Optimally-resilient Unconditionally-secure Asynchronous Multi-party Computation Revisited

Ashish Choudhury
International Institute of Information Technology Bangalore, India
ashish.choudhury@iiitb.ac.in

Abstract

In this paper, we present an *optimally-resilient*, unconditionally-secure *asynchronous multi-party computation* (AMPC) protocol for n parties, tolerating a *computationally unbounded* adversary, capable of corrupting up to $t < \frac{n}{3}$ parties. Our protocol needs a communication of $\mathcal{O}(n^4)$ field elements per multiplication gate. This is to be compared with previous best AMPC protocol (Patra et al, ICITS 2009) in the same setting, which needs a communication of $\mathcal{O}(n^5)$ field elements per multiplication gate. To design our protocol, we present a simple and highly efficient *asynchronous verifiable secret-sharing* (AVSS) protocol, which is of independent interest.

keywords: Byzantine faults, secret-sharing, unconditional-security, privacy.

2012 ACM Subject Classification Security and privacy → Information-theoretic techniques; Theory of computation → Distributed algorithms; Theory of computation → Cryptographic protocols

Keywords and phrases Verifiable Secret-sharing, Secure MPC, Fault-tolerance

Digital Object Identifier 10.4230/LIPIcs...

Funding *Ashish Choudhury*: This research is an outcome of the R & D work undertaken in the project under the Visvesvaraya PhD Scheme of Ministry of Electronics & Information Technology, Government of India, being implemented by Digital India Corporation (formerly Media Lab Asia).

1 Introduction

Secure *multi-party computation* (MPC) [22, 14, 7, 20] is a fundamental problem, both in cryptography as well as distributed computing. Informally a MPC protocol allows a set of n mutually-distrusting parties to perform a joint computation on their inputs, while keeping their inputs as private as possible, even in the presence of an adversary Adv who can corrupt any t out of these n parties. Ever since its inception, the MPC problem has been widely studied in various flavours (see for instance, [15, 13, 17, 16] and their references). While the MPC problem has been pre-dominantly studied in the *synchronous* communication model where the message delays are bounded by *known* constants, the progress in the design of efficient asynchronous MPC (AMPC) protocols is rather slow. In the latter setting, the communication channels may have arbitrary but finite delays and deliver messages in any arbitrary order, with the only guarantee that all sent messages are *eventually* delivered. The main challenge in designing a fully asynchronous protocol is that it is impossible for an honest party to distinguish between a slow but honest sender (whose messages are delayed) and a corrupt sender (who did not send any message). Hence, at any stage, a party cannot wait to receive messages from all the parties (to avoid endless waiting) and so communication from t (potentially honest) parties may have to be ignored.

In this work, we consider a setting where Adv is *computationally unbounded*. In this setting, we have two class of AMPC protocols. *Perfectly-secure* AMPC protocols give the security guarantees without any error, while *unconditionally-secure* AMPC protocols give the security guarantees with probability at least $1 - \epsilon_{\mathsf{AMPC}}$, where ϵ_{AMPC} is any given (non-zero) error parameter. The *optimal resilience* for perfectly-secure AMPC is $t < n/4$ [6], while that for unconditionally-secure AMPC it is $t < n/3$ [8]. While there are quite a few works which consider optimally-resilient perfectly-secure AMPC protocol [5, 19], not too much attention has been paid to the design of efficient unconditionally-secure AMPC protocol with the optimal resilience of $t < \frac{n}{3}$. In this work, we make inroads in this direction, by presenting a simple and efficient unconditionally-secure AMPC protocol.

1.1 Our Results and Comparison with the Existing Works

In any unconditionally-secure AMPC protocol (including ours), the function to be computed is abstracted as a publicly-known *ckt* over some finite field $\mathbb{F}$, consisting of addition and multiplication gates over $\mathbb{F}$ and the goal is to let the parties jointly and"securely" evaluate *ckt*. The field $\mathbb{F}$ is typically the Galois field $\mathrm{GF}(2^{\kappa})$, where κ depends upon[1] ϵ_{AMPC}. The *communication complexity* of any AMPC protocol is dominated by the communication needed to evaluate the multiplication gates in *ckt* (see the sequel for details). Consequently, the focus of any generic AMPC protocol is to improve the communication required for evaluating the multiplication gates in *ckt*. The following table summarizes the communication complexity of the existing AMPC protocols with the optimal resilience of $t < \frac{n}{3}$ and our protocol.

Reference	Communication Complexity (in bits) for Evaluating a Single Multiplication Gate
[8]	$\mathcal{O}(n^{11}\kappa^4)$
[18]	$\mathcal{O}(n^5\kappa)$
This paper	$\mathcal{O}(n^4\kappa)$

We follow the standard approach of shared circuit-evaluation, where each value during the evaluation of *ckt* is Shamir secret-shared [21] among the parties, with threshold t. Informally, a value s is said to be Shamir-shared with threshold t, if there exists some degree-t polynomial with s as its constant term and every party P_i holds a distinct evaluation of this polynomial as its share. In the AMPC protocol, each party P_i *verifiably* secret-shares its input for *ckt*. The verifiability here ensures that if the parties terminate this step, then some value is indeed Shamir secret-shared among the parties on the behalf of P_i. To verifiably secret-share its input, each party executes an instance of *asynchronous verifiable secret-sharing* (AVSS). Once the inputs of the parties are secret-shared, the parties then evaluate each gate in *ckt*, maintaining the following invariant: if the gate inputs are secret-shared, then the parties try to obtain a secret-sharing of the gate output. Due to the linearity of Shamir secret-sharing, maintaining the invariant for addition gates do not need any interaction among the parties. However, for maintaining the invariant for multiplication gates, the parties need to interact with each other and hence the onus is rightfully shifted to minimize this cost. For evaluating the multiplication gates, the parties actually deploy the standard Beaver's circuit-randomization technique [3]. The technique reduces the cost of evaluating a multiplication gate to that of publicly reconstructing two secret-shared values, provided the parties have access to a Shamir-shared random and multiplication triple (a, b, c), where $c = a \cdot b$. The shared multiplication triples are generated in advance in a bulk in a circuit-independent pre-processing phase, using the efficient framework proposed in [11]. The framework allows to efficiently and verifiably generate Shamir-shared random multiplication triples, using any given AVSS protocol. Once all the gates in *ckt* are evaluated and the circuit-output

[1] Instead of the Galois field, one can also use any sufficiently large field, to bound the error probability by ϵ_{AMPC}.

is available in a secret-shared fashion, the parties publicly reconstruct this value. Since all the values (except the circuit output) during the entire computation remains Shamir-shared with threshold t, the privacy of the computation follows from the fact that during the shared circuit-evaluation, for each value in ckt, Adv learns at most t shares, which are independent of the actual shared value. While the AMPC protocols of [8] and [18] also follow the above blue-print of shared circuit-evaluation, the difference is in the underlying AVSS protocol.

AVSS [6, 8] is a well-known and important primitive in secure distributed computing. On a very high level, an AVSS protocol enhances the security of Shamir secret-sharing against a *malicious* adversary (Shamir secret-sharing achieves its properties only in the *passive* adversarial model, where even the corrupt parties honestly follow protocol instructions). The existing unconditionally-secure AVSS protocols with $t < n/3$ [8, 18] need high communication. This is because there are significant number of obstacles in designing unconditionally-secure AVSS with exactly $n = 3t + 1$ parties (which is the least value of n with $t < n/3$). The main challenge is to ensure that *all honest* parties obtain their shares of the secret. We call an AVSS protocol guaranteeing this "completeness" property as *complete* AVSS. However, in the asynchronous model, it is impossible to directly get the confirmation of the receipt of the share from each party, as corrupt parties may never respond. To get rid off this difficulty, [8] introduces a "weaker" form of AVSS which guarantees that the underlying secret is verifiably shared only among a set of $n - t$ parties and up to t parties may not have their shares. To distinguish this type of AVSS from complete AVSS, the latter category of AVSS is termed an *asynchronous complete secret-sharing* (ACSS) in [8], while the weaker version of AVSS is referred as just AVSS[2]. Given any AVSS protocol, [8] shows how to design an ACSS protocol using n instances of AVSS. An AVSS protocol with $t < n/3$ is also presented in [8]. With a communication complexity of $\Omega(n^9\kappa)$ bits, the protocol is highly expensive. This AVSS protocol when used in their ACSS protocol requires a communication complexity $\Omega(n^{10}\kappa)$. Apart from being communication expensive, the AVSS of [8] involves a lot of asynchronous primitives such as ICP, A-RS, AWSS and Two & Sum AWSS. In [18], a simplified AVSS protocol with communication complexity $\mathcal{O}(n^3\kappa)$ bits is presented, based on only few primitives, namely ICP and AWSS. This AVSS is then converted into an ACSS in the same way as [8], making the communication complexity of their ACSS $\mathcal{O}(n^4\kappa)$ bits.

In this work, we further improve upon the communication complexity of the ACSS of [18]. We first design a new AVSS protocol with a communication complexity $\mathcal{O}(n^2\kappa)$ bits. Then using the approach of [8], we obtain an ACSS protocol with communication complexity $\mathcal{O}(n^3\kappa)$ bits. Our AVSS protocol is conceptually simpler and is based on just the ICP primitive and hence easy to understand. Moreover, since we avoid the usage of AWSS in our AVSS, we get a saving of $\Theta(n)$ in the communication complexity, compared to [18] (the AVSS of [18] invokes n instances of AWSS, which is not required in our AVSS).

Paper Organization: As the main contribution of this work is the design of a new AVSS protocol, we mainly focus on the AVSS protocol and the proof of its properties in Section 3. The upgradation from AVSS to ACSS follows the blueprint of [8, 18] and given in Section 4. In Section 5 we present a high level discussion of our AMPC protocol.

[2] We stress that the weaker form of AVSS is not sufficient for the shared circuit-evaluation. This is because the set of $n - t$ share-holders might be different for different shared values.

