Guaranteed Termination Asynchronous Complete Secret Sharing with Lower Communication and Optimal Resilience

Ying Cai[1,2], Chengyi Qin[1,2], and Mingqiang Wang[1,2]

[1] School of Mathematics, Shandong University, Jinan, 250100, China
[2] State Key Laboratory of Cryptography and Digital Economy Security, Shandong University, Jinan, 250100, China wangmingqiang@sdu.edu.cn

Abstract. Asynchronous Complete Secret Sharing (ACSS) is a foundational module for asynchronous networks, playing a critical role in cryptography. It is essential for Asynchronous Secure Multi-Party Computation (AMPC) and, with termination, is widely applied in Validated Asynchronous Byzantine Agreement (VABA) and Asynchronous Distributed Key Generation (ADKG) to support secure distributed systems.
Currently, there are relatively few statistical secure ACSS protocols that can guarantee termination, and their communication complexity is relatively high. To reduce communication complexity, we propose a new multi-receiver signature scheme, ARICP, which supports linear operations on signatures. Leveraging the ARICP scheme and the properties of symmetric polynomials, we propose an ACSS protocol that ensures termination and optimal resilience ($t < n/3$) with $\mathcal{O}(n^2\kappa)$ bits per sharing. Compared with the best-known result of ACSS protocols that guarantee termination [CP23], the amortized communication complexity of our protocol is reduced by a factor of $\mathcal{O}(n)$.

Keywords: Asynchronous Complete Secret Sharing, · Termination · Optimal Resilience.

1 Introduction

Secret sharing (SS) is a foundational cryptographic technique that allows a secret to be divided into multiple shares and distributed among several parties. The original secret can only be reconstructed when a sufficient number of these shares are combined correctly. Within this framework, Verifiable Secret Sharing (VSS) [CGM+85, GHV22, CD24] extends the functionality of secret sharing, allowing parties to verify the validity of their shares even in the case of a potentially malicious dealer. Complete Secret Sharing (CSS) further enhances verifiable secret sharing by ensuring completeness.

Prior research on secret sharing, such as [AKP20, KKK08, FGG+06], has largely focused on synchronous settings, assuming fixed message delivery times. However, this approach fails to account for real-world network delays, which are unpredictable and unbounded. To bridge this gap, Asynchronous Complete

Secret Sharing (ACSS) protocols have been developed. ACSS accommodates variable delays, ensuring secure and reliable secret sharing in asynchronous networks.

ACSS is known to be the main module used in the design of asynchronous secure multiparty computation (AMPC) [GLS24]. Furthermore, ACSS protocols with termination have a wide range of applications, such as Validated Asynchronous Byzantine Agreement (VABA) and Asynchronous Distributed Key Generation (ADKG). Both VABA and ADKG are crucial for building secure distributed systems, especially ADKG, which is an important tool for supporting various distributed applications [CFG+23, SLM+23, GLL+22, CPS20, GJM+21, KMS20].

Current research on statistical ACSS protocols with termination is limited, with most focusing on protocols that only guarantee output. Moreover, protocols with termination tend to have high complexity. Under statistical security, the best-known result [CP23] with both termination and output has an amortized communication complexity of $\mathcal{O}(n^3\kappa)$ bits, while the best-known result [JLS24] with output but no termination achieves $\mathcal{O}(n\kappa)$ bits. A significant gap in communication complexity remains between protocols with and without termination. Addressing this gap is of paramount importance, as it remains an open problem that invites further exploration and innovative solutions.

1.1 Contributions

Our main contribution is an ACSS protocol with termination and optimal resilience $t < n/3$ that has a communication complexity of $\mathcal{O}(n^2\kappa)$-bit for sharing N degree-t Shamir shares, where κ is the size of a field element. Our protocol has lower communication complexity, which partially addresses the problem of the significant communication complexity gap between protocols with and without termination.

- ACSS protocols that guarantee termination are widely applicable in various scenarios, including AMPC, VABA, and ADKG. However, currently, there are relatively few statistical secure ACSS protocols that can guarantee termination. We propose an ACSS protocol that guarantees termination and has broad application prospects.
- We propose a new multi-receiver signature scheme, ARICP, which supports linear operations on signatures. Unlike the AICP scheme for a single receiver in [CP23], our ARICP scheme uses random shares to enable multiple revelations without exposing the underlying signature. This holds independent significance. It not only safeguards the secrecy of the signature but also broadens its applicability to a wider range of scenarios. This advancement is crucial as it significantly enhances the efficiency of the ACSS protocol.
- To guarantee termination, we use the ARICP scheme and the properties of symmetric polynomials. This allows honest parties to reconstruct the polynomial in the protocol without requiring assistance from the dealer or continuous online presence during the reconstruction phase.

- Compared with ACSS protocols that only guarantee output, ACSS protocols that guarantee termination typically have higher communication complexity. Compared with [CP23], which also guarantees termination, our protocol reduces the amortized communication complexity by a factor of $\mathcal{O}(n)$, thereby significantly narrowing the gap in communication complexity between ACSS protocols with and without termination.

We summarize some works under statistical security settings in the asynchronous setting, as shown in Table 1. Where N denotes the number of secrets, n is the number of parties, κ is the size of a field element, and t is the number of corrupted parties.The amortized communication complexity is the communication complexity divided by the number of secrets.

Table 1: Works under statistical security settings.

Work	Optimal Resilience	Communication Complexity	Amortized communication complexity ($N >> 1$)	Guarantee Termination
[JLS24]	$t < n/3$	$\mathcal{O}(Nn\kappa + n^{12}\kappa^2)$ bits	$\mathcal{O}(n\kappa)$ bits	No
[CP23]	$t < n/3$	$\mathcal{O}(Nn^3\kappa + n^4\kappa^2 + n^5)$ bits	$\mathcal{O}(n^3\kappa)$ bits	Yes
this work	$t < n/3$	$\mathcal{O}(Nn^2\kappa + n^4\kappa^2 + n^4 \log n)$ bits	$\mathcal{O}(n^2\kappa)$ bits	Yes

1.2 Related Works

In other security settings, the problem of designing communication-efficient ACSS protocols has also been extensively studied.

In the perfect security setting, it is known that $t < n/4$ is necessary, as demonstrated in the work [BCG93]. Subsequent research, such as [SR00, BH07, CHP13, PCR15, CP17], has progressively improved the communication efficiency of perfect ACSS in this setting. Among these, the best-known result [CP17] has achieved a linear communication complexity of $\mathcal{O}(n\kappa)$ bits per sharing.

In the computational security setting targeting $t < n/3$ corrupt parties, the work [AJM+23] relies on discrete logarithm assumption and pairings to achieve an ACSS protocol with linear communication complexity. Additionally, the work [AVY24] eliminates the need for a trusted setup, attempting to use the discrete logarithm assumption and the random oracle (RO) model to achieve an amortized communication complexity of $\mathcal{O}(n\kappa)$ bits. These works have not only made progress in communication efficiency but also successfully implemented ACSS protocols with termination.

2 Preliminaries

2.1 Notations

First, we introduce the basic notations.

For two integers $i \leq j$, the set $[i, j]$ represents the sequence $\{i, i+1, \dots, j\}$. Similarly, for any integer $n \in [N]$, the set $[N]$ denotes $\{1, 2, \dots, N\}$. The security parameter is represented by κ. The symbol $\mathbb{F}$ denotes a finite field, specifically $|\mathbb{F}| = 2^{\Theta(\kappa)}$.

We assume that n is a polynomial function of κ, i.e., $n = \text{poly}(\kappa)$. Additionally, $\text{negl}(\kappa)$ represents a negligible function of κ, meaning it is smaller than any $\text{poly}(\kappa)/2^{\kappa}$ for sufficiently large κ. All polynomials discussed are defined over a field $\mathbb{F}$. A polynomial of degree d is expressed as $f(x) = \beta_0 + \beta_1 x + \dots + \beta_d x^d$, with each coefficient β_i being an element of $\mathbb{F}$. A bivariate polynomial of degree (c, d) is given by $F(x, y) = \sum_{i=0}^{c} \sum_{j=0}^{d} r_{i,j} x^i y^j$, where each coefficient $r_{i,j}$ is also an element of $\mathbb{F}$.

2.2 Definitions

In this section, we briefly introduce the basic definitions involved in this paper.

Definition 1. *(**Shamir Secret Sharing**[Bla79,Sha79]) A degree-t (or $2t$) Shamir sharing is a method of distributing a secret value s among n parties in such a way that any $t+1$ (or $2t+1$) of them can reconstruct the original secret, but no fewer can. This is achieved by using a polynomial $f(x)$ of degree at most t (or $2t$) over a finite field $\mathbb{F}$, where the secret value s is the evaluation of the polynomial at a specific point α_0, and each party P_i holds a share s_i, which is the evaluation of the polynomial at another distinct point α_i, where $i \in [n]$.*

Definition 2. *(**Packed Secret Sharing**[FY92]) A degree-t packed secret sharing is a method of distributing $d+1$ secret value $\boldsymbol{s} = (s_0, \dots, s_d)$ among n parties in such a way that any $t+1$ of them can reconstruct the original secrets, but no fewer can. This is achieved by using a polynomial $f(x)$ of degree at most t over a finite field $\mathbb{F}$, where the secret value $s_j = f(\alpha_{-j})$ for each $j \in [d]$ and $s_0 = f(\alpha_0)$, and each party P_i holds a share $f(\alpha_i), i \in [n]$.*

The definitions of AVSS and ACSS will be provided in Appendix A.

Definition 3. *(**Symmetric Polynomial**[BS17, CTM+05]) A symmetric polynomial is a special type of multivariate polynomial. If a polynomial $P(x_1, \dots, x_n)$ in n variables remains unchanged under any permutation of its variables, then it is called a symmetric polynomial. Specifically, for any permutation σ of n elements, the following holds: $P(x_{\sigma(1)}, \dots, x_{\sigma(n)}) = P(x_1, \dots, x_n)$.*

2.3 The Security Model

In our work, we follow the security model in [JLS24, CP23].

The UC Framework. We utilize the UC framework, introduced by Canetti [Can01], to define the security standards for our protocols. These standards are based on the contrast between the real and ideal world models [Can00]. To put it simply, we consider a protocol secure if its execution in the real world can be simulated in the ideal world.

The above provides a high-level description of the UC framework. The complete formal details will be provided in Appendix A.1.

2.4 Agreement Primitives

We need an A-Cast protocol to support message broadcasting in asynchronous networks. According to [Bra84], broadcasting an ℓ-bit message requires $\mathcal{O}(n^2\ell)$ bits of communication.

Additional definitions of the consistency primitives underlying the protocol have been provided in Appendix A.

3 Technical Overview

In this section, we provide a high-level overview of the main techniques used in the construction of ACSS.

We utilize the AICP scheme introduced in [CP23] to propose a multi-receiver linear signature scheme ARICP. Using this ARICP scheme and the properties of symmetric polynomials, the ACSS protocol we construct can achieve termination.

3.1 Overview of AICP

In this part, we briefly introduce the AICP scheme and discuss how to construct an ACSS protocol using this scheme.

Overview of Asynchronous Information-Checking Protocol (AICP). The AICP concept, first introduced in [PCR09], operates as a signature protocol among a sender Sd, an intermediary I, and a receiver R. It allows Sd to send and authenticate a message to I, who forwards it to R. Upon receiving the message and its signature, R can verify its origin as Sd. Importantly, the amortized communication complexity of AICP is $\mathcal{O}(1)$ bits per message bit, matching the cost of direct transmission.

We encode the message sent by Sd to a vector $\mathbf{s} \in \mathbb{F}^L$. At a high level, the previous AICP [PCR09] is achieved by the following three steps.

- **Step 1:** The sender Sd samples a random polynomial $f(x)$ of degree-$L + t\kappa$, where the highest L coefficients form the vector $\mathbf{s}$. For each party P_i, Sd randomly selects κ elements $\alpha_1^i, \ldots, \alpha_\kappa^i$ from field $\mathbb{F}$ and computes their corresponding verification points $(\alpha_1^i, f(\alpha_1^i)), \ldots, (\alpha_\kappa^i, f(\alpha_\kappa^i))$ on $f(x)$. Then, Sd sends $f(x)$ to I and distributes the verification points to each party.

The polynomial $f(x)$ together with the verification points can be considered as a signature on the vector $\mathbf{s}$. However, the verification points sent by certain party (verifier) may not be consistent with the polynomial, so I needs to check for consistency to determine whether to accept the polynomial of the sender Sd.

- **Step 2:** After each verifier receives verification points from the sender Sd, he randomly divides into two sets of the same size and sends one of the sets of points to the intermediary I. When the intermediary I receives a polynomial $f(x)$ from the sender Sd, I waits to receive the sets of verification points from each verifier. I accepts the polynomial from Sd if the set of verification points of at least $2t+1$ different verifiers is on $f(x)$.

Since each verifier's points are randomly divided into two parts of the same quantity, if one part is correct, then the other part will be correct as well with a high probability.

- **Step 3:** When I accepts $f(x)$, I sends $f(x)$ to the receiver R. Each verifier sends R of the remaining set of verification points. If at least one of the verification points sent by at least $t+1$ verifiers is on the $f(x)$ received from I, then R accepts $f(x)$.

When R accepts $f(x)$, R assumes that the $f(x)$ received from I is indeed from the sender Sd. I can fudges the polynomial $f(x)$ only if I guesses correctly at some verification point by some honest verifier. However, the field $\mathbb{F}$ is large enough that the probability of I guessing correctly is negligible.

From AICP Towards AVSS. In [CP23], an AVSS protocol is constructed using the AICP scheme. The basic steps of the entire protocol are as follows:

- **Step 1:** The Dealer D first encodes the secret share $[s]_t$ into a bivariate polynomial $F(x, y)$ of degree-(t, t) with randomness, such that the secret s is stored in $F(\alpha_0, \alpha_0)$. The goal is for each party P_i to learn $f_i(x) = F(x, \alpha_i)$ and $g_i(y) = F(\alpha_i, y)$. First, D distributes column polynomial to each party. Then, each party P_i signs the values of the column polynomial and publicly broadcasts the issuance of signatures to D by broadcasting SC_i message.

When the set $\mathcal{M}$, consisting of $n-t$ parties, has each party send a signature to the dealer D, all parties move to the next phase. To ensure all parties agree on the set $\mathcal{M}$, the dealer D must broadcast this set. The column polynomials of the first $t+1$ honest parties in the set $\mathcal{M}$ fully determine a bivariate polynomial $F(x, y)$. When D is honest, this bivariate polynomial matches the one D possesses.

- **Step 2:** D sends row polynomial $f_j(x)$ to each P_j, $j \in [n]$. In addition, for each $P_i \in \mathcal{M}$, D acts as an intermediary for the AICP and sends the signature $g_i(\alpha_j) = f_j(\alpha_i)$ to P_j. Each P_j needs to verify that $f_j(\alpha_i)$ does come from sender P_i for all $P_i \in \mathcal{M}$. If true, P_j accepts $f_j(x)$ and broadcasts OK_{P_j}. The sharing phase ends when $2t+1$ parties accept their row polynomial. The existence of the signature guarantees that the t-th polynomial $f_j(x)$ assigned by D to P_j is indeed the j-th row polynomial, lying on the bivariate polynomial submitted by D.

Let $\mathcal{W}$ represent the set of parties that have accepted their row polynomials. Owing to the inherent properties of the AICP, for each honest party $P_i \in \mathcal{M}$ and each honest party $P_j \in \mathcal{W}$, the condition $g_i(\alpha_j) = f_j(\alpha_i)$ holds true.

- **Step 3:** To enable a party P_k to reconstruct the bivariate polynomial $F(x, y)$, each party $P_j \in \mathcal{W}$ sends their row polynomial $f_j(x)$ along with the signatures from the parties in the set $\mathcal{M}$. Provided that the signatures from the parties in the set $\mathcal{M}$ are verified as legitimate, P_k will accept $f_j(x)$. Given that $\mathcal{W}$ encompasses at least $t+1$ honest parties, P_k will consequently accumulate no fewer than $t + 1$ row polynomials, thereby facilitating the reconstruction of the entire bivariate polynomial $F(x, y)$.

The AVSS protocol requires communication of at least $\mathcal{O}(n^2\kappa)$ bits.

From AVSS Towards ACSS. However, in such a AVSS protocol, there is a possibility that a certain honest party may not obtain their share. To achieve completeness, it is necessary to execute n instances of the AVSS protocol. Thus, the protocol's total communication complexity is at least $\mathcal{O}(n^3\kappa)$ bits. In this ACSS protocol, honest parties are allowed to terminate the construction after getting their shares.

Asynchronous Packed Information-Checking Protocol (APICP) scheme in [JLS24] extends the AICP scheme to enable checking linear combinations of signatures and revealing valid signatures multiple times to different receivers. However, relying solely on the APICP scheme, a malicious dealer may withhold signature polynomials from some honest parties, failing to meet completeness. Therefore, [JLS24] uses the authentication tag from [BFO12] to remove dealer D from the completion phase. However, this keeps all honest parties online to generate tags for those still reconstructing, thereby failing to satisfy termination.

