Unconditionally Secure Asynchronous Multiparty Computation with Quadratic Communication

Arpita Patra **Ashish Choudhary** **C. Pandu Rangan**
Department of Computer Science and Engineering
Indian Institute of Technology Madras
Chennai India 600036
Email:{ arpita,ashishc }@cse.iitm.ernet.in, rangan@iitm.ernet.in

Abstract

Secure multiparty computation (MPC) allows a set of n parties to securely compute an agreed function, even if up to t parties are under the control of an adversary. In this paper, we propose a new *Asynchronous secure multiparty computation* (AMPC) protocol that provides information theoretic security with $n = 4t + 1$, where t out of n parties can be under the influence of a *Byzantine (active)* adversary $\mathcal{A}_t$ having *unbounded computing power*. Our protocol communicates $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits per multiplication and involves a negligible error probability of $2^{-\Omega(\kappa)}$, where κ is the error parameter and $\mathbb{F}$ is the field over which the computation is carried out. The best known information theoretically secure AMPC with $n = 4t+1$ communicates $\mathcal{O}(n^3 \log |\mathbb{F}|)$ bits per multiplication and does not involve any error probability in computation. Though a negligible error probability is involved, our AMPC protocol provides the best communication complexity among all the known AMPC protocols providing information theoretic security. Moreover, the communication complexity of our AMPC is same as the communication complexity of the best known AMPC protocol with *cryptographic assumptions*. As a tool for our AMPC protocol, we propose a new method of efficiently generating $(t, 2t)$-*sharing* of multiple secrets concurrently in asynchronous setting, which is of independent interest.

Keywords: Multiparty Computation, Byzantine Adversary, Asynchronous Networks.

1 Introduction

Secure Multiparty Computation (MPC): Secure multiparty computation (MPC) [36] allows a set of n parties to securely compute an agreed function f, even if some of the parties are under the control of a centralized adversary. More specifically, assume that the agreed function f can be expressed as $f : (\{0,1\}^*)^n \rightarrow (\{0,1\}^*)^n$ and party P_i has input $x_i \in \{0,1\}^*$. At the end of the computation of f, P_i gets $y_i \in \{0,1\}^*$, where $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$. Moreover, the adversary should not get any information about the input and output of the honest parties, other than what can be inferred from the input and output of the corrupted parties. Notice that the problem becomes very easy if there exists a *trusted third party* (TTP), to whom each party can privately give his input x_i, who then computes the function f and then privately distribute the output y_i to individual parties. However, in reality, there does not exist any TTP. So the goal of any general MPC is to simulate the functionality of TTP, by executing a protocol among the n parties, such that at the end of the protocol, all (honest) parties receive correct outputs and the messages seen by the adversary during the protocol contain no *additional* information about the inputs and outputs of the honest parties, other than what can be computed from the inputs and outputs of the corrupted parties. MPC is one of the most important and fundamental problems in distributed computing. Over the past three decades, the problem has been studied extensively in different settings, depending upon whether the network is synchronous [36, 21, 7, 13, 33, 1, 23, 20, 22, 25, 3, 17, 27, 5, 26] or asynchronous [6, 8, 35, 32, 10, 28, 4, 30], the adversary is threshold [36, 21, 7, 13, 33, 1, 20, 22, 3, 17, 5, 23] or non-threshold [23, 16, 28, 2, 24], the adversary behavior is static [36, 21, 7, 13, 33, 1, 23, 20] or mobile [29], the security is cryptographic [21, 25, 26] or information theoretic [36, 7, 13, 33] and finally whether the protocol is perfect (i.e., without any error) [7, 22, 5] or allows a negligible error probability [34, 33, 14, 15, 17, 3, 31].

Any general MPC protocol computes the function f as follows: without loss of generality, the function f is specified by an arithmetic circuit over a finite field $\mathbb{F}$, consisting of input, linear (e.g. addition), multiplication, random and output gates. We may denote the number of gates of each type by c_I, c_L, c_M, c_R and c_O respectively. The parties then share their respective inputs among all the parties using some (linear) secret sharing technique. Roughly speaking, a secret sharing scheme allows a specific party called *dealer* D to share a secret $s \in \mathbb{F}$ among the n parties, such that if D is honest then adversary does not get any information about s by pooling the shares of the corrupted parties. Once input sharing is done, the circuit is computed gate by gate such that the output of the intermediate gates are always kept as secret and are properly shared/ distributed among the parties. Due to the linearity of used secret sharing scheme, the parties can locally compute their shares of linear gates without doing any communication. However, to evaluate a multiplication gate, the parties need to execute a protocol among themselves so as to generate the proper sharing of the output of the multiplication gate. Among all the different types of gates, evaluation of a multiplication gate requires the most communication complexity. So the communication complexity of any general MPC is usually given in terms of the communication complexity per multiplication gate. Once the sharing of outputs (of the function) are computed, they are reconstructed towards appropriate parties.

The MPC problem has been studied extensively over synchronous networks which assumes that there is a global clock and the delay of any message in the network channels is bounded. However, though theoretically impressive, such networks do not model adequately real life networks like Internet. So in this paper, we study MPC in asynchronous networks, tolerating a threshold adversary, having unbounded computing power, who can corrupt t out of the n parties in Byzantine fashion.

Asynchronous Networks: Asynchronous networks model real life networks like the Internet much better than their synchronous counterpart. Here the communication channels between the parties have arbitrary, yet finite delay (i.e the messages are guaranteed to reach eventually). To model this, the adversary is given the power to schedule the delivery of messages in the network. The inherent difficulty in designing a protocol in asynchronous network, comes from the fact that when a party does not receive an expected message then he cannot decide whether the sender is corrupted (and did not send the message at all) or the message is just delayed in the network. Therefore it is impossible to consider the inputs of all uncorrupted parties. So input of up to t (potentially honest) parties may get ignored because waiting for them could turn out to be endless. Moreover the tools that are applicable in synchronous settings cannot be deployed in asynchronous settings. Hence, designing asynchronous protocols require complete

new set of primitives. For an excellent introduction to asynchronous protocols, see [11].

Asynchronous Multiparty Computation (AMPC): Unlike MPC in synchronous networks, designing AMPC protocols has received very less attention due to their inherent difficulty. It is known that AMPC under *cryptographic assumptions* [25, 26] is possible iff $n \geq 3t + 1$. In information theoretic settings, AMPC with *zero error* (i.e., *perfectly secure* AMPC) is possible iff $n \geq 4t + 1$ [6], whereas AMPC with *negligible error probability* is possible iff $n \geq 3t + 1$ [8]. The communication complexities *per multiplication* of the best known AMPC protocols are given in the following table, where c_M denotes the number of multiplication gates in arithmetic circuit corresponding to the function f and IT denotes Information Theoretic security. Furthermore, for cryptographic AMPC, κ is the security parameter, while for information theoretic AMPC (with negligible error probability), κ is the error parameter.

Reference	Type of Security	Resilience	Communication Complexity in bits
[25]	Cryptographic	$t < n/3$ (optimal)	$\mathcal{O}(c_M n^3 \kappa)$
[26]	Cryptographic	$t < n/3$ (optimal)	$\mathcal{O}(c_M n^2 \kappa)$
[35]	IT (no error)	$t < n/4$ (optimal)	$\Omega(c_M n^5 \log(\lvert\mathbb{F}\rvert))$
[4]	IT (no error)	$t < n/4$ (optimal)	$\mathcal{O}(c_M n^3 \log(\lvert\mathbb{F}\rvert))$
[8]	IT (negligible error)	$t < n/3$ (optimal)	$\Omega(c_M n^{11} \kappa^4)$
[30]	IT (negligible error)	$t < n/3$ (optimal)	$\mathcal{O}(c_M n^5 \kappa)$
[32]	IT (negligible error)	$t < n/4$ (non-optimal)	$\mathcal{O}(c_M n^4 \kappa)$

Recently in [19], the authors have designed communication efficient MPC protocols over networks that exhibit partial asynchrony (where the network is synchronous up to certain point and becomes completely asynchronous after that) and hence we do not compare it with our MPC protocol, which is designed in completely asynchronous settings.

Our Contribution: We design an efficient information theoretic secure AMPC protocol with $n = 4t+1$ satisfying : (a) **Termination:** Every honest party terminates the protocol eventually, (b) **Correctness:** Every honest party receives the correct output at the end except with negligible error probability of $2^{-\Omega(\kappa)}$, where κ is the error parameter, (c) **Secrecy:** The adversary gets no information about the inputs and outputs of *honest* parties apart from what can be computed from the inputs and outputs of the corrupted parties. Our protocol communicates $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits per multiplication and involves a negligible error probability of $2^{-\Omega(\kappa)}$ only in **correctness**, where $\mathbb{F}$ is the field over which the computation is carried out. The best known AMPC with $n = 4t + 1$ communicates $\mathcal{O}(n^3 \log |\mathbb{F}|)$ bits per multiplication [4] and satisfies all the three properties, namely **termination**, **secrecy** and **correctness** perfectly (i.e without any error probability). Though a negligible error probability is involved in correctness, our AMPC protocol provides the best communication complexity among all the known AMPC protocols providing information theoretic security. Finally, the communication complexity of our AMPC protocol is same as the communication complexity of the best known AMPC protocol with *cryptographic assumptions* [26]. As a tool for our AMPC protocol, we propose a new and efficient method of generating $(t, 2t)$-*sharing* of multiple secrets concurrently in asynchronous settings, which is of independent interest.

2 Preliminaries

Model: We follow the network model of [6], where there is a set of n parties denoted by $\mathcal{P} = \{P_1, \ldots, P_n\}$, who are pairwise connected by secure asynchronous channels. An adversary $\mathcal{A}_t$ with *unbounded computing power* can control at most $t < \frac{n}{4}$ parties in Byzantine fashion and can make the corrupted parties to deviate from the protocol in any arbitrary manner. Moreover, the adversary is given the power to schedule messages over each channel. But he will have no access to the messages sent by honest parties. The function to be computed is specified by an arithmetic circuit over a finite field $\mathbb{F}$, consisting of input, linear (e.g. addition), multiplication, random and output gates. We denote the number of gates of each type by c_I, c_L, c_M, c_R and c_O respectively.

The Ground Field and The Extension Field: For the rest of the paper, we fix a finite field $\mathbb{F}$ with $|\mathbb{F}| \geq n$ over which most of our computations will be performed. We call $\mathbb{F}$ as the *Ground Field*. Any element from $\mathbb{F}$ can be represented by $\log |\mathbb{F}|$ bits. We also fix an extension field $\mathbb{E} \supset \mathbb{F}$ to be the smallest extension for which $|\mathbb{E}| \geq 2^{\kappa}$. Each element of $\mathbb{E}$ can be written down using $\mathcal{O}(\kappa)$ bits. We call $\mathbb{E}$ as *Extension Field*. Moreover, without loss of generality, we assume that $n = \text{poly}(\kappa)$. An AMPC protocol is called efficient if the computations and communication of the honest parties in the protocol is $\text{poly}(\kappa)$.

In our AMPC protocol, each honest party performs computations and communication which is poly(n, κ).

A-cast, Agreement on a Core Set (ACS)

A-Cast[12]: It is an asynchronous broadcast primitive, introduced and elegantly implemented by Bracha [9] with $n \geq 3t + 1$. From [9], A-Cast of b bits incurs a private communication of $\mathcal{O}(n^2 b)$ bits. Let Π be an asynchronous protocol initiated by a special party (called the sender), having input m (the message to be broadcast). We say that Π is a t-resilient A-cast protocol if the following holds:

• **Termination**: 1. If the sender is honest and all the honest parties participate in the protocol, then each honest party will eventually terminate the protocol.

2. Irrespective of the behavior of the sender, if any honest party terminates the protocol then each honest party will eventually terminate the protocol.

• **Correctness**: If the honest parties terminate the protocol then they have a common output m^*. Furthermore, if the sender is honest then $m^* = m$.

Agreement on Core Set (ACS)[4, 8]: It is a primitive presented in [6, 8]. It is used to determine a set of $n - t$ parties that correctly shared their values. More concretely, every party P_i starts the ACS protocol with an accumulative set of parties who from P_i's view point correctly shared their values. The output of the protocol is a set of at least $n - t$ parties, who correctly shared their values. The communication cost of ACS protocol is $\Omega(n^4 \log n)$ bits.

Definition 1 (d-Sharing [3]:) *A value $s \in \mathbb{F}$ is said to be d-shared among a set of parties $\overline{\mathcal{P}} \subseteq \mathcal{P}$ with $|\overline{\mathcal{P}}| \geq d+1$ if* **every honest** *$P_i \in \overline{\mathcal{P}}$ is holding a share s_i of s, such that there exists a degree d polynomial $f(x)$ over $\mathbb{F}$ with $f(0) = s$ and $f(j) = s_j$ for every $P_j \in \overline{\mathcal{P}}$. The vector of shares is called a d-sharing of s and is denoted by $[s]_d$. A set of shares (at least $d + 1$) is called d-consistent if these shares lie on a d degree polynomial.*

Definition 2 ($(t, 2t)$-Sharing [5]:) *A value s is said to be $(t, 2t)$-shared among the parties in $\mathcal{P}$, denoted as $[s]_{(t,2t)}$, if s is both t-shared and $2t$-shared among the parties in $\mathcal{P}$.*

3 AMPC Protocol Overview

Our AMPC protocol proceeds in three phases: preparation phase, input phase and computation phase. Every honest party will eventually complete each phase. In the preparation phase, $(t, 2t)$-sharing of $c_M + c_R$ random values will be generated. Each multiplication gate and random gate of the circuit will be associated with a $(t, 2t)$-sharing. In the input phase the parties t-share (commit) their inputs and agree on a core set of $n-t$ parties who correctly t-shared their inputs (every honest party will eventually get a share of the inputs of the parties in the core set). In the computation phase, based on the inputs of the parties in core set, the actual circuit will be computed gate by gate, such that the output of the intermediate gates are always kept as secret and are properly t-shared among the parties. Due to the linearity of the used t-sharing, the linear gates can be computed locally without communication. Each multiplication gate will be evaluated with the help of the $(t, 2t)$-sharing associated with it. For this, we adapt a technique from [17] used in synchronous settings.

4 Reconstruction of d-Sharing

Let $s \in \mathbb{F}$ be a secret which is d-shared among the parties in $\mathcal{P}$ by a degree-d polynomial $f(x)$, such that $f(0) = s$ and $d < n - 2t$. Let $P_\alpha \in \mathcal{P}$. We now give a protocol **d-Rec-Private**, which allows P_α to privately reconstruct $f(x)$ and hence $s = f(0)$ in asynchronous settings. Moreover, if P_α is honest then s remains secure. The high level idea of the protocol is as follows: every party P_i sends his share s_i of s to P_α. The shares may reach P_α in any arbitrary order. Moreover, up to t of the shares may be incorrect or missing. To reconstruct $f(x)$, P_α applies OEC (Online Error Correcting) [6] on the obtained s_i's to get the interpolation polynomial $f(x)$ and reconstructs the secret $s = f(0)$. Roughly speaking, the online error correction method enables P_α to recognize when the received shares define a unique degree-d interpolation polynomial. We call the reconstruction of s by P_α as the P_α-Private-Reconstruction of s.

Lemma 1 *For any secret s which is d-shared among the parties in $\mathcal{P}$, with $d < n - 2t$, protocol d-Rec-Private achieves the following properties tolerating any $\mathcal{A}_t$:*

– Termination: *If P_α is honest, then P_α will eventually terminate* d-Rec-Private.

– Correctness: *An honest P_α will always output s correctly at the end of the protocol.*

– Secrecy: *If P_α is honest then $\mathcal{A}_t$ obtains no information about s.*

Protocol d-Rec-Private *privately communicates $\mathcal{O}(n|\mathbb{F}|)$ bits.*

Protocol d-Rec-Private($\mathcal{P}, d, s, P_\alpha$): P_α-Private-Reconstruction of s

Code for P_i: Send s_i to P_α.

Code for P_α: Apply On-line Error Correcting (OEC) on the received s_i's to interpolate a unique degree-d polynomial $f(x)$ and output $s = f(0)$.

The following observations on d-Rec-Private will be required in the sequel:

(i) Protocol d-Rec-Private can be used to reconstruct t-sharing as well as $2t$-sharing of s which is shared among the parties in $\mathcal{P}$ (since $d < n - 2t$ holds good when $n = 4t + 1$ and $d = 2t$ as well as $d = t$).

(ii) Protocol d-Rec-Private can be used to reconstruct t-sharing of s which is t-shared among the parties in $\overline{\mathcal{P}} \subseteq \mathcal{P}$ with $|\overline{\mathcal{P}}| \geq 3t + 1$. That is if $\overline{\mathcal{P}}$ is any $3t + 1$ sized subset of $\mathcal{P}$, then d-Rec-Private($\overline{\mathcal{P}}, t, s, P_\alpha$) ensures successful P_α-Private-Reconstruction of s.

5 Generating $(t, 2t)$-Sharing

We now present a novel protocol, called (t,2t)-Share-Multiple that allows a dealer $D \in \mathcal{P}$ (dealer can be any party from $\mathcal{P}$) to $(t, 2t)$-share $\ell \geq 1$ secrets from $\mathbb{F}$ concurrently among the parties in $\mathcal{P}$ with error probability of $2^{-\Omega(\kappa)}$. For the ease of understanding, we first present a protocol, called (t,2t)-Share-Single that allows D to $(t, 2t)$-share a single secret among the parties in $\mathcal{P}$ with error probability of $2^{-\Omega(\kappa)}$. Protocol (t,2t)-Share-Single and (t,2t)-Share-Multiple may not terminate when D is corrupted. But if an honest party terminates these protocols, then eventually all the honest parties will also terminate the protocols. Moreover, if D is honest then the secret(s) remains secure after the execution of the protocols.

We now provide a broad-level overview of (t,2t)-Share-Single. For this, we follow an evolutionary approach, where we explain the problems faced at each step and the measures taken to tackle those problems. To begin with, D on having a secret s, selects random polynomials $F(x)$ and $G(x)$ over $\mathbb{F}$ of degree t and $2t$ respectively such that $F(0) = G(0) = s$. To $(t, 2t)$-share s among the parties in $\mathcal{P}$, every $P_i \in \mathcal{P}$ should hold $F(i)$ and $G(i)$ at the end of the protocol such that parties have verified that both $F(x)$ and $G(x)$ share same secret s. It is clear that allowing D to directly send $F(i)$ and $G(i)$ to party P_i can not be a solution. This is because, a corrupted D may send wrong values to parties i.e the values given to the parties do not define t and $2t$ degree polynomial respectively. Moreover the polynomials may not have common secret as their constant term. Also a corrupted D may choose not to deliver values to t potential honest parties and since the protocol is to be executed over asynchronous network, we can not wait for those t parties to receive their values (as this waiting may turn out to be endless).

To overcome the above problems, we follow a different strategy: D is asked to select a random degree-t polynomial $f^0(x)$ and $2n$ degree-t polynomials $f^1(x), \ldots, f^n(x), g^1(x), \ldots, g^n(x)$ over $\mathbb{F}$ such that:

(i) For $i = 1, \ldots, n$, the points $(0, f^0(i))(1, f^1(i)), \ldots (n, f^n(i))$ define a unique t polynomial.