2 Preliminaries, Definitions and Existing Tools

We assume a set of n parties $\mathcal{P} = \{P_1, \ldots, P_n\}$, connected by pair-wise private and authentic asynchronous channels. A *computationally unbounded* adversary Adv can corrupt any $t < n/3$ parties. We assume $n = 3t + 1$, so that $t = \Theta(n)$. In our protocols, all computation are done over a Galois field $\mathbb{F} = \mathrm{GF}(2^\kappa)$. The parties want to compute a function f over $\mathbb{F}$, represented by a publicly known arithmetic circuit ckt over $\mathbb{F}$. For simplicity and without loss of generality, we assume that each party $P_i \in \mathcal{P}$ has a single input $x^{(i)}$ for the function f and there is a single function output $y = f(x^{(1)}, \ldots, x^{(n)})$, which is supposed to be learnt by all the parties. Apart from the input and output gates, ckt consists of 2-input gates of the form $g = (x, y, z)$, where x and y are the inputs and z is the output. The gate g can be either an addition gate (i.e. $z = x + y$) or a multiplication gate (i.e. $z = x \cdot y$). The circuit ckt consists of c_M multiplication gates. We require $|\mathbb{F}| > n$. Additionally, we need the condition $\frac{n^5\kappa}{2^\kappa - (3c_M+1)} \leq \epsilon_{\mathsf{AMPC}}$ to hold. Looking ahead, this will ensure that the error probability of our AMPC protocol is upper bounded by ϵ_{AMPC}. We assume that $\alpha_1, \ldots, \alpha_n$ are distinct, non-zero elements from $\mathbb{F}$, where α_i is associated with P_i as the "evaluation point". By *communication complexity* of a protocol, we mean the total number of bits communicated by the honest parties in the protocol. While denoting the communication complexity, we use the term $\mathcal{BC}(\ell)$ to denote that ℓ bits are broadcasted in the protocol.

2.1 Definitions

A degree-d *univariate polynomial* is of the form $f(x) = a_0 + \ldots + a_d x^d$, where each $a_i \in \mathbb{F}$. A degree-(ℓ, m) *bivariate polynomial* $F(x, y)$ is of the form $F(x, y) = \sum_{i,j=0}^{i=\ell, j=m} r_{ij} x^i y^j$, where each $r_{ij} \in \mathbb{F}$. Let $f_i(x) \stackrel{\text{def}}{=} F(x, \alpha_i), g_i(y) \stackrel{\text{def}}{=} F(\alpha_i, y)$. We call $f_i(x)$ and $g_i(y)$ as i^{th} *row* and *column polynomial* respectively of $F(x, y)$ and often say that $f_i(x), g_i(y)$ lie on $F(x, y)$. We use the following well-known lemma, which states that if there are "sufficiently many" degree-t univariate polynomials which are "pair-wise consistent", then there exists a unique degree-(t, t) bivariate polynomial, passing through these univariate polynomials.

▶ **Lemma 1 (Pair-wise Consistency Lemma [10, 1]).** *Let $f_{i_1}(x), \ldots, f_{i_\ell}(x), g_{j_1}(y), \ldots, g_{j_m}(y)$ be degree-t polynomials where $\ell, m \geq t + 1$ and $i_1, \ldots, i_\ell, j_1, \ldots, j_m \in \{1, \ldots, n\}$. Moreover, let for every $i \in \{i_1, \ldots, i_\ell\}$ and every $j \in \{j_1, \ldots, j_m\}$, $f_i(\alpha_j) = g_j(\alpha_i)$ holds. Then there exists a unique degree-(t, t) bivariate polynomial, say $\overline{F}(x, y)$, such that the row polynomials $f_{i_1}(x), \ldots, f_{i_\ell}(x)$ and the column polynomials $g_{j_1}(y), \ldots, g_{j_m}(y)$ lie on $\overline{F}(x, y)$.*

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

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

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

Note that complete t-sharings are *linear*: given $[a]_t, [b]_t$, then $[a + b]_t = [a]_t + [b]_t$ and $[c \cdot a]_t = c \cdot [a]_t$ hold, for any public $c \in \mathbb{F}$.

▶ **Definition 3** (**Asynchronous Complete Secret Sharing (ACSS)** [8, 18]). *Let* CSh *be an asynchronous protocol, where there is a designated dealer* $\mathsf{D} \in \mathcal{P}$ *with a private input* $S = (s^{(1)}, \ldots, s^{(L)}) \in \mathbb{F}^L$*. Then* CSh *is a* $(1 - \epsilon_{\mathsf{ACSS}})$ *ACSS protocol for a given error parameter* ϵ_{ACSS}*, if the following requirements hold for every possible* Adv*.*

- **Termination***: Except with probability* ϵ_{ACSS}*, the following holds.* **(a)***: If* D *is honest and all honest parties participate in* CSh*, then each honest party eventually terminates* CSh*.* **(b)***: If some honest party terminates* CSh*, then every other honest party eventually terminates* CSh*.*
- **Correctness***: If the honest parties terminate* CSh*, then except with probability* ϵ_{ACSS}*, there exists some* $\overline{S} \in \mathbb{F}^L$ *which is completely* t*-shared, where* $\overline{S} = S$ *for an honest* D*.*
- **Privacy***: If* D *is honest, then the view of* Adv *during* CSh *is independent of* S*.*

We next give the definition of *asynchronous information-checking protocol* (AICP), which will be used in our ACSS protocol. An AICP involves three entities: a *signer* $\mathsf{S} \in \mathcal{P}$, an *intermediary* $\mathsf{I} \in \mathcal{P}$ and a *receiver* $\mathsf{R} \in \mathcal{P}$, along with the set of parties $\mathcal{P}$ acting as *verifiers*. Party S has a private input $\mathcal{S}$. An AICP can be considered as information-theoretically secure analogue of digital signatures, where S gives a "signature" on $\mathcal{S}$ to I, who eventually reveals it to R, claiming that it got the signature from S. The protocol proceeds in the following three phases, each of which is implemented by a dedicated sub-protocol.

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

▶ **Definition 4** (**AICP** [18]). *A triplet of protocols* $(\mathsf{Gen}, \mathsf{Ver}, \mathsf{RevPriv})$ *where* S *has a private input* $\mathcal{S} \in \mathbb{F}^L$ *for* Gen *is called a* $(1 - \epsilon_{\mathsf{AICP}})$*-secure AICP, for a given error parameter* ϵ_{AICP}*, if the following holds for every possible* Adv*.*

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

Note that we do not put any termination condition for AICP. Looking ahead, we use AICP as a primitive in our ACSS protocol and the termination conditions in our instantiation of ACSS ensure that the underlying instances of AICP also terminate.

Finally, we give the definition of two-level t-sharing with IC-signatures, which is the data structure generated by our AVSS protocol, as well as by the AVSS protocols of [8, 18]. This sharing is an enhanced version of t-sharing, where each share is further t-shared. Moreover, for the purpose of authentication, each second-level share is signed.

▶ **Definition 5** (**Two-level t-Sharing with IC-signatures** [18]). $S = (s^{(1)}, \ldots, s^{(L)})$ *is said to be two-level* t*-shared with IC-signatures if there exists a set* $\mathcal{C} \subseteq \mathcal{P}$ *with* $|\mathcal{C}| \geq n - t$ *and a set* $\mathcal{C}_j \subseteq \mathcal{P}$ *for each* $P_j \in \mathcal{C}$ *with* $|\mathcal{C}_j| \geq n - t$*, such that the following conditions hold.*

- *Each $s^{(k)} \in S$ is t-shared among $\mathcal{C}$, with each party $P_j \in \mathcal{C}$ holding its primary-share $s_j^{(k)}$.*
- *For each primary-share holder $P_j \in \mathcal{C}$, there exists a set of parties $\mathcal{C}_j \subseteq \mathcal{P}$, such that each primary-share $s_j^{(k)}$ is t-shared among $\mathcal{C}_j$, with each $P_i \in \mathcal{C}_j$ holding the secondary-share $s_{j,i}^{(k)}$ of the primary-share $s_j^{(k)}$.*
- *Each primary-share holder $P_j \in \mathcal{C}$ holds $\mathsf{ICSig}(P_i \to P_j, (s_{j,i}^{(1)}, \ldots, s_{j,i}^{(L)}))$, corresponding to each honest secondary-share holder $P_i \in \mathcal{C}_j$.*

We stress that the $\mathcal{C}_j$ sets might be different for each $P_j \in \mathcal{C}$. We finally define AMPC.

▶ **Definition 6 (Unconditionally-secure AMPC [8]).** *Let $f : \mathbb{F}^n \to \mathbb{F}$ be a publicly known function where each P_i has a private input $x^{(i)} \in \mathbb{F}$. Any AMPC consists of three stages. In the first stage, each P_i commits its input. Even if P_i is corrupt, if it completes this step, then it is committed to some value $\overline{x}^{(i)}$ (not necessarily $x^{(i)}$), where $\overline{x}^{(i)} = x^{(i)}$ for an honest P_i. Then the parties agree on a common subset, say $\mathcal{R}$, of $n - t$ committed inputs. In the last stage, the parties compute $f(\overline{x}^{(1)}, \ldots, \overline{x}^{(n)})$, where $\overline{x}^{(i)} = 0$ if $P_i \notin \mathcal{R}$.*

An asynchronous protocol Π among $\mathcal{P}$ for computing f is called a $(1-\epsilon_{\mathsf{AMPC}})$ unconditionally-secure AMPC protocol, if it satisfies the following conditions for every possible Adv.

- **Termination**: *If all honest parties participate in Π, then the honest parties eventually terminates Π with probability at least $1 - \epsilon_{\mathsf{AMPC}}$.*
- **Correctness**: *Honest parties output $f(\overline{x}^{(1)}, \ldots, \overline{x}^{(n)})$, with probability at least $1 - \epsilon_{\mathsf{AMPC}}$.*
- **Privacy**: *The view of the* Adv *is independent of the inputs of the honest parties in $\mathcal{R}$.*

2.2 Existing Asynchronous Protocols Used in Our ACSS protocol

We use the AICP protocol of [18] (see Appendix A for the details), where $\epsilon_{\mathsf{AICP}} \leq \frac{n\kappa}{2^\kappa-(L+1)}$ and where Gen, Ver and $\mathsf{RevPriv}$ has communication complexity of $\mathcal{O}((L + n\kappa)\kappa)$, $\mathcal{O}(n\kappa^2)$ and $\mathcal{O}((L + n\kappa)\kappa)$ bits respectively. In the AICP, any party in $\mathcal{P}$ can play the role of S, I and R. In the rest of the paper, we use the following terms which using the AICP of [18].

