Unconditionally Secure Asynchronous Multiparty Computation with Quadratic Communication Per Multiplication

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 *d-sharing* of multiple secrets concurrently in asynchronous setting, which is of independent interest, where $t \leq d \leq 2t$. In the literature, though there are protocols for generating t-sharing and $2t$-sharing separately, there is no generic protocol for generating *d-sharing* for the range $t \leq d \leq 2t$. Moreover, our protocol provides better communication complexity than the existing methods for generating $2t$-sharing.

Keywords: Multiparty Computation, Byzantine Adversary, Asynchronous Networks.

1 Introduction

Secure Multiparty Computation (MPC): Secure multiparty computation (MPC) [38] 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 $f : \mathbb{F}^n \rightarrow \mathbb{F}^n$ and party P_i has input $x_i \in \mathbb{F}$, where $\mathbb{F}$ is a finite field. At the end of the computation of f, P_i gets $y_i \in \mathbb{F}$, 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. 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 [38, 21, 7, 13, 34, 1, 23, 20, 22, 25, 3, 17, 27, 5, 26] or asynchronous [6, 8, 37, 33, 10, 28, 4, 31], the adversary is threshold [38, 21, 7, 13, 34, 1, 20, 22, 3, 17, 5, 23] or non-threshold [23, 16, 28, 2, 24], the adversary behavior is static [38, 21, 7, 13, 34, 1, 23, 20] or mobile [30], the security is cryptographic [21, 25, 26] or information theoretic [38, 7, 13, 34], whether the protocol is perfect (i.e., without any error) [7, 22, 5] or allows a negligible error probability [35, 34, 14, 15, 17,

[*]Financial Support from Microsoft Research India Acknowledged
[†]Financial Support from Infosys Technology India Acknowledged
[‡]Work Supported by Project No. CSE/05-06/076/DITX/CPAN on Protocols for Secure Communication and Computation, Sponsored by Department of Information Technology, Govt. of India

3, 32] etc. In any general MPC protocol, the function f is specified by an arithmetic circuit over $\mathbb{F}$, consisting of input, linear (e.g. addition), multiplication, random and output gates. *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.*

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* (denoted as "CC in bits/ Multiplication Gate" in the table) of the best known AMPC protocols are given in the following table, where 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	CC in bits/ Multiplication Gate
[25]	Cryptographic	$t < n/3$ (optimal)	$\mathcal{O}(c_M n^3 \kappa)$
[26]	Cryptographic	$t < n/3$ (optimal)	$\mathcal{O}(c_M n^2 \kappa)$
[37]	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)	$\Omega(c_M n^{11} \kappa^4)$
[31]	IT (negligible error)	$t < n/3$ (optimal)	$\mathcal{O}(c_M n^5 \kappa)$
[33]	IT (negligible error)	$t < n/4$ (non-optimal)	$\mathcal{O}(c_M n^4 \kappa)$

Recently in [39], 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). In another work, Damgaard et. al [19] has reported efficient MPC protocol over a network that assumes the concept of synchronization point i.e the network is asynchronous before and after the synchronization point. We do not compare the protocols of [39] and [19] with our AMPC protocol, as our protocol 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 except with negligible probability of $2^{-\Omega(\kappa)}$, where κ is the error parameter, (b) **Correctness:** Every honest party receives the correct output at the end, except with negligible error probability of $2^{-\Omega(\kappa)}$, (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 $2^{-\Omega(\kappa)}$ in **correctness** and **termination**, where $\mathbb{F}$ is the field over which the computation is carried out. Our AMPC is to be compared with the best known AMPC with $n = 4t + 1$ that 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). Moreover, 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 *d-sharing* of multiple secrets concurrently in asynchronous settings, where $t \leq d \leq 2t$. In the literature, though there are protocols for generating t-sharing and $2t$-sharing separately, there is no generic protocol for generating *d-sharing* for the range $t \leq d \leq 2t$. Moreover, our protocol provides better communication complexity than the existing methods for generating $2t$-sharing.

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 $n = \text{poly}(\kappa)$.

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 $n \geq 3t+1$. A-Cast of b bits incurs a private communication of $\mathcal{O}(n^2 b)$ bits [9]. 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 $f(i) = s_i$ for every honest $P_i \in \overline{\mathcal{P}}$. The vector of shares is called a d-sharing of s and is denoted by $[s]_d$. A set of values (at least $d+1$) is called d-consistent if these values 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 with very high probability. 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 evaluating multiplication gate, we adapt a technique from [17] used in synchronous settings.

4 Reconstruction of d-Sharing

Let a secret $s \in \mathbb{F}$ 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 $P_\alpha \in \mathcal{P}$ be a *receiver*. We now give a protocol **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) technique [6] on the received s_i's to get the polynomial $f(x)$ and reconstructs $s = f(0)$. Roughly speaking, the online error correcting 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.

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

Code for P_i: Send s_i to P_α.

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

Theorem 1 ([11]) *For any secret s which is correctly d-shared among the parties in $\mathcal{P}$, with $d < n - 2t$, protocol 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.*

— Communication Complexity: *Protocol Rec-Private privately communicates $\mathcal{O}(n \log |\mathbb{F}|)$ bits.*

Theorem 2 *(i) Protocol Rec-Private can be used to reconstruct t-sharing as well as $2t$-sharing of s which is shared among the parties in $\mathcal{P}$, where $|\mathcal{P}| = 4t + 1$.*

(ii) If $\overline{\mathcal{P}}$ is any $3t + 1$ sized subset of $\mathcal{P}$, such that s is t-shared among the parties in $\overline{\mathcal{P}}$, then Rec-Private($\overline{\mathcal{P}}, t, s, P_\alpha$) ensures successful P_α-Private-Reconstruction of s.

(iii) Let s be d-shared among the parties in $\mathcal{P}$, except with probability $2^{-\Omega(\kappa)}$, where $d < n - 2t$ and $n = 4t + 1$. Then Rec-Private satisfies the Termination *and* Correctness *properties mentioned in Theorem 1, except with probability $2^{-\Omega(\kappa)}$.*

For the description of OEC and proof of Theorem 2, see **APPENDIX A**.

5 Generating d-Sharing with $d < n - 2t$

We now present a novel protocol, called d-Share-MS, that allows a dealer $D \in \mathcal{P}$ (dealer can be any party from $\mathcal{P}$) to concurrently d-share $\ell \geq 1$ secrets from $\mathbb{F}$, among the parties in $\mathcal{P}$, except with error probability of $2^{-\Omega(\kappa)}$, where $t \leq d \leq 2t$. Protocol d-Share-MS achieves the following properties:

1. Termination: (a) If D is honest, then every honest party will eventually terminate d-Share-MS. (b) If D is corrupted and some honest party has terminated d-Share-MS, then all the honest parties will eventually terminate d-Share-MS, except with probability $2^{-\Omega(\kappa)}$.

2. Correctness: (a) If D is *honest*, then all the ℓ secrets will be correctly d-shared among the honest parties in $\mathcal{P}$. (b) If D is *corrupted* and the honest parties in $\mathcal{P}$ terminate d-Share-MS, then there are ℓ secrets that are properly d-shared among the honest parties in $\mathcal{P}$, except with probability $2^{-\Omega(\kappa)}$.

3. Secrecy: If D is honest, then $\mathcal{A}_t$ obtains no information about the secrets of D.

Comparison with Existing Protocols for t-sharing and $2t$-sharing in Asynchronous Settings: In [4], the authors have presented a protocol for generating t-sharing of ℓ secrets concurrently (the protocol is a simple extension of Canetti's [11] protocol for generating t-sharing of a single secret). The protocol of [4] requires private communication of $\mathcal{O}(\ell n^2 \log |\mathbb{F}|)$ bits and A-cast of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits. But protocol of [4] is not extendible to generate d-sharing for $d > t$. In order to generate $2t$-sharing of a *single* secret, [4] have used another protocol, which first generates t-sharing of $2t+1$ random secrets and then combine them to generate $2t$-sharing. This incurs a private communication of $\mathcal{O}(n^3 \log |\mathbb{F}|)$ bits and A-cast of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits, for a *single* secret. Our protocol for generating d-sharing with $t \leq d \leq 2t$ is *generic* and can generate d-sharing of ℓ secrets concurrently, with a private communication of $\mathcal{O}((\ell n^2 + n^3) \log |\mathbb{F}|)$ bits and A-cast of $\mathcal{O}(n^3 \kappa)$ bits. If ℓ is significantly large, then instead of generating $2t$-sharing of the individual secrets by executing ℓ instances of the protocol of [4], we can concurrently generate the $2t$-sharing of all the secrets by executing a single instance of our protocol, which will result in less communication overhead. Specifically, if $\ell = \Omega(n)$, which is the case in our AMPC protocol, then our protocol gains a factor of $\Omega(n)$, in generating $2t$-sharing of ℓ secrets, in comparison to the protocol of [4] (see Remark 1 and Remark 2 in Section 6 and Section 7 respectively). For the sake of completeness, we give some details of the existing protocols for generating $2t$-sharing in **APPENDIX B**.

For the ease of understanding, we first present a protocol, called d-Share-SS, that allows D to d-share a single secret among the parties in $\mathcal{P}$ except with error probability of $2^{-\Omega(\kappa)}$. Later we present protocol d-Share-MS which is simple extension of d-Share-SS. Our discussion will clearly show, that executing a *single* instance of d-Share-MS dealing with multiple secrets *concurrently*, is advantageous over executing *multiple* instances of d-Share-Single, dealing with single secret, in terms communication complexity. Thus protocol d-Share-MS harnesses the advantages offered by dealing with multiple secrets *concurrently*. The sole purpose of presenting d-Share-SS is to simplify the overall presentation of d-Share-MS.

We divide the structure of d-Share-SS into three main phases:

1. **Distribution by D:** As the name suggests, in this phase, D on having a secret s, distributes information to the parties in $\mathcal{P}$ in order to generate d-sharing of s.

2. **Verification & Agreement on CORE:** Here the parties in $\mathcal{P}$ jointly perform some computation and communication in order to verify consistency of the information distributed by D in **Distribution by D** phase. In case of successful verification, all the honest parties agree on a set of at least $3t+1$ parties called $CORE$.

3. **Generation of d-sharing of Secret:** In this phase, *only* the parties in $CORE$ communicates to every party in $\mathcal{P}$ and every party performs local computation (on the data received from the parties in $CORE$) to finally generate the d-sharing of secret s.

We now focus on the details of each of these phases.

5.1 Distribution by D

In this phase, D on having a secret s, selects a random bivariate polynomial $F(x, y)$ of degree d in x and t in y, such that $F(0,0) = s$. Let $f_i(x) = F(x, i)$, $p_i(y) = F(i, y)$. While all $f_i(x)$ polynomials are of degree-d, all $p_i(y)$ polynomials are of degree-t. We will call the $f_i(x)$ polynomials as *row polynomials* and $p_i(y)$ polynomials as *column polynomials*. Now D sends $f_i(x)$ to party P_i. In this phase, D also distributes some more information which will be used to keep his secret secure during **Verification & Agreement on CORE** phase. Precisely, D distributes the shares of $(t+1)n$ random polynomials of degree-t which will be used for *blinding* purpose in **Verification & Agreement on CORE** phase. We refer these polynomials as *blinding polynomials*, so that its purpose and role is clear to the reader. The reason for taking these many blinding polynomials will be clear in the next section.

Protocol Distr-SS($D, \mathcal{P}, s, d$)

CODE FOR D:

1. Select a random bivariate polynomial $F(x,y)$ of degree d in x and degree t in y, such that $F(0,0) = s$. Let $f_i(x) = F(x,i)$, $p_i(y) = F(i,y)$ for $0 \leq i \leq n$.
2. Select $(t+1)n$ degree-t, random, distinct *blinding polynomials*, over $\mathbb{F}$, denoted by $b^{(P_i,1)}(y), \ldots, b^{(P_i,t+1)}(y)$ for $i = 1, \ldots, n$.
3. Send the following to party P_i: (i) $f_i(x)$; (ii) $b^{(P_j,1)}(i), \ldots, b^{(P_j,t+1)}(i)$ for $j = 1, \ldots, n$.

In the sequel, we describe **Verification & Agreement on CORE** phase. If **Verification & Agreement on CORE** phase is successful, then at the end of **Generation of d-sharing of Secret** phase, the secret s will be d-shared using polynomials $f_0(x)$.

5.2 Verification & Agreement on CORE

As it is clear from the description of **Distribution by D** phase, if D behaves honestly in protocol Distr-SS, then j^{th} points on *all row polynomial* $f_i(x)$, corresponding to honest P_i's (i.e $f_i(j)$'s), should define degree-t *column polynomial* $p_j(y)$. For an honest D, this is obviously true. But for a corrupted D, we must ensure the above condition by enforcing some verification mechanism. While it may be difficult to ensure that the j^{th} points on *all* honest P_i's row polynomial $f_i(x)$ define a degree-t polynomial $p_j(y)$, it is easier to ensure the same for the honest parties in a set of at least $3t+1$ parties, say $CORE$ ($CORE \subseteq \mathcal{P}$). In fact, this is what our protocol attempts to achieve. Specifically, the protocol tries to identify a set of parties, called $CORE$, having the following property:

Property 1 (Desired Properties of $CORE$) *$CORE \subseteq \mathcal{P}$, such that $|CORE| \geq 3t+1$. Moreover, for $j = 1, \ldots, n$, the j^{th} points on all honest P_i's row polynomial $f_i(x)$, where $P_i \in CORE$, should define a degree-t column polynomial $p_j(y)$.*

Hence, we do not care about $f_i(x)$ possessed by P_i, for $P_i \notin CORE$. The verification mechanism and the construction of $CORE$ is the crux of protocol d-Share-SS.

An Informal Description: In our verification mechanism, every party has dual responsibility: (a) it acts as a verifier to verify certain consistency of the information distributed by D to the parties; (b) it also co-operates as a party, with other verifiers, in order to make the verification mechanism initiated by them, finishes successfully. So, we first concentrate on the part of communications and computations that is to be carried out with respect to a single verifier, say V (here V can be any party from $\mathcal{P}$). The goal of this part of communications and computations is to decide on a set of at least $3t+1$ parties, say $AgreeSet^V$, such that if V is honest, then $AgreeSet^V$ should satisfy all the desirable properties of $CORE$. That is $AgreeSet^V$ could be a eligible candidate for $CORE$, when V is honest. To implement this, we use the following protocol. In the protocol, we use the following notation:

Notation 1 *Given ρ polynomials, $C = \{c_1(x), \ldots, c_\rho(x)\}$ and a vector $R = (\zeta_1, \ldots, \zeta_\rho)$ of length ρ, we define $c(x)$ as the polynomial obtained by the linear combination of the polynomials in C with respect to R. That is, $c(x) = \sum_{i=1}^{\rho} \zeta_i . c_i(x)$. We capture this by: $c(x) =$ LinCombPoly(C, R). Similarly, we define $c =$ LinCombValue(C, R), where $C = \{c_1, \ldots, c_\rho\}$ and c is the linear combination of C, with respect to R.*

Protocol Single-Verifier-SS$(V, \mathcal{P}, s, d)$

i. CODE FOR P_i:

1. Wait to receive $f_i(x)$ and $b^{(P_j,1)}(i), \ldots, b^{(P_j,t+1)}(i)$ for $j = 1, \ldots, n$ from D.
2. After receiving, check whether $f_i(x)$ is a degree-d polynomial. If yes, then send a `Received-From-D` signal to V.

ii. CODE FOR V:

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

iii. CODE FOR D:

1. If $(r^{(V,\beta)}, ReceivedSet^{(V,\beta)})$ is received from the A-cast of V, then A-cast the polynomial $E^{(V,\beta)}(y)$, where $E^{(V,\beta)}(y) = \mathsf{LinCombPoly}(\mathcal{E}, R)$. Here $\mathcal{E} = \{b^{(V,\beta)}(y), p_1(y), \ldots, p_n(y)\}$ and $R = (1, r^{(V,\beta)}, \left(r^{(V,\beta)}\right)^2, \ldots, \left(r^{(V,\beta)}\right)^n)$.

iv. CODE FOR P_i:

1. If $(r^{(V,\beta)}, ReceivedSet^{(V,\beta)})$ is received from the A-cast of V, then check if $P_i \in ReceivedSet^{(V,\beta)}$. If yes, then A-cast $e_i^{(V,\beta)} = \mathsf{LinCombValue}(\Delta_i, R)$, where $\Delta_i = \{b^{(V,\beta)}(i), f_i(1), \ldots, f_i(n)\}$ and $R = (1, r^{(V,\beta)}, \left(r^{(V,\beta)}\right)^2, \ldots, \left(r^{(V,\beta)}\right)^n)$.
2. Say that **party P_j agrees with D with respect to** $(r^{(V,\beta)}, ReceivedSet^{(V,\beta)})$ if the following holds:
   (a) $E^{(V,\beta)}(y)$ is t degree polynomial, (b) $P_j \in ReceivedSet^{(V,\beta)}$ and (c) $e_j^{(V,\beta)} = E^{(V,\beta)}(j)$
   where $e_j^{(V,\beta)}$, $E^{(V,\beta)}(y)$ and $(r^{(V,\beta)}, ReceivedSet^{(V,\beta)})$ are obtained from the A-casts of P_j, D and V respectively.
3. With respect to $(r^{(V,\beta)}, ReceivedSet^{(V,\beta)})$, when there are $3t+1$ P_j's who agree with D, add all of them in a set $AgreeSet^{(V,\beta)}$.

We now prove the following lemmas for Protocol Single-Verifier-SS.

Lemma 1 *In protocol Single-Verifier-SS, if V is honest, then the points on the row polynomials, held by the honest parties in $AgreeSet^{(V,\beta)}$ (with $|AgreeSet^{(V,\beta)}| \geq 3t+1$), define n degree-t column polynomials, with very high probability. Moreover, the points on blinding polynomial $b^{(V,\beta)}(y)$ held by the honest parties in $AgreeSet^{(V,\beta)}$ will also lie on a t degree polynomial with very high probability.*

PROOF: The lemma is trivially true when D is honest. Hence we consider the case when D is corrupted. Let $H^{(V,\beta)}$ denote the set of honest parties in $AgreeSet^{(V,\beta)}$. First of all, since V is honest, he A-casts random $r^{(V,\beta)}$ only after listening `Received-From-D` signal from all the parties in $ReceivedSet^{(V,\beta)}$. Thus D has no knowledge of $r^{(V,\beta)}$, when he distributes the row polynomials and points on blinding polynomial $b^{(V,\beta)}(y)$ to the (honest) parties in $ReceivedSet^{(V,\beta)}$. Let $\overline{b^{(V,\beta)}(y)}$ denote the *minimum degree* polynomial, defined by the points on $b^{(V,\beta)}(y)$, held by the honest parties in $H^{(V,\beta)}$. Similarly, let $\overline{p_1(y)}, \ldots, \overline{p_n(y)}$ denote the *minimum degree* polynomials, defined by the row polynomials held by the parties in $H^{(V,\beta)}$. For convenience, we use an uniform notation for these $n+1$ polynomials. We denote them by $h^0(y), \ldots, h^n(y)$, respectively. Then the value $e_i^{(V,\beta)}$ A-casted by P_i is defined as $e_i^{(V,\beta)} = \sum_{j=0}^{n} \left(r^{(V,\beta)}\right)^j h^j(i)$.

We now claim that with very high probability, $h^0(y), \ldots, h^n(y)$ have degree-t. On the contrary, if we assume that at least one of the polynomials has degree more than t, then we can show that the minimum degree polynomial, say $h^{min}(y)$, defined by $e_i^{(V,\beta)}$'s for $P_i \in H^{(V,\beta)}$ will be of degree more than t, with very high probability. This will clearly imply $E^{(V,\beta)}(y) \neq h^{min}(y)$ and hence $e_i^{(V,\beta)} \neq E^{(V,\beta)}(i)$ for at least one $P_i \in H^{(V,\beta)}$. This is a contradiction as $e_i^{(V,\beta)} = E^{(V,\beta)}(i)$ holds for every $P_i \in Agree^{(V,\beta)}$ and $H^{(V,\beta)} \subseteq Agree^{(V,\beta)}$. This shows that our claim is true.

So we proceed to prove that $h^{min}(y)$ will be of degree more than t with very high probability, when one of $h^0(y), \ldots, h^n(y)$ has degree more than t. For this, we show that the following two holds:

1. We first show that $h^{def}(y) = \Sigma_{j=0}^{n} \left(r^{(V,\beta)}\right)^j h^j(y)$ will of degree more than t with very high probability, if one of $h^0(y), \ldots, h^n(y)$ has degree more than t.
2. We then show that $h^{min}(y) = h^{def}(y)$, implying that $h^{min}(y)$ will be of degree more than t with very high probability

The first point is easy to prove. If at least one of $h^0(y), \ldots, h^n(y)$, has degree more than t, then the linear combination of these polynomials, namely $h^{def}(y)$, can be written as $h^{def}(y) = h_1^{def}(y) + h_2^{def}(y)$, where

$h_1^{def}(y)$ contains all the coefficients of $h^{def}(y)$ having degree more than t, while $h_2^{def}(y)$ contains all the remaining coefficients of $h^{def}(y)$. Now $h^{def}(y)$ will be of degree t, if $h_1^{def}(y) = 0$, which can happen for at most n possible values of $r^{(V,\beta)}$. Since $r^{(V,\beta)}$ is selected randomly from $\mathbb{E}$, independent of $h^0(y), \ldots, h^n(y)$, the probability that $h_1^{def}(y) = 0$ is at most $\frac{n}{|\mathbb{E}|} \approx 2^{-\Omega(\kappa)}$.

For the second point, consider the difference polynomial $dp(y) = h^{def}(y) - h^{min}(y)$. Clearly, $dp(y) = 0$, for all $y = i$, where $P_i \in H^{(V,\beta)}$. Thus $dp(y)$ will have at least $|H^{(V,\beta)}|$ roots. On the other hand, maximum degree of $dp(y)$ could be $|H^{(V,\beta)}| - 1$. These two facts together imply that $dp(y)$ is the zero polynomial, implying that $h^{def}(y) = h^{min}(y)$. The complete formal proof of the lemma is given in **APPENDIX C**. □

Lemma 2 *In Single-Verifier-SS, V will A-Cast $(r^{V,\beta}, ReceivedSet^{(V,\beta)})$ at most $t+1$ times. Moreover, each $ReceivedSet^{(V,\beta)}$ will be unique. Furthermore, each $r^{V,\beta}$ will be unique with very high probability.*

Proof: First note that $ReceivedSet^{(V,1)} \geq 3t+1$. In the worst case $ReceivedSet^{(V,1)}$ may be exactly equal to $3t+1$ and potentially t honest parties may not be present in it. But for every β^{th} $(\beta > 1)$ receipt of `Received-From-D` signal, from a new party $P_\alpha \notin ReceivedSet^{(V,\beta-1)}$, V will construct $ReceivedSet^{(V,\beta)} = ReceivedSet^{(V,\beta-1)} \cup \{P_\alpha\}$. Since $n = 4t+1$, V will A-Cast $(r^{V,\beta}, ReceivedSet^{(V,\beta)})$ at most $t+1$ times. The uniqueness of $ReceivedSet^{(V,\beta)}$'s follows from the fact that $ReceivedSet^{(V,\beta-1)} \subset ReceivedSet^{(V,\beta)}$. Also, since each $r^{V,\beta}$ is randomly selected from $\mathbb{E}$, they are unique with very high probability. □

Lemma 3 *In protocol Single-Verifier-SS, if both V and D are honest, then for some β $(1 \leq \beta \leq t+1)$ $AgreeSet^{(V,\beta)}$ with $|AgreeSet^{(V,\beta)}| \geq 3t+1$ will be generated.*

Proof: When D is honest, every honest party in $\mathcal{P}$ will eventually send `Received-From-D` signal to V. Moreover, from previous lemma, V will A-Cast $ReceivedSet^{(V,\beta)}$ at most $t+1$ times. These two facts implies that eventually, there will exist a $\beta \in \{1, \ldots, t+1\}$, such that $ReceivedSet^{(V,\beta)}$ is bound to contain all the $3t+1$ honest parties. Moreover, each honest party from such $ReceivedSet^{(V,\beta)}$ will eventually enter into $AgreeSet^{(V,\beta)}$. □

Lemma 4 *In protocol Single-Verifier-SS, if D is honest, then $\mathcal{A}_t$ will have no information about s.*

Proof: Without loss of generality, let $\mathcal{A}_t$ controls $P_1, \ldots, P_t$. So $\mathcal{A}_t$ will know $f_1(x), \ldots, f_t(x)$. Since $F(x,y)$ is of degree d and t in x and y respectively, its constant term $F(0,0)$ will remain information theoretically secure. $\mathcal{A}_t$ may learn $E^{(V,\beta)}(y)$ for $\beta = 1, \ldots, t+1$. But each $E^{(V,\beta)}(y)$ is the linear combination of polynomials $b^{(V,\beta)}(y), p_1(y), \ldots, p_n(y)$. As $b^{(V,\beta)}(y)$ is completely random and independent of $p_1(y), \ldots, p_n(y)$, $E^{(V,\beta)}(y)$ will be completely random for $\mathcal{A}_t$. Moreover for every $\beta \in \{1, \ldots, t+1\}$, distinct $b^{(V,\beta)}(y)$ is used. Hence $\mathcal{A}_t$ obtains no information about s in Single-Verifier-SS. □

Need for $n(t+1)$ Blinding Polynomials: In protocol Distr-SS, D selects $n(t+1)$ *blinding polynomials*. This is because in **Verification and Agreement on Core Phase**, each of the n parties will act as a verifier and execute protocol Single-Verifier-SS. Moreover, from Lemma 2, each verifier will A-Cast at most $t+1$ $(r^{V,\beta}, ReceivedSet^{(V,\beta)})$'s. Furthermore, from Lemma 4, in order to main secrecy of s with respect to each $(r^{V,\beta}, ReceivedSet^{(V,\beta)})$, an honest D requires one blinding polynomial.

So far, we have concentrated on the part of communications and computations that has to be carried out with respect to a single verifier V. We also proved that if V is honest then Single-Verifier-SS can provide us with a candidate solution for $CORE$. But as we do not have the knowledge of the exact identities of the honest parties in $\mathcal{P}$, we can not pick a $AgreeSet^{(V,*)}$ for any particular V as $CORE$. Thus we need a special trick to construct $CORE$. Informally, we have to execute protocol Single-Verifier-SS for every verifier V and finally compute $CORE$ based upon all $AgreeSet^{(*,*)}$'s. Before presenting our protocol for **Verification & Agreement on CORE** phase, we prove the following lemmas which will help to grasp the part of code used for constructing $CORE$.

Lemma 5 *For an* **honest** *verifier V, the row polynomials held by honest parties in any two sets $AgreeSet^{(V,\beta)}$ and $AgreeSet^{(V,\gamma)}$ with $\beta \neq \gamma$, define the same set of degree-t column polynomials, namely $p_1(y), \ldots, p_n(y)$, with very high probability.*

Proof: By Lemma 1, for an honest V, the row polynomials held by the honest parties in $AgreeSet^{(V,\beta)}$, as well as in $AgreeSet^{(V,\gamma)}$, define degree-t column polynomials with very high probability. Let $\overline{p_1(y)}, \ldots, \overline{p_n(y)}$

and $\widehat{p_1(y)}, \ldots, \widehat{p_n(y)}$ be the degree-t column polynomials, defined by the row polynomials, held by the honest parties in $AgreeSet^{(V,\beta)}$ and $AgreeSet^{(V,\gamma)}$ respectively. We claim that these two sets of polynomials are identical. Since $AgreeSet^{(V,\beta)}$ and $AgreeSet^{(V,\gamma)}$ 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 6 *For any two* **honest** *verifiers V_α and V_δ, with $\alpha \neq \delta$, the column polynomials, defined by the row polynomials, held by honest parties in any two sets $AgreeSet^{(V_\alpha,\beta)}$ and $AgreeSet^{(V_\delta,\gamma)}$ are same, with very high probability.*

Proof: Follows using similar argument as in Lemma 5. □

The protocol for **Verification & Agreement on CORE** phase is given in Table 1.

Lemma 7 *If D is honest, then the row polynomials held by honest parties in $CORE$ define degree-t column polynomials. If D is corrupted, then the row polynomials held by honest parties in $CORE$ define degree-t column polynomials with very high probability. Moreover there can not be another set $\overline{CORE}$ containing $3t+1$ parties such that the row polynomials held by honest parties in $\overline{CORE}$ define a deferent set of degree-t column polynomials.*

Proof: If D is honest then the lemma is trivially true. We now prove the lemma for the case of a corrupted D. By the construction of $CORE$, every party in $CORE$ is guaranteed to be present in $AgreeSet$ of at least one honest verifier. By Lemma 5, corresponding to an honest verifier P_α, the row polynomials held by the honest parties in $AgreeSet^{P_\alpha}$ define t-degree column polynomials, say $p_1(y), \ldots, p_n(y)$. Moreover, by Lemma 6, the row polynomials held by the honest parties in the union of $AgreeSet^{P_\alpha}$'s, corresponding to all *honest* P_α's, also define $p_1(y), \ldots, p_n(y)$ with very high probability. This implies that the values held by the honest parties in $CORE$, define $p_1(y), \ldots, p_n(y)$.

Now we prove the second part of the lemma. Assume that there is another set, $\overline{CORE}$ containing $3t+1$ parties such that the row polynomials held by honest parties in $\overline{CORE}$ define a deferent set of degree-t column polynomials, say $\overline{p_1(y)}, \ldots, \overline{p_n(y)}$. Now since both $CORE$ and $\overline{CORE}$ are of size at least $3t+1$, they have $2t+1$ parties in common of which $t+1$ are honest. This implies that for $i = 1, \ldots, n$ polynomial $p_i(y)$ and $\overline{p_i(y)}$ has $t+1$ points in common. As both $p_i(y)$ and $\overline{p_i(y)}$ are of degree t, the above fact concludes that $p_i(y) = \overline{p_i(y)}$. Hence $p_i(y) = \overline{p_i(y)}$ for $i = 1, \ldots, n$. Hence the lemma. □

Protocol Ver-Agree-on-CORE-SS($D, \mathcal{P}, s, d$)

<u>Verification and $CORE$ Construction:</u>

i. Code for P_i:

1. Start executing Protocol Single-Verifier-SS($P_\alpha, \mathcal{P}, s, d$) for every verifier $P_\alpha \in \mathcal{P}$ parallely.
2. Add a verifier P_α to a set $ValidVerifier$ if at least one $AgreeSet^{(P_\alpha,\beta)}$ has been generated.

ii. Code for D:

1. 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.

<u>Agreement on CORE:</u> Code for P_i:

1. Wait to receive $CORE$ from the A-cast of D. Upon receiving, locally perform the same steps as specified for D 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 terminate.

Table 1: Protocol for Verification & Agreement on CORE phase

Once the parties agrees on a $CORE$ set, generation of d-sharing requires n private reconstructions. The formal protocol is given in the next section.

5.3 Generation of d-sharing of the Secret

Assuming that the honest parties in $\mathcal{P}$ have agreed upon a $CORE$, our protocol achieves d-sharing of s in the following way: Since $|CORE| \geq 3t+1$ and each $p_i(y)$ is t-shared among the parties in $CORE$, from the property (ii) of Theorem 2, each $p_i(0)$ can be P_i-Private-Reconstructed for $i = 1, \ldots, n$. Every P_i can then output $p_i(0)$ as the shares of D's secret s and this will complete the d-sharing. As $f_0(i) = p_i(0)$, D's secret s (which is equal to $F(0,0)$) is d-shared using degree-d polynomial $f_0(x) = F(x, 0)$.

Protocol Gen-d-Share-SS($D, \mathcal{P}, s, d$)

<u>For $j = 1, \ldots, n$, P_j-Private-Reconstruction of $p_j(0)$:</u> Code for P_i:

1. If $P_i \in CORE$, participate in Rec-Private($CORE, t, p_j(0), P_j$) for P_j-Private-Reconstruction of $p_j(0)$, for $j = 1, \ldots, n$.
2. As a receiver, participate in Rec-Private($CORE, t, p_i(0), P_i$) for P_i-Private-Reconstruction of $p_i(0)$.
3. Output $f_0(i) = p_i(0)$ as the i^{th} share of secret s and terminate. s is now d-shared using polynomial $f_0(x)$.

5.4 Final Protocol for Generating d-sharing: Protocol d-Share-SS

We now present protocol d-Share-SS.

Protocol d-Share-SS($D, \mathcal{P}, s, d$)

i. Code for D:

1. Execute Distr-SS($D, \mathcal{P}, s, d$).

ii. Code for P_i:

1. Participate in Ver-Agree-on-CORE-SS($D, \mathcal{P}, s, d$).
2. Upon termination of Ver-Agree-on-CORE-SS($D, \mathcal{P}, s, d$), participate in Gen-d-Share-SS($D, \mathcal{P}, s, d$).

Lemma 8 *In protocol d-Share-SS, if D is honest, then every honest party will eventually terminate Ver-Agree-on-CORE-SS. If D is corrupted and some honest party has terminated Ver-Agree-on-CORE-SS, then all the honest parties will eventually terminate Ver-Agree-on-CORE-SS.*

Proof: 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 Ver-Agree-on-CORE-SS. This proves first part of the lemma.

If some honest party P_i has terminated Ver-Agree-on-CORE-SS, then he must have checked the validity of $CORE$ received from the A-cast of D. In the same way, every other honest P_j will check the validity of $CORE$ and terminate Ver-Agree-on-CORE-SS. This proves second part of the lemma. $\square$

Lemma 9 *In protocol d-Share-SS, if D is honest, then every honest party will eventually terminate Gen-d-Share-SS.*

Proof: If D is honest, then from previous lemma, each honest party will terminate Ver-Agree-on-CORE-SS. Moreover, in this case, each honest party in $CORE$ will hold correct points on degree-t column polynomials. The rest follows from property (ii) of Theorem 2. $\square$

Lemma 10 *In protocol d-Share-SS, if D is corrupted and some honest party terminates Ver-Agree-on-CORE-SS, then every honest party will eventually terminate Gen-d-Share-SS, with very high probability.*

Proof: From Lemma 8, if D is corrupted and some honest party terminates Ver-Agree-on-CORE-SS, then all the honest parties will eventually terminate Ver-Agree-on-CORE-SS and agree on $CORE$ of size $3t+1$. The rest follows from property (iii) of Theorem 2 and the fact that, in this case, with very high probability, the honest parties in $CORE$ hold correct points on degree-t column polynomials. $\square$

Theorem 3 *The protocol d-Share-SS achieves the following properties:*

–Termination: *(a) If D is honest, then every honest party will eventually terminate d-Share-SS. (b) If D is corrupted and some honest party has terminated d-Share-SS, then all the honest parties will eventually terminate d-Share-SS with very high probability.*

–Correctness: *(a) If D is honest, then the secret s will be correctly d-shared among the honest parties* *in $\mathcal{P}$. (b) If D is corrupted and the honest parties in $\mathcal{P}$ terminate d-Share-SS, then there is a secret s that is properly d-shared among the honest parties in $\mathcal{P}$, except with probability $2^{-\Omega(\kappa)}$.*

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

–COMMUNICATION COMPLEXITY: *The protocol privately communicates $\mathcal{O}(n^3 \log(|\mathbb{F}|))$ bits and A-Cast $\mathcal{O}(n^3\kappa)$ bits, where $\kappa = \log(|\mathbb{E}|)$.*

PROOF: **Termination:** The proof follows from Lemma 9 and Lemma 10.

Correctness: Part (a) follows from Lemma 8 and Lemma 9. For part(b), if D is corrupted and the honest parties in $\mathcal{P}$ terminates d-Share-SS, then from Lemma 7, the row polynomials held by honest parties in $CORE$ define degree-t column polynomials, say $p_1(y), \ldots, p_n(y)$ with very high probability. Now by Theorem 2-(ii), P_i-Private-Reconstruction of $p_i(0)$ is possible for all $i = 1, \ldots, n$ with very high probability. This implies every $P_i \in \mathcal{P}$ will compute $f_0(i) = p_i(0)$. So s will be d-shared among the parties in $\mathcal{P}$ using $f_0(x)$ with very high probability.

Secrecy: We have to consider the case when D is honest. By Lemma 4, the polynomials $E^{(P_\alpha,\beta)}$ A-casted in Ver-Agree-on-CORE are completely random to $\mathcal{A}_t$ and hence can be ignored. Without loss of generality, let $\mathcal{A}_t$ controls $P_1, \ldots, P_t$. So at the end of d-Share-SS, $\mathcal{A}_t$ will know $f_1(x), \ldots, f_t(x), p_1(y), \ldots, p_t(y)$. The bivariate polynomial $F(x, y)$ can be interpolated using $t+1$ $f_i(x)$'s or $d+1$ $p_i(y)$'s. $\mathcal{A}_t$ knows $f_1(x), \ldots, f_t(x)$ and t more points on each of $f_i(x)$ for $t+1 \leq i \leq n$ from $p_1(y), \ldots, p_t(y)$. But the t points on $f_{t+2}(x), \ldots, f_n(x)$ are linearly dependent on the t points on $f_{t+1}(x)$. As $f_{t+1}(x)$ is a degree-d polynomial, $\mathcal{A}_t$ requires $d+1-t$ more points to completely interpolate $f_{t+1}(x)$. Thus $d+1-t$ coefficients of $F(x, y)$ and hence $f_0(x)$ remains secure. Therefore $F(0, 0) = f_0(0)$ is secure.

Communication Complexity: During **Distribution by D** phase, D privately communicates $\mathcal{O}((nd + n^3)\log(\mathbb{F}))$ bits. Since $t \leq d < n - 2t$, in our context $d = \mathcal{O}(n)$. During **Verification & Agreement on CORE** phase, the parties perform A-Cast of $\mathcal{O}(n^3 \log(|\mathbb{F}|) + n^2\kappa)$ bits. During **Generation of d-Sharing of Secret** phase, the parties perform private communication of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits. As $\log(|\mathbb{F}|) \leq \kappa$, overall the protocol involves a private communication of $\mathcal{O}(n^3 \log(\mathbb{F})$ bits and A-Cast of $\mathcal{O}(n^3\kappa)$ bits. □

We now present protocol d-Share-MS, that allows a dealer $D \in \mathcal{P}$ to concurrently generate d-sharing of $\ell \geq 1$ secrets from $\mathbb{F}$, denoted as $S = (s^1, \ldots, s^\ell)$. We need to extend protocol Distr-SS, Single-Verifier-SS, Ver-Agree-on-CORE-SS, and Gen-d-Share-SS for ℓ secrets. We refer them as Distr-MS, Single-Verifier-MS, Ver-Agree-on-CORE-MS and Gen-d-Share-MS. We present the protocols in **APPENDIX C**. The proofs for the properties of the protocols dealing with multiple secrets will be similar to the proofs of the protocols dealing with single secret. We now have the following theorem:

Theorem 4 *Protocol d-Share-MS satisfies the* TERMINATION, CORRECTNESS *and* SECRECY *property of Theorem 3. The protocol privately communicates $\mathcal{O}((\ell n^2 + n^3)\log |\mathbb{F}|)$ bits and A-cast $\mathcal{O}(n^3\kappa)$ bits where $\kappa = \log |\mathbb{E}|$.*

PROOF: The proof of the above theorem is given in **APPENDIX C**. □

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

We now present a novel protocol, called (t,2t)-Share-MS that allows a dealer $D \in \mathcal{P}$ (dealer can be any party from $\mathcal{P}$) to concurrently generate $(t, 2t)$-sharing of $\ell \geq 1$ secrets from $\mathbb{F}$. Protocol (t,2t)-Share-MS achieves the following properties:

1. TERMINATION: (a) If D is *honest*, then every honest party will eventually terminate (t,2t)-Share-MS. (b) If D is *corrupted* and some honest party terminates (t,2t)-Share-MS, then all the honest parties will also terminate the protocol, except with probability $2^{-\Omega(\kappa)}$.

2. CORRECTNESS: (a) If D is *honest*, then all the ℓ secrets are correctly $(t, 2t)$-shared among the parties in $\mathcal{P}$. (b) If the honest parties terminate (t,2t)-Share-MS, then there are ℓ secrets, that are correctly $(t, 2t)$-shared among the parties in $\mathcal{P}$, except with probability $2^{-\Omega(\kappa)}$.

3. SECRECY: $\mathcal{A}_t$ will have no information about the secrets of an honest D.

The idea of (t,2t)-Share-MS is as follows: D, on having ℓ secrets $S = (s^1, \ldots, s^\ell)$, invokes two instances of d-Share-MS, to t-share and $2t$-share S respectively. For an honest D, this is enough to generate $(t, 2t)$-sharing of S. But a corrupted D may t-share $\widehat{S} = (\widehat{s^1}, \ldots, \widehat{s^\ell})$ and $2t$-share $\overline{S} = (\overline{s^1}, \ldots, \overline{s^\ell})$ where $\widehat{S} \neq \overline{S}$. To verify whether $\widehat{S} = \overline{S}$, the values $d^l = \widehat{s^l} - \overline{s^l}$, for $l = 1, \ldots, \ell$, are P_i-Private-Reconstructed for every P_i. Every party P_i can then check whether $d^l = 0$ for $l = 1, \ldots, \ell$ and in case of all the d^l values are 0, every P_i knows that S is correctly $(t, 2t)$-shared.

Protocol (t,2t)-Share-MS($D, \mathcal{P}, S = (s^1, \ldots, s^\ell)$)

CODE FOR D:

1. Invoke d-Share-MS($D, \mathcal{P}, \widehat{S} = (\widehat{s^1}, \ldots, \widehat{s^\ell}), t$) and d-Share-MS($D, \mathcal{P}, \overline{S} = (\overline{s^1}, \ldots, \overline{s^\ell}), 2t$), such that $\widehat{S} = \overline{S}$.

CODE FOR P_i:

1. Participate in d-Share-MS($D, \mathcal{P}, \widehat{S}, t$) and d-Share-MS($D, \mathcal{P}, \overline{S}, 2t$).
2. Wait until both d-Share-MS($D, \mathcal{P}, \widehat{S}, t$) and d-Share-MS($D, \mathcal{P}, \overline{S}, 2t$) terminate. Participate in d-Rec-Private($\mathcal{P}, 2t, \widehat{s^l} - \overline{s^l}, P_j$) for P_j-Private-Reconstruction of $\widehat{s^l} - \overline{s^l}$ for $j = 1, \ldots, n$ and $l = 1, \ldots, \ell$.
3. As a receiver participate in d-Rec-Private($\mathcal{P}, 2t, \widehat{s^l} - \overline{s^l}, P_i$) for P_i-Private-Reconstruction of $\widehat{s^l} - \overline{s^l}$, for $l = 1, \ldots, \ell$.
4. If $\widehat{s^l} - \overline{s^l} = 0$ for all $l = 1, \ldots, \ell$, then output φ_i^l and χ_i^l, where φ_i^l and χ_i^l are the i^{th} shares of secret s^l, obtained during d-Share-MS($D, \mathcal{P}, \widehat{S}, t$) and d-Share-MS($D, \mathcal{P}, \overline{S}, 2t$) respectively and terminate.

Theorem 5 *Protocol (t,2t)-Share-MS satisfies* TERMINATION, CORRECTNESS *and* SECRECY *as mentioned above. (t,2t)-Share-MS privately communicates $\mathcal{O}((\ell n^2 + n^3) \log |\mathbb{F}|)$ bits and A-cast $\mathcal{O}(n^3 \kappa)$ bits.*

PROOF: The TERMINATION, CORRECTNESS and communication complexity follows from Theorem 4. We now prove the secrecy property. We show that the secret s^1 is secure. The same argument will hold for $s^2, \ldots, s^\ell$. Let s^1 be t-shared and $2t$-shared, using polynomial $f_0^1(x)$ and $F_0^1(x)$ of degree t and $2t$ respectively, by protocol d-Share-MS. Hence, from Theorem 4 and Theorem 3, $\mathcal{A}_t$ will have no information about *one* and $t+1$ coefficients of $f_0^1(x)$ and $F_0^1(x)$ respectively. In protocol (t,2t)-Share-MS, $F_0^1(x) - f_0^1(x)$ is privately reconstructed towards each party. Thus, $\mathcal{A}_t$ will also know $F^{diff}(x) = F_0^1(x) - f_0^1(x)$. It is easy to see that the higher order t coefficients of $F^{diff}(x)$ is same as the higher order t coefficients of $F_0^1(x)$, which will be now known to $\mathcal{A}_t$. In addition, the lower order $t + 1$ coefficients of $F^{diff}(x)$ are constituted by the difference of the corresponding lower order $t + 1$ coefficients of $F_0^1(x)$ and $f_0^1(x)$. These difference of lower order $t + 1$ coefficients of $F_0^1(x)$ and $f_0^1(x)$ will also be now known to $\mathcal{A}_t$. Moreover, from protocol d-Share-MS, $\mathcal{A}_t$ knows t distinct points on $F_0^1(x)$ and $f_0^1(x)$. These are the total information, which will be available to $\mathcal{A}_t$ during (t,2t)-Share-MS, about $f_0^1(x)$ and $F_0^1(x)$. It is easy to see that, inspite of all these information, $\mathcal{A}_t$ will have no information about $f_0^1(0) = F_0^1(0) = s^1$. □

Remark 1 (Advantage of Concurrently Sharing Multiple Secrets) *Notice that in the protocol, D could have individually t-shared and $2t$-shared each s^i, for $i = 1, \ldots, \ell$, by executing 2ℓ concurrent executions of d-Share-SS. This would incur a private communication of $\mathcal{O}(\ell n^3 \log(|\mathbb{F}|))$ bits and A-Cast of $\mathcal{O}(\ell n^3 \kappa)$ bits. Comparing this with Theorem 5, we find that concurrently sharing ℓ secrets using* **two** *instances of d-Share-MS, instead of* **individually** *sharing ℓ secrets using 2ℓ instances of d-Share-SS, offers the following two advantages: (a) We get an A-Cast communication, which is independent of ℓ; (b) If $\ell = \Omega(n)$, then the protocol involves less private communication. The second point will me more clear in the next section (see Remark 2).*

7 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. 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 its ability to extract randomness which has been exploited in [36, 17, 4].

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-MS($P_i, \mathcal{P}, S^i$)** 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-MS($P_j, \mathcal{P}, S^j$)**.

AGREEMENT ON A CORE-SET: CODE FOR P_i

1. Create an accumulative set $C^i = \emptyset$. Upon terminating **t-2t-Share-MS($P_j, \mathcal{P}, S^j$)**, 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.

Vandermonde Matrix and Randomness Extraction [17]: Let $\beta_1, \ldots, \beta_c$ be distinct 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 11 *Each honest party will eventually terminate PreparationPhase, except with probability $2^{-\Omega(\kappa)}$. 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 $O(((c_M + c_R)n^2 + n^4) \log |\mathbb{F}|)$ 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.*

PROOF: The termination property follows from the termination property of protocol **t-2t-Share-MS**. We now prove the communication complexity. In the protocol, each party executes an instance of **t-2t-Share-MS**, by acting as a dealer, to $(t, 2t)$ share $L = \frac{c_M + c_R}{n-2t}$ secrets. So substituting $\ell = L$ in Theorem 5, the total private communication of the protocol is $\mathcal{O}\left(\left(Ln^3 + n^4\right) \log(|\mathbb{F}|)\right)$ bits. Since $n - 2t = \Theta(n)$, the total private communication of the protocol will be $O(((c_M + c_R)n^2 + n^4) \log |\mathbb{F}|)$ bits. Moreover, the protocol will A-Cast $\mathcal{O}(n^4\kappa)$ bits. □

Remark 2 (Advantage of Concurrently Sharing Multiple Secrets) *In protocol PreparationPhase, each party could have individually t-shared and $2t$-shared each of the L secrets, by executing $2L$ instances of d-Share-SS. This would individually generate the $(t, 2t)$-sharing of each of the L secrets. However, as explained in Remark 1, this will result in a overall private communication of $\mathcal{O}((c_M + c_R)n^3 \log(|\mathbb{F}|))$ bits and A-Cast of $\mathcal{O}((c_M + c_R)n^3\kappa)$ bits in protocol PreparationPhase. This clearly shows that concurrently sharing $\ell > 1$ secrets, offers significant advantage, in comparison to individually sharing each of the ℓ secrets, in terms of communication complexity.*

8 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-share his input X_i by acting as a dealer and executing d-Share-MS. 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 d-Share-MS. In order to agree on a core set of parties whose instance of d-Share-MS have terminated and whose inputs will be taken into consideration for computation (of the circuit), one instance of ACS is invoked. At the end, everyone considers the t-sharing of all the inputs shared by parties, only in the core set.

Protocol InputPhase($\mathcal{P}$)

<u>Secret Sharing:</u> Code for P_i

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

<u>Agreement on a Core-Set:</u> Code for P_i

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

<u>Output Generation:</u> Code for P_i:

1. Waits until ACS completes with output C containing $n-t$ parties. Output the the shares corresponding to t-sharing of the inputs of the parties in C and terminate.

Lemma 12 *Each honest party will eventually terminate InputPhase and will correctly output t-sharing of the inputs of the parties in core set C with high probability. The protocol privately communicates $O((c_I n^2 + n^4) \log |\mathbb{F}|)$ 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.*

9 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 Rec-Private($\mathcal{P}, 2t, C, P_j$) for P_j-Private-Reconstruction of C for all $j = 1, \ldots, n$.
   (d) Participate in Rec-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 d-Rec-Private($\mathcal{P}, t, x, P_i$) for every $P_j \in \mathcal{P}$. Participate in Rec-Private($\mathcal{P}, t, x, P_i$) as a receiver to reconstruct x. Output x.

Lemma 13 *Each honest party will eventually terminate ComputationPhase with very high probability. 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 with probability $2^{-\Omega(\kappa)}$*

10 The AMPC Protocol

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

Theorem 6 *For every coalition of up to $t < n/4$ bad players, the protocol AMPC securely computes the circuit representing function f. Except with probability $2^{-\Omega(\kappa)}$, each honest party will terminate the protocol. The protocol privately communicates $\mathcal{O}(((c_I + c_M + c_R + c_O)n^2 + n^4) \log |\mathbb{F}|)$ 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 $2^{-\Omega(\kappa)}$.*

11 Conclusion

In this paper, we have designed an efficient information theoretic secure AMPC protocol with $n = 4t + 1$ that provides perfect secrecy (without any error). Moreover, the protocol satisfies correctness and termination property, except with 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 gate. 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 (error less) 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 ASI-ACRYPT 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] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error Correcting Codes*. North-Holland Publishing Company, 1978.

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

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

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

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

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

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

[36] K. Srinathan, A. Narayanan, and C. Pandu Rangan. Optimal perfectly secure message transmission. In *Proc. of Advances in Cryptology: CRYPTO 2004*, pages 545–561, 2004.

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

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

[39] Jesper Buus Nielsen Zuzana Beerliova-Trubiniova, Martin Hirt. Almost-asynchronous mpc with faulty minority. Cryptology ePrint Archive, Report 2008/416, 2008.

APPENDIX A: Online Error Correction and Properties of Protocol Rec-Private

The current description of OEC is taken from [11]. Consider the following scenario: Let s be a secret, which is d-shared among the parties in $\mathcal{P}$, by a degree-d polynomial $f(x)$, where $d < n - 2t$. Thus, each honest party P_i has the share $s_i = f(i)$. Let $R \in \mathcal{P}$ be a specific party in $\mathcal{P}$, which we call as *receiver*. R wants to reconstruct the polynomial $f(x)$ and get s. For this, he expects the honest parties, to send their share of s to R. Due to asynchronity of the network, the shares may arrive in any arbitrary order. Moreover, t of the shares may be wrong or missing. In such a scenario, *online error correction* (OEC)

[11] allows R to recover $f(x)$ from the received shares in an *online* fashion. Informally, the procedure allows R to identify, when the received shares define a unique degree-d polynomial.

Before describing OEC, we recall the definition of *generalized Reed-Solomon* (GRS) code [29]. Consider the following code over $\mathbb{F}$: A word $W = \{(i_1, a_1), \ldots, (i_l, a_l)\}$ over $\mathbb{F}$ is a codeword iff there exists a degree-d polynomial $F(x)$ over $\mathbb{F}$, such that $F(i_j) = a_j$, for $j = 1, \ldots, l$. This code is called GRS code. GRS codes have an efficient error correcting procedure, which can correct r errors in an input word $\overline{W}$, provided that $|\overline{W}| \geq d + 2r + 1$ (see [29]). Let EC be such a procedure.

We now describe the procedure for OEC, using the above notions of GRS codes. Informally, the procedure will run for at most $t + 1$ iterations. In r^{th} iteration, R will wait to receive shares of s from $d + t + r + 1$ parties. R will now assume that at most r shares are corrupted in the received shares and try to correct them using procedure EC. Now there are two possible cases:

1. There are at most r errors in the received shares. So in this case, EC will correctly output the original degree-d polynomial $f(x)$ and hence s by correcting r errors in the received shares. R can check the validity of output polynomial $f(x)$ by verifying that $d + t + 1$ received shares (other than ones which are corrected by EC) lie on $f(x)$. Finally, in this case, R terminates the OEC procedure.

2. There are more than r errors in the received shares. So in this case, EC may either fail to output any degree-d polynomial or may output an incorrect degree-d polynomial, say $f'(x)$. In the former case, R can easily identify that more than r errors are present in the received shares. However, even in the later case, R will identify that more than r errors are present in the received shares. This is because $d + t + 1$ received shares (other than ones which are corrected by EC) will not lie on $f'(x)$. Thus R will know that more honest shares will eventually come from the honest parties and hence proceed to the next iteration.

The protocol is formally given in the following table:

Protocol OEC

For $0 \leq r \leq t$, in iteration r, $\mathbf{R}$ does the following:

1. Let $\mathcal{W}$ denote the set of shares received by $\mathbf{R}$ and I_r denote the received shares in $\mathcal{W}$, when $\mathcal{W}$ contains $d+t+r+1$ shares.
2. Wait until $|\mathcal{W}| \geq d + t + r + 1$. Then apply EC to I_r to get the polynomial $\overline{f(x)}$ of degree d. If no polynomial is output, then skip the next step and proceed to next iteration.
3. If at least $d + t + 1$ shares in I_r (other than the ones which are corrected by EC) lie on $\overline{f(x)}$, then output $\overline{s} = \overline{f(0)}$ as the secret and terminate. Otherwise, proceed to the next iteration.

Theorem 7 ([11]) *In protocol OEC, if s is correctly d-shared among the parties in $\mathcal{P}$, where $d < n - 2t$ and $|\mathcal{P}| = n$, then* $\mathbf{R}$ *will eventually output s without any error.*

Proof: Suppose $\mathcal{A}_t$ corrupts $\hat{r} \leq t$ shares of s, during their transmission to $\mathbf{R}$. Since $d < n - 2t$, during $\hat{r}^{th}$ iteration, $\mathbf{R}$ will receive $d + t + \hat{r} + 1$ shares of s, of which $\hat{r}$ are corrupted. So from the properties of GRS codes [29] (as mentioned above), EC will correct $\hat{r}$ errors in the received shares and will output $\overline{f(x)}$ of degree d. