactly one* of the triples $(A^{(l)}(0), B^{(l)}(0), C^{(l)}(0))$ or $(X^{(l)}(0), Y^{(l)}(0), Z^{(l)}(0))$ is not a multiplication triple, then the output of the honest parties will be identically distributed in both the worlds, except with probability $\frac{1}{|\mathbb{F}|} = 2^{-\Omega(\kappa)}$. Namely, in the ideal-world, the honest parties output $\perp$, as $\mathcal{S}_{\mathsf{AMTSS}}$ provides polynomials of degree more than t as inputs to $\mathcal{F}_{\mathsf{AMTSS}}$ on the behalf of D. If the triple $(A^{(l)}(0), B^{(l)}(0), C^{(l)}(0))$ is a multiplication-triple, then clearly $\tau^{(l)}$ will be non-zero. Moreover, since $\tau^{(l)}$ is completely t-shared and reconstructed using OEC, it follows that irrespective of the behaviour of the corrupt parties, the honest parties eventually reconstruct the correct $\tau^{(l)}$ and hence the honest parties output $\perp$ even in the real-world. On the other hand, if $(A^{(l)}(0), B^{(l)}(0), C^{(l)}(0))$ is not a multiplication-triple but $(X^{(l)}(0), Y^{(l)}(0), Z^{(l)}(0))$ is a multiplication-triple, then $\tau^{(l)}$ will be zero, provided $r = 0$ holds, which can happen with probability $\frac{1}{|\mathbb{F}|}$. And in this case, the honest parties will not output $\perp$ in the real-world. Since there are ℓ possible values for l, it follows from the union bound that except with probability at most $\frac{\ell}{|\mathbb{F}|} \equiv 2^{-\Omega(\kappa)}$, the honest parties would reconstruct $\tau^{(l)} \neq 0$ and output $\perp$ in the real-world. $\square$

Finally from Claims 6.2-6.5, we conclude that

$$\left\{\mathrm{HYBRID}^{\mathcal{F}_{\mathsf{ACSS}},\mathcal{F}_{\mathsf{Rand}}}_{\Pi_{\mathsf{AMTSS}},\mathsf{Adv}(z)}(\{(A^{(l)}(\cdot), B^{(l)}(\cdot), C^{(l)}(\cdot))\}_{l\in\{1,\ldots,\ell\}})\right\}_{z\in\{0,1\}^\star} \overset{s}{\equiv} \left\{\mathrm{IDEAL}_{\mathcal{F}_{\mathsf{AMTSS}},\mathcal{S}_{\mathsf{AMTSS}}(z)}(\{(A^{(l)}(\cdot), B^{(l)}(\cdot), C^{(l)}(\cdot))\}_{l\in\{1,\ldots,\ell\}})\right\}_{z\in\{0,1\}^\star}$$

holds, thus proving the theorem. $\square$

In the protocol Π_{APrep}, the dealer shares 6ℓ degree-t polynomials via $\mathcal{F}_{\mathsf{ACSS}}$. Since the functionality $\mathcal{F}_{\mathsf{ACSS}}$ can be securely realized by the protocol Π_{ACSS} with statistical security, by substituting $L = 6\ell$ in Theorem 5.6, we get the following corollary of Theorem 6.1.

Corollary 6.6. *Protocol Π_{AMTSS} UC-securely realizes the functionality $\mathcal{F}_{\mathsf{AMTSS}}$ with statistical security in the $\mathcal{F}_{\mathsf{Rand}}$-hybrid model, in the presence of a static malicious adversary, corrupting at most $t < \frac{n}{3}$ parties. The protocol needs a communication of $\mathcal{O}(\ell \cdot n^3\kappa + n^4\kappa^2 + n^5)$ bits and one instance of $\mathcal{F}_{\mathsf{Rand}}$ is involved.*

6.2 Securely Realizing $\mathcal{F}_{\mathsf{APrep}}$ in $(\mathcal{F}_{\mathsf{AMTSS}}, \mathcal{F}_{\mathsf{ABA}})$-Hybrid Model

We next present a protocol Π_{APrep} (Fig 18), for securely realizing $\mathcal{F}_{\mathsf{APrep}}$ in the $(\mathcal{F}_{\mathsf{AMTSS}}, \mathcal{F}_{\mathsf{ABA}})$-hybrid model. The high level idea of the protocol is as follows. Each party generates complete t-sharing of c_M number of random multiplication-triples by invoking the functionality $\mathcal{F}_{\mathsf{AMTSS}}$.

To avoid an indefinite wait, the parties cannot afford to wait to receive output from all the n instances of $\mathcal{F}_{\mathsf{AMTSS}}$, as the *corrupt* dealers in $\mathcal{P}$ may not invoke the corresponding instance of $\mathcal{F}_{\mathsf{AMTSS}}$. Hence the parties next try to agree on a common subset $\mathcal{CS}$ of $n - t$ triple-providers, whose corresponding multiplication-triples are eventually completely t-shared. This is done by using the *agreement on a common-subset* (ACS) protocol of [13]. Informally, it requires the parties to call the functionality $\mathcal{F}_{\mathsf{ABA}}$ n times, where the j^{th} call is used to decide whether party $P_j \in \mathcal{CS}$. For this, the parties participate with an input 1 during the j^{th} call to $\mathcal{F}_{\mathsf{ABA}}$ if and only if P_j generates a complete t-sharing of its multiplication-triples. Upon receiving 1 as an output from $n - t$ instances of $\mathcal{F}_{\mathsf{ABA}}$, the parties participate with input 0 for the remaining calls to $\mathcal{F}_{\mathsf{ABA}}$ for which no input has been provided yet. The idea here is that since there are at least $n - t$ honest parties whose multiplication-triples are eventually completely t-shared, it implies that *all* honest parties eventually participate with vote 1 in at least $n - t$ calls to $\mathcal{F}_{\mathsf{ABA}}$. Consequently, all *honest* parties eventually receive an output 1 from $n - t$ instances of $\mathcal{F}_{\mathsf{ABA}}$. Once $n - t$ calls to $\mathcal{F}_{\mathsf{ABA}}$ respond back with output 1, *all* honest parties pass an input 0 to all the remaining calls to $\mathcal{F}_{\mathsf{ABA}}$ which are not yet initiated, ensuring that there is a participation of at least $n - t$ parties in all the n calls to $\mathcal{F}_{\mathsf{ABA}}$. This guarantees that all honest parties eventually receive an output in response to all the n calls to $\mathcal{F}_{\mathsf{ABA}}$. And at least $n - t$ of these (common) outputs will be 1, which corresponds to the indices of the parties in $\mathcal{CS}$. In the protocol, we use the notation $\mathsf{sid}_j \overset{\text{def}}{=} \mathsf{sid}||j$ to denote the j^{th} call to $\mathcal{F}_{\mathsf{ABA}}$ and $\mathcal{F}_{\mathsf{AMTSS}}$ during the session id sid.

For simplicity, let $n - t = 2k + 1$. Since $t < n/3$, it follows that $k \geq t$. Also, without loss of generality, let $P_1, \ldots, P_{2k+1}$ be the parties in $\mathcal{CS}$. Each party P_j in $\mathcal{CS}$ has shared c_M multiplication-triples. This is because P_j is included in $\mathcal{CS}$ only if the j^{th} instance of $\mathcal{F}_{\mathsf{ABA}}$ responds with an output 1, which can happen only if at least one *honest* party provides an input 1 to the the j^{th} instance of $\mathcal{F}_{\mathsf{ABA}}$, implying that the honest party has received valid shares of P_j's multiplication-triples from $\mathcal{F}_{\mathsf{AMTSS}}$ with sid_j; this ensures that every other honest party also eventually receives its respective shares of P_j's multiplication-triples from $\mathcal{F}_{\mathsf{AMTSS}}$. Moreover, the multiplication-triples shared by the honest parties in $\mathcal{CS}$ are uniformly random and there are at least $t + 1$ such *honest* parties in the set $\mathcal{CS}$. However, the exact identity of the honest parties in $\mathcal{CS}$ is not known. Hence the parties apply a "triple-extraction" protocol from [24] on the multiplication-triples of the parties in $\mathcal{CS}$ to output complete t-sharing of c_M multiplication-triples, which are uniformly random. For this, the parties divide the triples of the parties in $\mathcal{CS}$ into c_M batches, where the l^{th} batch consists of the l^{th} shared multiplication-triple of the parties in $\mathcal{CS}$. And then the triple-extraction is executed in parallel for all the c_M batches, where a uniformly random shared multiplication-triple is extracted from each batch. We explain the triple-extraction procedure for a single batch and the same idea is executed with all the c_M batches.

Consider a batch of completely t-shared multiplication-triples $\{[(a^{(1)}]_t, [b^{(1)}]_t, [c^{(1)}]_t), \ldots, ([a^{(2k+1)}]_t, [b^{(2k+1)}]_t, [c^{(2k+1)}]_t)\}$, where the triple $(a^{(j)}, b^{(j)}, c^{(j)})$ is shared by party $P_j \in \mathcal{CS}$ by invoking $\mathcal{F}_{\mathsf{AMTSS}}$. It follows that if P_j is *honest*, then the triple $(a^{(j)}, b^{(j)}, c^{(j)})$ is uniformly random. Moreover, the adversary's output from $\mathcal{F}_{\mathsf{AMTSS}}$ with sid_j is completely independent of $(a^{(j)}, b^{(j)}, c^{(j)})$, since the underlying sharing polynomials are random and of degree-t and adversary receives at most t shares for each polynomial. The parties first "transform" the triples $\{([a^{(j)}]_t, [b^{(j)}]_t, [c^{(j)}]_t)\}_{j \in \{1,\ldots,2k+1\}}$, into another set of multiplication-triples $\{([u^{(j)}]_t, [v^{(j)}]_t, [w^{(j)}]_t)\}_{j \in \{1,\ldots,2k+1\}}$, such that all the following hold:

- There exist polynomials, say $U(\cdot), V(\cdot)$ and $W(\cdot)$ of degree-k, k and $2k$ respectively, passing through the points $\{(j, u^{(j)})\}_{j \in \{1,\ldots,2k+1\}}$, $\{(j, v^{(j)})\}_{j \in \{1,\ldots,2k+1\}}$ and $\{(j, w^{(j)})\}_{j \in \{1,\ldots,2k+1\}}$ respectively, such that $W(\cdot) = U(\cdot) \cdot V(\cdot)$ holds.
- If $P_j \in \mathcal{CS}$ is *corrupt*, then adversary completely learns the multiplication-triple $(u^{(j)}, v^{(j)}, w^{(j)})$.
- If $P_j \in \mathcal{CS}$ is *honest*, then adversary's view is independent of the multiplication-triple $(u^{(j)}, v^{(j)}, w^{(j)})$.

From the properties of the transformation, it follows that adversary learns at most t distinct values on the triplet of polynomials $(U(\cdot), V(\cdot), W(\cdot))$, thus leaving $(k + 1) - t$ "degree of freedom" in these polynomials, from the view point of the adversary. Since $k \geq t$, we get $(k + 1) - t \geq 1$ and hence the triplet $(U(\beta), V(\beta), W(\beta))$ will be uniformly random from the view point of the adversary, where β is a distinct value, different from $1, \ldots, 2k + 1$. More specifically, there exists a one-to-one mapping between the $(k+1)-t$ values on the polynomials $(U(\cdot), V(\cdot), W(\cdot))$ unknown to the adversary and $(U(\beta), V(\beta), W(\beta))$. Moreover, the triplet $(U(\beta), V(\beta), W(\beta))$ will be a multiplication-triple, since $W(\cdot) = U(\cdot) \cdot V(\cdot)$ holds. Since $U(\beta), V(\beta)$ and $W(\beta)$ can be expressed as a publicly-known linear function (namely the Lagrange's function) of the points $\{(j, u^{(j)})\}_{j \in \{1,\ldots,2k+1\}}$, $\{(j, v^{(j)})\}_{j \in \{1,\ldots,2k+1\}}$ and $\{(j, w^{(j)})\}_{j \in \{1,\ldots,2k+1\}}$ respectively, it follows that once the transformation is done, the parties can locally compute the complete t-sharing of the output multiplication-triple $(U(\beta), V(\beta), W(\beta))$, from the complete t-sharing of transformed multiplication-triples. We next explain the transformation of the shared multiplication-triples $\{([a^{(j)}]_t, [b^{(j)}]_t, [c^{(j)}]_t)\}_{j \in \{1,\ldots,2k+1\}}$ into $\{([u^{(j)}]_t, [v^{(j)}]_t, [w^{(j)}]_t)\}_{j \in \{1,\ldots,2k+1\}}$.

The idea behind the transformation is the following: the parties first define the polynomials $U(\cdot)$ and $V(\cdot)$ and then compute their product (all in a shared fashion). To define $U(\cdot)$, the parties set $U(\cdot)$ to be the distinct degree-k polynomial, passing through the $k+1$ distinct points $\{(j, a^{(j)})\}_{j \in \{1,\ldots,k+1\}}$. Consequently, the parties set $[u^{(j)}]_t$ to be the same as $[a^{(j)}]_t$ for $j = 1, \ldots, k + 1$ (which can be done locally). Similarly, to define $V(\cdot)$, the parties set $V(\cdot)$ to be the distinct degree-k polynomial, passing through the $k + 1$ distinct points $\{(j, b^{(j)})\}_{j \in \{1,\ldots,k+1\}}$. Consequently, the parties set $[v^{(j)}]_t$ to be the same as $[b^{(j)}]_t$ for $j = 1, \ldots, k+1$ (which can be done locally). Since we need $W(\cdot) = U(\cdot) \cdot V(\cdot)$ to hold, the parties set $[w^{(j)}]_t$ to be the same as $[c^{(j)}]_t$ for $j = 1, \ldots, k+1$ (which can be done locally). However, the polynomial $W(\cdot)$ is not yet well defined, because it is of degree-$2k$ and so to uniquely define $W(\cdot)$, we need $2k+1$ distinct points. We stress that the parties cannot define $W(\cdot)$ to be the polynomial passing through the points $\{(j, c^{(j)})\}_{j \in \{1,\ldots,2k+1\}}$, as this will not ensure that $W(\cdot) = U(\cdot) \cdot V(\cdot)$ holds. This is because the values $\{a^{(j)}\}_{j\in\{k+2,\ldots,2k+1\}}$ and $\{b^{(j)}\}_{j\in\{k+2,\ldots,2k+1\}}$ need not lie on the polynomials $U(\cdot)$ and $V(\cdot)$, which are already defined.