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

We also use the *asynchronous broadcast* protocol of Bracha [9], which allows a *sender* $\mathbf{S} \in \mathcal{P}$ to identically send a message m to all the parties, even in the presence of Adv. If $\mathbf{S}$ is *honest*, then all honest parties eventually terminate with output m. If $\mathbf{S}$ is *corrupt* but some honest party terminates with an output $m^\star$, then eventually every other honest party terminates with output $m^\star$. The protocol has communication complexity $\mathcal{O}(n^2 \cdot \ell)$ bits, if sender's message m consists of ℓ bits. We use the term P_i *broadcasts* m to mean that P_i acts as $\mathbf{S}$ and invokes an instance of Bracha's protocol to broadcast m. Similarly, the term P_j *receives* m *from the broadcast of* P_i means that P_j (as a receiver) completes the execution of P_i's broadcast (namely the instance of broadcast protocol where P_i is $\mathbf{S}$), with m as output.

3 Verifiably Generating Two-Level t-sharing with IC Signatures

We present a protocol Sh, which will be used as a sub-protocol in our ACSS scheme. In the protocol, there exists a designated $\mathsf{D} \in \mathcal{P}$ with a private input $S \in \mathbb{F}^L$ and the goal is to *verifiably* generate a two-level t-sharing with IC signatures of S. The verifiability allows the parties to publicly verify if D behaved honestly, while preserving the privacy of S for an *honest* D. We first present the protocol Sh assuming that D has a single value for sharing, that is $L = 1$. The modifications needed to share L values are straight-forward.

To share s, D hides s in the constant term of a random degree-(t, t) bivariate polynomial $F(x, y)$. The goal is then to let D distribute the row and column polynomials of $F(x, y)$ to respective parties and then publicly verify if D has distributed consistent row and column polynomials to sufficiently many parties, which lie on a single degree-(t, t) bivariate polynomial, say $\bar{F}(x, y)$, which is considered as D's *committed* bivariate polynomial (if D is honest then $\bar{F}(x, y) = F(x, y)$ holds). Once the existence of an $\bar{F}(x, y)$ is confirmed, the next goal is to let each P_j who holds its row polynomial $\bar{F}(x, \alpha_j)$ lying on $\bar{F}(x, y)$, get signature on $\bar{F}(\alpha_i, \alpha_j)$ values from at least $n - t$ parties P_i. Finally, once $n - t$ parties P_j get their row polynomials signed, it implies the generation of two-level t-sharing of $\bar{s} = \bar{F}(0, 0)$ with IC signatures. Namely, $\bar{s}$ will be t-shared through degree-t column polynomial $\bar{F}(0, y)$. The set of signed row-polynomial holders P_j will constitute the set $\mathcal{C}$, where P_j holds the primary-share $\bar{F}(0, \alpha_j)$, which is the constant term of its row polynomial $\bar{F}(x, \alpha_j)$. And the set of parties P_i who signed the values $\bar{F}(\alpha_i, \alpha_j)$ for P_j constitute the $\mathcal{C}_j$ set with P_i holding the secondary-share $\bar{F}(\alpha_i, \alpha_j)$, thus ensuring that the primary-share $\bar{F}(0, \alpha_j)$ is t-shared among $\mathcal{C}_j$ through degree-t row polynomial $\bar{F}(x, \alpha_j)$. For a pictorial depiction of how the values on D's bivariate polynomial constitute the two-level t-sharing of its constant term, see Fig 1.

Figure 1 Two-level t-sharing with IC signatures of $s = F(0, 0)$. Here we assume that $\mathcal{C} = \{P_1, \ldots, P_{2t+1}\}$ and $\mathcal{C}_j = \{P_1, \ldots, P_{2t+1}\}$ for each $P_j \in \mathcal{C}$. Party P_j will possess all the values along the j^{th} row, which constitute the row polynomial $f_j(x) = F(x, \alpha_j)$. Column-wise, P_i possesses the values in the column labelled with P_i, which lie on the column polynomial $g_i(y) = F(\alpha_i, y)$. Party P_j will possess P_i's information-checking signature on the common value $f_j(\alpha_i) = F(\alpha_i, \alpha_j) = g_i(\alpha_j)$ between P_j's row polynomial and P_i's column polynomial, denoted by blue color.

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

The above stated goals are achieved in four stages, each of which is implemented by executing the steps in one of the highlighted boxes in Fig 2 (the purpose of the steps in each box appears as a comment outside the box). To begin with, D distributes the column polynomials to respective parties (the row polynomials are currently retained) and tries to get all the row polynomials signed by a *common set* $\mathcal{M}$ of $n - t$ column holders, by asking each of them to sign the common values between their column polynomials and row polynomials. That is, each P_i is given its column polynomial $g_i(y) = F(\alpha_i, y)$ and is asked to sign the values f_{ji} for $j = 1, \ldots, n$, where $f_{ji} = f_j(\alpha_i)$ and $f_j(x) = F(x, \alpha_j)$ is the j^{th} row polynomial. Party P_i signs the values $f_{1i}, \ldots, f_{ni}$ for D after verifying that all of them lie on its column polynomial $g_i(y)$ and then publicly announces the issuance of signatures to D by broadcasting a `MC` message (standing for "matched column"). Once a set $\mathcal{M}$ of $n - t$ parties broadcasts `MC` message, it confirms that the row polynomials held by D and the column polynomials of the parties in $\mathcal{M}$ together lie on a single degree-(t, t) bivariate polynomial (due to the pair-wise consistency Lemma 1). This also confirms that D is committed to a single (yet unknown) degree-(t, t) bivariate polynomial. The next stage is to let D distribute the row polynomials of this committed bivariate polynomial to individual parties.

To prevent a potentially corrupt D from distributing arbitrary polynomials to the parties as row polynomials, D actually sends the signed row polynomials to the individual parties, where the values on the row polynomials are signed by the parties in $\mathcal{M}$. Namely, to distribute the row polynomial $f_j(x)$ to P_j, D reveals the $f_j(\alpha_i)$ values to P_j, signed by the parties $P_i \in \mathcal{M}$. The presence of the signatures ensure that D reveals the correct $f_j(x)$ polynomial to P_j, as there are at least $t+1$ honest parties in $\mathcal{M}$, whose signed values uniquely define $f_j(x)$. Upon the receipt of correctly signed row polynomial, P_j publicly announces it by broadcasting a MR message (standing for "matched row"). The next stage is to let such parties P_j obtain "fresh" signatures on $n-t$ values of $f_j(x)$ by at least $n-t$ parties $\mathcal{C}_j$. We stress that the signatures of the parties in $\mathcal{M}$ on the values of $f_j(x)$, which are revealed by D cannot be "re-used" and hence $\mathcal{M}$ cannot be considered as $\mathcal{C}_j$, as IC-signatures are not "transferable" and those signatures were issued to D and not to P_j. We also stress that the parties in $\mathcal{M}$ cannot be now asked to re-issue fresh signatures on P_j's row polynomial, as corrupt parties in $\mathcal{M}$ may now not participate honestly during this process. Hence, P_j has to ask for the fresh signatures on $f_j(x)$ from every potential party.

The process of P_j getting $f_j(x)$ freshly signed can be viewed as P_j recommitting its received row polynomial to a set of $n-t$ column-polynomial holders. However, extra care has to be taken to prevent a potentially corrupt P_j from getting fresh signatures on arbitrary values, which do not lie in $f_j(x)$. This is done as follows. Party P_i on receiving a "signature request" for f_{ji} from P_j signs it, only if it lies on P_i's column polynomial; that is $f_{ji} = g_i(\alpha_j)$ holds. Then after receiving the signature from P_i, party P_j publicly announces the same. Now the condition for including P_i to $\mathcal{C}_j$ is that apart from P_j, there should exist at least $2t$ other parties P_k who has broadcasted MR messages and who also got their respective row polynomials signed by P_i. This ensures that there are total $2t+1$ parties who broadcasted MR messages and whose row polynomials are signed by P_i. Now among these $2t+1$ parties, at least $t+1$ parties P_k are honest, whose row polynomials $f_k(x)$ lie on D's committed bivariate polynomial. Since these $t+1$ parties got signature on $f_k(\alpha_i)$ values from P_i, this further implies that $f_k(\alpha_i) = g_i(\alpha_k)$ holds for these $t+1$ honest parties P_k, further implying that P_i's column polynomial $g_i(y)$ also lies on D's committed bivariate polynomial. Now since $f_{ji} = g_i(\alpha_j)$ holds for P_j as well, it implies that the value which P_j got signed by P_i is $g_i(\alpha_j)$, which is the same as $f_j(\alpha_i)$. Finally, If D finds that the set $\mathcal{C}_j$ has $n-t$ parties, then it includes P_j in the $\mathcal{C}$ set, indicating that P_j has recommitted the correct $f_j(x)$ polynomial.

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

We stress that there is *no* termination condition in Sh. The protocol will be used as a sub-protocol in our ACSS and terminating conditions of ACSS will ensure that all underlying instances of Sh terminate, if ACSS terminates. Protocol Sh is presented in Fig 2.

Comparison with the AVSS Protocol of [18]. The sharing phase protocol of the AVSS of [18] also uses a similar four-stage approach as ours. However, the *difference* is in the first two stages. Namely, to ensure that D is committed to a single bivariate polynomial, each row polynomial $f_j(x)$ is first shared by D using an instance of *asynchronous weak secret-sharing* (AWSS) and once the commitment is confirmed, each polynomial $f_j(x)$ is later reconstructed towards the corresponding designated party P_j. There are n instances of AWSS involved, where each such instance is further based on distributing shares lying on a degree-(t, t) bivariate polynomial. Consequently, the resultant AVSS protocol becomes involved. We do not involve any AWSS instances for confirming D's commitment to a single bivariate polynomial. Apart from giving us a saving of $\Theta(n)$ in communication complexity, it also makes the protocol conceptually much simpler.

We next proceed to prove the properties of protocol Sh protocol. In the proofs, we use the fact that the error probability of a single instance of AICP in Sh is ϵ_{AICP}, where $\epsilon_{\mathsf{AICP}} \leq \frac{n\kappa}{2^\kappa - 2}$, which is obtained by substituting $L = 1$ in the AICP of [18].

▶ **Lemma 7.** *In protocol* Sh*, if* D *is honest, then except with probability* $n^2 \cdot \epsilon_{\mathsf{AICP}}$*, all honest parties are included in the* $\mathcal{C}$ *set. This further implies that* D *eventually finds a valid* $\mathcal{C}$ *set.*

Proof. Since D is *honest*, each *honest* P_i eventually receives the degree-t column polynomial $g_i(y)$ from D. Moreover, P_i also receives the values f_{ji} from D for signing, such that $f_{ji} = g_i(\alpha_j)$ holds. Furthermore, P_i eventually gives the signatures on these values to D and broadcasts $\mathtt{MC}_i$. As there are at least $2t + 1$ honest parties who broadcast $\mathtt{MC}_i$, it implies that D eventually finds a set $\mathcal{M}$ of size $2t + 1$ and broadcasts the same.

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

Finally, consider an arbitrary pair of *honest* parties P_i, P_j. Since D is *honest*, the condition $f_j(\alpha_i) = g_i(\alpha_j)$ holds. Now P_i eventually receives $f_{ji} = f_j(\alpha_i)$ from P_j for signing and finds that $f_{ji} = g_i(\alpha_j)$ holds and hence gives the signature $\mathsf{ICSig}(P_i \to P_j, f_{ji})$ to P_j. Consequently, P_j eventually broadcasts $(\mathtt{SR}_j, P_i)$. As there are at least $2t + 1$ honest parties P_k, who eventually broadcast $(\mathtt{SR}_k, P_i)$, it follows that P_i is eventually included in the set $\mathcal{C}_j$. As there are at least $2t + 1$ honest parties, the set $\mathcal{C}_j$ eventually becomes of size $2t + 1$ and hence P_j is eventually included in $\mathcal{C}$. ◀

▶ **Lemma 8.** *In protocol* Sh*, if some honest party receives a valid* $\mathcal{C}$ *set from* D*, then every other honest party eventually receives the same valid* $\mathcal{C}$ *set from* D*.*

Proof. Since the $\mathcal{C}$ set is broadcasted, it follows from the properties of broadcast that all honest parties will receive the same $\mathcal{C}$ set, if at all D broadcasts any $\mathcal{C}$ set. Now it is easy to see that if a broadcasted $\mathcal{C}$ set is found to be valid by some *honest* party P_m, then it will be considered as valid by every other honest party. This is because in Sh the validity conditions for $\mathcal{C}$ which hold for P_m will eventually hold for every other honest party. ◀

Figure 2 Two-level secret-sharing with IC signatures of a single secret.

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

%Distribution of values and identification of signed column polynomials.

- **Distribution of Column Polynomials and Common Values on Row Polynomials by** D: The following code is executed only by D.
  - Select a random degree-(t, t) bivariate polynomial $F(x, y)$ over $\mathbb{F}$, such that $F(0, 0) = s$.
  - Send $g_j(y) = F(\alpha_j, y)$ to each $P_j \in \mathcal{P}$. And send $f_j(\alpha_i)$ to each $P_i \in \mathcal{P}$, where $f_j(x) = F(x, \alpha_j)$.
- **Signing Common Values on Row Polynomials for** D: Each $P_i \in \mathcal{P}$ (including D) executes the following code.
  - Wait to receive a degree-t column polynomial $g_i(y)$ and for $j = 1, \ldots, n$ the values f_{ji} from D.
  - On receiving the values from D, give $\mathsf{ICSig}(P_i \to \mathsf{D}, f_{ji})$ to D for $j = 1, \ldots, n$ and broadcast the message $\mathtt{MC}_i$, provided $f_{ji} = g_i(\alpha_j)$ holds for each $j = 1, \ldots, n$.
- **Identifying Signed Column Polynomials**: The following code is executed only by D:
  - Include P_i to an accumulative set $\mathcal{M}$ (initialized to $\emptyset$), if $\mathtt{MC}_i$ is received from the broadcast of P_i and D received $\mathsf{ICSig}(P_i \to \mathsf{D}, f_{ji})$ from P_i, for each $j = 1, \ldots, n$.
  - Wait till $|\mathcal{M}| = 2t + 1$. Once $|\mathcal{M}| = 2t + 1$, then broadcast $\mathcal{M}$.

% Distribution of signed row polynomials by D and verification by the parties.

- **Revealing Row Polynomials to Respective Parties**: for $j = 1, \ldots, n$, D reveals $\mathsf{ICSig}(P_i \to \mathsf{D}, f_{ji})$ to P_j, for each $P_i \in \mathcal{M}$.
- **Verifying the Consistency of Row Polynomials Received from** D: Each $P_j \in \mathcal{P}$ (including D) broadcasts $\mathtt{MR}_j$, if the following holds.
  - P_j received an $\mathcal{M}$ with $|\mathcal{M}| = 2t + 1$ from D and $\mathtt{MC}_i$ from each $P_i \in \mathcal{M}$.
  - P_j accepted $\{\mathsf{ICSig}(P_i \to \mathsf{D}, f_{ji})\}_{P_i \in \mathcal{M}}$ and $\{(\alpha_i, f_{ji})\}_{P_i \in \mathcal{M}}$ lie on a degree-t polynomial $f_j(x)$.

%Recommitment of row polynomials.

- **Getting Signatures on Row Polynomial**: Each $P_j \in \mathcal{P}$ (including D) executes the following.
  - If P_j has broadcast $\mathtt{MR}_j$, then for $i = 1, \ldots, n$, send $f_j(\alpha_i)$ to P_i for getting P_i's signature. Upon receiving $\mathsf{ICSig}(P_i \to P_j, f_{ji})$ from P_i, broadcast $(\mathtt{SR}_j, P_i)$, if $f_{ji} = f_j(\alpha_i)$ holds.
  - If P_i sent f_{ij} and has broadcast $\mathtt{MR}_i$, give $\mathsf{ICSig}(P_j \to P_i, f_{ij})$ to P_i, provided $f_{ij} = g_j(\alpha_i)$ holds.
- **Preparing the $\mathcal{C}_j$ Sets and $\mathcal{C}$ Set**: the following code is executed only by D.
  - Include P_i in $\mathcal{C}_j$ (initialized to $\emptyset$), if $(\mathtt{SR}_k, P_i)$ is received from the broadcast of at least $2t + 1$ parties P_k (including P_j) who have broadcasted the message $\mathtt{MR}_k$.
  - Include $P_j \in \mathcal{C}$ (initialized to $\emptyset$), if $|\mathcal{C}_j| \geq n - t$. Keep on including new parties P_i in $\mathcal{C}_j$ even after including P_j to $\mathcal{C}$, if the above conditions for P_i's inclusion to $\mathcal{C}_j$ are satisfied.

%Public announcement of $\mathcal{C}$ and verification. This code will be triggered by our ACSS protocol..

- **Publicly Announcing the $\mathcal{C}$ Set**: D broadcasts $\mathcal{C}$ and $\mathcal{C}_j$ for each $P_j \in \mathcal{C}$.
- **Verification of the $\mathcal{C}$ Set by the Parties**: Upon receiving $\mathcal{C}$ and $\mathcal{C}_j$ sets from the broadcast of D, each party $P_m \in \mathcal{P}$ checks if $\mathcal{C}$ is *valid* by checking if all the following conditions hold for $\mathcal{C}$.
  - $|\mathcal{C}| \geq n - t$ and each party $P_j \in \mathcal{C}$ has broadcast $\mathtt{MR}_j$.
  - For each $P_j \in \mathcal{C}$, $|\mathcal{C}_j| \geq n - t$. Moreover, for each $P_i \in \mathcal{C}_j$, the message $(\mathtt{SR}_k, P_i)$ is received from the broadcast of at least $2t + 1$ parties P_k (including P_j) who broadcasted $\mathtt{MR}_k$.

▶ **Lemma 9.** *Let $\mathcal{R}$ be the set of parties P_j, who broadcast $\mathtt{MR}_j$ messages during* Sh. *If $|\mathcal{R}| \geq 2t+1$, then except with probability $n^2 \cdot \epsilon_{\mathsf{AICP}}$, there exists a degree-$(t,t)$ bivariate polynomial, say $\overline{F}(x,y)$, where $\overline{F}(x,y) = F(x,y)$ for an honest* D*, such that the row polynomial $f_j(x)$ held by each honest $P_j \in \mathcal{R}$ satisfies $f_j(x) = \overline{F}(x, \alpha_j)$ and the column polynomial $g_i(y)$ held by each honest $P_i \in \mathcal{M}$ satisfies $g_i(y) = \overline{F}(\alpha_i, y)$.*

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

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

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

▶ **Lemma 10.** *In the protocol* Sh*, if* D *broadcasts a valid $\mathcal{C}$, then except with probability $n^2 \cdot \epsilon_{\mathsf{AICP}}$, there exists some $\overline{s} \in \mathbb{F}$, where $\overline{s} = s$ for an honest* D*, such that $\overline{s}$ is eventually two-level t-shared with IC signature.*

Proof. Since the $\mathcal{C}$ set is valid, it implies that the honest parties receive $\mathcal{C}$ and $\mathcal{C}_j$ for each $P_j \in \mathcal{C}$ from the broadcast of D, where $|\mathcal{C}| \geq n - t = 2t+1$ and $|\mathcal{C}_j| \geq n - t = 2t+1$. Moreover, the parties receive $\mathtt{MR}_j$ from the broadcast of each $P_j \in \mathcal{C}$. Since $|\mathcal{C}| \geq 2t+1$, it follows from Lemma 9, that except with probability $n^2 \cdot \epsilon_{\mathsf{AICP}}$, there exists a degree-$(t,t)$ bivariate polynomial, say $\overline{F}(x,y)$, where $\overline{F}(x,y) = F(x,y)$ for an honest D, such that the row polynomial $f_j(x)$ held by each *honest* $P_j \in \mathcal{C}$ satisfies $f_j(x) = \overline{F}(x, \alpha_j)$ and the column polynomial $g_i(y)$ held by each *honest* $P_i \in \mathcal{M}$ satisfies $g_i(y) = \overline{F}(\alpha_i, y)$. We define $\overline{s} = \overline{F}(0,0)$ and show that $\overline{s}$ is two-level t-shared with IC signatures.