3.2 Our ACSS Protocol

We propose an ARICP scheme, which can be regarded as a signature scheme for multiple receivers and with linear signatures, and using this scheme and the properties of symmetric polynomials, we construct an ACSS protocol with termination and lower communication complexity.

Asynchronous Random Information-Checking Protocol (ARICP). The key improvement from AICP to APICP is the ability to allow multiple receivers to receive signatures from the same sender. This step greatly assists honest parties in determining the same bivariate polynomials, thus avoiding the need to execute n instances as in [CP23], effectively reducing communication complexity. However, since there are multiple receivers involved, it is necessary to ensure that the sender's signature does not reveal the sender's column polynomials. In [JLS24], although the additional authentication tag technique can make the protocol satisfy completeness, it requires each party to generate authentication tags online for other parties, thus the protocol does not guarantee termination. In this work, we use random polynomials as mask to protect the sender's column polynomial, enabling multiple receivers to receive the sender's signature. Even if the adversary corrupts t parties, they cannot know the sender's true column polynomial. When reconstructing the row and column polynomials, the properties of symmetric polynomials generated by random polynomials can ensure that the protocol guarantees both completeness and termination.

At a high level, our ARICP scheme proceeds through the following steps:

- **Step 1:** Each party receives the column polynomials, random row and random column polynomials from D as messages that need to be signed, and stores them into three high-degree polynomials respectively, selects some random points from the polynomials to send to each party (verifier), and sends the three high-degree polynomials $\{H^{(j)}(y)\}_{j\in[3]}$ to I. Meanwhile, each verifier randomly divides points into $n+1$ sets of the same size and sends the $(n+1)$-th set of points to I.
- **Step 2:** When I receives the high-degree polynomials from the sender Sd, then verifies whether the polynomials sent by Sd are consistent with the points from each verifier.When at least $2t+1$ verifiers' points pass the verification, the intermediary I computes the sum of the first polynomial and the second polynomial, and the sum of the second polynomial and the third polynomial to generate two signature polynomials. Then, the signature polynomials are revealed to the receiver R. Moreover, each verifier computes the remaining verification points in the same manner and sends a new set of verification points to R.
- **Step 3:** When R receives the signature polynomials sent by I, then verifies whether the polynomials sent by I are consistent with the points from each verifier. If at least one of the verification points sent by at least $t+1$ verifiers pass the verification, then R accepts the signature polynomials.

In this ARICP signature scheme, the signature polynomials that I reveals to R masks the column polynomials of Sd. Moreover, in multiple reveals, the intermediary I cannot tamper with the signature polynomial since R receives verification points from the verifier, i.e., if I sends signature polynomials that are not consistent with the polynomials relation of Sd, then R will not accept them. Finally, for the same sender Sd, each receiver R receives the same signature polynomials.

Towards ACSS. The ACSS protocol with termination that we want should satisfy the following three conditions:

1. When the dealer is honest, the protocol must guarantee that all the honest parties will eventually receive their shares of the degree-t Shamir secret sharing.
2. When the dealer is corrupted, either all honest parties eventually get a valid degree-t Shamir secret sharing or no honest party gets his share.
3. Regardless of whether the dealer is corrupted or honest, when one honest party completes the sharing, all honest parties will also complete the sharing, i.e., there is no need for the honest party to be online all the time, and the honest party can go offline when he gets his share.

After D sends the column polynomials, random row and random column polynomials to each party, the protocol proceeds with the following steps:

- **Step 1:** While performing the signature scheme, we design each party to verify whether the random row and column polynomials sent by the dealers are on the same bivariate polynomial. Thus, each party needs to send a share of random row and column polynomials to each party. Therefore, honest parties can verify whether the random row and column polynomials they receive are on the same bivariate polynomials by consistency checking. For this, D needs to construct $\mathcal{V}, \{\mathcal{V}_j\}_{j\in[n]}, \mathcal{M}$ sets, where each set size is denoted at least $2t+1$. The random row and column polynomials of the $2t+1$ parties in the set $\mathcal{M}$ are on the same bivariate polynomial, and D considers that the column polynomials received by the parties in the set $\mathcal{M}$ are all correct and valid.
- **Step 2:** So far, because of the ARICP scheme, it is realized that for the same sender Sd , each receiver R receives the same signature polynomials. There are at least $t+1$ honest parties in the set $\mathcal{M}$. All honest parties reconstruct two polynomials P, Q in terms of the signature polynomials of the parties in the set $\mathcal{M}$. Each polynomial Q is the sum of the random bivariate polynomials and the bivariate polynomials with the secret in it. Each polynomial P is a polynomial denoting a symmetric polynomial representing the sum of a random column polynomial and a random row polynomial. In reconstructing the row polynomials, each party checks whether the column polynomial messages sent by the party in the set $\mathcal{M}$ and the random column polynomial messages received earlier are on each polynomial Q. If so, the party accepts the column polynomial messages from that party. After accepting the column polynomial messages and the random column polynomial messages from at least $t+1$ parties, the party can reconstruct its own row polynomials and random row polynomials.
- **Step 3:** In reconstructing the column polynomials, since there are at least $t+1$ honest parties in the set $\mathcal{M}$, each party computes its own random column polynomials, using each symmetric polynomial P. Interpolation is started upon receipt of row and random row messages from $2t+1$ parties, and it is verified whether the random column polynomials formed by interpolation is the same as the one previously computed. If so, the party accepts the row, random row messages from those $2t+1$ parties. After accepting the row polynomial messages and random row polynomial messages from at least $2t+1$ parties, each party can reconstruct its own row polynomials and random row polynomials.

When the dealer is honest, the corrupted parties only know the sum of the random row and random column polynomials of the honest parties but not the specific polynomials, the honest parties can verify whether the messages sent by the corrupted parties are valid or not. When dealer is corrupted, by the ARICP scheme and the set $\mathcal{M}$, all honest parties get the bivariate polynomials P, Q. Because it is a definite value, it is difficult for the adversary to go to forge it, so all the honest parties can distinguish whether the messages sent by the corrupted party are valid or not.

Throughout the protocol, when the dealer is honest, all honest parties end up getting their share of degree-t Shamir secret sharing. When the dealer is corrupted, either all honest parties end up getting a valid degree-t Shamir secret sharing, or no honest party gets his share. In addition, when reconstructing row and column polynomials, on the one hand, each honest party can complete the reconstruction without the dealer's assistance. On the other hand, once they have sent the necessary messages for reconstruction, they can go offline. Therefore, the ACSS protocol we construct guarantees completeness and termination.

4 The Asynchronous Random Information-Checking Protocol (ARICP)

In this section, we present the construction of ARICP. Our constructed ARICP satisfies the linear homomorphism property and multiple revelation.

4.1 The Functionality of ARICP

We present the functionality of ARICP in Fig. 1. ARICP has two phases, the initialization phase and the revealing phase. Where the revelation phase can be called at most n times. Before each invocation of the revelation phase, we assume that all parties agree to the $(\mathsf{Request}, \mathsf{ARICP}, R)$ request. This will be guaranteed by our ACSS protocol.

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

For fixed sender Sd and intermediary I:

Initialization Phase: Init$(n, (s^{(1)}, s^{(2)}, s^{(3)}))$

- The trusted party receives the identities of corrupted parties $\mathcal{C} \subset \mathcal{P}$.
- Upon receiving $(Sd, I, n, \mathsf{ARICP}, (s^{(1)}, s^{(2)}, s^{(3)}))$ from Sd , the trusted party sends a request-based delayed output $(s^{(1)}, s^{(2)}, s^{(3)})$ to I and set $\texttt{count} = n$.

Revelation Phase: Rev$(R, (s^{(1)}, s^{(2)}, s^{(3)}))$

- Each time the trusted party receives a request $(\mathsf{Request}, \mathsf{ARICP}, R)$, the trusted party does the following things and replace $\texttt{count}$ by $\texttt{count} - 1$, until $\texttt{count} = 1$.
  a) If $I \in \mathcal{C}$,the trusted party waits to receive the instruction from $\mathcal{S}$.
     - If $\mathcal{S}$ sends Ignore, the trusted party does nothing.
     - If $\mathcal{S}$ sends $\mathsf{Proceed}$, if $Sd \in \mathcal{C}$, the trusted party waits to receive $s', \overline{s'}$ from $\mathcal{S}$ and sends a request-based delayed output $s', \overline{s'}$ to the receiver

R. Otherwise, the trusted party sends a request-based delayed output $(s, \overline{s}) = (s^{(1)} + s^{(2)}, s^{(2)} + s^{(3)})$ to the receiver R.

b) If $I \notin \mathcal{C}$, the trusted party sends a request-based delayed output $(s, \overline{s}) = (s^{(1)} + s^{(2)}, s^{(2)} + s^{(3)})$ to the receiver R.

- If R is honest, R outputs the result received from the trusted party. Corrupted parties may output anything they want.

Fig. 1: Ideal functionality for ARICP

4.2 The Instantiation of ARICP

The protocol Π_{ARICP} consists of two sub-protocols Π_{Init} and Π_{Rev}. Each of the two sub-protocols is related to the initialization phase and the revelation phase.

Protocol Π_{Init}

Protocol Init$(Sd, I, n, 3, L, \kappa)$

Parameter: The sender Sd, intermediary I, revelation times n, 3 secret vectors, length of secret vector L and security parameter κ.

1. Sd receives his input $(s^{(1)}, s^{(2)}, s^{(3)})$ from the environment.
2. For each verifier $P_i \in \mathcal{P}$, Sd selects $(n+1)\kappa$ random elements in $\mathbb{F}$, denoted by $\alpha^i_1, \ldots, \alpha^i_{(n+1)\kappa}$,where $i \in [n]$.
3. Sd selects a random degree-$(L + t(n+1)\kappa)$ polynomial $H^{(m)}(y)$ whose the L highest coefficients are elements in $s^{(m)}$, where $m \in [3]$.
4. Sd sends $H^{(1)}(y), H^{(2)}(y), H^{(3)}(y)$ to I and verification points $z^i_j = (\alpha^i_j, H^{(1)}(\alpha^i_j), H^{(2)}(\alpha^i_j), H^{(3)}(\alpha^i_j))$ to each verifier P_i for $j \in [(n+1)\kappa]$.
5. Each verifier P_i randomly divides z^i_j into $n+1$ disjoints sets, where each set is of size κ, denoted by $Z^i_1, \ldots, Z^i_{(n+1)}$.
6. Each verifier P_i sends Z^i_{n+1} to I.
7. Upon receiving $H^{(1)}(y), H^{(2)}(y), H^{(3)}(y)$ from Sd, I checks whether these polynomials are all of degree-$(L + t(n+1)\kappa)$. If true, I does the following:
   a) Upon receiving Z^i_{n+1} from P_i and $|Z^i_{n+1}| = \kappa$, I checks whether the verification points in Z^i_{n+1} are all consistent with $H^{(1)}(y), H^{(2)}(y), H^{(3)}(y)$.
   b) If for at least $2t+1$ verifiers, the above condition is satisfied, then I accepts the signature $H^{(1)}(y), H^{(2)}(y), H^{(3)}(y)$.

Fig. 2: The protocol of the Π_{Init}

Π_{Init} is shown in Fig. 2. In this protocol, the sender Sd stores the vectors $s^{(1)}, s^{(2)}, s^{(3)}$ in three randomly sampled high-degree polynomials, which is equivalent to a signature. Then, Sd randomly samples verification points on these polynomials and distributes them to all parties, which are equivalent to verifiers. At the same time, Sd sends polynomials to the intermediary I. Upon receiving verification points from Sd, each verifier selects a set of verification points with set size κ and sends this set to the intermediary I. When I receives these polynomials (signatures), I verifies whether there are at least $2t+1$ verifiers whose sets of verification points are on these polynomials. If true, I accepts these polynomials, which is equivalent to I accepting the vectors over these polynomials. The communication complexity of this protocol is $\mathcal{O}(L\kappa + n^2\kappa^2)$ bits.

Protocol Π_{Rev}

Protocol Rev$(I, R, \texttt{count}, n, L, \kappa)$

Parameter: The intermediary I, receiver R, counter $\texttt{count}$, revelation times n, secret vector length L and security parameter κ.

1. If I accepts the signature $H^{(1)}(y), H^{(2)}(y), H^{(3)}(y)$ from Sd, he sends this $H(y) = H^{(1)}(y) + H^{(2)}(y), \overline{H}(y) = H^{(2)}(y) + H^{(3)}(y)$ to R.
2. Each verifier P_i sends $\tilde{Z}^i_{\texttt{count}} = \{(\alpha^i_j, H^{(1)}(\alpha^i_j) + H^{(2)}(\alpha^i_j), H^{(2)}(\alpha^i_j) + H^{(3)}(\alpha^i_j))\}_{z^i_j \in Z^i_{\texttt{count}}}$ to R, so $|\tilde{Z}^i_{\texttt{count}}| = \kappa$.
3. R dose the following:
   a) Upon receiving $\tilde{Z}^i_{count}$ from P_i and $|\tilde{Z}^i_{\texttt{count}}| = \kappa$, R checks whether there exists at least one point $(\alpha^i_j, \beta^i_{j,1}, \beta^i_{j,2}) \in \tilde{Z}^i_{\texttt{count}}$ satisfied $H(\alpha^i_j) = \beta^i_{j,1}$ and $\overline{H}(\alpha^i_j) = \beta^i_{j,2}$.
   b) If for at least $t+1$ verifiers, the above condition is satisfied, then R accepts the signature $H(y), \overline{H}(y)$. Let $s, \overline{s}$ respectively be the L highest coefficients of $H(y), \overline{H}(y)$, R outputs $s, \overline{s}$.

Fig. 3: The protocol of the Π_{Rev}

Π_{Rev} is shown in Fig. 3. In this protocol, the intermediary I sends the sum of the first polynomial and the second polynomial, and the sum of the second polynomial and the third polynomial to the receiver R. Each verifier processes the verification points in the same additive manner and sends κ verification points to the receiver R. R then checks if the points of at least $t+1$ verifiers satisfy the requirement that there exists at least one verification point on the polynomials. If true, R accepts the polynomial received from I. This is equivalent to R believes that the polynomials received from I are from the sender Sd. The protocol is executed at most n times in each ARICP instance. It is equivalent to each receiver R receives the same signature polynomials from I for the same sender Sd. The communication complexity of this protocol is $\mathcal{O}(L\kappa + n^2\kappa^2)$ bits.

Protocol Π_{ARICP}

Parameter: The sender Sd, intermediary I, counter $\texttt{count}$, revelation times n, secret vector length L and security parameter κ.

Initialization Phase: Init(n)

- All parties participate in Π_{Init}.

- All parties initialize a counter `count` $= n$.

Revelation Phase: Rev(R)

- All parties participate in $\mathit{\Pi}_{\mathsf{Rev}}$ with inputs R, and replace `count` by `count` $-$ 1.

Fig. 4: The protocol of the $\mathit{\Pi}_{\mathsf{ARICP}}$

$\mathit{\Pi}_{\mathsf{ARICP}}$ is shown in Fig. 4. Its communication complexity is $\mathcal{O}(Ln\kappa + n^3\kappa^2)$ bits. we will give a detailed complexity analysis in Appendix B.

Theorem 1. *The protocol $\mathit{\Pi}_{\mathsf{ARICP}}$ realizes $\mathcal{F}_{\mathsf{ARICP}}$ with statistically security and $\mathcal{O}(Ln\kappa + n^3\kappa^2)$-bit communication.*

We prove the theorem in Appendix B.

5 The Asynchronous Secret Sharing Protocol (ACSS)

In this section, we give our ACSS protocol $\mathit{\Pi}_{\mathsf{ACSS}}$ with termination and its communication complexity.

5.1 The Functionality of ACSS

In this part, We give the functionality of ACSS in Fig. 5. The Dealer is allowed to send degree-t sharing polynomials $q^{(1)}(x), \ldots, q^{(N)}(x)$ to $\mathcal{F}_{\mathsf{ACSS}}$. If the polynomials are valid, $\mathcal{F}_{\mathsf{ACSS}}$ will distribute the shares to the honest parties.

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

Public Input: $(\alpha_0, \alpha_1, \ldots, \alpha_n), N$.
Upon receiving $(\mathsf{Dealer}, \mathsf{ACSS}, q^{(1)}(\cdot), q^{(2)}(\cdot), \ldots, q^{(N)}(\cdot))$ from $D \in \mathcal{P}$, the trusted party does the following:

1. If all the polynomials $q^{(1)}(\cdot), q^{(2)}(\cdot), \ldots, q^{(N)}(\cdot)$ are degree-t, the trusted party sends an request-based delayed output $q^{(1)}(\alpha_i), q^{(2)}(\alpha_i), \ldots, q^{(N)}(\alpha_i)$ to each party $P_i \in \mathcal{P}$.
2. If any of the $q^{(1)}(\cdot), q^{(2)}(\cdot), \ldots, q^{(N)}(\cdot)$ is not a degree-t polynomial, the trusted party does nothing.

Fig. 5: Ideal functionality for ACSS

5.2 The Instantiation of ACSS with termination

The ACSS protocol $\mathit{\Pi}_{\mathsf{ACSS}}$ consists of three sub-protocols $\mathit{\Pi}_{\mathsf{Sh}}$,$\mathit{\Pi}_{\mathsf{Ver}}$ and $\mathit{\Pi}_{\mathsf{Comp}}$. Each of these three sub-protocols is related to Sharing Phase, Verification Phase and Completion Phase.

Protocol Π_{Sh}

Parameter: All parties agree on distinct public elements $\alpha_{-t}, \ldots, \alpha_{-1}, \alpha_0, \alpha_1, \ldots, \alpha_n$ in field $\mathbb{F}$, L' polynomial. Let $\mathcal{F}_{\mathsf{ARICP}}(Sd, I)$ denote $\mathcal{F}_{\mathsf{ARICP}}$ with sender Sd and intermediary I, and let L denote the vector length in $\mathcal{F}_{\mathsf{ARICP}}$.
Initialization: Let $L' = L/n$, be the number of polynomials packed in a single vector. Let $N = (t+1)L'$, be the number of input polynomials. Let n will be the number of revelations in $\mathcal{F}_{\mathsf{ARICP}}$.

Sharing Phase

Distributing column polynomials and random polynomials: Upon receiving the input $q^{(1)}(\cdot), \ldots, q^{(N)}(\cdot)$ from environment, for each $k \in [N]$, the dealer D compute $\lceil k/(t+1) \rceil = \ell$, $r \equiv k-1 \pmod{t+1}$, so $\ell \in [L']$ and $r \in [0, t]$. Dealer does the following:
For each $\ell \in [L']$:
1. D selects a random degree-$(t, 2t)$ bivariate polynomial $F^{(\ell)}(x, y)$, s.t. $F^{(\ell)}(x, \alpha_{-r}) = q^{(\ell)}(x)$ for each $r \in [0, t]$.
2. D selects a random degree-$(t, 2t)$ bivariate polynomial $R^{(\ell)}(x, y)$ and $R^{(\ell)}(x, y) \neq F^{(\ell)}(x, y)$.
3. For $i \in [n]$, D sends the column polynomials $g_i^{(\ell)}(y) = F^{(\ell)}(\alpha_i, y)$, random column polynomials $z_i^{(\ell)}(y) = F^{(\ell)}(\alpha_i, y)$ and random row polynomials $w_i^{(\ell)}(x) = F^{(\ell)}(x, \alpha_i)$ to each $P_i \in \mathcal{P}$.

Signing the polynomials and distributing share of random polynomials: Each $P_i \in \mathcal{P}$ does the following:
For each $\ell \in [L']$:
1. Wait to receive $\{g_i^{(\ell)}, z_i^{(\ell)}(y), w_i^{\ell}(x)\}_{\ell \in [L']}$ from D.
2. If $\{g_i^{(\ell)}, z_i^{(\ell)}(y)\}_{\ell \in [L']}$ are all of degree-$2t$ and $\{w_i^{(\ell)}(x)\}$ are all of degree-t, P_i broadcasts OK_i.
3. Send $z_i^{(\ell)}(\alpha_j), w_i^{(\ell)}(\alpha_j)$ to P_j, each $j \in [n]$.
4. Set $sign_{i,1}^{(\ell)} = (g_i^{(\ell)}(\alpha_1), \ldots, g_i^{(\ell)}(\alpha_n))$, $sign_{i,2}^{(\ell)} = (z_i^{(\ell)}(\alpha_1), \ldots, z_i^{(\ell)}(\alpha_n))$and $sign_{i,3}^{(\ell)} = (w_i^{(\ell)}(\alpha_1), \ldots, w_i^{(\ell)}(\alpha_n))$. Then the vector $sign_{i,m}^* = (sign_{i,m}^{(1)}, \ldots, sign_{i,m}^{(L')})$ for each $m \in [3]$ is a vector of size $n \cdot L' = L$.
5. Send $(\mathsf{Init}, \mathsf{ARICP}, n, (sign_{i,1}^*, sign_{i,2}^*, sign_{i,3}^*))$ to $\mathcal{F}_{\mathsf{ARICP}}(P_i, D)$.
6. Upon receiving $z_j^{(\ell)}(\alpha_i), w_j^{(\ell)}(\alpha_i)$ from P_j, check whether $z_j^{(\ell)}(\alpha_i) = w_i^{(\ell)}(\alpha_j)$ and $w_j^{(\ell)}(\alpha_i) = z_i^{(\ell)}(\alpha_j)$, if both true, P_i broadcasts $iAMj$.

Preparing the $\mathcal{V}_j$ sets and $\mathcal{V}$ set: D does the following:
For each $\ell \in [L']$:
1. Initialize sets $\mathcal{V}_j, \mathcal{V}$ to $\emptyset$, each $j \in [n]$.
2. Upon receiving $iAMj$ from P_i, include P_i to $\mathcal{V}_j$ and broadcast $\mathcal{V}_j$ when $|\mathcal{V}_j| \geq 2t+1$.
3. Upon $|\mathcal{V}_j| \geq 2t+1$, include P_j to $\mathcal{V}$ and broadcast $\mathcal{V}$ when $|\mathcal{V}| \geq 2t+1$.

Identifying the column polynomials and random polynomials: D does the following:

For each $\ell \in [L']$:

1. Initialize set $\mathcal{M}$ to $\emptyset$, each $j \in [n]$.
2. If $|\mathcal{M}| < 2t+1$, include P_i into the set $\mathcal{M}$ when:
   a) Received OK_i from P_i and $P_i \in \mathcal{V}$.
   b) Upon receiving $(P_i, \mathsf{ARICP}, (sign^*_{i,1}, sign^*_{i,2}, sign^*_{i,3}))$ from $\mathcal{F}_{\mathsf{ARICP}}(P_i, D)$, respectively verify $sign^{(\ell)}_{i,1}(y) = F^{(\ell)}(\alpha_i, y)$, $sign^{(\ell)}_{i,2}(y) = R^{(\ell)}(\alpha_i, y)$ and $sign^{(\ell)}_{i,3}(y) = R^{(\ell)}(y, \alpha_i)$ are true.
3. Broadcast $\mathcal{M}$ when $|\mathcal{M}| = 2t+1$.

Verify the set $\mathcal{M}$: Each parties moves to the next phase if the following conditions are met:

For each $\ell \in [L']$:

1. Received the set $\mathcal{M}$ from D and $|\mathcal{M}| = 2t+1$.
2. Received OK_h from all $P_h \in \mathcal{M}$.
3. Received $\mathcal{V}$ from D, $|\mathcal{V}| \geq 2t+1$ and $\mathcal{M} \subseteq \mathcal{V}$.
4. Received $\mathcal{V}_j$ from all $P_j \in \mathcal{V}$ and $iAMj$ from all $P_i \in \mathcal{V}_j$, each $|\mathcal{V}_j| \geq 2t+1$.

Fig. 6: The protocol of the Π_{Sh}

The Sharing Phase Π_{Sh} is shown in Fig. 6. During this phase, the dealer D first encodes $t+1$ degree-t polynomials into each degree-$(t, 2t)$ bivariate polynomial $F(x, y)$. Then, D randomly selects degree-$(t, 2t)$ bivariate polynomials $R(x, y)$ and each $R(x, y)$ is not equal to $F(x, y)$. D distributes degree-$2t$ column polynomials of bivariate polynomials $F(x, y)$, degree-$2t$ random column polynomials and degree-t random row polynomials of random bivariate polynomials $R(x, y)$ to all parties. Each party's secret share is $g(\alpha_{-t}), \ldots, g(\alpha_0)$, where g is his own column polynomial. Each party P_i uses the column polynomials, random column polynomials and random row polynomials received from D to execute $\mathcal{F}_{\mathsf{ARICP}}$, generate signatures and send them to $\mathcal{F}_{\mathsf{ARICP}}(P_i, D)$. At the same time, each party sends corresponding share of the random row polynomials and random column polynomials to each of the other parties. If the share of random row and column polynomials received by each party P_i from party P_j matches the random column and row polynomials received by itself from D, then P_i broadcasts $iAMj$ (i.e., P_i accepts the random information from P_j). If a party receives polynomials sent by D and the random row and random column polynomials of that party are on the same bivariate polynomial, then D includes that party to the set $\mathcal{M}$. When the size of the set $\mathcal{M}$ reaches $2t+1$, D broadcasts the set $\mathcal{M}$. All parties receive the set $\mathcal{M}$ set and go to verify it. Disregarding the communication complexity of Π_{ARICP}, the communication complexity of Π_{Sh} is $\mathcal{O}(Ln\kappa + n^4 \log n)$ bits.

The Verification Phase Π_{Ver} is shown in Fig. 7. In this phase, when party P_i receives polynomials from D, two bivariate polynomials can be obtained by using the two signature polynomials revealed by the parties in revelation phase in the set $\mathcal{M}$. Each P_i can verify that the sum of the column polynomials and random column polynomials, and the sum of the random column polynomials and random row polynomials that he receives from D correspond to the two bivariate polynomials. Disregarding the communication complexity of Π_{ARICP}, the communication complexity of Π_{Ver} is 0 bit.

Protocol Π_{Ver}

Verification Phase

For each $P_i \in \mathcal{P}$:

1. Upon receiving $\{g_i^{(\ell)}, z_i^{(\ell)}(y), w_i^{\ell}(x)\}_{\ell \in [L']}$ from D, $\{g_i^{(\ell)}, z_i^{(\ell)}(y)\}_{\ell \in [L']}$ are all of degree-$2t$ and $\{w_i^{(\ell)}(x)\}$ are all of degree-t.
2. Each party sends $(\mathsf{Request}, P_i, \mathsf{ARICP})$ from $\mathcal{F}_{\mathsf{ARICP}}(P_h, D)$, for all $P_h \in \mathcal{M}$.
3. Upon receiving $sign_h^*, \overline{sign_h}^*$ from $\mathcal{F}_{\mathsf{ARICP}}(P_h, D)$, for all $P_h \in \mathcal{M}$, P_i accepts the sum of $g_i^{(\ell)}(y)_{\ell \in [L']}$, $z^{(\ell)}(y)$ and $w^{(\ell)}(y)$ if the following hold for each $\ell \in [L']$.
   a) For each $P_h \in \mathcal{M}$, respectively parse $sign_h^*, \overline{sign_h}^*$ into $sign_h^{(\ell)}(y), \overline{sign_h}^{(\ell)}(y)$, each $sign_h^{(\ell)}(y), \overline{sign_h}^{(\ell)}(y)$ is a degree-$2t$ polynomial.
   b) There exists a degree-$(t, 2t)$ polynomial $Q^{(\ell)}(x, y)$ and degree-$(2t, 2t)$ symmetric polynomial $P^{(\ell)}(x, y)$ s.t. $Q^{(\ell)}(\alpha_h, y) = sign_h^{(\ell)}(y)$ and $P^{(\ell)}(\alpha_h, y) = \overline{sign_h}^{(\ell)}(y)$ for all $P_h \in \mathcal{M}$.
   c) Check whether $Q^{(\ell)}(\alpha_i, y) = g_i^{(\ell)}(y) + z_i^{(\ell)}(y)$ and $P^{(\ell)}(\alpha_i, y) = z_i^{(\ell)}(y) + w_i^{(\ell)}(y)$

Fig. 7: The protocol of the Π_{Ver}

The Completion Phase Π_{Comp} is shown in Fig. 8. In this phase, each party P_i first sends shares of its own column polynomials to the corresponding parties. Then, P_i uses the sum of the column polynomials and the random column polynomials to verify whether the column polynomial message sent by the parties in the set $\mathcal{M}$ satisfies the sum. If so, P_i accepts the column polynomial message. Since the set $\mathcal{M}$ consists of at least $t+1$ honest parties, when D is honest, there are two bivariate polynomials determined by the signature polynomials of the parties in the set $\mathcal{M}$ that necessarily correspond to the polynomials chosen by D. When D is corrupted, there are two bivariate polynomials determined by the two signature polynomials of the first $t+1$ honest parties in the set $\mathcal{M}$. The sum relation is fixed, and in the Sharing Phase, P_i has received the share of random column polynomials from the other parties, and the other parties must have to send correct and fixed column polynomial messages for P_i to accept them. After P_i accepts at least $t+1$ column polynomial messages from the parties in the set $\mathcal{M}$, he can interpolate to get his row polynomials. Of course, at the same time he can also interpolate to get his random row polynomials.

Protocol Π_{Comp}

Completion Phase

Reconstructing row polynomials:

For each $P_i \in \mathcal{P}$, $\ell \in [L']$ do:

1. Each P_i sends $g_i^{(\ell)}(\alpha_j)$ to P_j, $j \in [n]$.
2. Upon P_i receiving $g_j^{(\ell)}(\alpha_i)$ from P_j, P_i does the following:
   a) Check whether $P_j \in \mathcal{M}$.
   b) Check whether $g_j^{(\ell)}(\alpha_i) + z_j^{(\ell)}(\alpha_i) = Q^{(\ell)}(\alpha_j, \alpha_i)$, where $z_j^{(\ell)}(\alpha_i)$ received from P_j during the Sharing Phase.
   c) If both the above conditions are satisfied, P_i gets his row polynomials $f_i^{(\ell)}(x)$ and random row polynomials $w_i^{(\ell)}(x)$.

Reconstructing column polynomials:

For each $P_i \in \mathcal{P}$, $\ell \in [L']$ do:

1. Upon each P_i reconstructing $f_i^{(\ell)}(x)$ and $w_i^{(\ell)}(x)$, P_i sends $f_i^{(\ell)}(\alpha_j)$ and $w_i^{(\ell)}(\alpha_j)$ to P_j, $j \in [n]$.
2. Computing $\tilde{z}_i^{(\ell)}(y) = P^{(\ell)}(\alpha_i, y) - w_i^{(\ell)}(y)$.
3. Upon receiving $2t+1$ different P_j's $f_j^{(\ell)}(\alpha_i)$ and $w_j^{(\ell)}(\alpha_i)$, P_i does the following:
   a) Simultaneously interpolate two degree-$2t$ polynomials $g_i^{(\ell)}(y)$ and $z_i^{(\ell)}(y)$.
   b) Check whether $z_i^{(\ell)}(y) = \tilde{z}_i^{(\ell)}(y)$, if true then accepts $g_i^{(\ell)}(y)$ and $z_i^{(\ell)}(y)$, else return a).
4. Upon accepting $z_i^{(\ell)}(y)$, P_i outputs $\{g_i^{(\ell)}(\alpha_r)\}_{\ell\in[L'],r\in[-t,0]}$

Fig. 8: The protocol of the Π_{Comp}

Then, P_i sends its own share of row polynomials and random row polynomials to the corresponding parties and uses the bivariate symmetric polynomials (the sum of the random row polynomials and the random column polynomials) obtained in the previous phases to compute the random column polynomials that it ought to get. When P_i receives row and random row messages from $2t+1$ different parties, he goes to interpolate to compute his own column polynomials and random column polynomials. If the random column polynomials obtained by interpolation at this moment are equal to the random column polynomials that P_i is expected to compute itself, then P_i accepts these column polynomials and random column polynomials. Otherwise, P_i continues with interpolation calculations. Because the sum of the column polynomials and the random column polynomials is deterministic and valid, when P_i reconstructs the correct random column polynomials, he will surely get the correct column polynomials at the same time. Finally, all honest parties can get the corresponding row and column polynomials with each row and column polynomial on the same bivariate polynomial. In addition, after each honest party sends messages and reconstructs its own row and column polynomials, it does not need to help other honest parties online to reconstruct the polynomials. The communication complexity of the protocol Π_{Comp} is $\mathcal{O}(L\kappa)$ bits.

Based on the above construction, we obtain the following theorem.

Theorem 2. *Let κ denote the security parameter. For a finite field $\mathbb{F}$ of size $2^{\Theta(\kappa)}$, N degree-t Shamir sharings, a static malicious adversary and $t < n/3$ corrupted parties, there exists a protocol Π_{ACSS} UC-securely realizes $\mathcal{F}_{\mathsf{ACSS}}$ in the $\mathcal{F}_{\mathsf{ARICP}}$-hybrid model with statistical security. The protocol requires a communication of $\mathcal{O}(Nn^2\kappa + n^4\kappa^2 + n^4 \log n)$ bits.*

The security proof of this theorem will be given in Appendix C.2, and the detailed complexity analysis of our Π_{ACSS} will be provided in Appendix C.3.

Acknowledgments. This study was funded by the National Key Research and Development Program of China (grant number 2021YFA1000600), the National Natural Science Foundation of China (grant number 12441104), and Key Research and Development Program of Shandong province (grant number 2022CXGC020101). The study also received support from the General Fund (grant number 11140050070800).