(ii) For $i = 1, \ldots, n$, the points $(0, f^0(i))(1, g^1(i)), \ldots (n, g^n(i))$ define a unique $2t$ degree polynomial.

(iii) For $i = 0, \ldots, n$, $f^i(0) = F(i)$ and for $i = 1, \ldots, n$, $g^i(0) = G(i)$. Moreover, $f^0(0) = F(0) = G(0)$.

Given $F(x)$ and $G(x)$, D can always select the above polynomials in the following manner:

Generate($F(x), G(x)$)

1. D selects following degree-t random polynomials over $\mathbb{F}$:
   (a) a polynomial $f^0(x)$, such that $f^0(0) = F(0) = G(0) = s$
   (b) t polynomials $f^1(x), \ldots, f^t(x)$ such that for $i = 1, \ldots, t$, $f^i(0) = F(i)$
   (c) $2t$ polynomials $g^1(x), \ldots, g^{2t}(x)$, such that for $j = 1, \ldots, 2t$, $g^j(0) = G(j)$.
2. Using $f^0(x), f^1(x), \ldots, f^t(x)$, D constructs additional $3t + 1$ degree-t polynomials $f^{t+1}(x), \ldots, f^n(x)$, such that for $i = 0, \ldots, n$ the points $f^0(i), f^1(i), \ldots, f^n(i)$ lie on a unique t-degree polynomial. This also ensures that $f^i(0) = F(i)$.
3. Using $f^0(x), g^1(x), \ldots, g^{2t}(x)$, D constructs additional $2t+1$ degree-t polynomials $g^{2t+1}(x), \ldots, g^n(x)$, such that for $i = 0, \ldots, n$ the points $f^0(i), g^1(i), \ldots, g^n(i)$ lie on a unique $2t$-degree polynomial. This ensures that for $i = 1, \ldots, n$, $g^i(0) = G(i)$.

We call the polynomials $f^0(x), \ldots, f^n(x), g^1(x), \ldots, g^n(x)$ as *base polynomials*. Now D delivers i^{th} point on the base polynomials to P_i. Upon receiving these points, P_i checks whether $f^0(i), f^1(i), \ldots, f^n(i)$ lie on a unique t-degree polynomial and $f^0(i), g^1(i), \ldots, g^n(i)$ lie on a unique $2t$-degree polynomial. Now if the parties in $\mathcal{P}$ can agree on a set of at least $3t+1$ parties, say $CORE$, such that *every honest party in $CORE$ has checked the above conditions and it is ensured that honest parties in $CORE$ hold correct points on degree-t polynomials $f^0(x), \ldots, f^n(x), g^1(x), \ldots, g^n(x)$*, then every party in $\mathcal{P}$ can conclude that $f^0(0), f^1(0), \ldots, f^n(0)$ defines a t degree polynomial, say $F(x)$ and $f^0(0), g^1(0), \ldots, g^n(0)$ defines a $2t$ degree polynomial, say $G(x)$ with $F(0) = G(0) = f^0(0)$. This claim follows from the following argument: Let H be the set of honest parties in $CORE$. So $|H| \geq 2t+1$. Assume that for every $P_i \in H$, the points $((0, f^0(i)), \ldots, (n, f^n(i)))$ define a unique t degree polynomial, say $r^i(x)$. This implies that the points $f^0(0), \ldots, f^n(0)$ also define a unique t degree polynomial, say $F(x)$ as every honest $P_i \in H$ holds correct points on degree-t polynomials $f^0(x), \ldots, f^n(x)$ and $F(x)$ can be expressed as a linear combination of $r^i(x)$'s (corresponding to parties in H). Moreover $F(0)$ is same as $f^0(0)$. Similarly, assume that for every $P_i \in H$, the points $((0, f^0(i)), (1, g^1(i)), \ldots, (n, g^n(i)))$ define a unique $2t$ degree polynomial, say $q^i(x)$. Then the points $f^0(0), g^1(0), \ldots, g^n(0)$ also define a unique $2t$ degree polynomial, say $G(x)$ as every $P_i \in H$ holds correct points on degree-t polynomials $f^0(x), g^1(x), \ldots, g^n(x)$ and $G(x)$ can be expressed as a linear combination of $q^i(x)$'s. Moreover $G(0) = f^0(0) = F(0)$. So $F(x)$ and $G(x)$ will have common constant term.

Assuming that the honest parties in $\mathcal{P}$ have agreed upon such a $CORE$, our protocol achieves $(t, 2t)$-sharing in the following way: Since $CORE \geq 3t+1$, from the properties of protocol d-Rec-Private, both $f^i(0) = F(i)$ and $g^i(0) = G(i)$ can be P_i-Private-Reconstructed for $i = 1, \ldots, n$. Once this is done, every P_i can output $f^i(0)$ and $g^i(0)$ as the shares of D's committed secret and this will complete the $(t, 2t)$-sharing. Now the question that remains to answer is: *how do the parties in $\mathcal{P}$ decide whether there exists a $CORE$ set of size $3t+1$ such that the parties in it hold proper shares of degree-t base polynomials?*

To accomplish the above, every $P_i \in \mathcal{P}$ acts as a verifier and does the following: P_i waits to receive signals from at least $3t+1$ parties (denoted as $ReceivedSet^{(P_i,1)}$), saying that they have received their respective values on the base polynomials from D and checked the t-consistency as well as $2t$-consistency of the received values. P_i then selects a random element $r^{(P_i,1)}$ from extension field $\mathbb{E}$ and A-cast the tuple $(r^{(P_i,1)}, ReceivedSet^{(P_i,1)})$. On listening $r^{(P_i,1)}$, D computes and A-casts a linear combination of the base polynomials where the scalars (of the linear combination) are function of $r^{(P_i,1)}$, while every party in $ReceivedSet^{(P_i,1)}$ computes and A-casts the same linear combination of the $2n+1$ values on the base polynomials. Now everyone can check whether the linear combination of the values A-casted by parties in $ReceivedSet^{(P_i,1)}$ lie on the linear combination of the polynomial A-casted by D (ideally this should hold). Let us denote the set of parties whose values are consistent with D's A-casted polynomial as $AgreeSet^{(P_i,1)}$. If verifier P_i is honest and $|AgreeSet^{(P_i,1)}| \geq 2t+1$, then it contains at least $t+1$ honest parties and with very high probability they hold correct points on degree-t base polynomials. This is because of the random $r^{(P_i,1)}$, which is selected by P_i. But since $|ReceivedSet^{(P_i,1)}| = 3t+1$, $|AgreeSet^{(P_i,1)}|$ may contain only $2t+1$ parties even for an *honest* D, as t potentially corrupted parties from $ReceivedSet^{(P_i,1)}$ may not purposefully take part in the verification. This may bar to construct $CORE$ of size $3t+1$ even for an *honest* D.

To deal with this, every verifier P_i on receiving signals from new parties, selects new random number from $\mathbb{E}$ and A-cast the number along with updated set $ReceivedSet^{(P_i,*)}$. Thus for j^{th} receipt of signal from a new party P_α, P_i A-cast new $r^{(P_i,j)} \in_R \mathbb{E}$ and $ReceivedSet^{(P_i,j)}$ where $ReceivedSet^{(P_i,j)} = ReceivedSet^{(P_i,j-1)} \cup \{P_\alpha\}$. Thus every verifier P_i will A-cast a tuple "(random number, index set)" at most $t+1$ times. For every such A-cast, D and the parties in $\mathcal{P}$ do the same computation and communication as described earlier for $(r^{(P_i,1)}, ReceivedSet^{(P_i,1)})$. Now consider an *honest* verifier P_i and its two A-casts $(r^{(P_i,\alpha)}, ReceivedSet^{(P_i,\alpha)})$ and $(r^{(P_i,\beta)}, ReceivedSet^{(P_i,\beta)})$ and corresponding Agree Sets $AgreeSet^{(P_i,\alpha)}$ and $AgreeSet^{(P_i,\beta)}$. We argue that if both $AgreeSet^{(P_i,\alpha)}$ and $AgreeSet^{(P_i,\beta)}$ are of size at least $3t+1$ and are not totally same, then with very high probability the values held by the honest parties in both the sets define the same set of degree-t base polynomials. In other words, honest parties in $AgreeSet^{(P_i,\alpha)} \cup AgreeSet^{(P_i,\beta)}$ hold proper values on a single set of degree-t base polynomials. This is because by previous argument, for an honest verifier P_i, the set of honest parties in $AgreeSet^{(P_i,\alpha)}$ hold correct points on degree-t polynomials. The same holds for $AgreeSet^{(P_i,\beta)}$ as well. Now since both the sets are of size $3t+1$, there are at least $2t+1$ parties common between them out of which at least

Protocol (t,2t)-Share-Single($D, \mathcal{P}, s$)

D's Distribution: Code for D:

1. Select random polynomials $G(x)$ and $F(x)$ over $\mathbb{F}$ of degree $2t$ and t respectively, with $G(0) = F(0) = s$.
2. Compute $\mathcal{F} = (f^0(x), f^1(x), \ldots, f^n(x), g^1(x), \ldots, g^n(x)) = \mathsf{Generate}(F(x), G(x))$. We call the polynomials in $\mathcal{F}$ as *base polynomials*.
3. Select $(t+1)n$ degree-t random distinct *blinding polynomials* over $\mathbb{F}$, denoted by $p^{(i,1)}(x), \ldots, p^{(i,t+1)}(x)$ for $i = 1, \ldots, n$.
4. For $i = 1, \ldots, n$, deliver the following to party P_i:
   (a) $\Delta_i = \{f^0(0), f^1(i), \ldots, f^n(i), g^1(i), \ldots, g^n(i)\}$ and (b) $p^{(j,1)}(i), \ldots, p^{(j,t+1)}(i)$ for $j = 1, \ldots, n$.