To define the remaining k distinct points on the $W(\cdot)$ polynomial, the parties first "extend" the already defined polynomials $U(\cdot)$ and $V(\cdot)$ at k new points. Namely, the points $\{U(j)\}_{j\in\{k+2,\ldots,2k+1\}}$ and $\{V(j)\}_{j\in\{k+2,\ldots,2k+1\}}$ can be expressed as a publicly-known linear function (namely the Lagrange's function) of the points $\{U(j)\}_{j\in\{1,\ldots,k+1\}}$ and $\{V(j)\}_{j\in\{1,\ldots,k+1\}}$ respectively. Consequently, the parties (locally) compute complete t-sharings of the new points on the $U(\cdot)$ and $V(\cdot)$ polynomials, by applying the corresponding Lagrange's function on the complete t-sharings of $\{U(j)\}_{j\in\{1,\ldots,k+1\}}$ and $\{V(j)\}_{j\in\{1,\ldots,k+1\}}$ respectively. The parties next compute complete t-sharings of the product of these new points, by deploying the Beaver's method. Namely, to compute complete t-sharing of $U(j) \cdot V(j)$ by Beaver's method, the parties use the completely t-shared multiplication-triple $(a^{(j)}, b^{(j)}, c^{(j)})$ as the auxiliary triple, for $j = k+2, \ldots, 2k+1$. We stress that the un-transformed triples $\{(a^{(j)}, b^{(j)}, c^{(j)})\}_{j\in\{k+2,\ldots,2k+1\}}$ are not yet used till this step and there are k such "un-used" multiplication-triples, which can be deployed to compute the product of k new points on the $U(\cdot)$ and $V(\cdot)$ polynomials (in a shared fashion).

We note that in the protocol Π_{APrep}, the corrupt parties will have complete "control" on their shares of the resultant extracted multiplication-triples, although the resultant multiplication-triples will be random from the view point of the adversary. This is because the polynomials $(U(\cdot), V(\cdot), W(\cdot))$ defined during the triple-transformation are *deterministically* determined by the multiplication-triples, shared by the parties in $\mathcal{CS}$. While the *honest* parties in $\mathcal{CS}$ share uniformly random multiplication-triples using random polynomials, the adversary can decide which honest parties finally make it to $\mathcal{CS}$ by scheduling the messages of the parties during the instances of $\mathcal{F}_{\mathsf{ABA}}$ and $\mathcal{F}_{\mathsf{AMTSS}}$. Moreover, adversary can decide its sharing polynomials for $\mathcal{F}_{\mathsf{AMTSS}}$, based on the shares of the polynomials of the honest parties, that it learns from $\mathcal{F}_{\mathsf{AMTSS}}$. It is precisely to model this artefact, the ideal-world adversary is given the provision to dictate the shares that it wants for the corrupt parties, corresponding to the random multiplication-triples, picked by the functionality $\mathcal{F}_{\mathsf{APrep}}$ (see the formal description of $\mathcal{F}_{\mathsf{APrep}}$ in Fig 14).

We stress that we separately do not put any termination condition for any party in protocol Π_{APrep}. Since Π_{APrep} will be used as a sub-protocol in our AMPC protocol, the termination condition of our AMPC protocol will automatically trigger the termination of all the underlying instances of Π_{APrep}.

Protocol Π_{APrep}

Sharing Random Multiplication-Triples:

1. Randomly pick multiplication-triples $\{(a^{(i,l)}, b^{(i,l)}, c^{(i,l)})\}_{l\in\{1,\ldots,c_M\}}$, where $c^{(i,l)} = a^{(i,l)} \cdot b^{(i,l)}$ holds.
2. For $l = 1, \ldots, c_M$, pick random degree-t univariate polynomials $A^{(i,l)}(\cdot), B^{(i,l)}$ and $C^{(i,l)}(\cdot)$ respectively, such that $A^{(i,l)}(0) = a^{(i,l)}$, $B^{(i,l)}(0) = b^{(i,l)}$ and $C^{(i,l)}(0) = c^{(i,l)}$ holds.
3. Send $(\mathsf{Dealer}, \mathsf{AMTSS}, \mathsf{sid}_i, \{(A^{(i,l)}(\cdot), B^{(i,l)}(\cdot), C^{(i,l)}(\cdot))\}_{l\in\{1,\ldots,c_M\}})$ to $\mathcal{F}_{\mathsf{AMTSS}}$ as a dealer.

Agreeing on a Common Subset of Multiplication-Triple Providers: Initialize a set of local multiplication-triple providers $\mathcal{LP}_i$ to $\emptyset$ and a set of global multiplication-triple providers $\mathcal{GP}_i$ to $\emptyset$.

1. Request output from $\mathcal{F}_{\mathsf{AMTSS}}$ for each sid_j, where $j = 1, \ldots, n$. If $(P_j, \mathsf{Dealer}, \mathsf{AMTSS}, \mathsf{sid}_j, \{(a_i^{(j,l)}, b_i^{(j,l)}, c_i^{(j,l)})\}_{l\in\{1,\ldots,c_M\}})$ is received as an output from $\mathcal{F}_{\mathsf{AMTSS}}$ for sid_j where P_j is the dealer, then include P_j to $\mathcal{LP}_i$.
2. If P_j is included in $\mathcal{LP}_i$, then send $(\mathsf{vote}, \mathsf{sid}_j, 1)$ to $\mathcal{F}_{\mathsf{ABA}}$.
3. Request outputs from $\mathcal{F}_{\mathsf{ABA}}$ until receiving $(\mathsf{decide}, \mathsf{sid}_j, v_j)$ for every $j \in \{1, \ldots, n\}$.
4. Include P_j to $\mathcal{GP}_i$ if $(\mathsf{decide}, \mathsf{sid}_j, v_j)$ is received from $\mathcal{F}_{\mathsf{ABA}}$, such that $v_j = 1$.

5. If $|\mathcal{GP}_i| = n - t$, then stop executing step 1 under "**Agreeing on a Common Subset of Multiplication-Triple Providers**" and send $(\mathsf{vote}, \mathsf{sid}_j, 0)$ to $\mathcal{F}_{\mathsf{ABA}}$, for every $j \in \{1, \ldots, n\}$ such that $P_j \notin \mathcal{GP}_i$.
6. Once $(\mathsf{decide}, \mathsf{sid}_j, v_j)$ is received from $\mathcal{F}_{\mathsf{ABA}}$ for every $j \in \{1, \ldots, n\}$, set $\mathcal{CS} = \{P_j : v_j = 1\}$.
7. Wait until receiving$(P_j, \mathsf{Dealer}, \mathsf{AMTSS}, \mathsf{sid}_j, \{(a_i^{(j,l)}, b_i^{(j,l)}, c_i^{(j,l)})\}_{l \in \{1,\ldots,c_M\}})$ from $\mathcal{F}_{\mathsf{AMTSS}}$ for every $P_j \in \mathcal{CS}$.

Transforming the Shared Multiplication-Triples of the Dealers in $\mathcal{CS}$: Wait until the set $\mathcal{CS}$ of $n - t$ multiplication-triple providers is decided and the shares corresponding to the multiplication-triples of each dealer in $\mathcal{CS}$ is received. For simplicity, let $n - t = 2k + 1$. Moreover, without loss of generality, let $\mathcal{CS} = \{P_1, \ldots, P_{2k+1}\}$. For $l = 1, \ldots, c_M$, do the following:

1. For $j = 1, \ldots, k+1$, set $u_i^{(j,l)} := a_i^{(j,l)}$, $v_i^{(j,l)} := b_i^{(j,l)}$ and $w_i^{(j,l)} := c_i^{(j,l)}$.
2. For $j = k+2, \ldots, 2k+1$, compute $u_i^{(j,l)} = \mathsf{Lagrange}(k+1, \{(1, u_i^{(1,l)}), \ldots, (k+1, u_i^{(k+1,l)})\}, j)$ and $v_i^{(j,l)} = \mathsf{Lagrange}(k+1, \{(1, v_i^{(1,l)}), \ldots, (k+1, v_i^{(k+1,l)})\}, j)$.
3. For $j = k+2, \ldots, 2k+1$, compute $d_i^{(j,l)} \overset{\text{def}}{=} u_i^{(j,l)} - a_i^{(j,l)}$ and $e_i^{(j,l)} \overset{\text{def}}{=} v_i^{(j,l)} - b_i^{(j,l)}$.
4. For $j = k+2, \ldots, 2k+1$, send $(\mathsf{sid}_j, \{d_i^{(j,l)}, e_i^{(j,l)}\})$ to every party in $\mathcal{P}$.
5. For $j = k+2, \ldots, 2k+1$, keep receiving $(\mathsf{sid}_j, \{d_m^{(j,l)}, e_m^{(j,l)}\})$ from various parties P_m and keep executing the steps of OEC, till $d^{(j,l)}$ and $e^{(j,l)}$ are reconstructed.
6. For $j = k+2, \ldots, 2k+1$, compute $w_i^{(j,l)} = d^{(j,l)} \cdot e^{(j,l)} + d^{(j,l)} \cdot b_i^{(j,l)} + e^{(j,l)} \cdot a_i^{(j,l)} + c_i^{(j,l)}$.

Output Computation:

1. For $l = 1, \ldots, c_M$, compute $u_i^{(l)} = \mathsf{Lagrange}(k+1, \{(1, u_i^{(1,l)}), \ldots, (k+1, u_i^{(k+1,l)})\}, \beta)$, $v_i^{(l)} = \mathsf{Lagrange}(k+1, \{(1, v_i^{(1,l)}), \ldots, (k+1, v_i^{(k+1,l)})\}, \beta)$ and $w_i^{(l)} = \mathsf{Lagrange}(2k+1, \{(1, w_i^{(1,l)}), \ldots, (2k+1, w_i^{(2k+1,l)})\}, \beta)$, where $\beta \in \mathbb{F}$ is a non-zero value, different from $1, \ldots, 2k+1$.
2. Output $(\mathsf{triple\text{-}shares}, \mathsf{sid}, \{(u_i^{(\ell)}, v_i^{(\ell)}, w_i^{(\ell)})\}_{\ell \in \{1,\ldots,c_M\}})$.

Figure 18: Protocol for realizing $\mathcal{F}_{\mathsf{APrep}}$ in the $(\mathcal{F}_{\mathsf{AMTSS}}, \mathcal{F}_{\mathsf{ABA}})$-hybrid model. The above code is executed by every $P_i \in \mathcal{P}$.

We next prove the security of the protocol Π_{APrep} in the $(\mathcal{F}_{\mathsf{AMTSS}}, \mathcal{F}_{\mathsf{ABA}})$-hybrid model. We actually prove a stronger statement, namely that protocol Π_{APrep} is *perfectly-secure*, given the parties have access to ideal functionalities $\mathcal{F}_{\mathsf{AMTSS}}$ and $\mathcal{F}_{\mathsf{ABA}}$.

Theorem 6.7. *Protocol Π_{APrep} UC-securely realizes the functionality $\mathcal{F}_{\mathsf{APrep}}$ with perfect security in the $(\mathcal{F}_{\mathsf{AMTSS}}, \mathcal{F}_{\mathsf{ABA}})$-hybrid model, in the presence of a static malicious adversary, corrupting at most $t < \frac{n}{3}$ parties. The protocol needs a communication of $\mathcal{O}(c_M \cdot n^3 \kappa)$ bits.*

Proof. The communication complexity simply follows from the fact that in the protocol, $\mathcal{O}(c_M \cdot n)$ completely t-shared values are publicly reconstructed, while transforming the multiplication-triples, shared by the dealers in $\mathcal{CS}$, where reconstructing one such value needs a communication of $\mathcal{O}(n^2)$ field elements.

For security, let Adv be an arbitrary real-world adversary, attacking the protocol Π_{APrep} and let $\mathcal{Z}$ be an arbitrary environment. We show the existence of a simulator $\mathcal{S}_{\mathsf{APrep}}$, such that for any set of corrupted parties $\mathcal{C}$ with $|\mathcal{C}| \leq t$ and for all inputs, the output of all parties and the adversary in an execution of the protocol Π_{APrep} with Adv is *perfectly-indistinguishable* from the outputs in an execution with $\mathcal{S}_{\mathsf{APrep}}$ in the ideal model involving $\mathcal{F}_{\mathsf{APrep}}$. This further implies that $\mathcal{Z}$ cannot distinguish between the two executions. The simulator simulates the various phases of the protocol as per the steps given in Fig 19.

The high level idea of the simulator is as follows. The simulator itself performs the role of the ideal functionality $\mathcal{F}_{\mathsf{AMTSS}}$ and $\mathcal{F}_{\mathsf{ABA}}$ whenever required. Whenever Adv sends polynomials for sharing to $\mathcal{F}_{\mathsf{AMTSS}}$ on the behalf of a corrupt party, the simulator records these polynomials if they are valid (namely triplets of degree-t polynomials, whose constant terms are multiplication-triples). On the other hand, for the honest parties, the simulator randomly picks triplets of degree-t polynomials, whose constant terms are random multiplication-triples and distributes the shares of these polynomials as per $\mathcal{F}_{\mathsf{AMTSS}}$. To select the common triple-providers, the simulator itself performs the role of $\mathcal{F}_{\mathsf{ABA}}$ and simulates the honest parties as per the steps of the protocol and $\mathcal{F}_{\mathsf{ABA}}$. This allows the simulator learn the common subset of triple-providers $\mathcal{CS}$. Notice that the sharing polynomials of all the parties in $\mathcal{CS}$ will be available with the simulator: while the polynomials of the honest parties in $\mathcal{CS}$ are selected by the simulator itself, for every *corrupt* party P_j which makes it to $\mathcal{CS}$, at least one honest party P_i should participate with input 1 in the corresponding call to $\mathcal{F}_{\mathsf{ABA}}$, implying that the honest party P_i must have received valid shares from $\mathcal{F}_{\mathsf{AMTSS}}$, corresponding to the degree-t polynomials which P_j sent to $\mathcal{F}_{\mathsf{AMTSS}}$. Since in the simulation, the role of $\mathcal{F}_{\mathsf{AMTSS}}$ is played by the simulator itself, it implies that the polynomials provided by P_j will be known to the simulator.

Once the simulator learns $\mathcal{CS}$ and the sharing polynomials of the parties in $\mathcal{CS}$, it simulates the rest of the interaction between the honest parties and Adv as per the protocol steps, by itself playing the role of the honest parties. Moreover, the simulator also learns the shares of the corrupt parties, corresponding to the output multiplication-triples in the simulated execution. The simulator then communicates these shares on the behalf of the corrupt parties during its interaction with $\mathcal{F}_{\mathsf{APrep}}$. This ensures that the shares of the corrupt parties remain same in both the worlds.

Simulator $\mathcal{S}_{\mathsf{APrep}}$

$\mathcal{S}_{\mathsf{APrep}}$ constructs virtual real-world honest parties and invokes the real-world adversary Adv. The simulator simulates the view of Adv, namely its communication with $\mathcal{Z}$, the messages sent by the honest parties and the interaction with various functionalities. In order to simulate $\mathcal{Z}$, the simulator forwards every message it receives from $\mathcal{Z}$ to Adv and vice-versa. The simulator then simulates the various phases of the protocol as follows.

Sharing Random Multiplication-Triples:

- The simulator simulates the operations of the honest parties during this phase, by picking random input multiplication-triples $(\widetilde{a}^{(j,l)}, \widetilde{b}^{(j,l)}, \widetilde{c}^{(j,l)})$ for $l = 1, \ldots, c_M$, for every $P_j \notin \mathcal{C}$. Namely, when Adv requests output from $\mathcal{F}_{\mathsf{AMTSS}}$ (on the behalf of any party $P_i \in \mathcal{C}$) with sid_j for any $P_j \notin \mathcal{C}$, the simulator responds with an output $(P_j, \mathsf{Dealer}, \mathsf{AMTSS}, \mathsf{sid}_j, \{(\widetilde{A}^{(j,l)}(\alpha_i), \widetilde{B}^{(j,l)}(\alpha_i), \widetilde{C}^{(j,l)}(\alpha_i))\}_{l \in \{1,\ldots,c_M\}})$ on the behalf of $\mathcal{F}_{\mathsf{APrep}}$. Here $\widetilde{A}^{(j,l)}(\cdot), \ldots, \widetilde{C}^{(j,l)}(\cdot)$ are random degree-t polynomials, where $\widetilde{A}((j,l))(0) = \widetilde{a}^{(j,l)}$, $\widetilde{B}((j,l))(0) = \widetilde{b}^{(j,l)}$ and $\widetilde{C}((j,l))(0) = \widetilde{c}^{(j,l)}$ holds.
- Whenever Adv sends $(\mathsf{Dealer}, \mathsf{AMTSS}, \mathsf{sid}_i, \{(A^{(i,l)}(\cdot), B^{(i,l)}(\cdot), C^{(i,l)}(\cdot))\}_{l \in \{1,\ldots,c_M\}})$ to $\mathcal{F}_{\mathsf{AMTSS}}$ as a dealer, on the behalf of any $P_i \in \mathcal{C}$, the simulator records these input polynomials on the behalf of P_i, provided the polynomials $(A^{(i,l)}(\cdot), B^{(i,l)}(\cdot), C^{(i,l)}(\cdot))$ are triplets of degree-t polynomials, with their constant terms being a multiplication-triple.

Agreeing on a Common Subset of Multiplication-Triple Providers: The simulator simulates the interface to $\mathcal{F}_{\mathsf{ABA}}$ for Adv as per the steps of the protocol, by itself performing the role of $\mathcal{F}_{\mathsf{ABA}}$. When the first honest party completes this phase during the simulated execution, $\mathcal{S}_{\mathsf{APrep}}$ learns the set $\mathcal{CS}$.

Transforming the Shared Multiplication-Triples of the Dealers in $\mathcal{CS}$: Based on the complete t-sharing of the multiplication-triples of the parties in $\mathcal{CS}$, the simulator simulates the interaction with Adv for this phase, on the behalf of the honest parties, by itself performing the role of the honest parties[a].

Interaction with $\mathcal{F}_{\mathsf{APrep}}$: Let $\{(\widetilde{u}_i^{(l)}, \widetilde{v}_i^{(l)}, \widetilde{w}_i^{(l)})\}_{l \in \{1,\ldots,c_M, P_i \in \mathcal{C}\}}$ be the output shares of the parties in $\mathcal{C}$, during the simulated execution of Π_{APrep}. The simulator sends $(\mathsf{shares}, \mathsf{sid}, \{(\widetilde{u}_i^{(l)}, \widetilde{v}_i^{(l)}, \widetilde{w}_i^{(l)})\}_{l \in \{1,\ldots,c_M, P_i \in \mathcal{C}\}})$ to $\mathcal{F}_{\mathsf{APrep}}$, on the behalf of parties in $\mathcal{C}$.

[a]The messages which honest parties send to Adv during this phase are deterministically determined by the complete t-

sharing of the multiplication-triples of the parties in $\mathcal{CS}$. And in the simulated execution of Π_{APrep}, the simulator will know these complete t-sharings.

Figure 19: Simulator for the protocol Π_{APrep} where Adv corrupts the parties in set $\mathcal{C}$, where $|\mathcal{C}| \leq t$

We now prove a series of claims, which will help us to finally prove the theorem. During the proofs of these claims, we skip the session id sid. We first claim that in any execution of Π_{APrep}, a set $\mathcal{CS}$ is eventually generated.

Claim 6.8. In any execution of Π_{APrep}, a set $\mathcal{CS}$ is eventually generated, such that for every $P_j \in \mathcal{CS}$, there exist c_M multiplication-triples known to P_j, which are eventually completely t-shared.

Proof. We first show that there exist at least $n-t$ instances of $\mathcal{F}_{\mathsf{ABA}}$ in which all the honest parties obtain an output 1. For this, we consider the following two cases.

- If some *honest* party P_i has participated with vote input 0 in any instance of $\mathcal{F}_{\mathsf{ABA}}$ during step 5 of the agreement on multiplication-triple providers phase, then it implies that $|\mathcal{GP}_i| = n-t$ holds for P_i, which further implies that at least $n-t$ instances of $\mathcal{F}_{\mathsf{ABA}}$ responded with output 1, which is what we wanted to show.
- No honest party has participated with vote input 0 in any instance of $\mathcal{F}_{\mathsf{ABA}}$. In the protocol, each *honest* party P_j sends its multiplication-triples to be completely t-shared to $\mathcal{F}_{\mathsf{AMTSS}}$ and every honest party eventually receives shares of these triples as output from the corresponding instances of $\mathcal{F}_{\mathsf{AMTSS}}$. Hence, corresponding to every honest P_j, all honest parties eventually participate with vote input 1 in the j^{th} instance of $\mathcal{F}_{\mathsf{ABA}}$. As there are at least $n-t$ honest parties P_j, it follows that all honest parties eventually participate with vote inputs 1 in at least $n-t$ instances of $\mathcal{F}_{\mathsf{ABA}}$. Consequently, these $n-t$ instances of $\mathcal{F}_{\mathsf{ABA}}$ respond with an output 1.

We next show that all honest parties eventually receive an output from all the instances of $\mathcal{F}_{\mathsf{ABA}}$. Since the honest parties obtain an output 1 in at least $n-t$ instances of $\mathcal{F}_{\mathsf{ABA}}$, it thus follows that all honest parties eventually participate with some vote inputs in the remaining $\mathcal{F}_{\mathsf{ABA}}$ instances and hence will eventually obtain some output from these $\mathcal{F}_{\mathsf{ABA}}$ instances as well. Since the set $\mathcal{CS}$ corresponds to the $\mathcal{F}_{\mathsf{ABA}}$ instances in which the honest parties obtain 1 as the output, it thus follows that eventually the honest parties obtain some $\mathcal{CS}$ where $|\mathcal{CS}| \geq n-t$. Moreover, the set $\mathcal{CS}$ will be common, as it is based on the outcome of $\mathcal{F}_{\mathsf{ABA}}$ instances.

Now consider an arbitrary $P_j \in \mathcal{CS}$. This implies that the parties obtain 1 as the output during the j^{th} instance of $\mathcal{F}_{\mathsf{ABA}}$. This further implies that at least one *honest* party P_i participated in this $\mathcal{F}_{\mathsf{ABA}}$ instance with vote input 1. This is possible only if P_i received valid shares as output from the j^{th} instance of $\mathcal{F}_{\mathsf{AMTSS}}$, further implying that P_j has provided valid degree-t multiplication-triple sharing polynomials as inputs to $\mathcal{F}_{\mathsf{AMTSS}}$. It now follows easily that eventually, all honest parties will have their respective shares, corresponding to the multiplication-triples of P_j. □

We next show that the view generated by $\mathcal{S}_{\mathsf{APrep}}$ for Adv is identically distributed as Adv's view during the real execution of Π_{APrep}.

Claim 6.9. The view of Adv in the simulated execution with $\mathcal{S}_{\mathsf{APrep}}$ is identically distributed as the view of Adv in the real execution of Π_{APrep}.

Proof. We first note that in the real-world (during the real execution of Π_{APrep}), the view of Adv consists of the following:

(1) Polynomials $\{(A^{(i,l)}(\cdot), B^{(i,l)}(\cdot), C^{(i,l)}(\cdot))\}_{l \in \{1,\ldots,c_M\}}$ (if any) for $\mathcal{F}_{\mathsf{AMTSS}}$, corresponding to $P_i \in \mathcal{C}$.

(2) Shares $\{(a_i^{(j,l)}, b_i^{(j,l)}, c_i^{(j,l)})\}_{l \in \{1,\ldots,c_M\}}$, corresponding to $P_j \notin \mathcal{C}$ and $P_i \in \mathcal{C}$.

(3) Inputs of the various parties during the $\mathcal{F}_{\mathsf{ABA}}$ instances and the outputs from the $\mathcal{F}_{\mathsf{ABA}}$ instances.

(4) For $j = k+2, \ldots, 2k+1$, complete t-sharing of[13] $\{d^{(j,l)}, e^{(j,l)}\}_{l \in \{1,\ldots,c_M\}}$.

The polynomials in (1) are the inputs of Adv and hence they are identically distributed in both the worlds. So let us fix these polynomials. In the real-world every $P_j \notin \mathcal{C}$ picks its *triple-sharing* polynomials for $\mathcal{F}_{\mathsf{AMTSS}}$ uniformly at random, where as in the ideal-world, for every $P_j \notin \mathcal{C}$, the simulator picks the sharing polynomials on the behalf of P_j uniformly at random. Since $|\mathcal{C}| \leq t$, it follows from Lemma 2.4, that the distribution of the shares in (2) is identical in both the worlds. Specifically, conditioned on the shares in (2), the underlying multiplication-triples shared by the parties $P_j \notin \mathcal{C}$ are uniformly distributed. Since the partial view of Adv containing (1) and (2) are identically distributed, let us fix them.

Now conditioned on $(1)-(2)$, it is easy to see that partial view of Adv consisting of (3) are identically distributed in both the worlds. This is because the outputs of the $\mathcal{F}_{\mathsf{ABA}}$ instances are determined deterministically based on the inputs provided by the various parties in these $\mathcal{F}_{\mathsf{ABA}}$ instances. And the inputs of the parties in various $\mathcal{F}_{\mathsf{ABA}}$ instances depend upon the order in which various parties receive outputs from various $\mathcal{F}_{\mathsf{AMTSS}}$ instances, which is completely determined by Adv since message scheduling is under the control of Adv. Thus the partial view of Adv containing $(1)-(3)$ are identically distributed in both the worlds and so let us fix them. This also fixes the set $\mathcal{CS}$, which according to Claim 6.8 is guaranteed to be generated. For simplicity and without loss of generality, let $\mathcal{CS} = \{P_1, \ldots, P_{2k+1}\}$, where $|\mathcal{CS}| = n - t = 2k+1$ holds.

Finally to see that the partial view consisting of (4) are identically distributed in both the worlds (conditioned on $(1)-(3)$), we note that $j = k+2, \ldots, 2k+1$, the t-sharing of $\{d^{(j,l)}, e^{(j,l)}\}_{l \in \{1,\ldots,c_M\}}$ are completely determined by the t-sharing of $\{(a^{(j,l)}, b^{(j,l)}, c^{(j,l)})\}_{j \in \{1,\ldots,2k+1\}, l \in \{1,\ldots,c_M\}}$. And conditioned on $(1)-(3)$, the distribution of t-sharing of $\{(a^{(j,l)}, b^{(j,l)}, c^{(j,l)})\}_{j \in \{1,\ldots,2k+1\}, l \in \{1,\ldots,c_M\}}$ are identical in both the worlds. Specifically, for every $P_j \in \mathcal{CS} \cap \mathcal{C}$, the complete t-sharing of $\{(a^{(j,l)}, b^{(j,l)}, c^{(j,l)})\}_{l \in \{1,\ldots,c_M\}}$ are exactly the same in both the worlds, as these sharings are the same as (1). On the other hand, for every $P_j \notin \mathcal{C}$ such that $P_j \in \mathcal{CS}$, the complete t-sharings of $\{(a^{(j,l)}, b^{(j,l)}, c^{(j,l)})\}_{l \in \{1,\ldots,c_M\}}$ are randomly distributed, conditioned on the shares in (2). $\square$

Finally, we show that conditioned on the view of Adv, the outputs of the honest parties are identically distributed in both the worlds.

Claim 6.10. Conditioned on the view of Adv, the output of the honest parties are identically distributed in the real execution of Π_{APrep} involving Adv, as well as in the ideal execution involving $\mathcal{S}_{\mathsf{APrep}}$ and $\mathcal{F}_{\mathsf{APrep}}$.