References

[AJM+23] Ittai Abraham, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, and Gilad Stern. Bingo: Adaptivity and asynchrony in verifiable secret sharing and distributed key generation. CRYPTO 2023, 39-70, 2023.

[AKP20] Benny Applebaum, Eliran Kachlon, and Arpita Patra. The round complexity of perfect MPC with active security and optimal resiliency. In FOCS. IEEE, 1277-1284, 2020.

[AVY24] Nicolas Alhaddad, Mayank Varia, and Ziling Yang. Haven++: Batched and Packed Dual-Threshold Asynchronous Complete Secret Sharing with Applications. Cryptology ePrint Archive, 2024.

[BCG93] Michael Ben-Or, Ran Canetti, and Oded Goldreich. Asynchronous secure computation. STOC 1993. 52-61, 1993.

[Bea92] Donald Beaver. Efficient multiparty protocols using circuit randomization. CRYPTO 1991, 420-432, 1992.

[BFO12] Eli Ben-Sasson, Serge Fehr, and Rafail Ostrovsky. Near-linear unconditionally-secure multiparty computation with a dishonest minority. CRYPTO 2012, 663-680, 2012.

[BH07] Zuzana Beerliová-Trubíniová and Martin Hirt. Simple and efficient perfectly-secure asynchronous MPC. ASIACRYPT 2007, 376-392, 2007.

[Bla79] George Robert Blakley. Safeguarding cryptographic keys. In AFIPS National Computer Conference. IEEE, 313-317, 1979.

[Bra84] Gabriel Bracha. An asynchronou $[(n-1)/3]$-resilient consensus protocol. PODC 1984, 154-162, 1984.

[BS17] Ben Blum-Smith and Samuel Coskey. The fundamental theorem on symmetric polynomials: history's first whiff of Galois theory. The College Mathematics Journal, 48(1):18-29, 2017.

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. J. Cryptol., 13(1):143-202, 2000.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In FOCS. IEEE, 136-145, 2001.

[Can20] Ran Canetti. Universally composable security. J. ACM, 67(5):28:1-28:94, 2020.

[Can96] Ran Canetti. Studies in secure multiparty computation and applications. Scientific Council of The Weizmann Institute of Science, 1996.

[CD24] Ignacio Cascudo and Bernardo David. Publicly verifiable secret sharing over class groups and applications to DKG and YOSO. EUROCRYPT 2024, 216-248, 2024.

[CFG+23] Ran Cohen, Pouyan Forghani, Juan Garay, Rutvik Patel, and Vassilis Zikas. Concurrent asynchronous byzantine agreement in expected-constant rounds, revisited. TCC 2023, 422-451, 2023.

[CGM+85] Benny Choc, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In FOCS. IEEE, 383-395, 1985.

[CHP13] Ashish Choudhury, Martin Hirt, and Arpita Patra. Asynchronous multiparty computation with linear communication complexity. DISC 2013, 388-40, 2013.

[CP17] Ashish Choudhury and Arpita Patra. An efficient framework for unconditionally secure multiparty computation. IEEE Trans. Inf. Theory, 63(1):428-468, 2017.

[CP23] Ashish Choudhury and Arpita Patra. On the communication efficiency of statistically secure asynchronous MPC with optimal resilience. J. Cryptol., 36(2):13, 2023.

[CPS20] T.-H. Hubert Chan, Rafael Pass, and Elaine Shi. Sublinear-round byzantine agreement under corrupt majority. PKC 2020, 246-265, 2020.

[CTM+05] John N. Chiasson, Leon M. Tolbert, Keith J. McKenzie, and Zhong Du. Elimination of harmonics in a multilevel converter using the theory of symmetric polynomials and resultants. IEEE Trans. Control Syst. Technol., 13(2):216-223, 2005.

[FGG+06] Matthias Fitzi, Juan Garay, Shyamnath Gollakota, C. Pandu Rangan, and Kannan Srinathan. Round-optimal and efficient verifiable secret sharing. TCC 2006, vol 3876, 329-342, 2006.

[FY92] Matthew Franklin, Moti Yung. Communication complexity of secure computation. STOC 1992, 699-710, 1992.

[GHV22] Craig Gentry, Shai Halevi, and Vadim Lyubashevsky. Practical non-interactive publicly verifiable secret sharing with thousands of parties. EUROCRYPT 2022, 458-487, 2022.

[GJM+21] Kobi Gurkan, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, Gilad Stern, and Alin Tomescu. Aggregatable Distributed Key Generation. EUROCRYPT 2021, 147-176, 2021.

[GLL+22] Yingzi Gao, Yuan Lu, Zhenliang Lu, Qiang Tang, Jing Xu, and Zhenfeng Zhang. Efficient asynchronous byzantine agreement without private setups. ICDCS 2022, 246-257, 2022.

[GLS24] Vipul Goyal, Chen-Da Liu-Zhang, and Yifan Song. Towards achieving asynchronous MPC with linear communication and optimal resilience. CRYPTO 2024, 170-206, 2024.

[JLS24] Xiaoyu Ji, Junru Li, and Yifan Song. Linear-communication asynchronous complete secret sharing with optimal resilience. CRYPTO 2024, 418-453, 2024.

[KKK08] Jonathan Katz, Chiu-Yuen Koo, and Ranjit Kumaresan. Improving the round complexity of VSS in point-to-point networks. ICALP 2008, vol 5126, 499-510, 2008.

[KMS20] Eleftherios Kokoris Kogias, Dahlia Malkhi, and Alexander Spiegelman. Asynchronous Distributed Key Generation for Computationally-Secure Randomness, Consensus, and Threshold Signatures. CCS 2020, 1751-1767, 2020.

[PCR09] Arpita Patra, Ashish Choudhary, and C. Pandu Rangan. Efficient statistical asynchronous verifiable secret sharing with optimal resilience. ICITS 2009, vol 5973, 74-92, 2009.

[PCR15] Arpita Patra, Ashish Choudhury, and C. Pandu Rangan. Efficient asynchronous verifiable secret sharing and multiparty computation. J. Cryptol., 28(1):49-109, 2015.

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

[SLM+23] Shravan Srinivasan, Julian Loss, Giulio Malavolta, Kartik Nayak, Charalampos Papamanthou, and Sri Aravinda Krishnan Thyagarajan. Transparent batchable time-lock puzzles and applications to byzantine consensus. PKC 2023, 554-584, 2023.

[SR00] K. Srinathan and C. Pandu Rangan. Efficient asynchronous secure multiparty distributed computation. INDOCRYPT 2000, 117-129, 2000.

A Additional Definitions and Security Model Details

Here, we first provide the formal definitions of AVSS and ACSS.

Definition 4. *(**Asynchronous Verifiable Secret Sharing (AVSS)** [BCG93, Can96]). Let (Sh,Rec) be a pair of protocols in which a dealer $D \in \mathcal{P}$ shares a secret $s \in \mathbb{F}$ using Sh. We say that (Sh,Rec) is a t-resilient AVSS scheme with n parties if the following hold for every possible $\mathcal{A}_t$:*

- ***Termination:***
  - *a) If D is honest then each honest party will eventually terminate the protocol Sh.*
  - *b) If some honest party has terminated the protocol Sh, then irrespective of the behavior of D, each honest party will eventually terminate Sh.*
  - *c) If all honest parties have terminated Sh and all honest parties invoke the protocol Rec, then each honest party will eventually terminate Rec.*
- ***Correctness:***
  - *a) If D is honest then each honest party upon completing the protocol Rec, outputs the shared secret s.*
  - *b) If D is corrupted and some honest party has terminated Sh, then there exists a fixed $\overline{s} \in \mathbb{F}$, such that each honest party upon completing Rec, will output $\overline{s}$, irrespective of the behavior of the corrupted parties. This property is also known as the strong commitment property and we often say that D is committed to $\overline{s}$.*
- ***Secrecy:** If D is honest then the adversary's view during the protocol Sh reveals no information about s in the information theoretic sense. In other words, the adversary's view is identically distributed for all different values of s.*

Definition 5. *(**Asynchronous Complete Secret Sharing (ACSS)**) In the ACSS protocol, it specifically adheres to the Completeness Property as outlined in the following:*

- ***Completeness**[PCR09]: If any honest party successfully completes Sh, there must exist a degree-t polynomial P such that $P(0)$ equals the initial secret s, and each honest party will ultimately obtain their respective share s_i, which is precisely the value of the polynomial P evaluated at i, i.e., $s_i = P(i)$. Notably, when the dealer is also honest, this s is the very secret that the dealer initially intended to share.*

We give the formal definitions of agreement primitives here.

A-cast is an asynchronous broadcast protocol that allows a designated sender to transmit a message to all parties. If the sender is honest, all honest parties will eventually receive the message. If the sender is corrupted, any honest party that terminates with a message ensures that all other honest parties will also terminate with the same message, albeit with some delay. Bracha's implementation of A-cast is efficient and works well even when the sender's behavior is unpredictable.We state the formal functionality of A-Cast [Bra84] in Fig. 9. From [Bra84], broadcasting an ℓ-bit message requires $\mathcal{O}(n^2\ell)$-bit communication.

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

Upon receiving $(\mathsf{sender}, \mathsf{ACast}, m)$ from $P_j \in \mathcal{P}$, the trusted party sends an request-based delayed output (P_j, ACast, m) to each $P_i \in \mathcal{P}$.

Fig. 9: Ideal functionality for ACast

A.1 Additional Descriptions of the UC Security Model

Here, we provide the complete formal details of the UC Security Model.

Real World vs. Ideal World. In the context of secure multiparty computation, the ideal world and real world are two contrasting models used to define and evaluate the security of protocols.

In the ideal world, a trusted party or ideal functionality $\mathcal{F}$ performs computations on behalf of n dummy parties. The environment $\mathcal{Z}$ interacts with these parties and an ideal adversary $\mathcal{S}$, who cannot observe or delay communications between honest parties and $\mathcal{F}$. The ideal functionality models the desired behavior of the computation, receiving inputs only from the parties and $\mathcal{S}$, and providing outputs to them. The adversary can instruct the functionality to delay output delivery to honest parties by ignoring their requests, but this delay is limited to a polynomial number of times, ensuring eventual delivery. This model simplifies the analysis by assuming a trusted entity can perform secure computations without the risks of an actual adversary.

In contrast, the real world involves a set of n parties $(P_1, \ldots, P_n)$, an adversary $\mathcal{A}$, and an environment $\mathcal{Z}$. The environment provides inputs to honest parties, receives their outputs, and communicates with the adversary. The adversary is fully malicious, capable of corrupting up to t parties (where $t < n/3$), and controlling their behavior completely. Parties and the adversary are modeled as interactive Turing machines, with the protocol proceeding through a sequence of activations. Parties can perform local computations, output, or send messages, while the adversary can send messages on behalf of corrupted parties. They have access to a network of point-to-point (P2P) asynchronous and secure channels, with the adversary deciding the arrival time of messages. The protocol is considered secure if it can simulate the ideal world's outputs in this adversarial environment, despite the presence of a malicious adversary and without relying on a trusted party.

The security of a protocol is thus proven by demonstrating that the real-world execution, even in the presence of a malicious adversary and without a trusted party, can emulate the secure and efficient computation that would occur in the ideal world with a trusted entity.

Perfect Security vs. Statistical Security. We assert that a protocol Π achieves t-security with respect to functionality $\mathcal{F}$ if, for every adversary $\mathcal{A}$, there exists a simulator $\mathcal{S}$ within the ideal model that ensures the following: regardless of the adversary's control over up to t parties and for any given environment $\mathcal{Z}$, it satisfies that the output distributions of the real-world execution and the ideal-world execution are consistent. We refer to a protocol Π as statistically t-securely realizing function $\mathcal{F}$ if, for any adversary $\mathcal{A}$, there is a simulator $\mathcal{S}$ within the ideal model such that for any adversary managing up to t parties and any environment $\mathcal{Z}$, it satisfies that the output distributions of the real-world execution and the ideal-world execution are statistically consistent.(i.e., the total variation distance between the two distributions is no more than $\epsilon = negl(\kappa)$).

The Hybrid Model. In the $\mathcal{G}$-hybrid model, a protocol operates similarly to the real world, with the exception that participants have the ability to utilize an ideal functionality $\mathcal{G}$ for a particular task. Throughout the execution of the protocol, interactions with $\mathcal{G}$ are conducted as they would be in an ideal setting. The UC framework ensures that the ideal functionality within a hybrid model can be substituted by a protocol that securely realizes $\mathcal{G}$ in the UC model. This guarantee is provided by the composition theorem detailed in [Can01, Can20].

Theorem 3. *([Can01, Can20]) 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 $\mathcal{G}$.*

Hybrid Arguments. Hybrid arguments are a method of demonstrating that the outputs of a protocol executed in the real world and simulated in the ideal world are the same or very close to each other. This approach smoothly transitions from the real-world execution to the ideal-world simulation by constructing a series of step-by-step transition scenarios (i.e., "Hybrids"). If there is no significant difference between the outputs of two neighboring blends at each step of the transition, then we can assume that the outputs of the real world and the ideal world are the same, even if there are some minor differences, which are statistically negligible. In short, the mixing argument demonstrates the similarity between the real world and the ideal world by comparing a series of realistic situations that are getting closer and closer to the ideal situation.

B Security Proof of The Asynchronous Random Information-Checking Protocol

Proof. We prove the security of the ARICP protocol by constructing an ideal adversary $\mathcal{S}$ that interacts with the environment $\mathcal{Z}$ and ideal functionalities. $\mathcal{S}$ simulates honest parties and runs the real-world adversary $\mathcal{A}$. In the proof, $\mathcal{S}$ communicates with $\mathcal{A}$ on behalf of honest parties and ideal functionalities. Messages between $\mathcal{S}$ and $\mathcal{Z}$ are forwarded to $\mathcal{A}$ and vice versa. When an honest party needs to send a message, $\mathcal{S}$ informs $\mathcal{A}$, which provides the arrival time to help $\mathcal{S}$ delay outputs in the ideal world. For each request-based delayed output that needs to be sent to an honest party, we let $\mathcal{S}$ delay the output in default until we say $\mathcal{S}$ allows the functionality to send the output. We will show that the output in the ideal world is identically distributed to that in the real world by using hybrid arguments.

Construction of the ideal adversary $\mathcal{S}$ is as follows. If we say that $\mathcal{S}$ delivers a message, $\mathcal{S}$ just tells $\mathcal{A}$ that the message has been delivered. $\mathcal{S}$ may not be able to know the context of the message.

When Sd and I are honest:

Simulator $\mathcal{S}$

1. For each corrupted verifier P_i, $\mathcal{S}$ randomly samples $(n+1)\kappa$ verification points $z = (\alpha, \beta_1, \beta_2, \beta_3)$ from $\mathbb{F}^4$ and sends these verification points to P_i on behalf of Sd. Each verification point is corresponding to $(\alpha, H^{(1)}(\alpha), H^{(2)}(\alpha), H^{(3)}(\alpha))$, where $H^{(1)}(\cdot), H^{(2)}(\cdot), H^{(3)}(\cdot)$ are the polynomials generated by Sd. $\mathcal{S}$ aborts the simulation if the $t(n+1)\kappa$ verification points are the same.
2. For each verifier P_i, if P_i is corrupted, $\mathcal{S}$ waits to receive a verification set Z^i_{n+1} from P_i and checks that each point in the verification set is on all verification points sent by $\mathcal{S}$ on behalf of Sd to the corrupted parties. If P_i is honest, when I receives the verification set of P_i, $\mathcal{S}$ considers that I receives a correct verification set.
3. When I receives at least $2t+1$ correct verification sets, I initializes a counter $\texttt{count} = n$, then $\mathcal{S}$ allows $\mathcal{F}_{\mathsf{ARICP}}$ to send output to I.
4. For each revelation, if $\texttt{count} > 0$, $\mathcal{S}$ does the following and replaces $\texttt{count}$ by $\texttt{count} - 1$.
   - If R is honest:
     a) For each verifier P_i, if P_i is corrupted, $\mathcal{S}$ waits to receive a verification set $\tilde{Z}^i_{\texttt{count}}$ from P_i, then $\mathcal{S}$ follows the protocol to check the verification set from P_i on behalf of R. If P_i is honest, when R receives the verification set of P_i, $\mathcal{S}$ considers that R receives a correct verification set.
     b) Upon receiving $t+1$ correct verification sets, $\mathcal{S}$ allows $\mathcal{F}_{\mathsf{ARICP}}$ to send the output to R.
   - If R is corrupted:
     a) $\mathcal{S}$ waits to receive $s, \overline{s}$ from $\mathcal{F}_{\mathsf{ARICP}}$, when the polynomials $H(y), \overline{H}(y)$ is first revealed, $\mathcal{S}$ samples two random degree-$L+(n+1)t\kappa$ polynomials $H(y), \overline{H}(y)$ whose L highest coefficients form $s, \overline{s}$ and all the $(n+1)t\kappa$ new points are correspond to the polynomials.
     b) $\mathcal{S}$ reveals $H(y), \overline{H}(y)$. For each honest verifier P_i, $\mathcal{S}$ samples κ random elements $\alpha^i_1, \ldots, \alpha^i_\kappa$ in $\mathbb{F}$ and sends $\{(\alpha^i_j, H(\alpha^i_j), \overline{H}(\alpha^i_j))\}_{j\in[\kappa]}$ to R on behalf of P_i

Fig. 10: Simulator for the $\mathcal{F}_{\mathsf{ARICP}}$ when both Sd and I are honest

Hybrid arguments:

$\mathbf{Hyb}_0$: In this hybrid, $\mathcal{S}$ receives the inputs of honest parties and runs the protocol honestly. This corresponds to the real-world scenario.