Verification of D's Distribution: Code for P_i:

i. Code for P_i as a Party:

1. Wait to receive Δ_i and $p^{(j,1)}(i), \ldots, p^{(j,t+1)}(i)$ for $j = 1, \ldots, n$ from D.
2. After receiving, check whether (a) $((0, f^0(i)), (1, f^1(i)), \ldots, (n, f^n(i)))$ defines a unique t degree polynomial and (b) $((0, f^0(i)), (1, g^1(i)), \ldots, (n, g^n(i)))$ defines a unique $2t$ degree polynomial. If yes then send a `Received-From-D` signal to every other party P_j.

ii. Code for P_i as a Verifier:

1. Wait to obtain `Received-From-D` signal from $3t+1$ parties. Put the identities of the $3t+1$ parties in a set $ReceivedSet^{(P_i,1)}$. Select a random $r^{(P_i,1)} \in_R \mathbb{E}$ and A-cast $(r^{(P_i,1)}, ReceivedSet^{(P_i,1)})$.
2. For j^{th} $(j > 1)$ receipt of `Received-From-D` signal from a new party $P_\alpha \notin ReceivedSet^{(P_i,j-1)}$, construct $ReceivedSet^{(P_i,j)} = ReceivedSet^{(P_i,j-1)} \cup \{P_\alpha\}$, select a random $r^{(P_i,j)} \in_R \mathbb{E}$ and A-cast $(r^{(P_i,j)}, ReceivedSet^{(P_i,j)})$.

iii. Code for P_i as a Party:

1. If $(r^{(P_\alpha,\beta)}, ReceivedSet^{(P_\alpha,\beta)})$ is obtained from A-cast of some verifier $P_\alpha \in \mathcal{P}$, then do the following:
   (a) Check if $P_i \in ReceivedSet^{(P_\alpha,\beta)}$. If yes, then A-cast $v_i^{(P_\alpha,\beta)} = \mathsf{LinCombValue}(\{p^{(\alpha,\beta)}(i) \cup \Delta_i\}, R)$, where $R = (1, r^{(P_\alpha,\beta)}, \left(r^{(P_\alpha,\beta)}\right)^2, \ldots, \left(r^{(P_\alpha,\beta)}\right)^{2n+1})$.
   (b) **If P_i is the dealer** D, then A-cast polynomial $F^{(P_\alpha,\beta)}(x) = \mathsf{LinCombPoly}(\{p^{(\alpha,\beta)}(x) \cup \mathcal{F}\}, R)$, where $R = (1, r^{(P_\alpha,\beta)}, \left(r^{(P_\alpha,\beta)}\right)^2, \ldots, \left(r^{(P_\alpha,\beta)}\right)^{2n+1})$.

CORE Construction & Agreement on CORE: Code for P_i:

1. Say that **party P_j agrees with D with respect to** $(r^{(P_\alpha,\beta)}, ReceivedSet^{(P_\alpha,\beta)})$ if the following holds:
   (a) $F^{(P_\alpha,\beta)}(x)$ is t degree polynomial, (b) $P_j \in ReceivedSet^{(P_\alpha,\beta)}$ and (c) $v_j^{(P_\alpha,\beta)} = F^{(P_\alpha,\beta)}(j)$
   where $v_j^{(P_\alpha,\beta)}$, $F^{(P_\alpha,\beta)}(x)$ and $(r^{(P_\alpha,\beta)}, ReceivedSet^{(P_\alpha,\beta)})$ are obtained from the A-casts of P_j, D and P_α respectively.
2. With respect to $(r^{(P_\alpha,\beta)}, ReceivedSet^{(P_\alpha,\beta)})$, when there are $3t+1$ P_j's who agree with D, add all of them in a set $AgreeSet^{(P_\alpha,\beta)}$.
3. Add a verifier P_α to a set $ValidVerifier$ if at least one $AgreeSet^{(P_\alpha,\beta)}$ has been generated.
4. **If P_i is the dealer** D, then check whether $|ValidVerifier| \geq t+1$ and in case of 'yes' perform the following computation:
   (a) For every $P_\alpha \in ValidVerifier$, compute $AgreeSet^{P_\alpha} = \cup_\beta AgreeSet^{(P_\alpha,\beta)}$.
   (b) Compute $CORE = \{P_i \mid P_i$ belongs to $AgreeSet^{P_\alpha}$ for at least t+1 $P'_\alpha s$ in $ValidVerifier\}$.
   (c) If $|CORE| \geq 3t+1$, then A-cast $CORE$. Else delete $AgreeSet^{P_\alpha}$'s, $CORE$ and wait for more updates.
5. Wait to listen $CORE$ from the A-cast of D. Upon listening, locally perform the same steps as specified for D in previous step for obtaining $CORE$. If the computed $CORE$ is a superset of the one received from the A-cast of D, then agree on the $CORE$ as received from D and proceed to the next step.

For $j = 1, \ldots, n$, P_j-Private-Reconstruction of $f^j(0)$ and $g^j(0)$: Code for P_i:

1. If $P_i \in CORE$, participate in d-Rec-Private$(CORE, t, f^j(0), P_j)$ and d-Rec-Private$(CORE, t, g^j(0), P_j)$ for P_j-Private-Reconstruction of $f^j(0)$ and $g^j(0)$, respectively for $j = 1, \ldots, n$.
2. As a receiver participate in d-Rec-Private$(CORE, t, f^i(0), P_i)$ and d-Rec-Private$(CORE, t, g^i(0), P_i)$ for P_i-Private-Reconstruction of $f^i(0)$ and $g^i(0)$.
3. Output $F(i) = f^i(0)$ and $G(i) = g^i(0)$ as the i^{th} shares of secret s and terminate. s is now t-shared as well as $2t$-shared using polynomials $F(x)$ and $G(x)$, respectively

$t+1$ are honest. As all the polynomials are of degree t, it implies that the set of honest parties in $AgreeSet^{(P_i,\alpha)} \cup AgreeSet^{(P_i,\beta)}$ hold correct points on the *same* degree-t polynomials. Similarly, corresponding to any two distinct honest verifiers P_i and P_j, if $|AgreeSet^{(P_i,\alpha)}| \geq 3t+1$ and $|AgreeSet^{(P_j,\beta)}| \geq 3t+1$, then with very high probability, the honest parties in $AgreeSet^{(P_i,\alpha)}$ and $AgreeSet^{(P_j,\beta)}$ hold correct values on the same set of degree-t base polynomials. Now let $AgreeSet^{P_i} = \cup_\alpha AgreeSet^{(P_i,\alpha)}$. Then $CORE$ consists of $3t+1$ parties, where each party from $CORE$ is a part of at least $t+1$ $AgreeSet^{P_i}$'s. Among these $t+1$ $AgreeSet^{P_i}$'s, at least one is corresponding to some honest verifier P_i. This ensures that the values held by the honest parties in $CORE$ lie on degree-t base polynomials.

One last point regarding secrecy of the protocol: every time when D A-casts a linear combination of base polynomials with respect to the A-cast of the verifiers, the secrecy of the base polynomials may be breached. So to keep them secure, D chooses $(t+1)$ *blinding polynomials* corresponding to each verifier P_i and uses one of them without repetition, along with the base polynomials for taking linear combination, for each A-cast of verifier P_i.

The formal description of the protocol incorporates all the above ideas. In the protocol, we use the following notation: given ρ polynomials, $Q = \{q_1(x), \ldots, q_\rho(x)\}$ and a vector $R = (\zeta_1, \ldots, \zeta_\rho)$ of length ρ, we define $q(x)$ as the polynomial obtained by the linear combination of the polynomials in Q with respect to the vector R, where $q(x) = \sum_{i=1}^{\rho} \zeta_i.q_i(x)$. We capture this by: $q(x) = \mathsf{LinCombPoly}(Q, R)$. Similarly, we define $q = \mathsf{LinCombValue}(Q, R)$, where $Q = \{q_1, \ldots, q_\rho\}$ is a set of ρ values and q is their linear combination with respect to R.

Lemma 2 *For every* **honest** *verifier P_α, the values held by the honest parties in any $AgreeSet^{(P_\alpha,\beta)} \subseteq ReceivedSet^{(P_\alpha,\beta)}$ indeed define degree-t base polynomials with very high probability.*

Proof: The lemma is trivially true when D is honest. Hence we consider the case when D is corrupted. First of all, since P_α is honest, he A-casts $r^{(P_\alpha,\beta)}$ only after listening `Received-From-D` signal from all the parties in $ReceivedSet^{(P_\alpha,\beta)}$. Thus D has no knowledge of $r^{(P_\alpha,\beta)}$ when he distributes the values of polynomials to the (honest) parties in $ReceivedSet^{(P_\alpha,\beta)}$. Let $q^0(x), q^1(x), \ldots, q^{2n+1}(x)$ denote the the *blinding polynomial* (corresponding to $r^{(P_\alpha,\beta)}$) and the $2n+1$ base polynomials, defined by the values held by the honest parties in $AgreeSet^{(P_\alpha,\beta)}$. We will show that with very high probability, all these $(2n+2)$ polynomials will have degree t. On the contrary, let us assume that at least one of the polynomials has degree more than t.