Proof. Let View be an arbitrary view of Adv. And let $\{([a^{(j,l)}]_t, [b^{(j,l)}]_t, [c^{(j,l)}]_t)\}_{j \in \{1,\ldots,2k+1\}, l \in \{1,\ldots,c_M\}}$ be the complete t-sharing of the multiplication-triples as per View, shared by the parties in $\mathcal{CS}$, where $\mathcal{CS} = \{P_1, \ldots, P_{2k+1}\}$. From the proof of Claim 6.9, it follows that corresponding to every *honest* $P_j \in \mathcal{CS}$, the multiplication-triples $\{(a^{(j,l)}, b^{(j,l)}, c^{(j,l)})\}_{l \in \{1,\ldots,c_M\}}$ are uniformly distributed, conditioned on the shares of these multiplication-triples, as determined by View. Let us fix these multiplication-triples. Now consider an arbitrary $l \in \{1, \ldots, c_M\}$. We show that in the real-world, the output of the honest parties consists of complete t-sharing of the multiplication-triple $(U^{(l)}(\beta), V^{(l)}(\beta), W^{(l)}(\beta))$, where the following holds.

- $U^{(l)}(\cdot), V^{(l)}(\cdot)$ and $W^{(l)}(\cdot)$ are polynomials of degree-k, k and $2k$ respectively, such that $W^{(l)}(\cdot) = U^{(l)}(\cdot) \cdot V^{(l)}(\cdot)$ holds. Moreover, β is a non-zero value from $\mathbb{F}$, distinct from $1, \ldots, 2k+1$.
- The points $\{(j, a^{(j,l)})\}_{j \in \{1,\ldots,k+1\}}$ and $\{(j, b^{(j,l)})\}_{j \in \{1,\ldots,k+1\}}$ lie on $U^{(l)}(\cdot)$ and $V^{(l)}(\cdot)$ respectively.
- Conditioned on View, the multiplication-triple $(U^{(l)}(\beta), V^{(l)}(\beta), W^{(l)}(\beta))$ is uniformly distributed.

Consider the transformed shared triples $\{(u^{(j,l)}, v^{(j,l)}, w^{(j,l)})\}_{j \in \{1,\ldots,2k+1\}}$. From the protocol steps, it is easy to verify that the transformed triples are also completely t-shared. We next argue that the transformed triples are also multiplication-triples. This is obviously the case for the first $k+1$ transformed triples,

[13]This is under the assumption that $n - t = 2k+1$ and $\mathcal{CS} = \{P_1, \ldots, P_{2k+1}\}$ holds.

as they are exactly the same as the first $k+1$ multiplication-triples $\{(a^{(j,l)}, b^{(j,l)}, c^{(j,l)})\}_{j \in \{1,\ldots,k+1\}}$. Let $U^{(l)}(\cdot)$ and $V^{(l)}(\cdot)$ be the unique degree-k polynomials, passing through the points $\{(j, u^{(j,l)})\}_{j \in \{1,\ldots,k+1\}}$ and $\{(j, v^{(j,l)})\}_{j \in \{1,\ldots,k+1\}}$ respectively. From the protocol steps, it follows that $u^{(j,l)} = U^{(l)}(j)$ and $v^{(j,l)} = V^{(l)}(j)$ holds respectively, for $j \in \{k+2, \ldots, 2k+1\}$. Moreover, $w^{(j,l)} = u^{(j,l)} \cdot v^{(j,l)}$ holds, for $j \in \{k+2, \ldots, 2k+1\}$. This is because for every $j \in \{k+2, \ldots, 2k+1\}$, a complete t-sharing of $w^{(j,l)}$ is computed by taking the product of completely t-shared $u^{(j,l)}$ and $v^{(j,l)}$ using Beaver's method, by deploying the auxiliary completely t-shared multiplication-triple $(a^{(j,l)}, b^{(j,l)}, c^{(j,l)})$. Let $W^{(l)}(\cdot)$ be the unique degree-$2k$ polynomial, passing through the points $\{(j, w^{(j,l)})\}_{j \in \{1,\ldots,2k+1\}}$. It follows that $W^{(l)}(\cdot) = U^{(l)}(\cdot) \cdot V^{(l)}(\cdot)$ holds, thus guaranteeing that the transformed triples are indeed multiplication-triples. Now since $u^{(l)} = U^{(l)}(\beta)$, $v^{(l)} = V^{(l)}(\beta)$ and $w^{(l)} = W^{(l)}(\beta)$ holds, it follows that the l^{th} output triple $(u^{(l)}, v^{(l)}, w^{(l)})$ is indeed a multiplication-triple. Moreover, it is easy to see that parties will hold a complete t-sharing of $(u^{(l)}, v^{(l)}, w^{(l)})$.

We next show that conditioned on View, the multiplication-triple $(u^{(l)}, v^{(l)}, w^{(l)})$ is uniformly distributed. For this, we argue that View consists of only $|\mathcal{C}|$ distinct values on the polynomials $U^{(l)}(\cdot)$ and $V^{(l)}(\cdot)$ (and hence on $W^{(l)}(\cdot)$). For this, we show that the transformed multiplication-triple $(u^{(j,l)}, v^{(j,l)}, w^{(j,l)})$ (which is the same $(U^{(l)}(j), V^{(l)}(j), W^{(l)}(j)))$ will be included in View, if and only if $P_j \in \mathcal{C} \cap \mathcal{CS}$. So consider an arbitrary party $P_j \in \mathcal{CS}$. We consider two cases, depending upon whether P_j is honest or corrupt.

- *Case I: $P_j \notin \mathcal{C}$.* In this case, conditioned on View, the multiplication-triple $(a^{(j,l)}, b^{(j,l)}, c^{(j,l)})$ is uniformly distributed. If $P_j \in \{P_1, \ldots, P_{k+1}\}$, then it also implies that View will be independent of $(u^{(j,l)}, v^{(j,l)}, w^{(j,l)})$, as in this case, the complete t-sharing of $(u^{(j,l)}, v^{(j,l)}, w^{(j,l)})$ is exactly the same as the complete t-sharing of $(a^{(j,l)}, b^{(j,l)}, c^{(j,l)})$, generated by P_j.

  On the other hand, if $P_j \in \{P_{k+2}, \ldots, P_{2k+1}\}$, then the shared triple $(a^{(j,l)}, b^{(j,l)}, c^{(j,l)})$ is used as an auxiliary triple while computing a complete t-sharing of $w^{(j,l)} = u^{(j,l)} \cdot v^{(j,l)}$, from complete t-sharing of $u^{(j,l)}$ and $v^{(j,l)}$ by using Beaver's method. Also, the complete t-sharing of $u^{(j,l)}$ and $v^{(j,l)}$ in this case is computed *non-interactively* as a linear function of complete t-sharings of $\{u^{(j,l)}\}_{j \in \{1,\ldots,k+1\}}$ and $\{v^{(j,l)}\}_{j \in \{1,\ldots,k+1\}}$ respectively. Now since the auxiliary triple $(a^{(j,l)}, b^{(j,l)}, c^{(j,l)})$ is independent of $u^{(j,l)}$ and $v^{(j,l)}$ and since adversary's view point is independent of the auxiliary triple, it follows that during the computation of complete t-sharing of $w^{(j,l)}$, no additional information about $u^{(j,l)}, v^{(j,l)}$ (and hence $w^{(j,l)}$) is added to View. Specifically, adversary learns $d^{(j,l)} \overset{\text{def}}{=} u^{(j,l)} - a^{(j,l)}$ and $e^{(j,l)} \overset{\text{def}}{=} v^{(j,l)} - b^{(j,l)}$. But it follows that for every candidate value of $u^{(j,l)}$ and $v^{(j,l)}$, there exists a corresponding candidate value of $a^{(j,l)}$ and $b^{(j,l)}$ respectively, which is consistent with the $d^{(j,l)}$ and $e^{(j,l)}$ learnt by the adversary. Thus View will be independent of $(u^{(j,l)}, v^{(j,l)}, w^{(j,l)})$ in this case as well.
- *Case II: $P_j \in \mathcal{C}$.* In this case, the complete t-sharing of the multiplication-triple $(a^{(j,l)}, b^{(j,l)}, c^{(j,l)})$ will be present in View. If $P_j \in \{P_1, \ldots, P_{k+1}\}$, then this further implies that $(u^{(j,l)}, v^{(j,l)}, w^{(j,l)})$ is included in View as well. This is because in this case, the complete t-sharing of $(u^{(j,l)}, v^{(j,l)}, w^{(j,l)})$ is exactly the same as the complete t-sharing of $(a^{(j,l)}, b^{(j,l)}, c^{(j,l)})$, generated by P_j. On the other hand, if $P_j \in \{P_{k+2}, \ldots, P_{2k+1}\}$, then during the computation of complete t-sharing of $w^{(j,l)}$, the adversary learns the complete t-sharing of $d^{(j,l)}$ and $e^{(j,l)}$, through which Adv learns the corresponding $u^{(j,l)}$ and $v^{(j,l)}$ respectively. This is because in this case, the complete t-sharing of the auxiliary multiplication-triple $(a^{(j,l)}, b^{(j,l)}, c^{(j,l)})$ used in the Beaver's method will be known to the adversary.

Now given that Adv learns only $|\mathcal{C}|$ distinct transformed multiplication-triples (which are distinct values on the triplets of polynomials $(U^{(l)}(\cdot), V^{(l)}(\cdot), W^{(l)}(\cdot)))$, it follows that View will be independent of the output multiplication-triple $(u^{(l)}, v^{(l)}, w^{(l)})$. This is because, there exists a one-to-one mapping between the $k+1-|\mathcal{C}|$ multiplication-triples $(a^{(j,l)}, b^{(j,l)}, c^{(j,l)})$ shared by the *first* $k+1-|\mathcal{C}|$ *honest* parties $P_j \in \{P_1, \ldots, P_{k+1}\}$

and $(u^{(l)}, v^{(l)}, w^{(l)})$. Namely, from the view point of Adv, for every candidate value $(a^{(j,l)}, b^{(j,l)}, c^{(j,l)})$ of the multiplication-triples shared by the *first* $k+1-|\mathcal{C}|$ *honest* parties $P_j \in \{P_1, \ldots, P_{k+1}\}$, there exists a unique triplet of polynomials $(U^{(l)}(\cdot), V^{(l)}(\cdot), W^{(l)}(\cdot))$ where $W^{(l)}(\cdot) = U^{(l)}(\cdot) \cdot V^{(l)}(\cdot)$ holds with degree of $U^{(l)}(\cdot), V^{(l)}(\cdot)$ being k, which is consistent with View. Since the multiplication-triples shared by the *honest* parties P_j are uniformly distributed and independent of View, it follows that $(u^{(l)}, v^{(l)}, w^{(l)})$ is also uniformly distributed.

To complete the proof, we now show that conditioned on the shares $\{(u_i^{(l)}, v_i^{(l)}, w_i^{(l)})\}_{P_i \in \mathcal{C}}$ (which are determined by View), the honest parties output complete t-sharing of some random multiplication-triple in the ideal-world, where the t-sharings are consistent with the shares $\{(u_i^{(l)}, v_i^{(l)}, w_i^{(l)})\}_{P_i \in \mathcal{C}}$. However, this simply follows from the fact that in the ideal-world, the simulator $\mathcal{S}_{\mathsf{APrep}}$ sends the shares $\{(u_i^{(l)}, v_i^{(l)}, w_i^{(l)})\}_{P_i \in \mathcal{C}}$ to $\mathcal{F}_{\mathsf{APrep}}$ on the behalf of the parties in $\mathcal{C}$. And $\mathcal{F}_{\mathsf{APrep}}$ generates a random complete t-sharing of some random multiplication-triple, consistent with the shares provided by $\mathcal{S}_{\mathsf{APrep}}$. □

Finally from Claim 6.8-6.10, we conclude that:

$$\left\{\text{HYBRID}^{\mathcal{F}_{\mathsf{AMTSS}}, \mathcal{F}_{\mathsf{ABA}}}_{\Pi_{\mathsf{APrep}}, \mathsf{Adv}(z)}(\cdot)\right\}_{z \in \{0,1\}^\star} \equiv \left\{\text{IDEAL}_{\mathcal{F}_{\mathsf{APrep}}, \mathcal{S}_{\mathsf{APrep}}(z)}(\cdot)\right\}_{z \in \{0,1\}^\star}$$

holds, thus proving the theorem. □

In the protocol Π_{APrep}, there are n instances of $\mathcal{F}_{\mathsf{AMTSS}}$, each for sharing c_M number of multiplication-triples. Moreover, there are n instances of $\mathcal{F}_{\mathsf{ABA}}$ to decide the set of triple-providers. Since the functionality $\mathcal{F}_{\mathsf{AMTSS}}$ can be securely realized by protocol Π_{AMTSS} with statistical security, by substituting $\ell = c_M$ in Corollary 6.6, we get the following corollary of Theorem 6.7.

Corollary 6.11. *Protocol Π_{APrep} UC-securely realizes the functionality $\mathcal{F}_{\mathsf{APrep}}$ with statistical security in the $(\mathcal{F}_{\mathsf{Rand}}, \mathcal{F}_{\mathsf{ABA}})$-hybrid model, in the presence of a static malicious adversary, corrupting at most $t < \frac{n}{3}$ parties. The protocol needs a communication of $\mathcal{O}(c_M \cdot n^4\kappa + n^5\kappa^2 + n^6)$ bits and involves n instances of $\mathcal{F}_{\mathsf{Rand}}$ and n instances of $\mathcal{F}_{\mathsf{ABA}}$.*

7 The AMPC Protocol

Finally, we present our AMPC protocol in the $(\mathcal{F}_{\mathsf{APrep}}, \mathcal{F}_{\mathsf{ACSS}}, \mathcal{F}_{\mathsf{ACast}}, \mathcal{F}_{\mathsf{ABA}})$-hybrid model. We first recall the corresponding ideal functionality $\mathcal{F}_{\mathsf{AMPC}}$ (see Fig 21) from [26].

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

$\mathcal{F}_{\mathsf{AMPC}}$ proceeds as follows, running with parties $\mathcal{P} = \{P_1, \ldots, P_n\}$ and an adversary $\mathcal{S}$ and parametrized by an n-party function $f : \mathbb{F}^n \to \mathbb{F}$. For each party P_i, initialize an input value $x^{(i)} = \perp$.

- Upon receiving a message $(\mathsf{input}, \mathsf{sid}, v)$ from some $P_i \in \mathcal{P}$, if $\mathcal{CS}$ has not been recorded yet or if $P_i \in \mathcal{CS}$, set $x^{(i)} = v$.
- Upon receiving a message $(\mathsf{coreset}, \mathsf{sid}, \mathcal{CS})$ from $\mathcal{S}$, verify that $\mathcal{CS}$ is a subset of $\mathcal{P}$ of size $n - t$, else ignore the message. If $\mathcal{CS}$ has not been recorded yet, then record $\mathcal{CS}$ and for every $P_i \notin \mathcal{CS}$, set $x^{(i)} = 0$.
- If the $\mathcal{CS}$ has been recorded and the value $x^{(i)}$ has been set to a value different from $\perp$ for every $P_i \in \mathcal{CS}$, then compute $y \overset{\text{def}}{=} f(x^{(1)}, \ldots, x^{(n)})$ and generate a request-based delayed output $(\mathsf{output}, \mathsf{sid}, (\mathcal{CS}, y))$ for every $P_i \in \mathcal{P}$.

Figure 20: The ideal functionality for asynchronous secure multi-party computation.

The functionality waits for a core set $\mathcal{CS}$ of $n - t$ input providers from the adversary and function-inputs of the respective parties in $\mathcal{CS}$. Upon receiving, the functionality substitutes 0 as the function-input for remaining parties and computes the output of the function f and generates a request-based delayed output for the parties. We note that the ideal-world adversary *cannot* delay sending $\mathcal{CS}$ to the functionality indefinitely. This is because in the real-world AMPC protocol, the set $\mathcal{CS}$ is eventually decided, irrespective of the message scheduling by the adversary. Also note that in the functionality, it is possible that for some $P_i \not\in \mathcal{CS}$, the input is reset to 0, even though P_i provided its input to the functionality. Looking ahead, this models the scenario that in the real-world protocol, even if P_i is able to provide its input, P_i's inclusion to $\mathcal{CS}$ will finally depend upon message scheduling, which is under adversarial control.

Our AMPC protocol Π_{AMPC} for securely realizing $\mathcal{F}_{\mathsf{AMPC}}$ is presented in Fig 21. The high level idea of the protocol is as follows. The parties start with a pre-processing phase where they generate complete t-sharing of c_M number of random multiplication-triples by calling the functionality $\mathcal{F}_{\mathsf{APrep}}$. Next the parties proceed to the input phase, for generating a complete t-sharing of their respective function-inputs. For this, each party acts as a dealer and picks a random degree-t polynomial whose constant term is the function-input of the party and sends the polynomial to $\mathcal{F}_{\mathsf{ACSS}}$ for distributing shares on the polynomial. To avoid an indefinite wait, the parties next try to agree on a common subset $\mathcal{CS}$ of $n - t$ input-providers, whose corresponding function-inputs are eventually completely t-shared. This is done by using the *agreement on a common-subset* (ACS) protocol of [13]. The idea used here is exactly the same as used to decide the set of $n - t$ multiplication-triple providers in the protocol Π_{APrep} by invoking n instances of $\mathcal{F}_{\mathsf{ABA}}$, where the j^{th} instance is used to decide whether P_j should be included in $\mathcal{CS}$. We use the notation $\mathsf{sid}_j \overset{\text{def}}{=} \mathsf{sid}||j$ to denote the j^{th} call to $\mathcal{F}_{\mathsf{ABA}}$ and $\mathcal{F}_{\mathsf{ACSS}}$ during the session id sid.

Upon the completion of input phase, the parties will know a common subset $\mathcal{CS}$ of $n-t$ input providers whose function-inputs will be completely t-shared. For the parties not in $\mathcal{CS}$, the parties take a default complete t-sharing of 0 and proceed to the circuit-evaluation phase, where each gate is jointly evaluated in a t-shared fashion. Namely, the parties maintain the BGW-invariant for each gate of the circuit (as discussed in the introduction). The evaluation of the linear gates is non-interactive, while for the multiplication gates the parties deploy the Beaver's method, using the completely t-shared multiplication-triples generated in the pre-processing phase. In order to publicly reconstruct the masked gate-inputs during the evaluation of multiplication gates, the parties exchange their respective shares, corresponding to the t-sharing of the masked gate-inputs. In the protocol, to denote the shares associated with the ℓ^{th} multiplication gate, we use the notation $\mathsf{sid}_\ell \overset{\text{def}}{=} \mathsf{sid}||\ell$. Finally once the function-output is completely t-shared, the parties publicly reconstruct the same.

Unlike the synchronous communication setting, in protocol Π_{AMPC}, all honest parties may not be reconstructing the function-output at the same "time" and different parties may be at different phases of the protocol, as the protocol is executed asynchronously. Consequently, a party P_i upon reconstructing the function-output cannot afford to terminate immediately, as its presence and participation might be needed for the completion of various phases of the protocol by other honest parties. Consequently, we include a termination phase in the protocol, whose code is executed concurrently by every party throughout the protocol. Namely in this phase, as soon as a party reconstructs the function-output, it broadcasts it by calling $\mathcal{F}_{\mathsf{ACast}}$. Then as soon as a party receives the same function-output broadcasted by at least $t + 1$ different parties, it considers it to be function-output and terminates the protocol. Since at least one of the $t + 1$ parties who broadcast a common function-output is *honest*, it is guaranteed that the honest parties terminate with the correct output.

Protocol Π_{AMPC}

Pre-Processing Phase

Upon receiving the input $(\mathsf{prep}, \mathsf{sid}, P_i)$ from the environment, proceed as follows:

1. Send $(\mathsf{prep}, \mathsf{sid}, P_i)$ to the functionality $\mathcal{F}_{\mathsf{APrep}}$.

2. Request output from $\mathcal{F}_{\mathsf{APrep}}$ until receiving $(\mathsf{triple\text{-}shares}, \mathsf{sid}, \{(u_i^{(\ell)}, v_i^{(\ell)}, w_i^{(\ell)})\}_{\ell=1,\ldots,c_M})$ from $\mathcal{F}_{\mathsf{APrep}}$.

Input Phase

Once the output from $\mathcal{F}_{\mathsf{APrep}}$ is received then upon receiving the input $(\mathsf{input}, \mathsf{sid}, x^{(i)})$ from the environment, proceed as follows:

- **Secret-sharing of the Inputs and Collecting Shares of Other Inputs**:
  1. Select a random degree-t polynomial $q_i(\cdot)$, such that $q_i(0) = x^{(i)}$. Send $(\mathsf{Dealer}, \mathsf{ACSS}, 1, \mathsf{sid}_i, q_i(\cdot))$ to $\mathcal{F}_{\mathsf{ACSS}}$.
  2. Request output from $\mathcal{F}_{\mathsf{ACSS}}$ with sid_j, for every $j \in \{1, \ldots, n\}$, until receiving $(P_j, \mathsf{Dealer}, 1, \mathsf{sid}_j, q_j(\alpha_i))$, where P_j acts as the dealer.

- **Selecting Common Input-Providers**: Initialize a set of local input-providers $\mathcal{LP}_i$ to $\emptyset$ and a set of global input-providers $\mathcal{GP}_i$ to $\emptyset$.
  1. If $(P_j, \mathsf{Dealer}, 1, \mathsf{sid}_j, q_j(\alpha_i))$ is received from $\mathcal{F}_{\mathsf{ACSS}}$, then include P_j to $\mathcal{LP}_i$.
  2. If $P_j \in \mathcal{LP}_i$, then send $(\mathsf{vote}, \mathsf{sid}_j, 1)$ to $\mathcal{F}_{\mathsf{ABA}}$.
  3. Request outputs from $\mathcal{F}_{\mathsf{ABA}}$ until receiving $(\mathsf{decide}, \mathsf{sid}_j, v_j)$ for every $j \in \{1, \ldots, n\}$.
  4. Include P_j to $\mathcal{GP}_i$ if $(\mathsf{decide}, \mathsf{sid}_j, v_j)$ is received from $\mathcal{F}_{\mathsf{ABA}}$, such that $v_j = 1$.
  5. If $|\mathcal{GP}_i| = n - t$, then stop executing step 1 under "**Selecting Common Input-Providers**" and send $(\mathsf{vote}, \mathsf{sid}_j, 0)$ to $\mathcal{F}_{\mathsf{ABA}}$, for every $j \in \{1, \ldots, n\}$ such that $P_j \notin \mathcal{GP}_i$.
  6. Once $(\mathsf{decide}, \mathsf{sid}_j, v_j)$ is received from $\mathcal{F}_{\mathsf{ABA}}$ for every $j \in \{1, \ldots, n\}$, set $\mathcal{CS} = \{P_j : v_j = 1\}$.
  7. Wait until receiving $(P_j, \mathsf{Dealer}, \mathsf{sid}_j, q_j(\alpha_i))$ from $\mathcal{F}_{\mathsf{ACSS}}$ for every $P_j \in \mathcal{CS}$. For every $P_j \notin \mathcal{CS}$, set 0 as the share of the circuit-input of P_j.

Circuit-Evaluation Phase

Wait until the Input Phase is completed, resulting in a set $\mathcal{CS}$ of $n - t$ input providers and the shares for each input of the input-providers in $\mathcal{CS}$ and the shares of the random multiplication triples are received. Then evaluate each gate g in the circuit according to the topological ordering as follows, depending upon the type of g.

- **Linear Gate**: If g is a linear gate with inputs x, y and output z, then set $z_i \overset{\text{def}}{=} x_i + y_i$ as the share corresponding to z. Here x_i and y_i are the shares corresponding to gate-inputs x and y respectively for P_i.
- **Multiplication Gate**: If g is the ℓ^{th} multiplication gate with inputs x, y and output z, where $\ell \in \{1, \ldots, c_M\}$, then do the following:
  1. Set $d_i^{(\ell)} \overset{\text{def}}{=} x_i - u_i^{(\ell)}$ and $e_i^{(\ell)} \overset{\text{def}}{=} y_i - v_i^{(\ell)}$, where x_i and y_i are the shares corresponding to x and y respectively and $(u_i^{(\ell)}, v_i^{(\ell)}, w_i^{(\ell)})$ are the shares of the ℓ^{th} multiplication-triple.
  2. Send $(\mathsf{mult}, \mathsf{sid}_\ell, (d_i^{(\ell)}, e_i^{(\ell)}))$ to each $P_j \in \mathcal{P}$.
  3. Keep receiving $(\mathsf{mult}, \mathsf{sid}_\ell, (d_j^{(\ell)}, e_j^{(\ell)}))$ from various parties $P_j \in \mathcal{P}$ and execute the steps of OEC, till $d^{(\ell)}$ and $e^{(\ell)}$ are reconstructed.
  4. Upon reconstructing $d^{(\ell)}$ and $e^{(\ell)}$, set $z_i \overset{\text{def}}{=} d^{(\ell)} \cdot e^{(\ell)} + d^{(\ell)} \cdot v_i^{(\ell)} + e^{(\ell)} \cdot u_i^{(\ell)} + w_i^{(\ell)}$ as the share of z.
- **Output Gate**: If g is the output gate with output y, then do the following:
  1. Send $(\mathsf{output}, \mathsf{sid}, y_i)$ to each $P_j \in \mathcal{P}$, where y_i is the share corresponding to y.
  2. Keep receiving $(\mathsf{output}, \mathsf{sid}, y_j)$ from various parties $P_j \in \mathcal{P}$ and execute the steps of OEC, till y is reconstructed.

Termination Phase

Concurrently execute the following steps during the protocol:

1. If the circuit output y is computed, then send $(\mathsf{sender}, \mathsf{ACast}, \mathsf{sid}_i, y)$ to $\mathcal{F}_{\mathsf{ACast}}$.
2. Request output from $\mathcal{F}_{\mathsf{ACast}}$ with sid_j, for every $j \in \{1, \ldots, n\}$, until receiving $(P_j, \mathsf{ACast}, \mathsf{sid}_j, \star)$, where P_j acts as the sender.

3. If $(P_j, \mathsf{ACast}, \mathsf{sid}_j, y)$ is received from $\mathcal{F}_{\mathsf{ACast}}$ corresponding to at least $t+1$ senders P_j, then output $(\mathsf{output}, \mathsf{sid}, (\mathcal{CS}, y))$ and terminate.

Figure 21: The AMPC protocol in the $(\mathcal{F}_{\mathsf{APrep}}, \mathcal{F}_{\mathsf{ACSS}}, \mathcal{F}_{\mathsf{ACast}}, \mathcal{F}_{\mathsf{ABA}})$-hybrid model. The above steps are executed by every $P_i \in \mathcal{P}$

Intuitively, protocol Π_{AMPC} is secure, as the function-inputs of the corrupt parties in $\mathcal{CS}$ will be independent of the inputs of the honest parties. This is because the inputs of the honest parties are shared through random degree-t univariate polynomials, which are communicated privately to $\mathcal{F}_{\mathsf{ACSS}}$. And corresponding to the polynomial of every honest party, the corrupt parties learn at most t shares, revealing no information about the function-inputs of the honest parties. Moreover, the privacy of the gate-inputs during the evaluation of multiplication gates is preserved, as the corresponding multiplication-triples are random and not known to the adversary. We next rigorously formalize this intuition by giving a formal security proof. Namely, we prove that the protocol Π_{AMPC} is perfectly-secure, if the parties have access to ideal functionalities $\mathcal{F}_{\mathsf{APrep}}, \mathcal{F}_{\mathsf{ACSS}}, \mathcal{F}_{\mathsf{ACast}}$ and $\mathcal{F}_{\mathsf{ABA}}$.

Theorem 7.1. *Protocol Π_{AMPC} UC-securely realizes the functionality $\mathcal{F}_{\mathsf{AMPC}}$ with perfect security in the $(\mathcal{F}_{\mathsf{APrep}}, \mathcal{F}_{\mathsf{ACSS}}, \mathcal{F}_{\mathsf{ACast}}, \mathcal{F}_{\mathsf{ABA}})$-hybrid model, in the presence of a static malicious adversary, corrupting at most $t < \frac{n}{3}$ parties. The protocol needs a communication of $\mathcal{O}(c_M \cdot n^2\kappa)$ bits.*

Proof. The communication complexity simply follows from the fact that in the protocol, the parties interact with each other only during the evaluation of the multiplication gates and during the reconstruction of shared circuit-output. During the evaluation of a multiplication gate, each honest party has to send 2 shares to every other honest party. To reconstruct the circuit output, each honest party has to send its share to every other party.