$\mathbf{Hyb}_1$: In this hybrid, $\mathcal{S}$ first randomly selects $(n+1)\kappa$ verification points for each corrupted party, then samples two polynomials $H(y), \overline{H}(y)$ based on the $(n+1)t\kappa$ verification points and secret vectors $s^{(1)}, s^{(2)}, s^{(3)}$. Because the polynomials $H(y), \overline{H}(y)$ are of degree-$L+(n+1)t\kappa$ and each $\{s^{(m)}\}_{m\in[3]}$ in $\mathbb{F}^L$, these verification points are $(n+1)t\kappa+1$ -wise independent. i.e. For each corrupted party, the $(n+1)t\kappa$ verification points is still indistinguishable from random. Thus, it is equivalent to changing only the order in which verification points are prepared for the corrupted party, but not the distribution of verification points. Therefore, the output distributions of $\mathbf{Hyb}_1$ and $\mathbf{Hyb}_0$ are consistent.

$\mathbf{Hyb}_2$: In this hybrid, $\mathcal{S}$ aborts the simulation if the $t(n+1)\kappa$ verification points are the same. Notice that the probability that each two verification points are the same is $1/|\mathbb{F}|$ and the corrupted parties have $(n+1)t\kappa$ points, so the probability that the verification points are all the same is

$$\epsilon_1 < \frac{1}{|\mathbb{F}|} \cdot \frac{[(n+1)t\kappa]}{[(n+1)t\kappa - 1]} < \frac{(n+1)^2 t^2 \kappa^2}{2^\kappa},$$

which is negligible in κ. Therefore, the output distributions of $\mathbf{Hyb}_2$ and $\mathbf{Hyb}_1$ are statistically consistent.

$\mathbf{Hyb}_3$: In this hybrid, $\mathcal{S}$ doesn't check the verification points sent by himself on behalf of I to the honest verifier P_i. Instead, $\mathcal{S}$ considers that when I receives a verification set from P_i then I accepts it. Because Sd and P_i are honest, P_i always sends a correct verification set and I will always accept it. Therefore, the output distributions of $\mathbf{Hyb}_3$ and $\mathbf{Hyb}_2$ are consistent.

$\mathbf{Hyb}_4$: In this hybrid, changing the way $\mathcal{S}$ checks the verification points of each corrupted verifier P_i. $\mathcal{S}$ checks the verification points are whether on the verification points sent by Sd to corrupted verifier P_i. The output distribution changes only if the verification points sent to I by the corrupted verifier P_i are not among the points sent to verifier P_i by Sd but the verification set is still accepted by I. Since corrupted parties only have $(n+1)t\kappa$ verification points, each $\{H^m(\cdot)\}_{m\in[3]}$ is random for corrupted parties. So at any element α' different from the first element of the $(n+1)t\kappa$ verification points, $(H^{(1)}(\alpha'), H^{(2)}(\alpha'), H^{(3)}(\alpha'))$ is uniformly random in $\mathbb{F}^3$. Thus, the probability of each corrupted verifier P_i sends κ fake points $(\alpha', \beta'_1, \beta'_2, \beta'_3)$ and I accepts them in $\mathbf{Hyb}_2$ is at most $\kappa/|\mathbb{F}^3|$. There are at most $t\kappa$ verification points sent by corrupted parties, so the probability that the output distribution changes is at most

$$\epsilon_2 < \frac{t\kappa}{2^{3\kappa}},$$

which is negligible in κ. Therefore, the output distributions of $\mathbf{Hyb}_4$ and $\mathbf{Hyb}_3$ are statistically consistent.

$\mathbf{Hyb}_5$: In this hybrid, for each revelation, if R is corrupted, $\mathcal{S}$ doesn't sample $\{H^{(m)}(y)\}_{m\in[3]}$ at first, but at the first revelation, $\mathcal{S}$ randomly samples two degree-$L+(n+1)t\kappa$ polynomials $H(y),\overline{H}(y)$ whose L highest coefficients form $s,\overline{s}$ and all the $t(n+1)\kappa$ new points of corrupted parties are on the polynomials. Since $\{H^{(m)}(y)\}_{m\in[3]}$ is random for R, and thus $H(y),\overline{H}(y)$ is also random for R until the first revelation of $H(y),\overline{H}(y)$. Therefore, the output distributions of $\mathbf{Hyb}_5$ and $\mathbf{Hyb}_4$ are consistent.

$\mathbf{Hyb}_6$: In this hybrid, for each honest verifier P_i, $\mathcal{S}$ doesn't prepare $(n+1)\kappa$ verification points for P_i, but samples κ random elements $\alpha_1^i,\dots,\alpha_\kappa^i$ in $\mathbb{F}$, and sends $\{(\alpha_j^i,H(\alpha_j^i),\overline{h}(\alpha_j^i))\}_{j\in[\kappa]}$ to R. Since each verification element α of P_i is randomly selected and each point $(\alpha,H(\alpha),\overline{H}(\alpha))$ sent by P_i to R must also correspond to $H(y),\overline{H}(y)$ sent by I to R, it does not change the output distribution. Therefore, the output distributions of $\mathbf{Hyb}_6$ and $\mathbf{Hyb}_5$ are consistent.

$\mathbf{Hyb}_7$: In this hybrid, for each revelation, if R is honest, $\mathcal{S}$ changes the way it checks points for honest verifiers. $\mathcal{S}$ considers that when R receives a verification set from each honest verifier then R accepts it. Since Sd and I are honest, each honest verifier also always sends R the correct verification points, and R will always accept them. Therefore, the output distributions of $\mathbf{Hyb}_7$ and $\mathbf{Hyb}_6$ are consistent.

$\mathbf{Hyb}_8$: In this hybrid, changing the way R gets $s,\overline{s}$. When R receives $t+1$ correct verification sets, $\mathcal{S}$ sends $\mathsf{proceed}$ and $(s,\overline{s})$ to $\mathcal{F}_{\mathsf{ARICP}}$ and lets R receive $(s,\overline{s})$ from $\mathcal{F}_{\mathsf{ARICP}}$. The only difference between $\mathbf{Hyb}_7$ and $\mathbf{Hyb}_8$ is whether R receives $(s,\overline{s})$ from $\mathcal{S}$ or $\mathcal{F}_{\mathsf{ARICP}}$, but this does not change the output distribution. Therefore, the output distributions of $\mathbf{Hyb}_8$ and $\mathbf{Hyb}_7$ are consistent.

Note that $\mathbf{Hyb}_8$ is the ideal-world scenario, Π_{ARICP} statistically-securely computes $\mathcal{F}_{\mathsf{ARICP}}$.

When Sd and I are corrupted:

Simulator $\mathcal{S}$

1. For each honest verifier P_i, $\mathcal{S}$ waits to receive $(n+1)\kappa$ verification point on behalf of P_i. Then $\mathcal{S}$ randomly divides them into $n+1$ disjoint sets. Each set is of size κ, denoted by $Z_1^i,\dots,Z_{n+1}^i$. Then $\mathcal{S}$ sends Z_{n+1}^i to I on behalf of P_i.
2. $\mathcal{S}$ sets $s^{(1)}=s^{(2)}=s^{(3)}=0$ and sends $(\mathsf{Init},\mathsf{ARICP},(s^{(1)},s^{(2)},s^{(3)}))$ to $\mathcal{F}_{\mathsf{ARICP}}$ on behalf of Sd. Here 0 is the zero vector in $\mathbb{F}^L$.
3. $\mathcal{S}$ initializes a counter $\texttt{count}=n$.
4. For each revelation, if $\texttt{count}>0$, $\mathcal{S}$ does the following and replaces $\texttt{count}$ by $\texttt{count}-1$:
   - If verifier P_i is honest:
     a). $\mathcal{S}$ waits to receive $H(y),\overline{H}(y)$ from I.
     b). For each verifier P_i, if P_i is corrupted, $\mathcal{S}$ waits to receive a verification set $\tilde{Z}_{\texttt{count}}^i$ from P_i. If P_i is honest, $\mathcal{S}$ uses $Z_{\texttt{count}}^i$ to compute $\tilde{Z}_{\texttt{count}}^i$. In both cases, $\mathcal{S}$ checks whether at least one point in $\tilde{Z}_{\texttt{count}}^i$ is consistent with $H(y),\overline{H}(y)$. If true, $\mathcal{S}$ considers that R receives a correct verification set.

c). Upon receiving $t+1$ correct verification sets, let $s, \overline{s}$ be the L highest coefficients of $H(y), \overline{H}(y)$, $\mathcal{S}$ sends Proceed and $s, \overline{s}$ to $\mathcal{F}_{\mathsf{ARICP}}$ and allows $\mathcal{F}_{\mathsf{ARICP}}$ sends the output to R. When R receives $2t+1$ incorrect verification sets, he sends Ignore to $\mathcal{F}_{\mathsf{ARICP}}$.

- If R is corrupted: For each honest verifier P_i, $\mathcal{S}$ sends $\tilde{Z}^i_{\mathtt{count}}$ to R on behalf of P_i.

Fig. 11: Simulator for the $\mathcal{F}_{\mathsf{ARICP}}$ when both Sd and I are corrupted

Hybrid arguments:

Hyb$_0$: In this hybrid, $\mathcal{S}$ receives the inputs of honest parties and runs the protocol honestly. This corresponds to the real-world scenario.

Hyb$_1$: In this hybrid, changing the way R gets $s, \overline{s}$. When R receives $t+1$ correct verification sets, $\mathcal{S}$ sends proceed and $(s, \overline{s})$ to $\mathcal{F}_{\mathsf{ARICP}}$ and lets R receive $(s, \overline{s})$ from $\mathcal{F}_{\mathsf{ARICP}}$. The only difference between **Hyb**$_0$ and **Hyb**$_1$ is whether R receives $(s, \overline{s})$ from $\mathcal{S}$ or $\mathcal{F}_{\mathsf{ARICP}}$, but this does not change the output distribution. Therefore, the output distributions of **Hyb**$_1$ and **Hyb**$_0$ are consistent.

Note that **Hyb**$_1$ is the ideal-world scenario, $\mathit{\Pi}_{\mathsf{ARICP}}$ statistically-securely computes $\mathcal{F}_{\mathsf{ARICP}}$.

When Sd is honest and I is corrupted:

Simulator $\mathcal{S}$

1. $\mathcal{S}$ waits to receive $s^{(1)}, s^{(2)}, s^{(3)}$ from $\mathcal{F}_{\mathsf{ARICP}}$. $\mathcal{S}$ selects three random degree-$L+(n+1)t\kappa$ polynomials $H^{(1)}(y), H^{(2)}(y), H^{(3)}(y)$ whose L highest coefficients form $s^{(1)}, s^{(2)}, s^{(3)}$.
2. For each corrupted verifier P_i, $\mathcal{S}$ randomly samples $(n+1)\kappa$ verification elements. $\mathcal{S}$ sends $(n+1)\kappa$ verification points $(\alpha, H^{(1)}(\alpha), H^{(2)}(\alpha), H^{(3)}(\alpha))$ to P_i on behalf of Sd.
3. $\mathcal{S}$ sends $\{H^{(m)}(y)\}_{m\in[3]}$ to I on behalf of Sd. For each honest verifier P_i, $\mathcal{S}$ selects κ random elements in $\mathbb{F}$ and sends κ verification points $Z^i_{n+1} = \{(\alpha^i_j, H^{(1)}(\alpha^i_j), H^{(2)}(\alpha^i_j), H^{(3)}(\alpha^i_j))\}_{j\in[\kappa]}$ to I on behalf of P_i.
4. $\mathcal{S}$ initializes a counter $\mathtt{count} = n$.
5. For each revelation, if $\mathtt{count} > 0$, $\mathcal{S}$ does the following and replaces $\mathtt{count}$ by $\mathtt{count} - 1$:
   - If verifier P_i is honest:
     a) $\mathcal{S}$ receives $H'(y), \overline{H'}(y)$ from I.
     b) Let $s, \overline{s}$ be the L highest coefficients of $H(y), \overline{H}(y)$, $\mathcal{S}$ checks whether $H'(y) = H^{(1)}(y) + H^{(2)}(y)$ and $\overline{H'}(y) = H^{(2)}(y) + H^{(3)}(y)$. If true, $\mathcal{S}$ sends Proceed and $s, \overline{s}$ to $\mathcal{F}_{\mathsf{ARICP}}$. Otherwise, $\mathcal{S}$ sends Ignore to $\mathcal{F}_{\mathsf{ARICP}}$.
     c) If $H'(y) = H^{(1)}(y) + H^{(2)}(y)$ and $\overline{H'}(y) = H^{(2)}(y) + H^{(3)}(y)$ are true. For each corrupted verifier P_i, $\mathcal{S}$ follows the protocol to check the verification set $\tilde{Z}^i_{\mathtt{count}}$ from P_i on behalf of R. For each honest

verifier P_i, when R receives the verification set of P_i, $\mathcal{S}$ considers that R receives a correct verification set.

d) Upon receiving $t+1$ correct verification sets, $\mathcal{S}$ allows $\mathcal{F}_{\mathsf{ARICP}}$ to send the output to R.

- If R is corrupted: For each honest verifier P_i, $\mathcal{S}$ samples κ random elements $\alpha_1^i, \ldots, \alpha_\kappa^i$ in $\mathbb{F}$ and sends $\{(\alpha_j^i, H(\alpha_j^i), \overline{H}(\alpha_j^i))\}_{j\in[\kappa]}$ to R on behalf of P_i

Fig. 12: Simulator for the $\mathcal{F}_{\mathsf{ARICP}}$ when Sd is honest and I is corrupted

Hybrid arguments:

$\mathbf{Hyb}_0$: In this hybrid, $\mathcal{S}$ receives the inputs of honest parties and runs the protocol honestly. This corresponds to the real-world scenario.

$\mathbf{Hyb}_1$: In this hybrid, $\mathcal{S}$ changes the way it gets $s^{(1)}, s^{(2)}, s^{(3)}$. Instead of using inputs from the honest parties, $\mathcal{S}$ waits to receive $s^{(1)}, s^{(2)}, s^{(3)}$ from $\mathcal{F}_{\mathsf{ARICP}}$. The only difference between $\mathbf{Hyb}_1$ and $\mathbf{Hyb}_0$ is how $\mathcal{S}$ gets $s^{(1)}, s^{(2)}, s^{(3)}$, which doesn't change the output distribution. Therefore, the output distributions of $\mathbf{Hyb}_1$ and $\mathbf{Hyb}_0$ are consistent.

$\mathbf{Hyb}_2$: In this hybrid, for each honest verifier P_i, $\mathcal{S}$ doesn't sample $(n+1)\kappa$ random elements at first. Instead, when each P_i needs to send Z_{n+1}^i or $\tilde{Z}_{\texttt{count}}^i$, $\mathcal{S}$ will randomly sample κ elements in $\mathbb{F}$ to compute $\tilde{Z}_{\texttt{count}}^i$. Thus, it is equivalent to changing only the order in which random elements are generated for honest parties, but not the output distribution. Therefore, the output distributions of $\mathbf{Hyb}_2$ and $\mathbf{Hyb}_1$ are consistent.