Then we show that the polynomial $q(x) = \sum_{i=0}^{2n+1} q^i(x) \left(r^{(P_\alpha,\beta)}\right)^i$ will be of degree t with negligible probability. Let m be such that $q^m(x)$ has maximal degree among $q^0(x), \ldots, q^{2n+1}(x)$, and let t_m be the degree of $q^m(x)$. Then according to the condition, $t_m > t$. Note that $t_m < |H|$, where H is the set of honest parties in $AgreeSet^{(P_\alpha,\beta)}$. This is because given $|H|$ values, the maximum degree polynomial that we can define using them is $|H|-1$. Now each polynomial $q^i(x)$ can be written as $q^i(x) = c^i_{t_m} x^{t_m} + q'^i(x)$ where $q'^i(x)$ has degree lower than t_m. By assumption $c^m_{t_m} \neq 0$. It implies that the vector $(c^0_{t_m}, \ldots, c^{2n+1}_{t_m})$ is not a complete 0 vector. Hence $c_{t_m} = \mathsf{LinCombValue}(\{c^0_{t_m}, \ldots, c^{2n+1}_{t_m}\}, R)$ will be zero with probability $\frac{2n+1}{|\mathbb{E}|} \approx 2^{-\Omega(\kappa)}$ (which is negligible) where $R = (1, r^{(P_\alpha,\beta)}, \left(r^{(P_\alpha,\beta)}\right)^2, \ldots, \left(r^{(P_\alpha,\beta)}\right)^{2n+1})$. This is because the vector $(c^0_{t_m}, \ldots, c^{2n+1}_{t_m})$ may be considered as the set of coefficients of a $2n+1$ degree polynomial, say $\mu(x)$, and hence the value c_{t_m} is the value of $\mu(x)$ evaluated at $r^{(P_\alpha,\beta)}$. Now c_{t_m} will be zero if $r^{(P_\alpha,\beta)}$ happens to be one of the $2n+1$ roots of $\mu(x)$ (since degree of $\mu(x)$ is at most $2n+1$).

Now since $r^{(P_\alpha,\beta)}$ is chosen randomly from $\mathbb{E}$ by P_α, independent of the polynomials $q^0(x), \ldots, q^{2n+1}(x)$, the probability that it is a root of $\mu(x)$ is $\frac{2n+1}{|\mathbb{E}|} \approx 2^{-\Omega(\kappa)}$. So with very high probability c_{t_m} which is the t^{th}_m coefficient of $q(x)$ is non-zero. This implies $q(x)$ will be of degree at least $t_m > t$.

Note that the values $v_i^{(P_\alpha,\beta)}$A-casted by the parties in H define the polynomial $q(x)$ which is of degree more than t. Moreover, the same values also lie on the t degree polynomial $F^{(P_\alpha,\beta)}(x)$, A-casted by D. Thus the difference polynomial $dp(x) = q(x) - F^{(P_\alpha,\beta)}(x)$ will have zero at all i where $P_i \in H$. Since $dp(x)$ has $|H|$ roots, it must have degree at least $|H|$. But since $q(x)$ has degree t_m, $dp(x)$ will have degree at most t_m which is less than $|H|$. These two argument together implies that $dp(x)$ is zero-polynomial. Thus $q(x)$ must be same as $F^{(P_\alpha,\beta)}(x)$ which is of degree t. Hence with very high probability all the $q^i(x)$'s are of degree t. □

Lemma 3 *For every* **honest** *verifier P_α, the values held by honest parties in any two sets $AgreeSet^{(P_\alpha,\beta)}$ and $AgreeSet^{(P_\alpha,\gamma)}$ with $\beta \neq \gamma$, define the same degree-t base polynomials with very high probability.*

PROOF: By Lemma 2, the values held by the honest parties in $AgreeSet^{(P_\alpha,\beta)}$ and $AgreeSet^{(P_\alpha,\gamma)}$ define degree-t base polynomials $\overline{f^0(x)},\ldots,\overline{f^n(x)},\overline{g^1(x)},\ldots,\overline{g^n(x)}$ and $\widehat{f^0(x)},\ldots,\widehat{f^n(x)},\widehat{g^1(x)},\ldots,\widehat{g^n(x)}$ respectively with high probability. We claim that these two sets of polynomials are identical. Since $AgreeSet^{(P_\alpha,\beta)}$ and $AgreeSet^{(P_\alpha,\gamma)}$ are of size at least $3t+1$, there are at least $2t+1$ common parties between them out of which at least $t+1$ are honest. Since all the polynomials are of degree t, any $t+1$ points completely and uniquely define them and hence these two set of polynomials can not be different while having $t+1$ common values. □

Lemma 4 *For any two* **honest** *verifiers P_α and P_δ, the base polynomials defined by the values held by honest parties in any two sets $AgreeSet^{(P_\alpha,\beta)}$ and $AgreeSet^{(P_\delta,\gamma)}$ are same with very high probability.*

PROOF: Follows from the proof of Lemma 3.

Theorem 1 *The protocol (t,2t)-Share-Single achieves the following properties:*

–TERMINATION: *(a) If D is honest, then every honest party will eventually terminate (t,2t)-Share-Single. (b) If D is corrupted and some honest party has terminated (t,2t)-Share-Single, then all the honest parties will eventually terminate (t,2t)-Share-Single.*

–CORRECTNESS: *(a) If D is honest then each honest party will eventually hold proper $(t,2t)$ share of s. (b) If D is corrupted and some honest party has terminated (t,2t)-Share-Single, then there exists a unique value $\overline{s}\in\mathbb{F}$ which is $(t,2t)$-shared among the parties in $\mathcal{P}$ except with an error probability of $2^{-\Omega(\kappa)}$.*

–SECRECY: *If D is honest, then $\mathcal{A}_t$ obtains no information about the secret s*

PROOF: **Termination:** When D is honest, then eventually for every honest verifiers P_α, the set $AgreeSet^{P_\alpha}$ will contain all the honest parties in $\mathcal{P}$. Since there are at least $3t+1$ honest verifiers, $CORE$ will eventually contain all the honest parties in $\mathcal{P}$. Thus when D is honest, every honest party will eventually terminate (t,2t)-Share-Single. This proves part (a).

If some honest party P_i has terminated (t,2t)-Share-Single, then he must have checked the validity of $CORE$ obtained from the A-cast of D. Moreover, P_i would have privately reconstructed $F(i)$ and $G(i)$. In the same way, every other honest party P_j will check the validity of $CORE$, privately reconstruct $F(j),G(j)$ and eventually terminate (t,2t)-Share-Single. This proves part (b).

Correctness: If D is honest then his secret s will be correctly $(t,2t)$-shared among the parties. This is because $CORE$ contains at least $3t+1$ parties out of which t can be corrupted. Moreover, every honest party in $CORE$ holds proper values of the polynomials $f^0(x),\ldots,f^n(x),g^1(x),\ldots,g^n(x)$. So by the property of d-Rec-Private, P_i-Private-Reconstruction of both $f^i(0)$ and $g^i(0)$ is possible for all $i=1,\ldots,n$. This implies every $P_i\in\mathcal{P}$ has $F(i)=f^i(0)$ and $G(i)=g^i(0)$ which are the shares of s. So s is $(t,2t)$-shared among the parties in $\mathcal{P}$ using $F(x)$ and $G(x)$.

We now prove the correctness when D is corrupted. By the computation of $CORE$, every party in $CORE$ is guaranteed to present in $AgreeSet$ of at least one honest party. By Lemma 3, with high probability, the values held by all the honest parties in $AgreeSet^{P_\alpha}$ define t-degree base polynomials, say $\overline{f^0(x)},\ldots,\overline{f^n(x)},\overline{g^1(x)},\ldots,\overline{g^n(x)}$. Moreover, by Lemma 4, the values held by the honest parties in the union of $AgreeSet^{P_\alpha}$'s for all *honest* P_α also define the same set of t-degree base polynomials with very high probability. This implies that the values held by all the honest parties in $CORE$, define $\overline{f^0(x)},\ldots,\overline{f^n(x)},\overline{g^1(x)},\ldots,\overline{g^n(x)}$. Now as in the case of honest D, every $P_i\in\mathcal{P}$ will hold $\overline{f^i(0)}=\overline{F(i)}$ and $\overline{g^i(0)}=\overline{G(i)}$ after P_i-Private-Reconstruction of $\overline{f^i(0)}$ and $\overline{g^i(0)}$. So $\overline{s}=\overline{F(0)}=\overline{G(0)}$ is $(t,2t)$-shared among the parties in $\mathcal{P}$ using $\overline{F(x)}$ and $\overline{G(x)}$ with very high probability.

Secrecy: We have to consider the case when D is honest. Without loss of generality, let $\mathcal{A}_t$ controls $P_1,\ldots,P_t$. So $\mathcal{A}_t$ will know $f^1(x),f^t(x),g^1(x),\ldots,g^t(x)$ and hence $F(1),\ldots,F(t)$ and $G(1),\ldots,G(t)$. Moreover, $\mathcal{A}_t$ will also know t distinct points on $f^0(x),f^{t+1}(x),\ldots,f^n(x),g^{t+1}(x),\ldots,g^n(x)$. Since these polynomials are of degree-t, $\mathcal{A}_t$ lacks one point on each of them to uniquely interpolate them. Notice that each time D A-Casts a linear combination of $f^0(x),\ldots,f^n(x),g^1(x),\ldots,g^n(x)$, he uses a random degree-t *blinding polynomial*. Thus it holds that $\mathcal{A}_t$ lacks one point on $f^0(x),f^{t+1}(x),\ldots,f^n(x),g^{t+1}(x),\ldots,g^n(x)$ to interpolate them and hence $s=F(0)=G(0)$ will remain information theoretically secure. □

We now present protocol (t,2t)-Share-Multiple that allows a dealer $D\in\mathcal{P}$ to concurrently generate $(t,2t)$-sharing of ℓ secrets from $\mathbb{F}$, denoted as $S=(s^1,\ldots,s^\ell)$. Due to its similarity with (t,2t)-Share-Single, we present the protocol in **APPENDIX A**. We now have the following theorem:

Theorem 2 *Protocol (t,2t)-Share-Multiple satisfies the same properties as given in Theorem 1. The protocol privately communicates $\mathcal{O}(\ell n^2 \log |\mathbb{F}|)$ bits and A-cast of $\mathcal{O}(n^3\kappa)$ bits where $\kappa = \log |\mathbb{E}|$.*

A Brief Discussion on Existing Protocols for Generating $(t, 2t)$-sharing: In [18], Damgard et. al. have proposed a protocol that generates $(t, 2t)$-sharing of ℓ secrets concurrently in *synchronous settings* with $n = 3t+1$ parties, conditioned on the event that all the parties correctly follow the protocol steps; i.e., behave honestly. If at least one party behaves in a corrupted manner, then the protocol of [18] fails to generate the $(t, 2t)$-sharing and terminates with a pair of parties in which at least one is corrupted. By allowing such susceptibility to the bad behavior of the corrupted parties, their protocol involves a communication complexity of $\mathcal{O}(\ell n \log |\mathbb{F}| + poly(n, \kappa))$ bits, where κ is the error probability of the protocol. The protocol of [17] cannot be directly adapted to asynchronous settings.