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

Consider an *arbitrary* $P_j \in \mathcal{C}$. We claim that corresponding to each honest $P_i \in \mathcal{C}_j$, party P_j holds the signature $\mathsf{ICSig}(P_i \rightarrow P_j, f_{ji})$, where $f_{ji} = \overline{F}(\alpha_i, \alpha_j)$. The claim is trivially true for an *honest* P_j. This is because $f_j(\alpha_i) = \overline{F}(\alpha_i, \alpha_j)$ and P_j includes P_i in the set $\mathcal{C}_j$ only after receiving the signature $\mathsf{ICSig}(P_i \to P_j, f_{ji})$ from P_i, such that the condition $f_{ji} = f_j(\alpha_i)$ holds. We next show that the claim is true, even for a *corrupt* $P_j \in \mathcal{C}$. For this, we show that for each *honest* $P_i \in \mathcal{C}_j$, the column polynomial $g_i(y)$ held by P_i satisfies the condition that $g_i(y) = \overline{F}(\alpha_i, y)$. The claim then follows from the fact that P_i gives the signature $\mathsf{ICSig}(P_i \to P_j, f_{ji})$ to P_j, only after verifying that the condition $f_{ji} = g_i(\alpha_j)$ holds.

So consider a *corrupt* $P_j \in \mathcal{C}$ and an *honest* $P_i \in \mathcal{C}_j$. We note that P_j is allowed to include P_i to $\mathcal{C}_j$, only if at least $2t + 1$ parties P_k (including P_j) who have broadcasted $\mathtt{MR}_k$, has broadcast $(\mathtt{SR}_k, P_i)$. Let $\mathcal{H}$ be the set of such *honest* parties P_k. For each $P_k \in \mathcal{H}$, the row polynomial $f_k(x)$ held by P_k satisfies the condition $f_k(x) = \overline{F}(x, \alpha_k)$ (follows from the proof of Lemma 9). Furthermore, for each $P_k \in \mathcal{H}$, the condition $f_k(\alpha_i) = g_i(\alpha_k)$ holds, where $g_i(y)$ is the degree-t column polynomial held by the honest P_i. This is because P_k broadcasts $(\mathtt{SR}_k, P_i)$, only after receiving the signature $\mathsf{ICSig}(P_i \to P_k, f_{ki})$ from P_i, such that $f_{ki} = f_k(\alpha_i)$ holds for P_k and P_i gives the signature to P_k only after verifying that $f_{ki} = g_i(\alpha_k)$ holds for P_i. Now since $|\mathcal{H}| \geq t + 1$ and $g_i(\alpha_k) = f_k(\alpha_i) = \overline{F}(\alpha_i, \alpha_k)$ holds for each $P_k \in \mathcal{H}$, it follows that the column polynomial $g_i(y)$ held by P_i satisfies the condition $g_i(y) = \overline{F}(\alpha_i, y)$. This is because both $g_i(y)$ and $\overline{F}(\alpha_i, y)$ are degree-t polynomials and two *different* degree-t polynomials can have at most t common values. ◀

▶ **Lemma 11.** *If* D *is honest then in protocol* Sh*, the view of* Adv *is independent of* s*.*

Proof. Without loss of generality, let $P_1, \ldots, P_t$ be under the control of Adv. We claim that throughout the protocol Sh, the adversary learns only t row polynomials $f_1(x), \ldots, f_t(x)$ and t column polynomials $g_1(y), \ldots, g_t(y)$. The lemma then follows from the standard property of degree-(t, t) bivariate polynomials [12, 18, 2]. We next proceed to prove the claim.

During the protocol Sh, the adversary gets $f_1(x), \ldots, f_t(x)$ and $g_1(y), \ldots, g_t(y)$ from D. Consider an arbitrary party $P_i \in \{P_1, \ldots, P_t\}$. Now corresponding to each *honest* party P_j, party P_i receives $f_{ji} = f_j(\alpha_i)$ for signature, both from D, as well as from P_j. However the value f_{ji} is already known to P_i, since $f_{ji} = g_i(\alpha_j)$ holds. Next consider an arbitrary pair of *honest* parties P_i, P_j. These parties exchange f_{ji} and f_{ij} with each other over the pair-wise secure channel and hence nothing about these values are learnt by the adversary. Party P_i gives the signature $\mathsf{ICSig}(P_i \to D, f_{ji})$ to D and $\mathsf{ICSig}(P_i \to P_j, f_{ji})$ to P_j and from the privacy property of AICP, the view of the adversary remains independent of the signed values. Moreover, even after D reveals $\mathsf{ICSig}(P_i \to D, f_{ji})$ to P_j, the view of the adversary remains independent of f_{ji}, which again follows from the privacy property of AICP. ◀

▶ **Lemma 12.** *The communication complexity of* Sh *is* $\mathcal{O}(n^3\kappa^2) + \mathcal{BC}(n^2)$ *bits.*

Proof. In the protocol D distributes n row and column polynomials. There are $\Theta(n^2)$ instances of AICP, each dealing with $L = 1$ value. In addition, D broadcasts a $\mathcal{C}$ set and $\mathcal{C}_j$ sets, each of which can be represented by a n-bit vector. ◀

We finally observe that D's computation in the protocol Sh can be recast as if D wants to share the degree-t polynomial $\bar{F}(0, y)$ among a set of parties $\mathcal{C}$ of size at least $n - t$ by giving each $P_j \in \mathcal{C}$ the share $\bar{F}(0, \alpha_j)$. Here $\bar{F}(x, y)$ is the degree-(t, t) bivariate polynomial committed by D, which is the same as $F(x, y)$ for an *honest* D (see the pictorial representation in Fig 1 and the proof of Lemma 10). If D is *honest*, then adversary learns at most t shares of the polynomial $F(0, y)$, corresponding to the corrupt parties in $\mathcal{C}$ (see the proof of Lemma 11). In the protocol, apart from $P_j \in \mathcal{C}$, every other party P_j who broadcasts the message $\mathtt{MR}_j$ also receives its share $\bar{F}(0, \alpha_j)$, lying on $\bar{F}(0, y)$, as the row polynomial received by every such P_j also lies on $\bar{F}(x, y)$. Based on these observations, we propose the following alternate notation for invoking the protocol Sh, where the input for D is a degree-t polynomial, instead of a value. This notation will later simplify the presentation of our ACSS protocol.

▶ **Notation 13 (Sharing Polynomial Using Protocol Sh).** *We use the notation* $\mathsf{Sh}(\mathsf{D}, r(\cdot))$*, where* $r(\cdot)$ *is some degree-t polynomial possessed by* D*, to denote that* D *invokes the protocol* Sh *by picking a degree-(t,t) bivariate polynomial* $F(x,y)$*, which is otherwise a random polynomial, except that* $F(0,y) = r(\cdot)$*. If* D *broadcasts a valid* $\mathcal{C}$*, then it implies that there exists some degree-t polynomial, say* $\bar{r}(\cdot)$*, where* $\bar{r}(\cdot) = r(\cdot)$ *for an honest* D*, such that each* $P_j \in \mathcal{C}$ *holds a primary-share* $\bar{r}(\alpha_j)$*. We also say that* P_j *(who need not be a member of* $\mathcal{C}$ *set) receives a share* r_j *during* $\mathsf{Sh}(\mathsf{D}, r(\cdot))$ *from* D *to denote that* P_j *receives a degree-t signed row polynomial from* D *with* r_j *as its constant term and has broadcast* $\mathtt{MR}_j$ *message.*

3.1 Designated Reconstruction of Two-level t-shared Values

Let s be a value which has been two-level t-shared with IC signatures by protocol Sh, with parties knowing a valid $\mathcal{C}$ set and respective $\mathcal{C}_j$ sets for each $P_j \in \mathcal{C}$. Then protocol RecPriv (see Fig 3) allows the reconstruction of s by a designated party R. Protocol RecPriv will be used as a sub-protocol in our ACSS protocol. In the protocol, each party $P_j \in \mathcal{C}$ reveals its primary-share to R. Once R receives $t+1$ “valid" primary-shares, it uses them to reconstruct s. For the validation of primary-shares, each party P_j actually reveals the secondary-shares, signed by the parties in $\mathcal{C}_j$. The presence of at least $t+1$ *honest* parties in $\mathcal{C}_j$ ensures that a potentially *corrupt* P_j fails to reveal incorrect primary-share. The properties of RecPriv are

Figure 3 Reconstruction of a two-level t-shared value by a designated party.

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

- **Revealing the signed secondary-shares**: Each $P_j \in \mathcal{C}$ executes the following code.
  - Corresponding to each $P_i \in \mathcal{C}_j$, reveal $\mathsf{ICSig}(P_i \rightarrow P_j, f_{ji})$ to R.
- **Verifying the signatures and reconstruction**: The following code is executed only by R.
  - Include party $P_j \in \mathcal{C}$ to a set $\mathcal{K}$ (initialized to $\emptyset$), if all the following holds:
    - R accepted $\mathsf{ICSig}(P_i \rightarrow P_j, f_{ji})$, corresponding to each $P_i \in \mathcal{C}_j$.
    - The values $\{(\alpha_i, f_{ji})\}_{P_i \in \mathcal{C}_j}$ lie on a degree-t polynomial, say $f_j(x)$.
  - Wait till $|\mathcal{K}| = t+1$. Then interpolate a degree-t polynomial, say $g_0(y)$, using the values $\{\alpha_j, f_j(0)\}_{P_j \in \mathcal{K}}$. Output s and terminate, where $s = g_0(0)$.

stated in Lemma 14, which simply follows from its informal discussion and formal steps and the fact that there are $\Theta(n^2)$ instances of RevPriv, each dealing with $L = 1$ value.

▶ **Lemma 14.** *Let s be two-level shared with IC-signatures. Then in protocol* RecPriv*, the following hold for every possible* Adv*, if all honest parties participate, where* $\epsilon_{\mathsf{AICP}} \leq \frac{n\kappa}{2^{\kappa}-2}$.