$\mathbf{Hyb}_3$: In this hybrid, for each honest R, $\mathcal{S}$ adds an addition condition to check whether $H'(y) = H^{(1)}(y) + H^{(2)}(y)$ and $\overline{H'}(y) = H^{(2)}(y) + H^{(3)}(y)$, where $H'(y), \overline{H'}(y)$ is received from I. If the condition is satisfied, $\mathcal{S}$ sends $\mathsf{Proceed}$ and $s, \overline{s}$ to $\mathcal{F}_{\mathsf{ARICP}}$. Otherwise, $\mathcal{S}$ sends Ignore to $\mathcal{F}_{\mathsf{ARICP}}$. The only difference between $\mathbf{Hyb}_1$ and $\mathbf{Hyb}_0$ is when $H'(y) \neq H^{(1)}(y) + H^{(2)}(y)$ or $\overline{H'}(y) \neq H^{(2)}(y) + H^{(3)}(y)$, but $\mathcal{S}$ can still receive at least $t+1$ correct verification sets, that means $\mathcal{S}$ accepts $H'(y), \overline{H'}(y)$ sent by I. i.e. At this time, there exists at least one verification point in honest verifier P_i's $\tilde{Z}_{\texttt{count}}^i$ is corresponding to $H'(y), \overline{H'}(y)$. For each honest verifier, during each revelation, which is equivalent to I correctly guessing one of the κ random elements selected by $\mathcal{S}$. The probability that happens is

$$\epsilon_3 = Pr[\mathcal{S} \text{ accepts } H'(y), \overline{H'}(y) \mid H'(y) \neq H^{(1)}(y) + H^{(2)}(y) \text{ or } \overline{H'}(y) \neq H^{(2)}(y) + H^{(3)}(y)] < \frac{\kappa}{\mathbb{F}} \cdot \frac{\kappa-1}{\mathbb{F}-1} \cdots \frac{1}{\mathbb{F}-\kappa+1} = \prod_{j=0}^{\kappa-1} \frac{\kappa-j}{\mathbb{F}-j} \leq (\frac{\kappa}{2^\kappa})^\kappa,$$

which is negligible in κ. Now consider that for all honest parties and for the maximum number of revelations, the probability that the $H'(y), \overline{H'}(y)$ forged by I is accepted by $\mathcal{S}$ is $(2t+1)n \cdot \epsilon_3$, which is still negligible. Therefore, the output distributions of $\mathbf{Hyb}_3$ and $\mathbf{Hyb}_2$ are statistically consistent.

Hyb$_4$: In this hybrid, for each revelation, if R is honest, for each honest verifier P_i, $\mathcal{S}$ doesn't check the verification set of P_i. $\mathcal{S}$ considers that when R receives a verification set from each honest verifier then R accepts it. Since Sd and P_i are honest, $H'(y) = H^{(1)}(y) + H^{(2)}(y)$ and $\overline{H'}(y) = H^{(2)}(y) + H^{(3)}(y)$, P_i always sends a correct verification set, so R always accepts it. Therefore, the output distributions of **Hyb**$_4$ and **Hyb**$_3$ are consistent.

Hyb$_5$: In this hybrid, changing the way R gets $s, \overline{s}$. When R receives $t+1$ correct verification sets, the condition $H'(y) = H^{(1)}(y) + H^{(2)}(y)$ and $\overline{H'}(y) = H^{(2)}(y) + H^{(3)}(y)$ is satisfied, $\mathcal{S}$ lets R receive $(s, \overline{s})$ from $\mathcal{F}_{\mathsf{ARICP}}$. At this time, $(s, \overline{s})$ computed from $\mathcal{S}$ is equal to $(s, \overline{s})$ output from $\mathcal{F}_{\mathsf{ARICP}}$. Therefore, the output distributions of **Hyb**$_5$ and **Hyb**$_4$ are consistent.

Note that **Hyb**$_5$ is the ideal-world scenario, Π_{ARICP} statistically-securely computes $\mathcal{F}_{\mathsf{ARICP}}$.

When Sd is corrupted and I is honest:

Simulator $\mathcal{S}$

1. For each honest verifier P_i, $\mathcal{S}$ waits to receive $(n+1)\kappa$ verification points from Sd. When P_i receives $(n+1)\kappa$ verification points, $\mathcal{S}$ randomly divides them into $n+1$ disjoint sets on behalf of P_i. Each set is of size κ, denoted by $Z_1^i, \ldots, Z_{n+1}^i$.
2. $\mathcal{S}$ receives $H^{(1)}(y), H^{(2)}(y), H^{(3)}(y)$ from Sd on behalf of I, and does the following:
   a) $\mathcal{S}$ checks $\{H^{(m)}(y)\}_{m\in[3]}$ are all of degree-$L + (n+1)t\kappa$. If true, $\mathcal{S}$ lets $s^{(1)}, s^{(2)}, s^{(3)}$ be the vector of the L highest coefficients of $H^{(1)}(y), H^{(2)}(y), H^{(3)}(y)$. Otherwise, $\mathcal{S}$ aborts the simulation.
   b) For each verifier P_i:
      – If P_i is honest, $\mathcal{S}$ utilizes the set Z_{n+1}^i divided by itself, and when it is satisfied that all points in Z_{n+1}^i and at least $n\kappa+1$ of the points receives by P_i from I correspond to $\{H^{(m)}(y)\}_{m\in[3]}$, $\mathcal{S}$ considers that I receives a correct verification set.
      – If P_i is corrupted, $\mathcal{S}$ waits to receive a verification set from P_i. When all the points in the verification set correspond to $\{H^{(m)}(y)\}_{m\in[3]}$, $\mathcal{S}$ considers that I received a correct verification set.
   c) When I receives $2t+1$ correct verification sets, $\mathcal{S}$ initializes a counter $\texttt{count} = n$ and sends $(\mathsf{Init}, \mathsf{ARICP}, (s^{(1)}, s^{(2)}, s^{(3)}))$ to $\mathcal{F}_{\mathsf{ARICP}}$. Then, $\mathcal{S}$ allows $\mathcal{F}_{\mathsf{ARICP}}$ sends the output to I. Otherwise, $\mathcal{S}$ doesn't continue.
3. $\mathcal{S}$ computes $H(y) = H^{(1)}(y) + H^{(2)}(y)$, $\overline{H}(y) = H^{(2)}(y) + H^{(3)}(y)$. For each revelation, if $\texttt{count} > 0$, $\mathcal{S}$ does the following and replaces $\texttt{count}$ by $\texttt{count} - 1$:
   - If R is honest:
      a) For each corrupted verifier P_i, $\mathcal{S}$ waits to receive a verification set $\tilde{Z}_{\texttt{count}}^i$ from P_i. For each honest verifier P_i, $\mathcal{S}$ uses $\tilde{Z}_{\texttt{count}}^i$ created by himself. In both cases, $\mathcal{S}$ follows the protocol to check the verification set from P_i on behalf of R.

b) Upon receiving $t+1$ correct verification sets, $\mathcal{S}$ lets $s, \overline{s}$ be the L highest coefficients of $H(y), \overline{H}(y)$ and allows $\mathcal{F}_{\mathsf{ARICP}}$ to send the output $s, \overline{s}$ to R.

- If R is corrupted: $\mathcal{S}$ sends $H(y), \overline{H}(y)$ to R on behalf of I. For each honest verifier P_i, $\mathcal{S}$ sends $\tilde{Z}^i_{\mathtt{count}}$to R on behalf of P_i.

Fig. 13: Simulator for the $\mathcal{F}_{\mathsf{ARICP}}$ when Sd is corrupted and I is honest

Hybrid arguments:

$\mathbf{Hyb}_0$: In this hybrid, $\mathcal{S}$ receives the inputs of honest parties and runs the protocol honestly. This corresponds to the real-world scenario.

$\mathbf{Hyb}_1$: In this hybrid, changing the way I receives $s^{(1)}, s^{(2)}, s^{(3)}$. When I receives $2t+1$ correct verification sets, $\mathcal{S}$ sends $(Init, \mathsf{ARICP}, (s^{(1)}, s^{(2)}, s^{(3)}))$ to $\mathcal{F}_{\mathsf{ARICP}}$. Then, I will receive $s^{(1)}, s^{(2)}, s^{(3)}$ from $\mathcal{F}_{\mathsf{ARICP}}$ rather than being computed by himself. However, this does not change the output distribution. Therefore, the output distributions of $\mathbf{Hyb}_1$ and $\mathbf{Hyb}_0$ are consistent.

$\mathbf{Hyb}_2$: In this hybrid, $\mathcal{S}$ adds an addition condition to check the verification points of honest verifiers. $\mathcal{S}$ needs to check at least $n\kappa+1$ of the points received by P_i from I correspond to $\{H^{(m)}(y)\}_{m\in[3]}$. If true, $\mathcal{S}$ considers that Z^i_{n+1} is a correct verification set. The output distribution only changes if P_i receives fewer than $n\kappa+1$ correct points from Sd but P_i still provides a correct verification set Z^i_{n+1} to I. The probability is

$$\epsilon_4 \leq \prod_{j=0}^{\kappa-1} \frac{n\kappa - j}{(n+1)\kappa - j} \leq (1 - \frac{1}{n+1})^{\kappa},$$

since $0 < (1 - \frac{1}{n+1}) < 1$, the probability ϵ_4 is negligible in κ. Since $\mathcal{S}$ needs to receive at least $t+1$ honest P_i's correct $\tilde{Z}^i_{\mathtt{count}}$, so the probability that the output distribution changes is at most $(t+1)\epsilon_4$, which is still negligible in κ. Therefore, the output distributions of $\mathbf{Hyb}_2$ and $\mathbf{Hyb}_1$ are statistically consistent.

Note that $\mathbf{Hyb}_2$ is the ideal-world scenario, Π_{ARICP} statistically-securely computes $\mathcal{F}_{\mathsf{ARICP}}$.

Then we compute the communication complexity of our protocol. Π_{ARICP} takes 3 vectors as inputs and each vector is of size L, n is an input parameter. All parties execute the Initialization Phase only once, while the Revelation Phase can be executed for at most n times.

During the Initialization Phase: Sd sends 3 degree-$(L+t(n+1)\kappa)$ polynomials to I, which requires $\mathcal{O}(L\kappa+n(n+1)\kappa^2)$-bit communication. Sd also sends $(n+1)\kappa$ evaluation points to each verifier, resulting in a communication of $\mathcal{O}((n+1)n\kappa^2)$. Each verifier sends a set of size κ to I, resulting in a communication of $\mathcal{O}(n\kappa^2)$ bits. Therefore, the total communication cost is $\mathcal{O}(L\kappa + n^2\kappa^2)$ bits during the Initialization Phase.

During the Revelation Phase: Each time sending two degree-$(L + t(n + 1)\kappa)$ polynomial from I to R requires communication of $\mathcal{O}(L\kappa + n(n + 1)\kappa^2)$ bits.

Each verifier sends a set of κ field elements to I, resulting in a communication of $\mathcal{O}(n\kappa^2)$ bits. Therefore, the total communication cost is $\mathcal{O}(L\kappa + n^2\kappa^2)$ bits during the Revelation Phase. Since the Revelation Phase can be executed for at most T times, the total communication cost is $\mathcal{O}(nL\kappa + n^3\kappa^2)$ bits.

Therefore, Π_{ARICP} requires communication of $\mathcal{O}(nL\kappa + n^3\kappa^2)$ bits. □

C Proof of The main Theorem about Our ACSS Protocol

C.1 Construction of Π_{ACSS}

In this section, we present our construction of Π_{ACSS} as follows.

Protocol Π_{ACSS}

All parties execute $\Pi_{\mathsf{Sh}}, \Pi_{\mathsf{Ver}}$ and Π_{Comp} in order.

Fig. 14: The protocol of the Π_{ACSS}

C.2 Security Proof

Proof. We prove the Theorem 2 by constructing an ideal adversary $\mathcal{S}$. $\mathcal{S}$ needs to interact with the environment $\mathcal{Z}$ and with the ideal functionalities. $\mathcal{S}$ constructs virtual real-world honest parties and runs the real-world adversary $\mathcal{A}$. For simplicity, we just let $\mathcal{S}$ communicate with $\mathcal{A}$ on behalf of honest parties and the ideal functionality of sub-protocols in our proof. In order to simulate the communication with $\mathcal{Z}$, every message that $\mathcal{S}$ receives from $\mathcal{Z}$ is sent to $\mathcal{A}$, and likewise, every message sent from $\mathcal{A}$ sends to $\mathcal{Z}$ is forwarded by $\mathcal{S}$. Each time an honest party needs to send a message to another honest party, $\mathcal{S}$ will tell $\mathcal{A}$ that a message has been delivered such that $\mathcal{A}$ can tell $\mathcal{S}$ the arrival time of this message to help $\mathcal{S}$ instruct the functionalities to delay the outputs in the ideal world. For each request-based delayed output that needs to be sent to an honest party, we let $\mathcal{S}$ delay the output in default until we say $\mathcal{S}$ allows the functionality to send the output. We will show that the output in the ideal world is identically distributed to that in the real world by using hybrid arguments. Construction of the ideal adversary $\mathcal{S}$ is as follows.

When D is honest:

Simulator $\mathcal{S}$

Sharing Phase

1. For each corrupted P_j, $\mathcal{S}$ receives $q^{(1)}(\alpha_j), \dots, q^{(N)}(\alpha_j)$ from $\mathcal{F}_{\mathsf{ACSS}}$.
2. For each corrupted P_j, $r \in [0, t]$ and $\ell \in [L']$, $\mathcal{S}$ selects a random degree-$2t$ column polynomial $g_j^{(\ell)}(y)$, s.t. $g_j^{(\ell)}(\alpha_{-r}) = q^{(k)}(\alpha_j)$.
3. For each corrupted P_j and $\ell \in [L']$, $\mathcal{S}$ selects a random degree-$2t$ polynomial $z_j^{(\ell)}(y)$ and $z_j^{(\ell)}(y) \neq g_j^{(\ell)}(y)$. If not, $\mathcal{S}$ aborts the simulation.

4. For each corrupted P_i and $\ell \in [L']$, $\mathcal{S}$ respectively selects two random degree-t polynomials $f_i^{(\ell)}(x), w_i^{(\ell)}(x)$, s.t. $f_i^{(\ell)}(\alpha_j) = g_j^{(\ell)}(\alpha_i)$ and $w_i^{(\ell)}(\alpha_j) = z_j^{(\ell)}(\alpha_i)$ for each corrupted party P_j on behalf of D.
5. For each corrupted P_j and $\ell \in [L']$, $\mathcal{S}$ sends $z_j^{(\ell)}(y), g_j^{(\ell)}(y), w_j^{(\ell)}(x)$ to P_j on behalf of D.
6. $\mathcal{S}$ initializes sets $\mathcal{V}_i, \mathcal{V}$ to $\emptyset$, each $j \in [n]$. For each $P_i \in \mathcal{P}$ and $\ell \in [L']$:
   - If P_i is honest:
     a) When P_i receives $z_i^{(\ell)}(y), g_i^{(\ell)}(y), w_i^{(\ell)}(x)$ from D, $\mathcal{S}$ broadcasts OK_i on behalf of P_i.
     b) $\mathcal{S}$ sends $\{z_i^{(\ell)}(\alpha_j), w_i^{(\ell)}(\alpha_j)\}_{\ell \in [L']}$ to P_j on behalf of P_i, each $j \in [n]$.
     c) $\mathcal{S}$ waits to receives $z_j^{(\ell)}(\alpha_i), w_j^{(\ell)}(\alpha_i)$ from P_j on behalf of P_i. If P_j is honest, $\mathcal{S}$ broadcasts $iAMj$ on behalf of P_i. Else, $\mathcal{S}$ on behalf of P_i checks whether the points sent by P_j to P_i are points with share α_i in the polynomials $z_j^{(\ell)}(y), g_j^{(\ell)}(y)$ distributed to P_j by $\mathcal{S}$ on behalf of D. If so, $\mathcal{S}$ broadcasts $iAMj$ on behalf of P_i.
     d) $\mathcal{S}$ delivers an initialization request to $\mathcal{F}_{\mathsf{ARICP}}(P_i, D)$ on behalf of P_i and emulates $\mathcal{F}_{\mathsf{ARICP}}$ to send the output to P_i.
     e) $\mathcal{S}$ follows the protocol to construct $\mathcal{V}_j$, $\mathcal{V}$ and $\mathcal{M}$ sets.
   - If P_i is corrupted:
     a) $\mathcal{S}$ sends $\{z_j^{(\ell)}(\alpha_i), w_j^{(\ell)}(\alpha_i)\}_{\ell \in [L']}$ to P_i on behalf of each honest P_j.
     b) $\mathcal{S}$ emulates $\mathcal{F}_{\mathsf{ARICP}}(P_i, D)$ to receive $(\mathsf{Init}, \mathsf{ARICP}, n, (sign^*_{i,1}, sign^*_{i,2}, sign^*_{i,3}))$ from P_i. Then, $\mathcal{S}$ follows the protocol to check$(sign^*_{i,1}, sign^*_{i,2}, sign^*_{i,3})$ of P_i on behalf of D. If so, $\mathcal{S}$ emulates $\mathcal{F}_{\mathsf{ARICP}}(P_i, D)$ to send the output to D and follows the protocol to construct $\mathcal{V}_j$, $\mathcal{V}$ and $\mathcal{M}$ sets.
7. $\mathcal{S}$ broadcasts the set $\mathcal{M}$ on behalf of D when $|\mathcal{M}| = 2t+1$.
8. For each honest P_j, $\mathcal{S}$ waits until P_j receives the set $\mathcal{M}$, $\{OK_i\}_{P_i \in \mathcal{M}}$, $\mathcal{V}$, $\{\mathcal{V}_j\}_{P_j \in \mathcal{V}}$ and then starts the next phase of the simulation.