Later in [4], the authors have generated $(t, 2t)$-sharing of a single secret in *asynchronous settings* from $3t + 1$ t-sharing of random values in asynchronous settings. Briefly, the authors have done the following: Let $[r^0]_t, \ldots, [r^{3t}]_t$ be the t-sharing of $3t + 1$ random values. Let $p(x)$ be the t-degree polynomial defined by the $t + 1$ coefficients $r^0, \ldots, r^t$. Let $q(x)$ be the $2t$-degree polynomial defined by the $2t + 1$ coefficients $r^0, r^{t+1} \ldots, r^{3t}$. It is to be noted that both $p(x)$ and $q(x)$ have common constant term (which is r^0). Now the parties jointly perform some computation such that every party P_i receives $p(i)$ and $q(i)$ at the end. This ensures that r^0 is $(t, 2t)$-shared among the parties. To generate t-sharing of $3t + 1$ random values, the authors in [4] have used a protocol, which involves a private communication of $\mathcal{O}(n^3 \log |\mathbb{F}|)$ bits and A-Cast of $\mathcal{O}(n^2 \log(|\mathbb{F}|))$ bits. Thus the protocol of [4] requires a private communication of $\mathcal{O}(n^3 \log |\mathbb{F}|)$ bits and A-Cast of $\mathcal{O}(n^2 \log(|\mathbb{F}|))$ bits to generate $(t, 2t)$ sharing of a single secret. Moreover, the protocol of [4] does not involve any error probability.

From the above discussion, we conclude that our protocol offers the most efficient way of generating $(t, 2t)$ sharing of ℓ secrets concurrently in asynchronous settings.

6 Preparation Phase

The goal of the preparation phase is to generate correct $(t, 2t)$-sharing of $c_M + c_R$ *secret* random values. We now present a protocol called PreparationPhase which achieves the same.

Protocol PreparationPhase($\mathcal{P}$)

Secret Sharing: Code for P_i:

1. Select $L = \frac{c_M + c_R}{n-2t}$ random secret elements $(s^{(i,1)}, \ldots, s^{(i,L)})$ from $\mathbb{F}$. As a dealer, invoke (t,2t)-Share-Multiple$(P_i, \mathcal{P}, S^i)$ to generate $(t, 2t)$-sharing of $S^i = (s^{(i,1)}, \ldots, s^{(i,L)})$.
2. For $j = 1, \ldots, n$, participate in (t,2t)-Share-Multiple$(P_j, \mathcal{P}, S^j)$.

Agreement on a Core-Set: Code for P_i

1. Create an accumulative set $C^i = \emptyset$. Upon terminating (t,2t)-Share-Multiple$(P_j, \mathcal{P}, S^j)$, include P_j in C^i.
2. Take part in ACS with the accumulative set C^i as input.

Generation of Random $(t, 2t)$-sharing: Code for P_i:

1. Wait until ACS completes with output C containing $n - t$ parties. Obtain the i^{th} shares $\varphi_i^{(j,1)}, \ldots, \varphi_i^{(j,L)}$ corresponding to t-sharing of S_j and i^{th} shares $\phi_i^{(j,1)}, \ldots, \phi_i^{(j,L)}$ corresponding to $2t$-sharing of S^j for every $P_j \in C$. Without loss of generality, let $C = \{P_1, \ldots, P_{n-t}\}$.
2. Let V denotes a $(n - t) \times (n - 2t)$ publicly known *Vandermonde Matrix*.
   (a) For every $k \in \{1, \ldots, L\}$, let $(r^{(1,k)}, \ldots, r^{(n-2t,k)}) = (s^{(1,k)}, \ldots, s^{(n-t,k)})V$.
   (b) Locally compute i^{th} shares corresponding to t-sharing of $r^{(1,k)}, \ldots, r^{(n-2t,k)}$ as $(\varsigma_i^{(1,k)}, \ldots, \varsigma_i^{(n-2t,k)}) = (\varphi_i^{(1,k)}, \ldots, \varphi_i^{(n-t,k)})V$.
   (c) Locally compute i^{th} shares corresponding to $2t$-sharing of $r^{(1,k)}, \ldots, r^{(n-2t,k)}$ as $(\sigma_i^{(1,k)}, \ldots, \sigma_i^{(n-2t,k)}) = (\phi_i^{(1,k)}, \ldots, \phi_i^{(n-t,k)})V$ and terminate.

The values $r^{(1,1)}, \ldots, r^{(n-2t,1)}, \ldots, r^{(1,L)}, \ldots, r^{(n-2t,L)}$ denotes the $c_M + c_R$ random secrets which are $(t, 2t)$-shared.

PreparationPhase asks individual party to act as a dealer and to $(t, 2t)$-share $\frac{c_M + c_R}{n-2t}$ random secrets. Then an instance of ACS protocol is executed to agree on a core set of $n - t$ parties who have correctly $(t, 2t)$-shared $\frac{c_M + c_R}{n-2t}$ random secrets. Now out of these $n - t$ parties, at least $n - 2t$ are honest. Hence the random secrets that are $(t, 2t)$-shared by these $n - 2t$ honest parties are truly random and unknown

to $\mathcal{A}_t$. So if we consider the $(t, 2t)$-sharing done by the honest parties (each of them has done $\frac{c_M+c_R}{n-2t}$ $(t, 2t)$-sharing) in core set, then we will get $\frac{c_M+c_R}{n-2t} * (n-2t) = c_M + c_R$ random $(t, 2t)$-sharing. For this, we use *Vandermonde Matrix* [17] and it's ability to extract randomness which has been exploited in [17, 4].

Vandermonde Matrix and Randomness Extraction [17]: Let $\beta_1, \ldots, \beta_c$ be distinct elements and publicly known elements from $\mathbb{F}$. We denote an $(r \times c)$ Vandermonde matrix by $V^{(r,c)}$, where for $1 \leq i \leq c$, the i^{th} column of $V^{(r,c)}$ is $(\beta_i^0, \ldots, \beta_i^{r-1})^T$. The idea behind extracting randomness using $V^{(r,c)}$ is as follows: without loss of generality, assume that $r > c$. Moreover, let $(x_1, \ldots, x_r)$ be such that (a) *any* c elements of it are chosen uniformly at random from $\mathbb{F}$ and are unknown to adversary $\mathcal{A}_t$, (b) the remaining $r - c$ elements are chosen with an arbitrary distribution from $\mathbb{F}$, independent of the c elements, and are also known to $\mathcal{A}_t$. Now if we compute $(y_1, \ldots, y_c) = (x_1, \ldots, x_r)V$, then $(y_1, \ldots, y_c)$ is an uniformly random vector of length c unknown to $\mathcal{A}_t$, extracted from $(x_1, \ldots, x_r)$ [17, 4].

Lemma 5 *Each honest party will eventually terminate* PreparationPhase. *The protocol correctly generates $(t, 2t)$-sharing of $c_M + c_R$ secret random values, except with error probability of $2^{-\Omega(\kappa)}$ by privately communicating $O((c_M + c_R)n^2 \log |\mathbb{F}|)$ bits,* A-Casting *$\mathcal{O}(n^4\kappa)$ bits and executing one invocation to* ACS. *Moreover, $\mathcal{A}_t$ will have no information about the random values.*

7 Input Phase

In protocol InputPhase, each $P_i \in \mathcal{P}$ acts as a dealer to t-share his input X_i containing c_i elements from $\mathbb{F}$. So total number of inputs $c_I = \sum_{i=1}^{n} c_i$. To achieve this, party P_i $(t, 2t)$-share his input X_i by acting as a dealer and executing (t,2t)-Share-Multiple. The asynchrony of the network does not allow the parties to wait for more than $n - t = 3t + 1$ parties to complete their instance of (t,2t)-Share-Multiple. In order to agree on a core set of parties whose instance of (t,2t)-Share-Multiple has terminated and whose inputs will be taken into consideration for computation (of the circuit), one instance of ACS is invoked. At the end, everyone ignores the $2t$-sharing and considers only t-sharing from the $(t, 2t)$-sharing of all the inputs shared by parties in core set.

Protocol InputPhase($\mathcal{P}$)

Secret Sharing: Code for P_i

1. Having input X_i, invoke (t,2t)-Share-Multiple$(P_i, \mathcal{P}, X_i)$, as a dealer to generate $(t, 2t)$-sharing of X_i.
2. For every $j = 1, \ldots, n$, participate in (t,2t)-Share-Multiple$(P_j, \mathcal{P}, X_j)$.

Agreement on a Core-Set: Code for P_i

1. Create an accumulative set $C^i = \emptyset$. Upon terminating (t,2t)-Share-Multiple$(P_j, \mathcal{P}, X_j)$ with dealer P_j, add P_j in C^i.
2. Participate in ACS with the accumulative set C^i as input.