For security, let Adv be an arbitrary real-world adversary, attacking protocol Π_{AMPC} and let $\mathcal{Z}$ be an arbitrary environment. We show the existence of a simulator $\mathcal{S}_{\mathsf{AMPC}}$, such that for any set of corrupted parties $\mathcal{C}$ with $|\mathcal{C}| \leq t$ and for all inputs, the output of all parties and the adversary in an execution of the real protocol with Adv is identical to the output in an execution with $\mathcal{S}_{\mathsf{AMPC}}$ involving $\mathcal{F}_{\mathsf{AMPC}}$ in the ideal model. This further implies that $\mathcal{Z}$ cannot distinguish between the two executions. The steps of the simulator are given in Fig 22.

The high level idea of the simulator is as follows. During the simulated execution, the simulator itself performs the role of the ideal functionality $\mathcal{F}_{\mathsf{APrep}}, \mathcal{F}_{\mathsf{ACSS}}, \mathcal{F}_{\mathsf{ABA}}$ and $\mathcal{F}_{\mathsf{ACast}}$ whenever required. Performing the role of $\mathcal{F}_{\mathsf{APrep}}$ allows the simulator to learn the complete t-sharings of all the associated multiplication-triples. During the input phase, whenever Adv sends any polynomial to $\mathcal{F}_{\mathsf{ACSS}}$ on the behalf of a corrupt party, the simulator checks if it is a degree-t polynomial and accordingly records this polynomial on the behalf of the corrupt party. Notice that this allows the simulator to implicitly learn the function-input of the corresponding corrupt party. On the other hand, for the honest parties, the simulator picks arbitrary values as their function-inputs and simulates the steps of the honest parties as per the protocol. To select the common input-providers during the simulated execution, the simulator itself performs the role of $\mathcal{F}_{\mathsf{ABA}}$ and simulates the honest parties as per the steps of the protocol and $\mathcal{F}_{\mathsf{ABA}}$. This allows the simulator learn the common subset of input-providers $\mathcal{CS}$, which the simulator passes to the functionality $\mathcal{F}_{\mathsf{AMPC}}$. Notice that the function-inputs for each corrupt party in $\mathcal{CS}$ will be available with the simulator. This is because for every corrupt party P_j which makes it to $\mathcal{CS}$, at least one honest party P_i should participate with input 1 in the corresponding call to $\mathcal{F}_{\mathsf{ABA}}$, implying that the honest party P_i must have received its share from $\mathcal{F}_{\mathsf{ACSS}}$, corresponding to the degree-t polynomial which P_j sent to $\mathcal{F}_{\mathsf{ACSS}}$. Since in the simulation, the role of $\mathcal{F}_{\mathsf{ACSS}}$ is played by the simulator, it implies that the polynomial (and hence its constant term) provided by P_j will be known to the simulator. Hence along with $\mathcal{CS}$, the simulator can send the corresponding function-inputs of the corrupt parties in $\mathcal{CS}$ to $\mathcal{F}_{\mathsf{AMPC}}$. Upon receiving the function-output, the simulator simulates the steps of the honest parties for the gate evaluations as per the protocol. Finally, for the output gate, the simulator arbitrarily computes a complete t-sharing of the function-output y received from $\mathcal{F}_{\mathsf{AMPC}}$, which are consistent with the shares which corrupt parties hold for the output-gate sharing. Then on the behalf of the honest parties, the simulator sends the shares corresponding to the above complete t-sharing of y. This ensures that in the simulated execution, Adv learns function-output y. For the termination phase, the simulator sends y as the response from $\mathcal{F}_{\mathsf{ACast}}$ on the behalf of honest senders.

Simulator $\mathcal{S}_{\mathsf{AMPC}}$

$\mathcal{S}_{\mathsf{AMPC}}$ constructs virtual real-world honest parties and invokes the real-world adversary Adv. The simulator simulates the view of Adv, namely its communication with $\mathcal{Z}$, the messages sent by the honest parties and the interaction with various functionalities. In order to simulate $\mathcal{Z}$, the simulator $\mathcal{S}_{\mathsf{AMPC}}$ forwards every message it receives from $\mathcal{Z}$ to Adv and vice-versa. The simulator then simulates the various phases of the protocol as follows.

Pre-Processing Phase

Simulating the call to $\mathcal{F}_{\mathsf{APrep}}$: The simulator honestly simulates the steps of $\mathcal{F}_{\mathsf{APrep}}$, by itself playing the role of $\mathcal{F}_{\mathsf{APrep}}$. Namely, it receives from Adv, the shares corresponding to the parties in $\mathcal{C}$ for each of the multiplication-triples and then generates complete t-sharing of c_M random multiplication-triples $\{(\widetilde{u}^{(\ell)}, \widetilde{v}^{(\ell)}, \widetilde{w}^{(\ell)})\}_{\ell \in \{1,\ldots,c_M\}}$, consistent with the shares corresponding to the parties in $\mathcal{C}$, as provided by Adv. At the end of simulation of this phase, the simulator will know the entire vector of shares, corresponding to the complete t-sharing of all multiplication-triples.

Input Phase

- The simulator simulates the operations of the honest parties during the input phase, by randomly picking $\widetilde{x}^{(j)}$ as the input, for every $P_j \notin \mathcal{C}$. Namely, when Adv requests output from $\mathcal{F}_{\mathsf{ACSS}}$ (on the behalf of any party P_i in $\mathcal{C}$) with sid_j for any $P_j \notin \mathcal{C}$, the simulator responds with an output $(P_j, \mathsf{Dealer}, \mathsf{sid}_j, \widetilde{q}_j(\alpha_i))$ on the behalf of $\mathcal{F}_{\mathsf{ACSS}}$, where $\widetilde{q}_j(\cdot)$ is a random degree-t polynomial, such that $\widetilde{q}_j(0) = \widetilde{x}^{(j)}$.
- Whenever Adv sends $(P_i, \mathsf{Dealer}, \mathsf{sid}_i, q_i(\cdot))$ to $\mathcal{F}_{\mathsf{ACSS}}$ on the behalf of any $P_i \in \mathcal{C}$, the simulator performs the role of $\mathcal{F}_{\mathsf{ACSS}}$. Moreover, if $q_i(\cdot)$ is found to be a degree-t polynomial, then the simulator records the input $x^{(i)} \stackrel{\text{def}}{=} q_i(0)$ on the behalf of P_i.
- When the simulation reaches the "Selecting Common Input-Providers" stage, the simulator simulates the interface to $\mathcal{F}_{\mathsf{ABA}}$ to Adv, by itself performing the role of $\mathcal{F}_{\mathsf{ABA}}$. When the first honest party completes the simulated input phase, $\mathcal{S}_{\mathsf{AMPC}}$ learns the set $\mathcal{CS}$.

Interaction with $\mathcal{F}_{\mathsf{AMPC}}$: Once $\mathcal{S}_{\mathsf{AMPC}}$ learns $\mathcal{CS}$, it sends to $\mathcal{F}_{\mathsf{AMPC}}$ inputs values $x^{(i)}$ that it has recorded on the behalf of each $P_i \in \mathcal{C} \cap \mathcal{CS}$ and the set of input-providers $\mathcal{CS}$. Upon receiving back the output y from $\mathcal{F}_{\mathsf{AMPC}}$, the simulator starts the simulation of circuit-evaluation phase.

Circuit-Evaluation Phase

The simulator simulates the evaluation of each gate g in the circuit in topological order as follows:

- **Linear Gate**: Since this step involves local computation, the simulator does not have to simulate any messages on the behalf of the honest parties. The simulator locally adds the complete t-sharings corresponding to the gate-input and obtain the complete t-sharing corresponding to the gate-output.
- **Multiplication Gate**: If g is the ℓ^{th} multiplication gate in the circuit, then the simulator takes the complete t-sharing of the ℓ^{th} multiplication triple $(\widetilde{u}^{(\ell)}, \widetilde{v}^{(\ell)}, \widetilde{w}^{(\ell)})$ and computes the messages of the honest parties as per the steps of the protocol (by considering the complete t-sharing of the above multiplication-triple and complete t-sharing of the gate-inputs) and sends them to Adv on the behalf of the honest parties. Once the simulation of the gate-evaluation is done, the simulator will know the complete t-sharing corresponding to the gate-output.
- **Output Gate**: Let $[\widetilde{y}]_t = (\widetilde{y}_1, \ldots, \widetilde{y}_n)$ be the complete t-sharing, corresponding to the output gate, available with $\mathcal{S}_{\mathsf{AMPC}}$ during the simulated circuit-evaluation. The simulator computes a degree-t univariate polynomial, say $\widetilde{f}(\cdot)$, such that $\widetilde{f}(0) = y$ and $\widetilde{f}(\alpha_i) = \widetilde{y}_i$ holds for each[a] $P_i \in \mathcal{C}$. The simulator then computes the shares $y_i \stackrel{\text{def}}{=} \widetilde{f}(\alpha_i)$ for each $P_i \notin \mathcal{C}$ and sends these y_i values on the behalf of P_i to Adv.

Termination Phase

The simulator simulates the termination phase as follows:

- If Adv request an output from $\mathcal{F}_{\mathsf{ACast}}$ (on the behalf of any party in $\mathcal{C}$) for any sid^2_j, where $P_j \not\in \mathcal{C}$, then P_j responds with an output $(P_j, \mathsf{ACast}, \mathsf{sid}^2_j, y)$.

[a]If $|\mathcal{C}| = t$, then the polynomial is fixed, else the simulator picks $(t+1) - |\mathcal{C}|$ additional random points and interpolates $\widetilde{f}(\cdot)$,

Figure 22: Simulator for the protocol Π_{AMPC} where Adv corrupts the parties in set $\mathcal{C}$, where $|\mathcal{C}| \leq t$

We next prove a sequence of claims, which helps us to show that the joint distribution of the parties are identical in both the real-world, as well as ideal-world. During the proofs of these claims, we skip the session id sid. We first claim that in any execution of Π_{AMPC}, a set $\mathcal{CS}$ is eventually generated. This automatically implies that the honest parties eventually possess complete t-sharing of c_M multiplication-triples generated by $\mathcal{F}_{\mathsf{APrep}}$, as well as complete t-sharing of the inputs of the parties in $\mathcal{CS}$

Claim 7.2. In any execution of Π_{AMPC}, a set $\mathcal{CS}$ is eventually generated, such that for every $P_j \in \mathcal{CS}$, there exists some $x^{(j)}$ known to P_j, which is eventually completely t-shared.

Proof. The proof of this claim is similar to the proof of Claim 6.8 (except that instances of $\mathcal{F}_{\mathsf{AMTSS}}$ are replaced by $\mathcal{F}_{\mathsf{ACSS}}$) and so we skip the formal details. □

We next show that the view generated by $\mathcal{S}_{\mathsf{AMPC}}$ for Adv is identically distributed as Adv's view during the real execution of Π_{AMPC}.

Claim 7.3. The view of Adv in the simulated execution with $\mathcal{S}_{\mathsf{AMPC}}$ is identically distributed as the view of Adv in the real execution of Π_{AMPC}.

Proof. It is easy to see that the view of Adv during the pre-processing phase is identically distributed in both the executions. This is because in both the executions, Adv receives no messages from the honest parties and the steps of $\mathcal{F}_{\mathsf{APrep}}$ are executed by the simulator itself in the simulated execution. Namely, in both the executions, Adv's view consists of the shares of c_M random multiplication-triples, corresponding to the parties in $\mathcal{C}$. So let us fix these shares. Now conditioned on these shares, for the input phase, Adv learns the shares $\{q_j(\alpha_i)\}_{P_j \not\in \mathcal{C}, P_i \in \mathcal{C}}$ during the real execution, corresponding to the parties $P_j \not\in \mathcal{C}$, while in the simulated execution it learns the shares $\{\widetilde{q}_j(\alpha_i)\}_{P_j \not\in \mathcal{C}, P_i \in \mathcal{C}}$. The polynomial $q_j(\cdot)$ as well as $\widetilde{q}_j(\cdot)$ are both random degree-t univariate polynomials, where $q_j(0) = x^{(j)}$ (the real function-input of P_j) and $\widetilde{q}_j(0) = \widetilde{x}^{(j)}$ (the simulated function-input of P_j). Since $|\mathcal{C}| \leq t$, it follows from the properties of degree-t polynomials (see Lemma 2.4), that the distributions of the shares $\{q_j(\alpha_i)\}_{P_j \not\in \mathcal{C}, P_i \in \mathcal{C}}$ and $\{\widetilde{q}_j(\alpha_i)\}_{P_j \not\in \mathcal{C}, P_i \in \mathcal{C}}$ are identical and so let us fix these shares. Since the role of $\mathcal{F}_{\mathsf{ABA}}$ is played by the simulator itself, it follows easily that the view of Adv during the selection of the set $\mathcal{CS}$ is identically distributed in both the real as well as the simulated execution.

During the evaluation of linear gates, no communication is involved. During the evaluation of multiplication gates, in the simulated execution, the simulator will know the complete t-sharing associated with gate-inputs and also the complete t-sharing of the associated multiplication-triple. Hence the simulator correctly sends the shares corresponding to the $d^{(\ell)}$ and $e^{(\ell)}$ values as per the protocol on the behalf of the honest parties. This ensures that the Adv's view during the evaluation of multiplications gates is identically distributed, both in the real execution, as well as in the simulated execution.

For the output gate, the shares received by Adv in the real execution from the honest correspond to a complete t-sharing of the function-output y. From the steps of $\mathcal{S}_{\mathsf{AMPC}}$, it is easy to see that the same holds even in the simulated execution, as $\mathcal{S}_{\mathsf{AMPC}}$ sends to Adv shares corresponding to a complete t-sharing of y, which are consistent with the shares held by Adv. Hence the Adv's view is identically distributed in both the executions during the evaluation of output gate. Finally, it is easy to see that Adv's view is identically distributed in both the executions during the termination phase. This is because $\mathcal{S}_{\mathsf{AMPC}}$ plays the role of $\mathcal{F}_{\mathsf{ACast}}$ and also the role of honest parties, with their inputs for $\mathcal{F}_{\mathsf{ACast}}$ being the function-output y. $\square$

We next claim that conditioned on the view of Adv, the output of the honest parties are identically distributed in both the worlds.

Claim 7.4. Conditioned on the view of Adv, the output of the honest parties are identically distributed in the real execution of Π_{AMPC} involving Adv, as well as in the ideal execution involving $\mathcal{S}_{\mathsf{AMPC}}$ and $\mathcal{F}_{\mathsf{AMPC}}$.

Proof. Let View be an arbitrary view of Adv. And let $\mathcal{CS}$ be the set of input-providers as determined by View (from Claim 7.2, such a set $\mathcal{CS}$ is bound to exist). Moreover, according to View, for every $P_i \in \mathcal{CS}$, there exists some input $x^{(i)}$, such that the parties hold a complete t-sharing of $x^{(i)}$. Furthermore, from Claim 7.3, if $P_i \in \mathcal{C}$ then the corresponding complete t-sharing is included in View while for $P_i \notin \mathcal{C}$, the corresponding $x^{(i)}$ is uniformly distributed, conditioned on the shares of $x^{(i)}$ available with Adv as determined by View. Let us fix the $x^{(i)}$ values corresponding to the parties in $\mathcal{CS}$ and denote by $\vec{x}$ the vector of values $x^{(i)}$, where $x^{(i)} = 0$ if $P_i \notin \mathcal{CS}$.

It is easy to see that in the ideal-world, the output of the honest parties is y, where $y \overset{\text{def}}{=} f(\vec{x})$. This is because $\mathcal{S}_{\mathsf{AMPC}}$ provides the identity of $\mathcal{CS}$ along with the inputs $x^{(i)}$ corresponding to $P_i \in (\mathcal{CS} \cap \mathcal{C})$ to $\mathcal{F}_{\mathsf{AMPC}}$. We now show that the honest parties output y even in the real-world. For this, we argue that all the values during the circuit-evaluation phase of the protocol are correctly t-shared. Since the evaluation of linear gates need only local computation, it follows that the output of the linear gates will be completely t-shared. During the evaluation of a multiplication gate, the honest parties will hold a complete t-sharing of the corresponding $d^{(\ell)}$ and $e^{(\ell)}$ values, as during the pre-processing phase, all the multiplication-triples are generated in a completely t-shared fashion, since they are computed and distributed by $\mathcal{F}_{\mathsf{APrep}}$. During the reconstruction of $d^{(\ell)}$ and $e^{(\ell)}$, the honest parties send correct shares which are eventually delivered and hence any incorrect share sent by the corrupt parties can be error-corrected using OEC. This will automatically imply that the honest parties eventually hold a complete t-sharing of y and reconstruct it correctly. This is because even if the corrupt parties send incorrect shares during the reconstruction of y, they can be error-corrected using the steps of OEC. Moreover, in the termination phase, at most t *corrupt* parties can send *incorrect* function-output for broadcasting to $\mathcal{F}_{\mathsf{ACast}}$, thus guaranteeing that all honest parties output the correct function-output. $\square$

Finally from Claim 7.2 and Claim 7.4, we conclude that:

$$\left\{\mathrm{HYBRID}^{\mathcal{F}_{\mathsf{APrep}},\mathcal{F}_{\mathsf{ACSS}},\mathcal{F}_{\mathsf{ABA}},\mathcal{F}_{\mathsf{ACast}}}_{\Pi_{\mathsf{AMPC}},\mathsf{Adv}(z)}(\vec{x})\right\}_{z\in\{0,1\}^\star,\vec{x}\in\mathbb{F}^n} \equiv \left\{\mathrm{IDEAL}_{\mathcal{F}_{\mathsf{AMPC}},\mathcal{S}_{\mathsf{AMPC}}(z)}(\vec{x})\right\}_{z\in\{0,1\}^\star,\vec{x}\in\mathbb{F}^n}$$

holds, thus proving the theorem. $\square$

From Corollary 6.11, protocol Π_{APrep} securely realizes $\mathcal{F}_{\mathsf{APrep}}$ with statistical security in the $(\mathcal{F}_{\mathsf{Rand}}, \mathcal{F}_{\mathsf{ABA}})$-hybrid model, with a communication complexity of $\mathcal{O}(c_M \cdot n^4\kappa + n^5\kappa^2 + n^6)$ bits and involves n instances of $\mathcal{F}_{\mathsf{Rand}}$ and n instances of $\mathcal{F}_{\mathsf{ABA}}$. In protocol Π_{AMPC}, during the Input Phase, the functionality $\mathcal{F}_{\mathsf{ABA}}$ is invoked n times and each party shares a single field element (namely its function-input) by calling $\mathcal{F}_{\mathsf{ACSS}}$. By realizing $\mathcal{F}_{\mathsf{ACSS}}$ with protocol Π_{ACSS} with statistical security, the input phase will cost a communication of $\mathcal{O}(n^5\kappa^2 + n^6)$ bits and will involve n instances of $\mathcal{F}_{\mathsf{ABA}}$. During the termination phase, the instances of $\mathcal{F}_{\mathsf{ACast}}$ can be realized using Bracha's reliable broadcast protocol and this will cost a communication of $\mathcal{O}(n^3\kappa)$ bits.

While there are several well-known methods to securely realize the functionality $\mathcal{F}_{\mathsf{Rand}}$, a simple and standard method will be the following: each party $P_i \in \mathcal{P}$ picks a random value $r^{(i)}$ and generates a complete t-sharing of $r^{(i)}$ by picking a random degree-t polynomial $R^{(i)}(\cdot)$ with $R^{(i)}(0) = r^{(i)}$ and sending the polynomial $R^{(i)}(\cdot)$ to $\mathcal{F}_{\mathsf{ACSS}}$. The parties then use the same idea as used in the input phase of Π_{AMPC} to agree on a common subset $\mathcal{CS}$ of $n-t$ parties P_i, whose corresponding $r^{(i)}$ values are completely t-shared. This will require n instances of $\mathcal{F}_{\mathsf{ABA}}$. Finally the parties set $r \overset{\text{def}}{=} \sum_{P_i \in \mathcal{CS}} r^{(i)}$ and locally generates the complete t-sharing $[r]_t \overset{\text{def}}{=} \sum_{P_i \in \mathcal{CS}} [r^{(i)}]_t$, followed by publicly reconstructing the t-sharing $[r]_t$. It is straight forward to see that the resultant t-sharing $[r]_t$ is a completely random t-sharing with adversary having control over the shares of the corrupt parties in the sharing (a formal security proof that the protocol realizes $\mathcal{F}_{\mathsf{Rand}}$ in the $(\mathcal{F}_{\mathsf{ACSS}}, \mathcal{F}_{\mathsf{ABA}})$-hybrid model can be easily given. The simulation strategy will be similar as used to design the simulator for the protocol Π_{APrep}). By securely realizing $\mathcal{F}_{\mathsf{ACSS}}$ with protocol Π_{ACSS}, we get that $\mathcal{F}_{\mathsf{Rand}}$ can be securely realized in the $\mathcal{F}_{\mathsf{ABA}}$-hybrid model with a communication complexity of $\mathcal{O}(n^4\kappa^2 + n^5)$ bits, where n instances of $\mathcal{F}_{\mathsf{ABA}}$ are involved.

All the instances of $\mathcal{F}_{\mathsf{ABA}}$ throughout the protocol Π_{AMPC} can be securely realized by using the almost-surely terminating ABA protocols of [1, 6], whose communication complexity is polynomial in n and κ and independent of c_M.[14] So overall, we get the following theorem.

Theorem 7.5. *Protocol Π_{AMPC} UC-securely realizes the functionality $\mathcal{F}_{\mathsf{AMPC}}$ with statistical security in the $\mathcal{F}_{\mathsf{ABA}}$-hybrid model, in the presence of a static malicious adversary, corrupting at most $t < \frac{n}{3}$ parties. The protocol has communication complexity $\mathcal{O}(c_M \cdot n^4\kappa + n^5\kappa^2 + n^6)$ bits. In addition, $3n$ instances of ABA protocol are involved.*

8 Conclusion and Future Directions

In this paper, we presented a new protocol for statistically-secure asynchronous MPC (AMPC) with the optimal resilience of $t < n/3$, which significantly improve upon the communication complexity of the only known statistically-secure asynchronous AMPC protocol of [13] with $t < n/3$. To design our protocol, we presented a new and conceptually simpler instantiation of asynchronous complete secret-sharing (ACSS). Unlike the previous MPC protocols in the asynchronous communication setting, we formally proved the security of our AMPC protocol in the more rigorous UC framework.

Even though we have significantly improved over the communication complexity of the protocol of [13], our protocol still requires a higher communication, compared to the synchronous MPC protocols and AMPC protocols with perfect security. A natural question is to further investigate the communication complexity of optimally-resilient AMPC protocols with statistical security. Another interesting direction is to derive lower bounds on the communication complexity of statistically-secure AMPC protocols.

References

[1] I. Abraham, D. Dolev, and J. Y. Halpern. An Almost-surely Terminating Polynomial Protocol for Asynchronous Byzantine Agreement with Optimal Resilience. In *PODC*, pages 405–414. ACM, 2008.

[2] I. Abraham, D. Dolev, and G. Stern. Revisiting Asynchronous Fault Tolerant Computation with Optimal Resilience. In *PODC*, pages 139–148. ACM, 2020.

[3] B. Applebaum, E. Kachlon, and A. Patra. The Resiliency of MPC with Low Interaction: The Benefit of Making Errors (Extended Abstract). In *TCC*, volume 12551 of *Lecture Notes in Computer Science*, pages 562–594. Springer, 2020.

[14] Even though the security of the ABA protocols of [1, 6] are based on the property-based definition of ABA, a simulation-based proof can be presented using standard techniques.

[4] G. Asharov and Y. Lindell. A Full Proof of the BGW Protocol for Perfectly Secure Multiparty Computation. *J. Cryptology*, 30(1):58–151, 2017.

[5] Hagit Attiya and Jennifer Welch. *Distributed Computing: Fundamentals, Simulations, and Advanced Topics*, volume 19. John Wiley & Sons, 2004.

[6] L. Bangalore, A. Choudhury, and A. Patra. The Power of Shunning: Efficient Asynchronous Byzantine Agreement Revisited. *J. ACM*, 67(3):1–59, 2020.

[7] D. Beaver. Efficient Multiparty Protocols Using Circuit Randomization. In *CRYPTO*, volume 576 of *Lecture Notes in Computer Science*, pages 420–432. Springer, 1991.

[8] Z. Beerliová-Trubíniová and M. Hirt. Efficient Multi-party Computation with Dispute Control. In *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 305–328. Springer, 2006.

[9] Z. Beerliová-Trubíniová and M. Hirt. Simple and Efficient Perfectly-Secure Asynchronous MPC. In *ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 376–392. Springer, 2007.

[10] Z. Beerliová-Trubíniová and M. Hirt. Perfectly-Secure MPC with Linear Communication Complexity. In *TCC*, volume 4948 of *Lecture Notes in Computer Science*, pages 213–230. Springer, 2008.

[11] M. Ben-Or, R. Canetti, and O. Goldreich. Asynchronous Secure Computation. In *STOC*, pages 52–61. ACM, 1993.

[12] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract). In *STOC*, pages 1–10. ACM, 1988.

[13] M. Ben-Or, B. Kelmer, and T. Rabin. Asynchronous Secure Computations with Optimal Resilience (Extended Abstract). In *PODC*, pages 183–192. ACM, 1994.

[14] E. Ben-Sasson, S. Fehr, and R. Ostrovsky. Near-Linear Unconditionally-Secure Multiparty Computation with a Dishonest Minority. In *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 663–680. Springer, 2012.

[15] G. Bracha. An Asynchronous [(n-1)/3]-Resilient Consensus Protocol. In *PODC*, pages 154–162. ACM, 1984.

[16] R. Canetti. *Studies in Secure Multiparty Computation and Applications*. PhD thesis, Weizmann Institute, Israel, 1995.

[17] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *FOCS*, pages 136–145. IEEE Computer Society, 2001.

[18] R. Canetti. Universally Composable Security. *J. ACM*, 67(5):28:1–28:94, 2020.

[19] R. Canetti and T. Rabin. Fast Asynchronous Byzantine agreement with Optimal Resilience. In *STOC*, pages 42–51. ACM, 1993.

[20] A. Chandramouli, A. Choudhury, and A. Patra. A Survey on Perfectly-Secure Verifiable Secret-Sharing. *IACR Cryptol. ePrint Arch.*, page 445, 2021.

[21] D. Chaum, C. Crépeau, and I. Damgård. Multiparty Unconditionally Secure Protocols (Extended Abstract). In *STOC*, pages 11–19. ACM, 1988.

[22] A. Choudhury. Improving the Efficiency of Optimally-Resilient Statistically-Secure Asynchronous Multi-party Computation. In *INDOCRYPT*, volume 12578 of *Lecture Notes in Computer Science*, pages 810–831. Springer, 2020.

[23] A. Choudhury, M. Hirt, and A. Patra. Asynchronous Multiparty Computation with Linear Communication Complexity. In *DISC*, volume 8205 of *Lecture Notes in Computer Science*, pages 388–402. Springer, 2013.

[24] A. Choudhury and A. Patra. An Efficient Framework for Unconditionally Secure Multiparty Computation. *IEEE Trans. Information Theory*, 63(1):428–468, 2017.

[25] A. Choudhury, A. Patra, and D. Ravi. Round and Communication Efficient Unconditionally-Secure MPC with $t < n/3$ in Partially Synchronous Network. In *ICITS*, volume 10681 of *Lecture Notes in Computer Science*, pages 83–109. Springer, 2017.

[26] R. Cohen. Asynchronous Secure Multiparty Computation in Constant Time. In *PKC*, volume 9615 of *Lecture Notes in Computer Science*, pages 183–207. Springer, 2016.

[27] S. Coretti, J. A. Garay, M. Hirt, and V. Zikas. Constant-Round Asynchronous Multi-Party Computation Based on One-Way Functions. In *ASIACRYPT*, volume 10032 of *Lecture Notes in Computer Science*, pages 998–1021, 2016.

[28] R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin. Efficient Multiparty Computations Secure Against an Adaptive Adversary. In *EUROCRYPT*, volume 1592 of *Lecture Notes in Computer Science*, pages 311–326. Springer, 1999.

[29] R. Cramer, I. Damgård, and U. M. Maurer. General Secure Multi-party Computation from any Linear Secret-Sharing Scheme. In *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 316–334. Springer Verlag, 2000.

[30] I. Damgård and J. B. Nielsen. Scalable and Unconditionally Secure Multiparty Computation. In *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 572–590. Springer Verlag, 2007.

[31] I. Damgård, V. Pastro, N. P. Smart, and S. Zakarias. Multiparty Computation from Somewhat Homomorphic Encryption. In *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 643–662. Springer, 2012.

[32] M. J. Fischer, N. A. Lynch, and M. Paterson. Impossibility of Distributed Consensus with One Faulty Process. *J. ACM*, 32(2):374–382, 1985.

[33] M. Fitzi. *Generalized communication and security models in Byzantine agreement*. PhD thesis, ETH Zurich, Zürich, Switzerland, 2003.

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

[35] O. Goldreich, S. Micali, and A. Wigderson. How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. In *STOC*, pages 218–229. ACM, 1987.

[36] V. Goyal, Y. Liu, and Y. Song. Communication-Efficient Unconditional MPC with Guaranteed Output Delivery. In *CRYPTO*, volume 11693 of *Lecture Notes in Computer Science*, pages 85–114. Springer, 2019.

[37] V. Goyal, Y. Song, and C. Zhu. Guaranteed Output Delivery Comes Free in Honest Majority MPC. In *CRYPTO*, volume 12171 of *Lecture Notes in Computer Science*, pages 618–646. Springer, 2020.

[38] M. Hirt. *Multi-Party Computation: Efficient Protocols, General Adversaries, and Voting*. PhD thesis, ETH Zurich, September 2001. Reprint as vol. 3 of *ETH Series in Information Security and Cryptography*, ISBN 3-89649-747-2, Hartung-Gorre Verlag, Konstanz, 2001.

[39] J. Katz, U. Maurer, B. Tackmann, and V. Zikas. Universally Composable Synchronous Computation. In *TCCs*, volume 7785 of *Lecture Notes in Computer Science*, pages 477–498. Springer, 2013.

[40] Y. Lindell. Secure Multiparty Computation (MPC). Cryptology ePrint Archive, Report 2020/300, 2020.

[41] Nancy A Lynch. *Distributed algorithms*. Morgan Kaufmann, 1996.

[42] A. Patra. Studies on Verifiable Secret Sharing, Byzantine Agreement and Multiparty Computation. *IACR Cryptol. ePrint Arch.*, 2010:280, 2010.

[43] A. Patra, A. Choudhary, and C. Pandu Rangan. Efficient Statistical Asynchronous Verifiable Secret Sharing with Optimal Resilience. In *ICITS*, volume 5973 of *Lecture Notes in Computer Science*, pages 74–92. Springer, 2009.

[44] A. Patra, A. Choudhury, and C. Pandu Rangan. Asynchronous Byzantine Agreement with Optimal Resilience. *Distributed Comput.*, 27(2):111–146, 2014.

[45] A. Patra, A. Choudhury, and C. Pandu Rangan. Efficient Asynchronous Verifiable Secret Sharing and Multiparty Computation. *J. Cryptology*, 28(1):49–109, 2015.

[46] A. Patra and D. Ravi. On the Power of Hybrid Networks in Multi-Party Computation. *IEEE Trans. Information Theory*, 64(6):4207–4227, 2018.

[47] Marshall Pease, Robert Shostak, and Leslie Lamport. Reaching Agreement in the Presence of Faults. *Journal of the ACM (JACM)*, 27(2):228–234, 1980.

[48] T. Rabin. Robust Sharing of Secrets When the Dealer is Honest or Cheating. *J. ACM*, 41(6):1089–1109, 1994.

[49] T. Rabin and M. Ben-Or. Verifiable Secret Sharing and Multiparty Protocols with Honest Majority (Extended Abstract). In *STOC*, pages 73–85. ACM, 1989.

[50] A. Shamir. How to Share a Secret. *Commun. ACM*, 22(11):612–613, 1979.

[51] A. C. Yao. Protocols for Secure Computations (Extended Abstract). In *FOCS*, pages 160–164. IEEE Computer Society, 1982.

A Bracha's ACast Protocol

We first recall the formal steps of Bracha's ACast protocol from [16]. The protocol is presented in Fig 23.

Protocol Π_{ACast}

1. If $P_i = P_S$, then on input m, send $(\texttt{msg}, P_S, \mathsf{sender}, \mathsf{ACast}, \mathsf{sid}, m)$ to all the parties.
2. Upon receiving $(\texttt{msg}, P_S, \mathsf{sender}, \mathsf{ACast}, \mathsf{sid}, m)$ from P_S, send $(\texttt{echo}, P_S, \mathsf{sender}, \mathsf{ACast}, \mathsf{sid}, m)$ to all the parties.
3. Upon receiving $n - t$ messages $(\texttt{echo}, P_S, \mathsf{sender}, \mathsf{ACast}, \mathsf{sid}, m^\star)$ that agree on the value of $m^\star$, send $(\texttt{ready}, P_S, \mathsf{sender}, \mathsf{ACast}, \mathsf{sid}, m^\star)$ to all the parties.
4. Upon receiving $t + 1$ messages $(\texttt{ready}, P_S, \mathsf{sender}, \mathsf{ACast}, \mathsf{sid}, m^\star)$ that agree on the value of $m^\star$, send $(\texttt{ready}, P_S, \mathsf{sender}, \mathsf{ACast}, \mathsf{sid}, m^\star)$ to all the parties.
5. Upon receiving $n - t$ messages $(\texttt{ready}, P_S, \mathsf{sender}, \mathsf{ACast}, \mathsf{sid}, m^\star)$ that agree on the value of $m^\star$, output $(P_S, \mathsf{ACast}, \mathsf{sid}, m^\star)$.

Figure 23: Bracha's asynchronous reliable broadcast protocol for session id sid. The above code is executed by every $P_i \in \mathcal{P}$ including P_S.

Theorem A.1. *Protocol Π_{ACast} UC-securely realizes the ideal functionality $\mathcal{F}_{\mathsf{ACast}}$ with perfect security in the presence of any static malicious adversary, corrupting at most $t < n/3$. The protocol incurs a communication complexity of $\mathcal{O}(n^2 \cdot |m|)$ bits, where $|m|$ denotes the number of bits in the message m.*

Proof. The communication complexity trivially follows from the protocol steps, since each party needs to send m to every other party. For security, let Adv be an arbitrary real-world adversary, attacking the protocol in Fig 23 and let $\mathcal{Z}$ be an arbitrary environment. We show the existence of a simulator $\mathcal{S}_{\mathsf{ACast}}$, such that for any set of corrupted parties $\mathcal{C}$ with $|\mathcal{C}| \leq t$, the output of all parties and the adversary in an execution of Π_{ACast} with Adv is identical to the output in an execution with $\mathcal{S}_{\mathsf{ACast}}$ involving $\mathcal{F}_{\mathsf{ACast}}$ in the ideal model. This further implies that $\mathcal{Z}$ cannot distinguish between the two executions. The simulator constructs virtual real-world honest parties and invokes the real-world adversary Adv. The simulator simulates the environment and the honest parties towards Adv as follows, In order to simulate $\mathcal{Z}$, the simulator $\mathcal{S}_{\mathsf{ACast}}$ forwards every message it receives from $\mathcal{Z}$ to Adv and vice-versa. To simulate the honest parties, we consider the following two cases, depending upon whether the sender P_S is under the control of Adv or not.

Case I: P_S is honest. In this case, the simulator $\mathcal{S}_{\mathsf{ACast}}$ first interacts with the ideal functionality $\mathcal{F}_{\mathsf{ACast}}$ and receives the output m from the functionality. The simulator then plays the role of P_S with input m, as well as the role of the honest parties and interact with Adv as per the steps of Π_{ACast}.

It is easy to see that that view of Adv is identical, both in the real-world as well as in the ideal-world. This is because only P_S has the input in the protocol and in the ideal-world, $\mathcal{S}_{\mathsf{ACast}}$ plays the role of P_S as per Π_{ACast} after learning the input of P_S from $\mathcal{F}_{\mathsf{ACast}}$. Now conditioned on the view of Adv, we show that the outputs of the honest parties are identical in the real-world and ideal-world. So consider an arbitrary View of Adv. Conditioned on View, all honest parties eventually obtain a request-based delayed output m in the ideal-world, where m is the input of P_S as per View. We show that even in the real-world, all honest parties eventually output m. This is because all honest parties complete steps $2 - 5$ in the protocol, even if the corrupt parties do not send their messages, as there are at least $n - t$ honest parties, whose messages are eventually selected for delivery. Moreover, Adv may send at most t $\texttt{echo}$ messages for m', where $m' \neq m$, on the behalf of corrupt parties. Similarly, Adv may send at most t $\texttt{ready}$ messages for m', where $m' \neq m$, on the behalf of corrupt parties. Consequently, no honest party ever generates a $\texttt{ready}$ message for m', neither in step 3, nor in step 4. Thus the output of the honest parties are *identically* distributed in both the worlds. Consequently, in this case, we conclude that $\left\{\text{REAL}_{\Pi_{\mathsf{ACast}}, \mathsf{Adv}(z), \mathcal{Z}}(m)\right\}_{m,z \in \{0,1\}^\star} \equiv \left\{\text{IDEAL}_{\mathcal{F}_{\mathsf{ACast}}, \mathcal{S}_{\mathsf{ACast}}(z), \mathcal{Z}}(m)\right\}_{m,z \in \{0,1\}^\star}$ holds, thus completing the proof for the case when P_S is *honest*.

Case II: P_S is corrupt. In this case, the simulator $\mathcal{S}_{\mathsf{ACast}}$ first plays the role of the honest parties and interacts with Adv, as per the protocol Π_{ACast}. If in the simulated execution, $\mathcal{S}_{\mathsf{ACast}}$ finds that some *honest* party, say P_h, outputs $m^\star$, then $\mathcal{S}_{\mathsf{ACast}}$ interacts with the functionality $\mathcal{F}_{\mathsf{ACast}}$ by sending $m^\star$ as the input to $\mathcal{F}_{\mathsf{ACast}}$, on the behalf of P_S. Else $\mathcal{S}_{\mathsf{ACast}}$ does not provide any input to $\mathcal{F}_{\mathsf{ACast}}$ on the behalf of P_S.

It is easy to see that the view of Adv is identically distributed, both in the real-world as well as ideal-world. This is because only P_S has the input in the protocol which is under the control of Adv and $\mathcal{S}_{\mathsf{ACast}}$ plays the role of the honest parties, exactly as per the protocol Π_{ACast}. We next show that conditioned on the view of Adv, the output of the honest parties are identically distributed in both the worlds.

Let View be an arbitrary view of Adv, corresponding to some execution of Π_{ACast}. Now there are two possible case. If according to View, no honest party obtains an output during the execution of Π_{ACast}, then the honest parties do not obtain any output in the ideal-world as well. This is because in this case, the simulator $\mathcal{S}_{\mathsf{ACast}}$ does not provide any input on the behalf of P_S to $\mathcal{F}_{\mathsf{ACast}}$. On the other hand, consider the case when according to View, some *honest* party P_h outputs $m^\star$. In this case, in the ideal-world, all honest parties eventually obtain an output $m^\star$ since $\mathcal{S}_{\mathsf{ACast}}$ provides $m^\star$ as the input to $\mathcal{F}_{\mathsf{ACast}}$ on the behalf of P_S. We next show that even in the real-world, all honest parties eventually obtain the output $m^\star$, thus showing that the output of the honest parties are identically distributed.

Since P_h obtains the output $m^\star$, it implies that it receives $n-t$ `ready` messages for $m^\star$ during step 5 of the protocol. Let $\mathcal{H}$ be the set of *honest* parties whose `ready` messages are received by P_h during step 5. It is easy to see that $|\mathcal{H}| \geq t+1$. The `ready` messages of the parties in $\mathcal{H}$ are eventually delivered to every honest party and hence each honest party (including P_h) eventually executes step 4 and sends a `ready` message for $m^\star$. As there are at least $n-t$ honest parties, it follows that eventually $n-t$ `ready` messages for $m^\star$ are delivered to every honest party (irrespective of whether Adv sends all the required messages), consequently guaranteeing that all honest parties eventually obtain some output. To complete the proof of the claim, we show that this output is the same as $m^\star$.

On contrary, let $P_{h'}$ be another honest party, different from P_h, who outputs $m^{\star\star} \neq m^\star$. This implies that $P_{h'}$ received `ready` messages for $m^{\star\star}$ from at least $t+1$ *honest* parties during step 5 of the protocol. Now from the protocol steps, it follow that an honest party generates a `ready` message for some potential m, only if it receives $n-t$ `echo` messages for the m during step 3 or $t+1$ `ready` messages for the m (one of which has to come from an honest party) during step 4. So all in all, in order that $n-t$ `ready` messages are eventually generated for some potential m during step 5, it must be the case that some honest party has to receive $n-t$ `echo` messages for m during step 2 and generate a `ready` message for m. Since P_h receives $n-t$ `ready` messages for $m^\star$, some honest party must have received $n-t$ `echo` messages for $m^\star$, at most t of which could come from the corrupt parties. Similarly, since $P_{h'}$ receives $n-t$ `ready` messages for $m^{\star\star}$, some honest party must have received $n-t$ `echo` messages for $m^{\star\star}$. However, since $n-t > 2t$, it follows that in order that $n-t$ `echo` messages are produced for both $m^\star$ as well as $m^{\star\star}$, it must be the case that some honest party must have generated an `echo` message, both for $m^\star$, as well as $m^{\star\star}$ during step 2, which is impossible. This is because an honest party executes step 2 at most once and hence generates an `echo` message at most once.

Consequently, $\left\{\text{REAL}_{\Pi_{\mathsf{ACast}},\mathsf{Adv}(z)}(m)\right\}_{m,z\in\{0,1\}^\star} \equiv \left\{\text{IDEAL}_{\mathcal{F}_{\mathsf{ACast}},\mathcal{S}_{\mathsf{ACast}}(z)}(m)\right\}_{m,z\in\{0,1\}^\star}$ holds even in this case, thus completing the proof. $\square$