Fig. 15: Part-(1/3) of the simulator for the $\mathcal{F}_{\mathsf{ACSS}}$ when D is honest

Simulator $\mathcal{S}$

Verification Phase

- For each honest P_i:
  1. For each honest $P_h \in \mathcal{M}$, $\mathcal{S}$ emulates $\mathcal{F}_{\mathsf{ARICP}}(P_h, D)$ to send the output to the receiver P_i. For each corrupted $P_h \in \mathcal{M}$, $\mathcal{S}$ faithfully emulates $\mathcal{F}_{\mathsf{ARICP}}(P_h, D)$.

2. When P_i receives the output from $\mathcal{F}_{\mathsf{ARICP}}(P_h, D)$ for all $P_h \in \mathcal{M}$, $\mathcal{S}$ considers that P_i accepts the sum of $\{g_i^{(\ell)}(y), z_i^{(\ell)}(y)\}_{\ell \in [L']}$ and $\{z_i^{(\ell)}(y), w_i^{(\ell)}(x)\}_{\ell \in [L']}$, then starts the simulation of P_i in the next phase.

- For each corrupted P_i: For all $P_h \in \mathcal{M}$, $\mathcal{S}$ faithfully emulates $\mathcal{F}_{\mathsf{ARICP}}(P_h, D)$.

Fig. 16: Part-(2/3) of the simulator for the $\mathcal{F}_{\mathsf{ACSS}}$ when D is honest

Simulator $\mathcal{S}$

Completion Phase

Reconstructing row polynomials:
For each $P_i \in \mathcal{P}$:

- If P_i is honest:
  1. $\mathcal{S}$ on behalf of the P_i follows the protocol to send $\{g_i^{(\ell)}(\alpha_j)\}_{\ell \in [L']}$ to P_j, $j \in [n]$.
  2. $\mathcal{S}$ waits to receive each P_j's $\{g_j^{(\ell)}(\alpha_i)\}_{\ell \in [L']}$.
     a) If P_j is honest and $P_j \in \mathcal{M}$, When P_i receives $\{g_j^{(\ell)}(\alpha_i)\}_{\ell \in [L']}$ from P_j, $\mathcal{S}$ considers that P_i accepts P_j's $\{g_j^{(\ell)}(\alpha_i)\}_{\ell \in [L']}$.
     b) If P_j is corrupted and $P_j \in \mathcal{M}$, $\mathcal{S}$ on behalf of P_i checks whether these points $\{g_j^{(\ell)}(\alpha_i)\}_{\ell \in [L']}$ are points whose share is α_i on the polynomials $\{g_j^{(\ell)}(y)\}_{\ell \in [L']}$ sent to corrupted P_j by $\mathcal{S}$ on behalf of D. If true, $\mathcal{S}$ considers that P_i accepts P_j's $\{g_j^{(\ell)}(\alpha_i)\}_{\ell \in [L']}$.
  3. When P_i accepts $t+1$ different P_j's $\{g_j^{(\ell)}(\alpha_i)\}_{\ell \in [L']}$, $\mathcal{S}$ considers that P_i gets his $\{f_i^{(\ell)}(x), w_i^{(\ell)}(x)\}_{\ell \in [L']}$.
- If P_i is corrupted: $\mathcal{S}$ follows the protocol to send $\{g_j^{(\ell)}(\alpha_i)\}_{\ell \in [L']}$ to P_i on behalf of each honest P_j.

Reconstructing column polynomials:
For each $P_i \in \mathcal{P}$:

- If P_i is honest:
  1. When P_i reconstructs $\{f_i^{(\ell)}(x)\}_{\ell \in [L']}$ and $\{w_i^{(\ell)}(x)\}_{\ell \in [L']}$, $\mathcal{S}$ on behalf of P_i follows the protocol to send $\{f_i^{(\ell)}(\alpha_j), w_i^{(\ell)}(\alpha_j)\}_{\ell \in [L']}$ to P_j, $j \in [n]$.
  2. $\mathcal{S}$ waits to receive each P_j's $\{f_j^{(\ell)}(\alpha_i), w_j^{(\ell)}(\alpha_i)\}_{\ell \in [L']}$ to P_j.
     a) If P_j is honest, When $P[i]$ receives $\{f_j^{(\ell)}(\alpha_i), w_j^{(\ell)}(\alpha_i)\}_{\ell \in [L']}$ from P_j, $\mathcal{S}$ considers that P_i accepts P_j's $\{f_j^{(\ell)}(\alpha_i), w_j^{(\ell)}(\alpha_i)\}_{\ell \in [L']}$.
     b) If P_j is corrupted, $\mathcal{S}$ on behalf of P_i checks whether these points $\{f_j^{(\ell)}(\alpha_i), w_j^{(\ell)}(\alpha_i)\}_{\ell \in [L']}$ are points whose share is α_i on the polyno-

mials $\{f_j^{(\ell)}(x), w_i^{(\ell)}(x)\}_{\ell\in[L']}$ prepared for corrupted P_j by $\mathcal{S}$. If true, $\mathcal{S}$ considers that P_i accepts P_j's $\{f_j^{(\ell)}(\alpha_i), w_j^{(\ell)}(\alpha_i)\}_{\ell\in[L']}$.

3. When P_i accepts $2t+1$ different P_j's $\{f_j^{(\ell)}(\alpha_i), w_j^{(\ell)}(\alpha_i)\}_{\ell\in[L']}$, $\mathcal{S}$ considers that P_i gets his $\{g_i^{(\ell)}(y), z_i^{(\ell)}(y)\}_{\ell\in[L']}$. Then, $\mathcal{S}$ allows $\mathcal{F}_{\mathsf{ACSS}}$ to send the output to P_i.

- If P_i is corrupted: $\mathcal{S}$ follows the protocol to send $\{f_j^{(\ell)}(\alpha_i), w_j^{(\ell)}(\alpha_i)\}_{\ell\in[L']}$ to P_i on behalf of each honest P_j.

Fig. 17: Part-(3/3) of the simulator for the $\mathcal{F}_{\mathsf{ACSS}}$ when D is honest

Hybrid arguments:

Hyb$_0$: In this hybrid, $\mathcal{S}$ receives the inputs of honest parties and runs the protocol honestly. This corresponds to the real-world scenario.

Hyb$_1$: In this hybrid, during the Sharing Phase, for each corrupted P_j, $\mathcal{S}$ changes the order of generation of the bivariate polynomials $\{F^{(\ell)}(x,y), R^{(\ell)}(x,y)\}_{\ell\in[L']}$ respectively. First, $\mathcal{S}$ samples the column polynomials, random column polynomials, row polynomials and random row polynomials of the corrupted parties separately, and then samples $\{F^{(\ell)}(x,y), R^{(\ell)}(x,y)\}_{\ell\in[L']}$ based on the inputs of D and the polynomials of the corrupted parties separately. Since only the order of generation of the two bivariate polynomials is changed, but not the output distribution. Therefore, the output distributions of **Hyb**$_1$ and **Hyb**$_0$ are consistent.

Hyb$_2$: In this hybrid, during the Sharing Phase, $\mathcal{S}$ aborts the simulation if $g_i^{(\ell)}(y) = z_i^{(\ell)}(y)$ for each $\ell \in [L']$ i.e. $F^{(\ell)}(x,y) = R^{(\ell)}(x,y)$. Note that the probability that $g_i^{(\ell)}(y) = z_i^{(\ell)}(y)$ for each $\ell \in [l']$, corrupted P_i is $\frac{2L'|\mathbb{F}|}{|\mathbb{F}|^{2t}} = \frac{L'}{2^{2t\kappa-\kappa-1}}$. There are at most t corrupted parties, so the probability that the output distribution changes is at most

$$\epsilon_1 = \frac{tL'}{2^{2t\kappa-\kappa-1}},$$

which is negligible in κ. Therefore, the output distributions of **Hyb**$_2$ and **Hyb**$_1$ are statistically consistent.

Hyb$_3$: In this hybrid, during the Sharing Phase, $\mathcal{S}$ doesn't check the degree of the polynomials $\{g_i^{(\ell)}(y), z_i^{(\ell)}(y), w_i^{(\ell)}(x)\}_{\ell\in[L']}$ sent to each honest P_i by himself on behalf f D. Instead, $\mathcal{S}$ considers that when P_i receives polynomials $\{g_i^{(\ell)}(y), z_i^{(\ell)}(y), w_i^{(\ell)}(x)\}_{\ell\in[L']}$ from D then P_i broadcasts OK_i. Since D and P_i are honest, D always sends correct $\{g_i^{(\ell)}(y), z_i^{(\ell)}(y), w_i^{(\ell)}(x)\}_{\ell\in[L']}$ to P_i and P_i will always broadcast OK_i. Therefore, the output distributions of **Hyb**$_3$ and **Hyb**$_2$ are consistent.

Hyb$_4$: In this hybrid, during the Sharing Phase, if P_i is honest, $\mathcal{S}$ changes the way P_i checks the $\{z_j^{(\ell)}(\alpha_i), w_j^{(\ell)}(\alpha_i)\}_{\ell\in[L']}$ sent by the honest P_j. Instead, $\mathcal{S}$ considers that when P_i receives $\{z_j^{(\ell)}(\alpha_i), w_j^{(\ell)}(\alpha_i)\}_{\ell\in[L']}$ from P_j then broadcasts

$iAMj$. Since D, P_i and P_j are honest, P_j always sends correct random points to P_i and P_i will always accepts them. Therefore, the output distributions of $\mathbf{Hyb}_4$ and $\mathbf{Hyb}_3$ are consistent.

$\mathbf{Hyb}_5$: In this hybrid, during the Sharing Phase, if P_i is honest, $\mathcal{S}$ changes the way P_i checks the $\{z_j^{(\ell)}(\alpha_i), w_j^{(\ell)}(\alpha_i)\}_{\ell\in[L']}$ sent by the corrupted P_j. Instead, if these points $\{z_j^{(\ell)}(\alpha_i), w_j^{(\ell)}(\alpha_i)\}_{\ell\in[L']}$ sent by P_j to P_i are points with share α_i on polynomials $\{z_j^{(\ell)}(y), w_j^{(\ell)}(x)\}_{\ell\in[L']}$ sent by $\mathcal{S}$ on behalf of D to P_j, then $\mathcal{S}$ considers that P_i accepts the points sent by P_j. Since D and P_i are honest, P_i always has correct $\{z_i^{(\ell)}(\alpha_j), w_i^{(\ell)}(\alpha_j)\}_{\ell\in[L']}$ i.e. $\{w_j^{(\ell)}(\alpha_i), z_j^{(\ell)}(\alpha_i)\}_{\ell\in[L']}$. This step is actually equivalent to the check in the protocol. Therefore, the output distributions of $\mathbf{Hyb}_5$ and $\mathbf{Hyb}_4$ are consistent.

$\mathbf{Hyb}_6$: In this hybrid, during the Sharing Phase, $\mathcal{S}$ doesn't check the $(sign^*_{i,1}, sign^*_{i,2}, sign^*_{i,3})$ generated by the polynomials $\{g_i^{(\ell)}(y), z_i^{(\ell)}(y), w_i^{(\ell)}(x)\}_{\ell\in[L']}$ sent by himself on behalf of D to each honest P_i. Instead, $\mathcal{S}$ considers that when D receives $(sign^*_{i,1}, sign^*_{i,2}, sign^*_{i,3})$ from each honest P_i then D accepts them. Since D and P_i are honest, P_i always sends correct $(sign^*_{i,1}, sign^*_{i,2}, sign^*_{i,3})$ and D will always accepts them. Therefore, the output distributions of $\mathbf{Hyb}_6$ and $\mathbf{Hyb}_5$ are consistent.

$\mathbf{Hyb}_7$: In this hybrid, during the Verification Phase, for each honest P_i, $\mathcal{S}$ changes the way P_i checks the output of $\mathcal{F}_{\mathsf{ARICP}}(P_h, D)$ for all $P_h \in \mathcal{M}$. $\mathcal{S}$ considers that when P_i receives the output of $\mathcal{F}_{\mathsf{ARICP}}(P_h, D)$ for all $P_h \in \mathcal{M}$ then P_i accepts the sum of $\{g_i^{(\ell)}(y), z_i^{(\ell)}(y)\}_{\ell\in[L']}$ and $\{z_i^{(\ell)}(y), w_i^{(\ell)}(x)\}_{\ell\in[L']}$ sent by D. Since D and P_i are honest, P_i always receives correct $\{g_i^{(\ell)}(y), z_i^{(\ell)}(y), w_i^{(\ell)}(x)\}_{\ell\in[L']}$ and P_i will always accepts them. Therefore, the output distributions of $\mathbf{Hyb}_7$ and $\mathbf{Hyb}_6$ are consistent.

$\mathbf{Hyb}_8$: In this hybrid, during the Completion Phase, when reconstruct the row polynomials, for each honest P_i, $\mathcal{S}$ changes the way P_i checks the $\{g_j^{(\ell)}(\alpha_i)\}_{\ell\in[L']}$ sent by each honest P_j and $P_j \in \mathcal{M}$. Instead, $\mathcal{S}$ considers that when P_i receives $\{g_j^{(\ell)}(\alpha_i)\}_{\ell\in[L']}$ from P_j then accepts them. Since D and P_i honest, P_i always receives correct $\{g_j^{(\ell)}(\alpha_i)\}_{\ell\in[L']}$ (correspond to $P^{(\ell)}(x,y), Q^{(\ell)}(x,y)$) from honest $P_j \in \mathcal{M}$, P_i will always accepts them. Therefore, the output distributions of $\mathbf{Hyb}_8$ and $\mathbf{Hyb}_7$ are consistent.

$\mathbf{Hyb}_9$: In this hybrid, during the Completion Phase, when reconstruct the row polynomials, for each honest P_i, $\mathcal{S}$ changes the way P_i checks the $\{g_j^{(\ell)}(\alpha_i)\}_{\ell\in[L']}$ sent by each corrupted P_j and $P_j \in \mathcal{M}$. Instead, if these points $\{g_j^{(\ell)}(\alpha_i)\}_{\ell\in[L']}$ sent by P_j to P_i are points with share α_i on polynomials $\{g_j^{(\ell)}(y)\}_{\ell\in[L']}$ sent by $\mathcal{S}$ on behalf of D to each corrupted P_j, then $\mathcal{S}$ considers that P_i accepts the points sent by P_j. Since polynomials $\{P^{(\ell)}(x,y), Q^{(\ell)}(x,y)\}_{\ell\in[L']}$ are correct, this step is actually equivalent to the check in the protocol. Therefore, the output distributions of $\mathbf{Hyb}_9$ and $\mathbf{Hyb}_8$ are consistent.

Hyb$_{10}$: In this hybrid, during the Completion Phase, when reconstruct the column polynomials, for each honest P_i, $\mathcal{S}$ changes the way P_i checks the $\{f_j^{(\ell)}(\alpha_i), w_j^{(\ell)}(\alpha_i)\}_{\ell\in[L']}$ sent by each honest P_j. Instead, $\mathcal{S}$ considers that when P_i receives $\{f_j^{(\ell)}(\alpha_i), w_j^{(\ell)}(\alpha_i)\}_{\ell\in[L']}$ from each honest P_j then accepts them. Since D and P_j honest, P_i always receives correct $\{f_j^{(\ell)}(\alpha_i), w_j^{(\ell)}(\alpha_i)\}_{\ell\in[L']}$ (correspond to $P^{(\ell)}(x,y), Q^{(\ell)}(x,y)$) from each honest P_j, P_i will always accepts them. Therefore, the output distributions of **Hyb**$_{10}$ and **Hyb**$_9$ are consistent.

Hyb$_{11}$: In this hybrid, during the Completion Phase, when reconstruct the column polynomials, for each honest P_i, $\mathcal{S}$ changes the way P_i checks the $\{f_j^{(\ell)}(\alpha_i), w_j^{(\ell)}(\alpha_i)\}_{\ell\in[L']}$ sent by each corrupted P_j. Instead, if these points $\{f_j^{(\ell)}(\alpha_i), w_j^{(\ell)}(\alpha_i)\}_{\ell\in[L']}$ sent by each corrupted P_j to P_i are points with share α_i on polynomials $\{f_j^{(\ell)}(x), w_j^{(\ell)}(x)\}_{\ell\in[L']}$ prepared by $\mathcal{S}$ on behalf of D to each corrupted P_j, then $\mathcal{S}$ considers that P_i accepts the points sent by P_j. Since polynomials $\{P^{(\ell)}(x,y), Q^{(\ell)}(x,y)\}_{\ell\in[L']}$ are correct, this step is actually equivalent to the check in the protocol. Therefore, the output distributions of **Hyb**$_{11}$ and **Hyb**$_{10}$ are consistent.

Note that **Hyb**$_{11}$ is the ideal-world scenario, Π_{ACSS} statistically-securely computes $\mathcal{F}_{\mathsf{ACSS}}$ when D is honest.