Output Generation: Code for P_i:

1. Waits until ACS completes with output C containing $n - t$ parties. Ignore the local shares corresponding to inputs of the parties not in C. Ignore the shares corresponding to $2t$-sharing of the inputs of the parties in C, output only the the shares corresponding to t-sharing of the inputs of the parties in C and terminate.

Lemma 6 *Each honest party will eventually terminate* InputPhase. *Moreover, each honest party will correctly output t-sharing of the inputs of the parties in core set C with high probability. The protocol privately communicates $O(c_I n^2 \log |\mathbb{F}|)$ bits,* A-Casts *$\mathcal{O}(n^4\kappa)$ bits and requires one invocation to* ACS. *Furthermore, $\mathcal{A}_t$ will have no information the inputs of the honest parties in C.*

8 Computation Phase

Once the input phase is over, in the computation phase, the circuit is evaluated gate by gate, where all inputs and intermediate values are t-shared among the parties. As soon as a party holds his shares of the input values of a gate, he joins the computation of the gate.

Due to the linearity of the secret-sharing scheme, linear gates can be computed locally simply by applying the linear function to the shares, i.e. for any linear function $c = f(a, b)$, the sharing $[c]_t$ is computed by letting every party P_i to compute $c_i = f(a_i, b_i)$, where a_i, b_i and c_i are the i^{th} shares of a, b and c respectively. With every random gate, one random $(t, 2t)$-sharing (from the preparation phase) is

associated, whose t-sharing is directly used as outcome of the random gate. With every multiplication gate, one random $(t, 2t)$-sharing (from the preparation phase) is associated, which is then used to compute t-sharing of the product, following the technique of Damgard et. al. [17] in synchronous settings. Given a $(t, 2t)$-sharing of a secret random value r (i.e., $[r]_{(t,2t)}$), the technique of Damgard et. al. [17] allows to evaluate a multiplication gate at the cost of one reconstruction. Let $z = xy$, where x, y are the inputs of the multiplication gate which are t-shared, i.e. $[x]_t, [y]_t$. Now for computing $[z]_t$, the t-sharing of z, the parties compute $[C]_{2t} = [x]_t.[y]_t + [r]_{2t}$. Then C is P_i-Private-Reconstructed for every $P_i \in \mathcal{P}$. Now every party defines $[C]_t$ as the default sharing of C, e.g., the constant degree-0 polynomial C and computes $[z]_t = [C]_t - [r]_t$. The security follows from the fact that r is random and independent of x and y [17, 4].

Protocol ComputationPhase($\mathcal{P}$)

For every gate in the circuit: Code for P_i
Wait until the i^{th} share of each of the inputs of the gate is available. Now depending on the type of the gate, proceed as follows:

1. **Input Gate:** $[s]_t = \mathsf{IGate}([s]_t)$: There is nothing to be done here.
2. **Linear Gate:** $[z]_t = \mathsf{LGate}([x]_t, [y]_t, \ldots)$: Compute $z_i = \mathsf{LGate}(x_i, y_i, \ldots)$, the i^{th} share of $z = \mathsf{LGate}(x, y, \ldots)$, where $x_i, y_i, \ldots$ denotes i^{th} share of $x, y, \ldots$.
3. **Multiplication Gate:** $[z]_t = \mathsf{MGate}([x]_t, [y]_t, [r]_{(t,2t)})$:
   (a) Let $[r]_{(t,2t)}$ be the random $(t, 2t)$-sharing associated with the multiplication gate. Also let $(\varphi_1, \ldots, \varphi_n)$ and $(\phi_1, \ldots, \phi_n)$ denote the t-sharing and $2t$-sharing of r, respectively.
   (b) Compute $C_i = x_i.y_i - \phi_i$ the i^{th} share of C which is now $2t$-shared.
   (c) Participate in $\mathsf{d\text{-}Rec\text{-}Private}(\mathcal{P}, 2t, C, P_j)$ for P_j-Private-Reconstruction of C for all $j = 1, \ldots, n$.
   (d) Participate in $\mathsf{d\text{-}Rec\text{-}Private}(\mathcal{P}, 2t, C, P_i)$ as a receiver to reconstruct C. Compute $z_i = C - \varphi_i$, the i^{th} share of z.
4. **Random Gate:** $[R]_t = \mathsf{RGate}([r]_{(t,2t)})$: Let $[r]_{(t,2t)}$ be the random $(t, 2t)$-sharing associated with the random gate. Also let $(\varphi_1, \ldots, \varphi_n)$ denote the t-sharing of r. Assign $R_i = \varphi_i$ as the i^{th} share of $R(= r)$.
5. **Output Gate:** $x = \mathsf{OGate}([x]_t)$: Participate in $\mathsf{d\text{-}Rec\text{-}Private}(\mathcal{P}, t, x, P_i)$ for every $P_j \in \mathcal{P}$. Participate in $\mathsf{d\text{-}Rec\text{-}Private}(\mathcal{P}, t, x, P_i)$ as a receiver to reconstruct x. Output x.

Lemma 7 *Each honest party will eventually terminate ComputationPhase. Given $(t, 2t)$-sharing of $c_M + c_R$ secret random values, the protocol computes the outputs of the circuit secretly by privately communicating $O(n^2(c_M + c_O) \log |\mathbb{F}|)$ bits. The outputs of the circuit will be correct except probability $2^{-\Omega(\kappa)}$*

9 The AMPC Protocol

Now our new AMPC protocol AMPC for evaluating function f is: (1). Invoke PreparationPhase (2). Invoke InputPhase (3). Invoke ComputationPhase.

Theorem 3 *For every coalition of up to $t < n/4$ bad players, the protocol AMPC securely computes the circuit representing function f and terminates always. AMPC privately communicates $\mathcal{O}((c_I + c_M + c_R + c_O)n^2 \log |\mathbb{F}|)$ bits, A-Casts $\mathcal{O}(n^4\kappa)$ bits and requires 2 invocations to ACS. The protocol correctly computes the circuit except with error probability of $2^{-\Omega(\kappa)}$.*

10 Conclusion

In this paper, we have designed an efficient information theoretic secure AMPC protocol with $n = 4t + 1$ that satisfies **Termination** and **Secrecy** perfectly (without any error) and achieves **Correctness** except with negligible error probability of $2^{-\Omega(\kappa)}$, where κ is the error parameter. Our protocol attains quadratic communication ($\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits) per multiplication. Our AMPC protocol provides the best communication complexity among all the known AMPC protocols providing information theoretic security. The key factor behind the communication efficiency of our AMPC protocol is a new and efficient method of generating $(t, 2t)$-*sharing* of multiple secrets *concurrently* in asynchronous settings.

It would be interesting to see whether it is possible to further reduce the communication complexity of the AMPC protocol with $n = 4t+1$ by using techniques such as player elimination [22]. Moreover, one can try to design an AMPC protocol with $n = 4t + 1$ that is perfect (errorless) in all respects, namely termination, secrecy and correctness while maintaining quadratic communication.

References

[1] D. Beaver. Efficient multiparty protocols using circuit randomization. In *Proc. of CRYPTO 1991*, volume 576 of *LNCS*, pages 420–432. Springer Verlag, 1991.

[2] Z. Beerliová-Trubíniová, M. Fitzi, M. Hirt, U. M. Maurer, and V. Zikas. MPC vs. SFE: Perfect security in a unified corruption model. In *TCC*, pages 231–250, 2008.

[3] Z. Beerliová-Trubíniová and M. Hirt. Efficient multi-party computation with dispute control. In *Proc. of TCC*, pages 305–328, 2006.

[4] Z. Beerliová-Trubíniová and M. Hirt. Simple and efficient perfectly-secure asynchronous MPC. In *ASIACRYPT*, pages 376–392, 2007.

[5] Z. Beerliová-Trubíniová and M. Hirt. Perfectly-secure MPC with linear communication complexity. In *Proc. of TCC 2008*, volume 4948 of *LNCS*, pages 213–230. Springer Verlag, 2008.

[6] M. Ben-Or, R. Canetti, and O. Goldreich. Asynchronous secure computation. In *STOC*, pages 52–61, 1993.

[7] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. of 20th ACM STOC*, pages 1–10, 1988.

[8] M. BenOr, B. Kelmer, and T. Rabin. Asynchronous secure computations with optimal resilience. In *PODC*, pages 183–192, 1994.

[9] G. Bracha. An asynchronous $\lfloor (n-1)/3 \rfloor$-resilient consensus protocol. In 3^{rd} *ACM PODC*, pages 154 – 162, 1984.

[10] C. Cachin, K. Kursawe, A. Lysyanskaya, and R. Strobl. Asynchronous verifiable secret sharing and proactive cryptosystems. In *ACM Conference on Computer and Communications Security*, pages 88–97, 2002.

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

[12] R. Canetti and T. Rabin. Fast asynchronous Byzantine agreement with optimal resilience. In *Proc. of STOC 1993*, pages 42–51. ACM, 1993.

[13] D. Chaum, C. Crpeau, and I. Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Proc. of FOCS 1988*, pages 11–19, 1988.

[14] R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin. Efficient multiparty computations secure against an adaptive adversary. In *Proc. of EUROCRYPT 1999*, volume 1592 of *LNCS*, pages 311–326. Springer Verlag, 1999.

[15] R. Cramer, I. Damgård, and S. Fehr. On the cost of reconstructing a secret, or VSS with optimal reconstruction phase. In *CRYPTO*, pages 503–523, 2001.

[16] R. Cramer, I. Damgård, and U. M. Maurer. General secure multi-party computation from any linear secret-sharing scheme. In *EUROCRYPT*, pages 316–334, 2000.

[17] I. Damgård and J. B. Nielsen. Scalable and unconditionally secure multiparty computation. In *Proc. of CRYPTO*, volume 4622 of *LNCS*, pages 572–590. Springer Verlag, 2007.