- **Termination***: An honest* R *terminates, except with probability* $n^2 \cdot \epsilon_{\mathsf{AICP}}$*.*
- **Correctness***: Except with probability* $n^2 \cdot \epsilon_{\mathsf{AICP}}$*, an honest* R *outputs* s*.*
- **Communication Complexity***: The communication complexity is* $\mathcal{O}(n^3\kappa^2)$ *bits.*

The computations done by the parties in RecPriv can be recast as if parties enable a designated R to reconstruct a degree-t polynomial $r(\cdot)$, which has been shared by D by executing an instance $\mathsf{Sh}(\mathsf{D}, r(\cdot))$ of Sh (see Notation 13). This is because in RecPriv, party R recovers the entire column polynomial $g_0(y)$, which is the same as $F(0,y)$ and as discussed in Notation 13, to share $r(\cdot)$, the dealer D executes Sh by setting $F(0,y)$ to $r(\cdot)$. Based on this discussion, we propose the following alternate notation for reconstructing a shared polynomial by R using RecPriv, which will later simplify the presentation of our ACSS protocol.

▶ **Notation 15 (Reconstructing a Shared Polynomial Using RecPriv).** *Let $r(\cdot)$ be a degree-t polynomial which has been shared by* D *by executing an instance* $\mathsf{Sh}(\mathsf{D}, r(\cdot))$ *of* Sh*. Then* $\mathsf{RecPriv}(\mathsf{D}, r(\cdot), \mathsf{R})$ *denotes that the parties execute the steps of the protocol* RecPriv *to enable* R *reconstruct $r(0)$, which implicitly allows* R *to reconstruct the entire polynomial $r(\cdot)$.*

3.2 Protocols CSh and RecPriv for L Polynomials

To share L number of degree-t polynomials $r^{(1)}(\cdot), \ldots, r^{(L)}(\cdot)$, D can execute L independent instances of Sh (as per Notation 13). This will cost a communication of $\mathcal{O}(L \cdot n^3 \kappa^2) + \mathcal{BC}(L \cdot n^2)$ bits. Instead, by making slight modifications, we achieve a communication complexity of $\mathcal{O}(L \cdot n^2 \kappa + n^3 \kappa^2) + \mathcal{BC}(n^2)$ bits. In the modified protocol, each P_i while issuing signatures to any party, issues a *single* signature on all the required values, on the behalf of all the L instances. For instance, as part of recommitment of row polynomials, party P_j will have L row polynomials (one from each Sh instance) and there will be L common values on these polynomials between P_i and P_j, so P_i needs to sign L values for P_j. Party P_i issues signature on the common values on all these L polynomials simultaneously and for this only one instance of AICP is executed, instead of L instances. Thus all instances of AICP now deal with L values and the error probability of single such instance will be ϵ_{AICP} where $\epsilon_{\mathsf{AICP}} \leq \frac{n\kappa}{2^\kappa - (L+1)}$. To make the broadcast complexity independent of L, each P_j broadcasts a *single* $\mathtt{MR}_j, \mathtt{MC}_j$ and $(\mathtt{SR}_j, P_i)$ message, if the conditions for broadcasting these messages are satisfied with respect to each Sh instance. Finally, each P_j recommits all its L row polynomials to a common set $\mathcal{C}_j$ and similarly D constructs a single $\mathcal{C}$ set with respect to each value in S. We call the resultant protocol as $\mathsf{MSh}(\mathsf{D}, (r^{(1)}(\cdot), \ldots, r^{(L)}(\cdot)))$.

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

4 Asynchronous Complete Secret Sharing

We now design our ACSS protocol CSh by using protocols Sh and RecPriv as sub-protocols, following the blueprint of [18]. We first explain the protocol assuming D has a single secret for sharing. The modifications for sharing L values are straight forward.

To share a value $s \in \mathbb{F}$, D hides s in the constant term of a random degree-(t, t) bivariate polynomial $F(x, y)$ where $s = F(0, 0)$ and distributes the column polynomial $g_i(y) = F(\alpha_i, y)$ to every P_i. D also invokes n instances of our protocol Sh, where the j^{th} instance Sh_j is used to share the row polynomial $f_j(x) = F(x, \alpha_j)$ (this is where we use our interpretation of sharing degree-t univariate polynomial using Sh as discussed in Notation 13). Party P_i upon receiving a share f_{ji} from D during the instance Sh_j checks if it lies on its column polynomial (that is if $f_{ji} = g_i(\alpha_j)$ holds) and if this holds for all the n instances of Sh, P_i broadcasts a $\mathtt{MC}$ message. This indicates that all the row polynomials of D are pair-wise consistent with the column polynomial $g_i(y)$. The goal is then to let D publicly identify a set of $2t+1$ parties, say $\mathcal{W}$, such that $\mathcal{W}$ constitutes a common $\mathcal{C}$ set in all the n Sh instances and such that each party in $\mathcal{W}$ has broadcast $\mathtt{MC}$ message. If D is honest, then such a common $\mathcal{W}$ set is eventually obtained, as there are at least $2t + 1$ honest parties, who constitute a potential common $\mathcal{W}$ set. This is because if D keeps on running the Sh instances, then eventually every honest party is included in the $\mathcal{C}$ sets of individual Sh instances. The idea here is that if such a common $\mathcal{W}$ is obtained, then it guarantees that the row polynomials held by D are pair-wise consistent with the column polynomials of the parties in $\mathcal{W}$, implying that the row polynomials of D lie on a single degree-(t, t) bivariate polynomial. Moreover, each of these row polynomials is shared among the common set of parties $\mathcal{W}$. The next goal is then to let each party P_j obtain the j^{th} row polynomial held by D, for which the parties execute an instance of the protocol RecPriv (here we use our interpretation of using RecPriv to enable designated reconstruction of a shared degree-t polynomial). We stress that once the common set $\mathcal{W}$ is publicly identified, *each* P_j obtains the desired row polynomial, even if D is *corrupt*, as the corresponding RecPriv instance terminates for P_j even for a corrupt D. Once the parties obtain their respective row polynomials, the constant term of these polynomials constitute a complete t-sharing of D's value. For the formal details of CSh, see Fig 4.

Figure 4 Complete sharing of a single secret.

$\underline{\mathsf{CSh}(\mathsf{D}, s)}$

- **Distribution of Column Polynomials and Sharing of Row Polynomials by** D:
  - D selects a random degree-(t,t) bivariate polynomial $F(x,y)$ over $\mathbb{F}$, such that $F(0,0) = s$.
  - For $i = 1, \ldots, n$, D sends the column polynomial $g_i(y) = F(\alpha_i, y)$ to P_i.
  - For $j = 1, \ldots, n$, D executes an instance $\mathsf{Sh}_j = \mathsf{Sh}(\mathsf{D}, f_j(x))$, where $f_j(x) = F(x, \alpha_j)$.
- **Pair-wise Consistency Check**: Each $P_i \in \mathcal{P}$ (including D) executes the following code.
  - Wait to receive a degree-t column polynomial $g_i(y)$ from D.
  - Participate in the instances $\mathsf{Sh}_1, \ldots, \mathsf{Sh}_n$.
  - If a share f_{ji} is received from D during the instance Sh_j, then broadcast the message $\mathtt{MC}_j$, if the condition $f_{ji} = g_i(\alpha_j)$ holds for each $j = 1, \ldots, n$.
- **Construction of $\mathcal{W}$ and Announcement**: The following code is executed only by D.
  - Let $\mathcal{C}^{(j)}$ denote the instance of $\mathcal{C}$ set constructed during the instance Sh_j. Keep updating the $\mathcal{C}^{(j)}$ sets till a set $\mathcal{W} = \mathcal{C}^{(1)} \cap \ldots \cap \mathcal{C}^{(n)}$ is obtained, where $|\mathcal{W}| = n - t$ and $\mathtt{MC}_i$ message is received from the broadcast of each party $P_i \in \mathcal{W}$.
  - Once a set $\mathcal{W}$ satisfying the above conditions are obtained, broadcast $\mathcal{W}$.
- **Verification of $\mathcal{W}$**: Each party $P_j \in \mathcal{P}$ (including D) executes the following code.
  - Upon receiving $\mathcal{W}$ from the broadcast of D, check if $\mathcal{W}$ is a valid $\mathcal{C}$ set for each of the instances $\mathsf{Sh}_1, \ldots, \mathsf{Sh}_n$ and if $\mathtt{MC}_i$ message is received from the broadcast of each $P_i \in \mathcal{W}$.
  - If the set $\mathcal{W}$ satisfies the above conditions, then invoke an instance $\mathsf{RecPriv}_j = \mathsf{RecPriv}(\mathsf{D}, f_j(x))$ to reconstruct the row polynomial $f_j(x)$. Participate in the instances $\mathsf{RecPriv}_k$, for $k = 1, \ldots, n$.
- **Share Computation and Termination**: Each party $P_j \in \mathcal{P}$ (including D) does the following.
  - Wait to terminate the instance $\mathsf{RecPriv}_j$ and obtain the row polynomial $f_j(x)$.
  - Upon terminating $\mathsf{RecPriv}_j$, output the share $s_j = f_j(0)$ and terminate the protocol CSh.

To generate a complete t-sharing of $S = (s^{(1)}, \ldots, s^{(L)})$, the parties execute the steps of the protocol CSh independently L times with the following modifications: corresponding to each party P_j, D will now have L number of degree-t row polynomials to share. Instead of executing L instances of Sh to share them, D shares all of them simultaneoulsy by executing an instance MSh_j of MSh. Similarly, each party P_i broadcasts a single $\mathtt{MC}_i$ message, if the conditions for broadcasting the $\mathtt{MC}_i$ message is satisfied for P_i in all the L instances. The proof of the following theorem follows from [18] and the fact that there are n instances of MSh and RecPriv, each dealing with L polynomials. For details, see Appendix B.

▶ **Theorem 16.** *Let* $\epsilon_{\mathsf{AICP}} \leq \frac{n\kappa}{2^\kappa - (L+1)}$. *Then* CSh *constitutes a* $(1 - \epsilon_{\mathsf{ACSS}})$ *ACSS protocol, with communication complexity* $\mathcal{O}(L \cdot n^3 \kappa + n^4 \kappa^2) + \mathcal{BC}(n^3)$ *bits where* $\epsilon_{\mathsf{ACSS}} \leq n^3 \cdot \epsilon_{\mathsf{AICP}}$.

5 The AMPC Protocol

Our AMPC protocol is obtained by directly plugging in our protocol CSh in the generic framework of [11]. The protocol has a circuit-independent pre-processing phase and a circuitdependent computation phase. During the pre-processing phase, the parties generate c_M number of completely t-shared, random and private multiplication triples (a, b, c), where $c = a \cdot b$. For this, each party first verifiably shares c_M number of random multiplication triples by executing CSh with $L = 3c_M$. As the triples shared by corrupt parties may not be random, the parties next apply a "secure triple-extraction" procedure to output c_M number of completely t-shared multiplication triples, which are truly random and private. The error probability ϵ_{AMPC} of the pre-processing phase will be $\frac{n^5\kappa}{2^\kappa-(3c_M+1)}$ and its communication complexity will be $\mathcal{O}(c_M n^4\kappa + n^4\kappa^2) + \mathcal{BC}(n^4)$ bits (as there are n instances of CSh).

During the computation phase, each party P_i generates a complete t-sharing of its input $x^{(i)}$ by executing an instance of CSh. As the corrupt parties may not invoke their instances of CSh, to avoid endless wait, the parties agree on a common subset of $n-t$ CSh instances which eventually terminate for every one. For this, the parties execute an instance of *agreement on common-subset* (ACS) primitive [10, 8]. The parties then securely evaluate each gate in *ckt*, as discussed in Section 1. As the AMPC protocol is standard and obtained using the framework of [11], we refer to [11] for the proof of the following theorem.

▶ **Theorem 17.** *Let $\mathbb{F} = GF(2^\kappa)$ and $f : \mathbb{F}^n \rightarrow \mathbb{F}$ be a function, expressed as a circuit over $\mathbb{F}$ consisting of c_M multiplication gates. Then there exists a $(1-\epsilon_{\mathsf{AMPC}})$ unconditionally-secure AMPC protocol, tolerating Adv, where $\epsilon_{\mathsf{AMPC}} \leq \frac{n^5\kappa}{2^\kappa-(3c_M+1)}$. The communication complexity for evaluating the multiplication gates is $\mathcal{O}(c_M n^4\kappa + n^4\kappa^2) + \mathcal{BC}(n^4)$ bits.*

References

1. G. Asharov and Y. Lindell. A Full Proof of the BGW Protocol for Perfectly Secure Multiparty Computation. *J. Cryptology*, 30(1):58–151, 2017.
2. L. Bangalore, A. Choudhury, and A. Patra. Almost-Surely Terminating Asynchronous Byzantine Agreement Revisited. In *PODC*, pages 295–304. ACM, 2018.
3. D. Beaver. Efficient Multiparty Protocols Using Circuit Randomization. In *CRYPTO*, volume 576 of *Lecture Notes in Computer Science*, pages 420–432. Springer, 1991.
4. Z. Beerliová-Trubíniová and M. Hirt. Efficient Multi-party Computation with Dispute Control. In *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 305–328. Springer, 2006.
5. Z. Beerliová-Trubíniová and M. Hirt. Simple and Efficient Perfectly-Secure Asynchronous MPC. In *ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 376–392. Springer, 2007.
6. M. Ben-Or, R. Canetti, and O. Goldreich. Asynchronous Secure Computation. In *STOC*, pages 52–61. ACM, 1993.
7. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract). In *STOC*, pages 1–10. ACM, 1988.
8. M. Ben-Or, B. Kelmer, and T. Rabin. Asynchronous Secure Computations with Optimal Resilience (Extended Abstract). In *PODC*, pages 183–192. ACM, 1994.
9. G. Bracha. An Asynchronous [(n-1)/3]-Resilient Consensus Protocol. In *PODC*, pages 154–162. ACM, 1984.
10. R. Canetti. *Studies in Secure Multiparty Computation and Applications*. PhD thesis, Weizmann Institute, Israel, 1995.
11. A. Choudhury and A. Patra. An Efficient Framework for Unconditionally Secure Multiparty Computation. *IEEE Trans. Information Theory*, 63(1):428–468, 2017.
12. R. Cramer and I. Damgård. *Multiparty Computation, an Introduction. Contemporary Cryptography*. Birkhåuser Basel, 2005.
13. M. Fitzi. *Generalized communication and security models in Byzantine agreement*. PhD thesis, ETH Zurich, Zürich, Switzerland, 2003.

14 O. Goldreich, S. Micali, and A. Wigderson. How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. In *STOC*, pages 218–229. ACM, 1987.
15 M. Hirt. *Multi party computation: efficient protocols, general adversaries, and voting*. PhD thesis, ETH Zurich, Zürich, Switzerland, 2001.
16 Y. Lindell. Secure Multiparty Computation (MPC). Cryptology ePrint Archive, Report 2020/300, 2020.
17 A. Patra. Studies on Verifiable Secret Sharing, Byzantine Agreement and Multiparty Computation. *IACR Cryptol. ePrint Arch.*, 2010:280, 2010.
18 A. Patra, A. Choudhary, and C. Pandu Rangan. Efficient Statistical Asynchronous Verifiable Secret Sharing with Optimal Resilience. In *ICITS*, volume 5973 of *Lecture Notes in Computer Science*, pages 74–92. Springer, 2009.
19 A. Patra, A. Choudhury, and C. Pandu Rangan. Efficient Asynchronous Verifiable Secret Sharing and Multiparty Computation. *J. Cryptology*, 28(1):49–109, 2015.
20 T. Rabin and M. Ben-Or. Verifiable Secret Sharing and Multiparty Protocols with Honest Majority (Extended Abstract). In *STOC*, pages 73–85. ACM, 1989.
21 A. Shamir. How to Share a Secret. *Commun. ACM*, 22(11):612–613, 1979.
22 A. C. Yao. Protocols for Secure Computations (Extended Abstract). In *FOCS*, pages 160–164. IEEE Computer Society, 1982.

A The Asynchronous Information Checking Protocol (AICP) of [18]

The AICP of [18] is an adaptation of the synchronous ICP of [4] to the asynchronous setting. Let $\mathcal{S} = (s^{(1)}, \ldots, s^{(L)}) \in \mathbb{F}^L$ be the private input of S. The high level idea of the protocol is as follows: during the distribution phase, S gives $\mathcal{S}$ along with some *authentication tag* to I and a corresponding information-theoretic *verification tag* to each individual verifier. The tags with respect to a verifier P_i are computed by picking a random $y \in \mathbb{F}$ and fitting a degree-L polynomial $f(x)$ passing through $L+1$ distinct points $(0, y), (1, s^{(1)}), \ldots, (L, s^{(L)})$. The authentication tag is set to y, while the verification tag is set to (u, v), where u is randomly chosen from $\mathbb{F} \setminus \{0, \ldots, L\}$ and $v = f(u)$. Later, during the revelation phase, I provides $\mathcal{S}$ and the verification tags to R and the verifiers provide the authentication tags to R and if the revealed $\mathcal{S}$ and verification tags are found to be consistent with "sufficiently large" authentication tags, then $\mathcal{S}$ is accepted, else it is rejected.

A problem with the above approach is that if S is *corrupt*, then it can distribute inconsistent data to I and the verifiers, which will later lead to the rejection of revealed $\mathcal{S}$. To get around this problem, a cut-and-choose technique is deployed, where instead of providing a single verification and authentication tag with respect to each verifier, S provides 2κ number of authentication tags to I and corresponding 2κ verification tags are given to each verifier P_i. Then during the authentication phase, P_i randomly reveals κ number of verification tags to I and if these verification tags are found to be consistent with $\mathcal{S}$ and the corresponding authentication tags (which is considered as cut-and-choose being successful for P_i), then with a high probability, it is ensured that *at least one* of the remaining undisclosed κ verification tags held by P_i is consistent with $\mathcal{S}$ and the corresponding authentication tag held by I.

In the protocol, the cut-and-choose step is executed independently between I and each individual verifier P_i. Party I sets $\mathsf{V}_{\mathsf{S},\mathsf{I}}$ to 1 as soon as it finds that the cut-and-choose test is successful for a set $\mathcal{R}$ of $n - t = 2t + 1$ verifiers. Later, during the revelation phase, R accepts the $\mathcal{S}$ revealed by I, if there are at least $|\mathcal{R}| - t = t + 1$ verifiers from the set $\mathcal{R}$, such that each of these $t + 1$ verifiers produce at least one consistent verification tag from their list of undisclosed verification tags. The protocol is formally presented in Fig 5.

We refer to [18] for the proof of the following theorem.

Figure 5 The AICP of [18].

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

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

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

- Each $P_i \in \mathcal{P}$ (including S, I) on receiving the verification tags $z_1^{(i)}, \ldots, z_{2\kappa}^{(i)}$ from S randomly partitions the index set $\{1, \ldots, 2\kappa\}$ into two equal halves I_i and $\bar{I}_i$ of size κ. Party P_i then sends the index sets $I_i, \bar{I}_i$ and the verification tags $z_j^{(i)}$ for each $j \in I_i$ to I.
- Party I upon receiving the index sets $I_i, \bar{I}_i$ and the verification tags $z_j^{(i)}$ for each $j \in I_i$ from P_i verifies if for *every* $j \in I_i$, the $L+2$ points $(0, y_j^{(i)}), (1, s^{(1)}), \ldots, (L, s^{(L)}), z_j^{(i)}$ lie on a degree-L polynomial. If the verification is successful, then I includes verifier P_i to a set $\mathcal{R}$, which is initialized to $\emptyset$.
- If $|\mathcal{R}| = 2t+1$, then I sets $\mathsf{V}_{\mathsf{S},\mathsf{I}} = 1$ and $\mathsf{ICSig}(\mathsf{S} \rightarrow \mathsf{I}, \mathcal{S}) = (\mathcal{S}, \{\bar{I}_i\}_{P_i \in \mathcal{R}}, \{y_j^{(i)}\}_{P_i \in \mathcal{R}, j \in \bar{I}_i})$.

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

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

▶ **Theorem 18** ([18]). *Protocols* $(\mathsf{Gen}, \mathsf{Ver}, \mathsf{RevPriv})$ *constitute a* $(1 - \epsilon_{\mathsf{AICP}})$*-secure AICP, where* $\epsilon_{\mathsf{AICP}} = \frac{n\kappa}{2^\kappa - (L+1)}$. *The communication complexity of* $\mathsf{Gen}, \mathsf{Ver}$ *and* $\mathsf{RevPriv}$ *is* $\mathcal{O}(L\kappa + n\kappa^2)$, $\mathcal{O}(n\kappa^2)$ *and* $\mathcal{O}(L\kappa + n\kappa^2)$ *bits respectively.*

B Properties of Our ACSS Protocol

We first prove the properties of the protocol CSh, assuming that $L = 1$ (see Fig 4). In the following proofs, $\epsilon_{\mathsf{AICP}} = \frac{n\kappa}{2^\kappa - 2}$, which is obtained by substituting $L = 1$ in the AICP.

▶ **Lemma 19** (Termination for an Honest D). *In protocol* CSh, *if* D *is honest, then* *except with probability $n^3 \cdot \epsilon_{\mathsf{AICP}}$, all honest parties eventually terminate the protocol.*

Proof. From Lemma 7, it follows that all honest parties are eventually included in the $\mathcal{C}$ set $\mathcal{C}^{(j)}$ constructed by D during the instance Sh_j, except with probability $n^2 \cdot \epsilon_{\mathsf{AICP}}$. This implies that all honest parties are eventually included in the sets $\mathcal{C}^{(1)}, \ldots, \mathcal{C}^{(n)}$, except with probability $n^3 \cdot \epsilon_{\mathsf{AICP}}$. Moreover, since D is honest, for every pair of honest parties P_i, P_j, the condition $f_i(\alpha_j) = g_j(\alpha_i)$ and $f_j(\alpha_i) = g_i(\alpha_j)$ hold. As there are at least $2t+1$ honest parties, this implies that eventually D finds a common set $\mathcal{W}$ of size $2t+1$, such that $\mathcal{W}$ constitutes a valid $\mathcal{C}$ set for all the instances $\mathsf{Sh}_1, \ldots, \mathsf{Sh}_n$ and each party P_i has broadcast $\mathtt{MC}_i$ message. Upon the broadcast of $\mathcal{W}$, each honest party eventually validates it and invokes the instances $\mathsf{RecPriv}_1, \ldots, \mathsf{RecPriv}_n$. From Lemma 14, the instance $\mathsf{RecPriv}_j$ eventually terminates for an *honest* P_j, except with probability $n^2 \cdot \epsilon_{\mathsf{AICP}}$. As there are at most n honest parties, it follows that except with probability $n^3 \cdot \epsilon_{\mathsf{AICP}}$, all honest parties eventually terminate their designated RecPriv instance and hence terminate CSh. ◀

▶ **Lemma 20 (Termination for a Corrupt D).** *In protocol* CSh*, if an honest party terminates* CSh*, then except with probability $n^3 \cdot \epsilon_{\mathsf{AICP}}$, all honest parties eventually terminate the protocol.*

Proof. Let P_j be an *honest* party who terminates CSh. This implies that P_j receives a set $\mathcal{W}$ of size $2t+1$, such that $\mathcal{W}$ constitutes a valid $\mathcal{C}$ set for all the instances $\mathsf{Sh}_1, \ldots, \mathsf{Sh}_n$. Since $\mathcal{W}$ is broadcast by D, every other honest party eventually receives the same valid $\mathcal{W}$ set from D. Party P_j also terminates its designated $\mathsf{RecPriv}_j$ instance. This implies that for any other honest P_k, the corresponding designated $\mathsf{RecPriv}_k$ instance eventually terminates for P_k, except with probability $n^2 \cdot \epsilon_{\mathsf{AICP}}$ (follows from Lemma 14). As there are at most n honest parties, it follows that all honest parties eventually terminate their designated RecPriv instance and hence terminate CSh. ◀

▶ **Lemma 21 (Correctness).** *In protocol* CSh*, if the honest parties terminate, then except with probability $n^3 \cdot \epsilon_{\mathsf{AICP}}$, there exists some $\overline{s} \in \mathbb{F}$, where $\overline{s} = s$ for an honest* D*, such that $\overline{s}$ is completely t-shared.*

Proof. Since the honest parties terminate CSh, it implies that they receive a set $\mathcal{W}$ from the broadcast of D, which constitutes a valid $\mathcal{C}$ set for the instances $\mathsf{Sh}_1, \ldots, \mathsf{Sh}_n$. Moreover, each honest P_j terminates its designated $\mathsf{RecPriv}_j$ instance with a degree-t polynomial. From Lemma 14, it follows that the degree-t polynomial reconstructed by P_j during $\mathsf{RecPriv}_j$ is the same degree-t polynomial, shared by D during the instance Sh_j, except with probability $n^2 \cdot \epsilon_{\mathsf{AICP}}$. As there are at most n honest parties, it follows that except with probability $n^3 \cdot \epsilon_{\mathsf{AICP}}$, the degree-$t$ polynomials reconstructed by the honest parties in their designated RecPriv instance are the same, as shared by D during the corresponding Sh instance. To complete the proof, we claim that the polynomials shared by D during the instances $\mathsf{Sh}_1, \ldots, \mathsf{Sh}_n$ lie on a single degree-(t,t) bivariate polynomial, except with probability $n^3 \cdot \epsilon_{\mathsf{AICP}}$.

The claim is true with probability 1 for an *honest* D, as it shares the row polynomial $f_j(x) = F(x, \alpha_j)$ during the instance Sh_j, for all $j = 1, \ldots, n$. So we next prove the claim for the case of a *corrupt* D. Consider an arbitrary instance Sh_j. Since $\mathcal{W}$ is a valid $\mathcal{C}$ set for the instance Sh_j, it follows from Lemma 10 that except with probability $n^2 \cdot \epsilon_{\mathsf{AICP}}$, D shared some degree-$t$ polynomial, say $\bar{f}_j(x)$ among the parties in $\mathcal{W}$ during the instance Sh_j. This implies that except with probability $n^3 \cdot \epsilon_{\mathsf{AICP}}$, the polynomials shared by D among $\mathcal{W}$ during the instances $\mathsf{Sh}_1, \ldots, \mathsf{Sh}_n$ are all degree-t polynomials, say $\bar{f}_1(x), \ldots, \bar{f}_n(x)$. Since every party P_i in $\mathcal{W}$ has broadcast $\mathtt{MC}_i$ message, it follows that if P_i is *honest*, then $\bar{f}_j(\alpha_i) = \bar{g}_i(\alpha_j)$ holds for all $j = 1, \ldots, n$; here $\bar{g}_i(y)$ is the degree-t column polynomial received by P_i. As there are at least $t+1$ honest parties P_i in $\mathcal{W}$, it implies that the degree-t row polynomials $\bar{f}_1(x), \ldots, \bar{f}_n(x)$ are pair-wise consistent with $t+1$ degree-t column polynomials, implying that the polynomials $\bar{f}_1(x), \ldots, \bar{f}_n(x)$ lie on a single degree-(t,t) bivariate polynomial, say $\bar{F}(x,y)$ (follows from Lemma 1). ◀

▶ **Lemma 22 (Privacy).** *In protocol* CSh*, if* D *is honest, then the view of the adversary* Adv *is independent of s.*

Proof. Without loss of generality, let $P_1, \ldots, P_t$ be under the control of Adv. We claim that throughout the protocol, Adv only learns the degree-t row polynomials $f_1(x), \ldots, f_t(x)$ and degree-t column polynomials $g_1(y), \ldots, g_t(y)$. The lemma then follows from the standard properties of degree-(t,t) bivariate polynomial [1] and the fact that the polynomial $F(x,y)$ is randomly chosen by D.

The column polynomials $g_1(y), \ldots, g_t(y)$ are given by Adv, while the row polynomials $f_1(x), \ldots, f_t(x)$ are obtained during the instances $\mathsf{RecPriv}_1, \ldots, \mathsf{RecPriv}_t$. For any $P_j \in \{P_{t+2}, \ldots, P_n\}$, the adversary learns the values $f_j(\alpha_1), \ldots, f_j(\alpha_t)$ during the instance Sh_j and these values are independent of the share $s_j \stackrel{\text{def}}{=} f_j(0)$ held by P_j (follows from Lemma 11). Moreover, the values $f_j(\alpha_1), \ldots, f_j(\alpha_t)$ are already known to Adv, as they lie on the column polynomials held by Adv. ◀

▶ **Lemma 23 (Communication Complexity).** *The communication complexity of* CSh *is $\mathcal{O}(n^4\kappa^2) + \mathcal{BC}(n^3)$ bits.*

Proof. The lemma follows from Lemma 12, Lemma 14 and the fact that there are n instances of Sh and RecPriv in the protocol. ◀

The following theorem finally follows from Lemma 19-23.

▶ **Theorem 24.** *Let $\epsilon_{\mathsf{AICP}} \leq \frac{n\kappa}{2^\kappa - 2}$. Then* CSh *constitutes a $(1-\epsilon_{\mathsf{ACSS}})$ ACSS protocol, with communication complexity $\mathcal{O}(n^4\kappa^2) + \mathcal{BC}(n^3)$ bits where $\epsilon_{\mathsf{ACSS}} \leq n^3 \cdot \epsilon_{\mathsf{AICP}}$.*

B.1 Properties of Protocol CSh for Sharing L Values

The procedure for sharing L values using CSh is outlined in Section 4. The resultant protocol involves n instances of MSh, each sharing L number of degree-t polynomials. Also n instances of RecPriv, each reconstructing L number of degree-t polynomials are involved. Moreover, each instance of underlying AICP deals with L values and error probability ϵ_{AICP} of a single instance is $\epsilon_{\mathsf{AICP}} = \frac{n\kappa}{2^\kappa - (L+1)}$. The proof of the following theorem now follows similar to the proof of Lemma 24.

Theorem 16. *Let $\epsilon_{\mathsf{AICP}} \leq \frac{n\kappa}{2^\kappa - (L+1)}$. Then* CSh *constitutes a $(1-\epsilon_{\mathsf{ACSS}})$ ACSS protocol, with communication complexity $\mathcal{O}(L \cdot n^3\kappa + n^4\kappa^2) + \mathcal{BC}(n^3)$ bits where $\epsilon_{\mathsf{ACSS}} \leq n^3 \cdot \epsilon_{\mathsf{AICP}}$.*