When D is corrupted:

Simulator $\mathcal{S}$

Sharing Phase

1. For each $P_i \in \mathcal{P}$:
   - If P_i is honest:
     a) When P_i receives $\{z_i^{(\ell)}(y), g_i^{(\ell)}(y), w_i^{(\ell)}(x)\}_{\ell\in[L']}$ from D, $\mathcal{S}$ broadcasts OK_i on behalf of P_i if these polynomials are of degree-$2t, 2t, t$.
     b) $\mathcal{S}$ sends $\{z_i^{(\ell)}(\alpha_j), w_i^{(\ell)}(\alpha_j)\}_{\ell\in[L']}$ to P_j on behalf of P_i, each $j \in [n]$.
     c) $\mathcal{S}$ follows the protocol to compute $(sign^*_{i,1}, sign^*_{i,2}, sign^*_{i,3})$ of P_i on behalf of P_i.
     d) $\mathcal{S}$ sends $(\mathsf{Init}, \mathsf{ARICP}, n, (sign^*_{i,1}, sign^*_{i,2}, sign^*_{i,3}))$ to $\mathcal{F}_{\mathsf{ARICP}}(P_i, D)$ on behalf of P_i.
     e) $\mathcal{S}$ faithfully emulates $\mathcal{F}_{\mathsf{ARICP}}(P_i, D)$.
   - If P_i is corrupted:
     a) $\mathcal{S}$ sends $\{z_j^{(\ell)}(\alpha_i), w_j^{(\ell)}(\alpha_i)\}_{\ell\in[L']}$ to P_i on behalf of each honest P_j.
     b) $\mathcal{S}$ faithfully emulates $\mathcal{F}_{\mathsf{ARICP}}(P_i, D)$.
2. For each honest P_j, when P_j receives $\mathcal{M}$, $\mathcal{V}$ and $\{\mathcal{V}_j\}_{P_j\in\mathcal{V}}$, $\mathcal{S}$ follows the protocol to check these sets. If proceeds then starts the next phase of the simulation.
3. Let $\mathcal{H}$ be the first $t+1$ honest parties in the set $\mathcal{M}$. For each $\ell \in [L']$, $\mathcal{S}$ reconstructs a degree-$(t, 2t)$ bivariate polynomial $Q^{(\ell)}(x,y)$ and a degree-$(2t, 2t)$ symmetric bivariate polynomial $P^{(\ell)}(x,y)$ s.t. $Q^{(\ell)}(\alpha_i, y) =$

$sign_{i,1}^{(\ell)}(y) + sign_{i,2}^{(\ell)}(y)$ and $P^{(\ell)}(\alpha_i, y) = sign_{i,2}^{(\ell)}(y) + sign_{i,3}^{(\ell)}(y)$ for each $P_i \in \mathcal{H}$.

4. For each $\ell \in [L']$, $\mathcal{S}$ computes each corrupted P_j's $\hat{sign}_j^{(\ell)}(y) = Q^{(\ell)}(\alpha_j, y)$ and $\overline{\hat{sign}}_j^{(\ell)}(y) = P^{(\ell)}(\alpha_j, y)$.

Fig. 18: Part-(1/3) of the simulator for the $\mathcal{F}_{\mathsf{ACSS}}$ when D is corrupted

Simulator $\mathcal{S}$

Verification Phase

- For each honest P_i:
  1. When P_i receives $\{g_i^{(\ell)}(y), z_i^{(\ell)}(y), w_i^{(\ell)}(x)\}_{\ell \in [L']}$ and these polynomials are of degree-$2t, 2t, t$. Then, $\mathcal{S}$ follows the protocol to compute $\{sign_i^{(\ell)}(y), \overline{sign}_i^{(\ell)}(y)\}_{\ell \in [L']}$ on behalf of P_i.
  2. For each honest $P_h \in \mathcal{M}$, $\mathcal{S}$ faithfully emulates $\mathcal{F}_{\mathsf{ARICP}}(P_i, D)$.
  3. $\mathcal{S}$ does the following:
     a) For each $\ell \in [L']$, $\mathcal{S}$ checks whether $sign_i^{(\ell)}(y) = Q^{(\ell)}(\alpha_i, y)$.
     b) For each $\ell \in [L']$, $\mathcal{S}$ checks whether $\overline{sign}_i^{(\ell)}(y) = P^{(\ell)}(\alpha_i, y)$.
     c) If both the above checks pass, $\mathcal{S}$ accepts $\{sign_i^{(\ell)}(y), \overline{sign}_i^{(\ell)}(y)\}_{\ell \in [L']}$ on behalf of P_i. Otherwise, $\mathcal{S}$ rejects these polynomials.
- For each corrupted P_i: For all $P_h \in \mathcal{M}$, $\mathcal{S}$ faithfully emulates $\mathcal{F}_{\mathsf{ARICP}}(P_h, D)$.

Fig. 19: Part-(2/3) of the simulator for the $\mathcal{F}_{\mathsf{ACSS}}$ when D is corrupted

Simulator $\mathcal{S}$

Completion Phase

Reconstructing row polynomials:
For each $P_i \in \mathcal{P}$:

- If P_i is honest:
  1. $\mathcal{S}$ on behalf of the P_i follows the protocol to send $\{g_i^{(\ell)}(\alpha_j)\}_{\ell \in [L']}$ to P_j, $j \in [n]$.
  2. $\mathcal{S}$ waits to receive each $P_j \in \mathcal{M}$'s $\{g_j^{(\ell)}(\alpha_i)\}_{\ell \in [L']}$.
     a) If $P_j \in \mathcal{H}$, When P_i receives $\{g_j^{(\ell)}(\alpha_i)\}_{\ell \in [L']}$ from P_j, $\mathcal{S}$ considers that P_i accepts P_j's $\{g_j^{(\ell)}(\alpha_i)\}_{\ell \in [L']}$.
     b) If $P_j \notin \mathcal{H}$ and P_j is corrupted, for each $\ell \in [L']$, $\mathcal{S}$ on behalf of P_i checks whether $g_j^{(\ell)}(\alpha_i) + z_j^{(\ell)}(\alpha_i) = \hat{sign}_j^{(\ell)}(\alpha_i)$.

3. When P_i accepts $t+1$ different P_j's $\{g_j^{(\ell)}(\alpha_i)\}_{\ell\in[L']}$, $\mathcal{S}$ considers that P_i gets his $\{f_i^{(\ell)}(x), w_i^{(\ell)}(x)\}_{\ell\in[L']}$.

- If P_i is corrupted: $\mathcal{S}$ follows the protocol to send $\{g_j^{(\ell)}(\alpha_i)\}_{\ell\in[L']}$ to P_i on behalf of each honest P_j.

Reconstructing column polynomials:
For each $P_i \in \mathcal{P}$:

- If P_i is honest:
  1. When P_i reconstructs $\{f_i^{(\ell)}(x)\}_{\ell\in[L']}$ and $\{w_i^{(\ell)}(x)\}_{\ell\in[L']}$, $\mathcal{S}$ on behalf of P_i follows the protocol to send $\{f_i^{(\ell)}(\alpha_j), w_i^{(\ell)}(\alpha_j)\}_{\ell\in[L']}$ to P_j, $j \in [n]$.
  2. For each $\ell \in [L']$, $\mathcal{S}$ computes $\tilde{z}_i^{\ell}(y) = P^{(\ell)}(\alpha_i, y) - w_i^{(\ell)}(y)$ on behalf of P_i.
  3. $\mathcal{S}$ waits to receive each P_j's $\{f_j^{(\ell)}(\alpha_i), w_j^{(\ell)}(\alpha_i)\}_{\ell\in[L']}$ to P_j.
     a) If P_j is honest, When P_i receives $\{f_j^{(\ell)}(\alpha_i), w_j^{(\ell)}(\alpha_i)\}_{\ell\in[L']}$ from P_j, $\mathcal{S}$ considers that P_i accepts P_j's $\{f_j^{(\ell)}(\alpha_i), w_j^{(\ell)}(\alpha_i)\}_{\ell\in[L']}$.
     b) If P_j is corrupted, for each $\ell \in [L']$, $\mathcal{S}$ follows the protocol to checks whether $f_j^{(\ell)}(\alpha_i) + w_j^{(\ell)}(\alpha_i) = Q^{(\ell)}(\alpha_i, \alpha_j)$.
  4. When $2t+1$ different P_j's $\{f_j^{(\ell)}(\alpha_i), w_j^{(\ell)}(\alpha_i)\}_{\ell\in[L']}$ passes the check, $\mathcal{S}$ interpolates $\{g_i^{(\ell)}(y), z_i^{(\ell)}(y)\}_{\ell\in[L']}$. If $z_i^{(\ell)}(y) = \tilde{z}_i^{\ell}(y)$ for each $\ell \in [L']$, $\mathcal{S}$ considers that P_i accepts the $\{g_i^{(\ell)}(y), z_{(\ell)}^{i}(y)\}_{\ell\in[L']}$.
- If P_i is corrupted: $\mathcal{S}$ follows the protocol to send $\{f_j^{(\ell)}(\alpha_i), w_j^{(\ell)}(\alpha_i)\}_{\ell\in[L']}$ to P_i on behalf of each honest P_j.

Fig. 20: Part-(3/3) of the simulator for the $\mathcal{F}_{\mathsf{ACSS}}$ when D is corrupted

Hybrid arguments:

$\mathbf{Hyb}_0$: In this hybrid, $\mathcal{S}$ receives the inputs of honest parties and runs the protocol honestly. This corresponds to the real-world scenario.

$\mathbf{Hyb}_1$: In this hybrid, during the Sharing Phase, $\mathcal{S}$ does the additional thing as following: Let $\mathcal{H}$ be the first $t+1$ honest parties in the set $\mathcal{M}$, for each $\ell \in [L']$, $\mathcal{S}$ reconstructs a degree-$(t, 2t)$ bivariate polynomial $Q^{(\ell)}(x, y)$ and a degree-$(2t, 2t)$ symmetric polynomial $P^{(\ell)}(x, y)$ s.t. $Q^{(\ell)}(\alpha_i, y) = sign_{i,1}^{(\ell)}(y) + sign_{i,2}^{(\ell)}(y)$ and $P^{(\ell)}(\alpha_i, y) = sign_{i,2}^{(\ell)}(y) + sign_{i,3}^{(\ell)}(y)$ for each $P_i \in \mathcal{H}$. Since $|\mathcal{H}| = t+1$, $\mathcal{S}$ can reconstruct a degree-$(t, 2t)$ bivariate polynomial $Q^{(\ell)}(x, y)$ and for a degree-$(2t, 2t)$ symmetric bivariate polynomial, $\mathcal{S}$ can also reconstruct $P^{(\ell)}(x, y)$. Since $\mathcal{S}$ does not do anything with these polynomials, it does not change the output distribution. Therefore, the output distributions of $\mathbf{Hyb}_1$ and $\mathbf{Hyb}_0$ are consistent.

$\mathbf{Hyb}_2$: In this hybrid, during the Completion Phase, When reconstructs the row polynomials, for each honest P_i, if $P_j \in \mathcal{H}$, $\mathcal{S}$ doesn't check $\{g_j^{(\ell)}(\alpha_i)\}_{\ell\in[L']}$

from P_j. $\mathcal{S}$ considers that when P_i receives $\{g_j^{(\ell)}(\alpha_i)\}_{\ell\in[L']}$ from P_j then P_i accepts them. Since $P_j \in \mathcal{H}$ is honest, P_j always sends points $\{g_j^{(\ell)}(\alpha_i)\}_{\ell\in[L']}$ that correspond to the polynomial $\{Q^{(\ell)}(x,y), P^{(\ell)}(x,y)\}_{\ell\in[L']}$ and $\mathcal{H} \subset \mathcal{M}$, P_i will always accepts P_j's $\{g_j^{(\ell)}(\alpha_i)\}_{\ell\in[L']}$. Therefore, the output distributions of $\mathbf{Hyb}_2$ and $\mathbf{Hyb}_1$ are consistent.

$\mathbf{Hyb}_3$: In this hybrid, during the Completion Phase, when reconstructs the row polynomials, for each honest P_i, $\mathcal{S}$ changes the way P_i checks the $\{g_j^{(\ell)}(\alpha_i)\}_{\ell\in[L']}$ sent by each corrupted P_j and $P_j \in \mathcal{M}$. Instead, $\mathcal{S}$ uses the pre-computed $\{\hat{sign}_j^{(\ell)}(y)\}_{\ell\in[L']}$ to check the whether $g_j^{(\ell)}(\alpha_i)+z_j^{(\ell)}(\alpha_i) = \hat{sign}_j^{(\ell)}(\alpha_i)$ for each $\ell \in [L']$. Since polynomials $\{P^{(\ell)}(x,y), Q^{(\ell)}(x,y)\}_{\ell\in[L']}$ are correct, this step is actually equivalent to the check in the protocol. Therefore, the output distributions of $\mathbf{Hyb}_3$ and $\mathbf{Hyb}_2$ are consistent.

$\mathbf{Hyb}_4$: In this hybrid, during the Completion Phase, when reconstructs the column polynomials, for each honest P_i, $\mathcal{S}$ changes the way P_i checks the $\{f_j^{(\ell)}(\alpha_i), w_j^{(\ell)}(\alpha_i)\}_{\ell\in[L']}$ sent by each honest P_j. Instead, $\mathcal{S}$ considers that when P_i receives $\{f_j^{(\ell)}(\alpha_i), w_j^{(\ell)}(\alpha_i)\}_{\ell\in[L']}$ from each honest P_j then accepts them. Since When P_i reconstructed $\{f_j^{(\ell)}(x), w_j^{(\ell)}(x)\}_{\ell\in[L']}$ and P_j honest, P_i always receives each honest P_j's $\{f_j^{(\ell)}(\alpha_i), w_j^{(\ell)}(\alpha_i)\}_{\ell\in[L']}$ that correspond to $P^{(\ell)}(x,y), Q^{(\ell)}(x,y))$, P_i will always accepts them. Therefore, the output distributions of $\mathbf{Hyb}_4$ and $\mathbf{Hyb}_3$ are consistent.

Note that $\mathbf{Hyb}_4$ is the ideal-world scenario, Π_{ACSS} statistically-securely computes $\mathcal{F}_{\mathsf{ACSS}}$ when D is corrupted. $\square$

C.3 Analysis of the Communication Complexity

Let's summarize the parameters in our Π_{ACSS} system. D categorizes his N input polynomials into L' groups. Within each group, every $t+1$ polynomials are combined to be represented by a single bivariate polynomial. Therefore, $N = (t+1)L'$. The $\mathcal{F}_{\mathsf{ARICP}}$ function will accept 3 vectors as input, where each vector has a length of $L = L' \cdot n$.

During the Sharing Phase: Since there are L' groups here with n parties. For each group, D dispatches a column polynomial of degree-$2t$ to each $P_i \in \mathcal{P}$. So, the total communication is $\mathcal{O}(L'n^2\kappa)$ bits across all parties. Subsequently, each $P_i \in \mathcal{P}$ broadcasts OK_i and $iAMj$ messages (with each OK_i and $iAMj$ requiring $\mathcal{O}(\log n)$ bits for encoding). Since each $P_i \in \mathcal{P}$ broadcasts $iAMj$ at most n times, the total communication of the broadcast is $\mathcal{O}(n^4 \log n)$ bits. Ultimately, D broadcasts three sets $\mathcal{V}, \{\mathcal{V}_j\}_{j\in[n]}$ and $\mathcal{M}$. Each set consumes $\mathcal{O}(n^4 \log n)$ bits. Here we ignore the communication of $\mathcal{F}_{\mathsf{ARICP}}$, so the Sharing Phase requires $\mathcal{O}(L'n^2\kappa + n^4 \log n)$ bits communication. Since $L'n = L$, so the communication overhead is $\mathcal{O}(Ln\kappa + n^4 \log n)$ bits.

During the verification Phase: If $\mathcal{F}_{\mathsf{ARICP}}$ is not taken into account, there is no need for any communication here, so the communication overhead is 0 bit.

During the Completion Phase: Each $P_i \in \mathcal{P}$ needs to sends $g_i^\ell(\alpha_j), w_i^\ell(\alpha_j)$ and $f_i^\ell(\alpha_j)$ to each P_j, resulting in a total communication of $\mathcal{O}(L'n\kappa)$ bits. Since $L'n = L$, so the communication overhead is $\mathcal{O}(L\kappa)$ bits.

In the end, we consider the communication cost of ARICP. For each $P_h \in \mathcal{M}$, according to Theorem 2, the communication of realizing each $\mathcal{F}_{\mathsf{ARICP}}(P_h, D)$ is $\mathcal{O}((n+1)(L\kappa+n^2\kappa^2))$ bits. Therefore, the total communication cost is $\mathcal{O}(n^2L\kappa+n^4\kappa^2)$ bits.

Therefore, the protocol Π_{ACSS} requires communication of $\mathcal{O}(n^4\kappa^2 + n^2L\kappa + n^4\log n)$ bits. Since $(L = nL' = \mathcal{O}(N))$, the protocol Π_{ACSS} requires communication of $\mathcal{O}(Nn^2\kappa + n^4\kappa^2 + n^4\log n)$ bits.