[18] I. Damgård and J. B. Nielsen. Scalable and unconditionally secure multiparty computation. In *Proc. of CRYPTO*, pages 572–590, 2007.

[19] I. Damgrd, M. Geisler, M. Krigaard, and J. Buus Nielsen. Asynchronous multiparty computation: Theory and implementation. Cryptology ePrint Archive, Report 2008/415, 2008.

[20] R. Gennaro, M. O. Rabin, and T. Rabin. Simplified VSS and fact-track multiparty computations with applications to threshold cryptography. In *PODC*, pages 101–111, 1998.

[21] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proc. of 19th ACM STOC*, pages 218–229, 1987.

[22] M. Hirt, U. Maurer, and B. Przydatek. Efficient secure multiparty computation. In *Proc. of ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 143–161. Springer Verlag, 2000.

[23] M. Hirt and U. M. Maurer. Complete characterization of adversaries tolerable in secure multi-party computation. In *PODC*, pages 25–34, 1997.

[24] M. Hirt, U. M. Maurer, and V. Zikas. MPC vs. SFE : Unconditional and computational security. In *ASIACRYPT*, pages 1–18, 2008.

[25] M. Hirt, J. B Nielsen, and B. Przydatek. Cryptographic asynchronous multi-party computation with optimal resilience (extended abstract). In *EUROCRYPT*, pages 322–340, 2005.

[26] M. Hirt, J. B Nielsen, and B. Przydatek. Asynchronous multi-party computation with quadratic communication. In *ICALP (2)*, pages 473–485, 2008.

[27] J. Katz and C. Y. Koo. Round-efficient secure computation in point-to-point networks. In *Proc. of EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 311–328. Springer Verlag, 2007.

[28] M. V. N. Ashwin Kumar, K. Srinathan, and C. Pandu Rangan. Asynchronous perfectly secure computation tolerating generalized adversaries. In *ACISP*, pages 497–512, 2002.

[29] R. Ostrovsky and M. Yung. How to withstand mobile virus attacks. In *Proc. of 10th PODC*, pages 51–61. ACM Press, 1991.

[30] A. Patra, A. Choudhary, and C. Pandu Rangan. Efficient asynchronous multiparty computation with optimal resilience. Cryptology ePrint Archive, Report 2008/425, 2008.

[31] A. Patra, A. Choudhary, and C. Pandu Rangan. Round efficient unconditionally secure multiparty computation protocol. In *INDOCRYPT*, pages 185–199, 2008.

[32] B. Prabhu, K. Srinathan, and C. Pandu Rangan. Trading players for efficiency in unconditional multiparty computation. In *SCN*, pages 342–353, 2002.

[33] T. Rabin. Robust sharing of secrets when the dealer is honest or cheating. *J. ACM*, 41(6):1089–1109, 1994.

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

[35] K. Srinathan and C. Pandu Rangan. Efficient asynchronous secure multiparty distributed computation. In *INDOCRYPT*, pages 117–129, 2000.

[36] A. C. Yao. Protocols for secure computations. In *Proc. of 23rd IEEE FOCS*, pages 160–164, 1982.

11 APPENDIX A: Protocol (t,2t)-Share-Multiple

Protocol (t,2t)-Share-Multiple($D, \mathcal{P}, S$)

D's Distribution: Code for D:

1. For $l = 1, \ldots, \ell$, corresponding to every s^l, select random polynomials $F^l(x)$ and $G^l(x)$ over $\mathbb{F}$ of degree t and $2t$ respectively, such that $G^l(0) = F^l(0) = s^l$.
2. For $l = 1, \ldots, \ell$, compute $(f^{(l,0)}(x), f^{(l,1)}(x), \ldots, f^{(l,n)}(x), g^{(l,1)}(x), \ldots, g^{(l,n)}(x)) = \mathsf{Generate}(G^l(x), F^l(x))$. Denote $\mathcal{F} = \cup_l \{f^{(l,0)}(x), f^{(l,1)}(x), \ldots, f^{(l,n)}(x), g^{(l,1)}(x), \ldots, g^{(l,n)}(x)\}$. We call the polynomials in $\mathcal{F}$ as *base polynomials*.
3. Select $(t+1)n$ degree-t random *blinding polynomials* over $\mathbb{F}$, denoted by $p^{(i,1)}(x), \ldots, p^{(i,t+1)}(x)$ for $i = 1, \ldots, n$.
4. For $i = 1, \ldots, n$, deliver the following to party P_i:
   (a) the i^{th} point on the polynomials in $\mathcal{F}$, namely $\Delta_i = \cup_l \{f^{(l,0)}(i), f^{(l,1)}(i), \ldots, f^{(l,n)}(i), g^{(l,1)}(i), \ldots, g^{(l,n)}(i))\}$ and
   (b) the i^{th} point on the polynomials $p^{(j,1)}(x), \ldots, p^{(j,t+1)}(x)$, namely $p^{(j,1)}(i), \ldots, p^{(j,t+1)}(i)$ for $j = 1, \ldots, n$.

Verification of D's Distribution: Code for P_i:

i. Code for P_i as a Party:

1. Wait to receive Δ_i and $p^{(j,1)}(i), \ldots, p^{(j,t+1)}(i)$ for $j = 1, \ldots, n$ from D.
2. For every $l = 1, \ldots, \ell$, check whether (a) $((0, f^{(l,0)}(i)), (1, f^{(l,1)}(i)), \ldots, (n, f^{(l,n)}(i)))$ defines a unique t degree polynomial and (b) $((0, f^{(l,0)}(i)), (1, g^{(l,1)}(i)), \ldots, (n, g^{(l,n)}(i)))$ defines a unique $2t$ degree polynomial. If yes then send a `Received-From-D` signal to every other party P_j.

ii. Code for P_i as a Verifier:

1. Wait to obtain `Received-From-D` signal from $3t + 1$ parties. Put the identities of the $3t + 1$ parties in a set $ReceivedSet^{(P_i,1)}$. Select a random $r^{(P_i,1)} \in_R \mathbb{E}$ and A-cast $(r^{(P_i,1)}, ReceivedSet^{(P_i,1)})$.
2. For j^{th} $(j > 1)$ receipt of `Received-From-D` signal from new party $P_\alpha \notin ReceivedSet^{(P_i,j-1)}$, construct $ReceivedSet^{(P_i,j)} = ReceivedSet^{(P_i,j-1)} \cup \{P_\alpha\}$, select a random $r^{(P_i,j)} \in_R \mathbb{E}$ and A-cast $(r^{(P_i,j)}, ReceivedSet^{(P_i,j)})$.

iii. Code for P_i as a Party:

1. If $(r^{(P_\alpha,\beta)}, ReceivedSet^{(P_\alpha,\beta)})$ is obtained from A-cast of some verifier $P_\alpha \in \mathcal{P}$, then do the following:
   (a) Check if $P_i \in ReceivedSet^{(P_\alpha,\beta)}$. If yes, then A-cast $v_i^{(P_\alpha,\beta)} = \mathsf{LinCombValue}(\{p^{(\alpha,\beta)}(i) \cup \Delta_i\}, R)$, where $R = (1, r^{(P_\alpha,\beta)}, \left(r^{(P_\alpha,\beta)}\right)^2, \ldots, \left(r^{(P_\alpha,\beta)}\right)^{2n\ell+\ell})$.
   (b) **If P_i is the dealer** D, then A-cast polynomial $F^{(\alpha,\beta)}(x) = \mathsf{LinCombPoly}(\{p^{(\alpha,\beta)}(x) \cup \mathcal{F}\}, R)$, where $R = (1, r^{(P_\alpha,\beta)}, \left(r^{(P_\alpha,\beta)}\right)^2, \ldots, \left(r^{(P_\alpha,\beta)}\right)^{2n\ell+\ell})$.

CORE Construction & Agreement on CORE: Code for P_i: Same as in (t,2t)-Share-Single.

For $j = 1, \ldots, n$, P_j-Private-Reconstruction of $f^{(1,j)}(0), \ldots, f^{(\ell,j)}(0)$ and $g^{(1,j)}(0), \ldots, g^{(\ell,j)}(0)$: Code for P_i:

1. If $P_i \in CORE$, then for $l = 1, \ldots, \ell$, participate in d-Rec-Private$(CORE, t, f^{(l,j)}(0), P_j)$ for P_j-Private-Reconstruction of $f^{(1,j)}(0), \ldots, f^{(\ell,j)}(0)$ for $j = 1, \ldots, n$.
2. If $P_i \in CORE$, then for $l = 1, \ldots, \ell$, participate in d-Rec-Private$(CORE, t, g^{(l,j)}(0), P_j)$ for P_j-Private-Reconstruction of $g^{(1,j)}(0), \ldots, g^{(\ell,j)}(0)$ for $j = 1, \ldots, n$.
3. As a receiver participate in d-Rec-Private$(CORE, t, f^{(l,i)}(0), P_i)$ for $l = 1, \ldots, \ell$ for P_i-Private-Reconstruction of $f^{(1,i)}(0), \ldots, f^{(\ell,i)}(0)$.
4. As a receiver participate in d-Rec-Private$(CORE, t, g^{(l,i)}(x), P_i)$ for $l = 1, \ldots, \ell$ for P_i-Private-Reconstruction of $g^{(1,i)}(0), \ldots, g^{(\ell,i)}(0)$.
5. For $l = 1, \ldots, \ell$, output $F^l(i) = f^{(l,i)}(0)$ and $G^l(i) = g^{(l,i)}(0)$ as the i^{th} share of secret s^l and terminate. For $l = 1, \ldots, \ell$, every s^l is now t-shared as well as $2t$ shared using polynomials $F^l(x)$ and $G^l(x)$, respectively.

Table 1: Generating $(t, 2t)$-sharing of secret $S = \{s^1, \ldots, s^\ell\}$ with $n = 4t + 1$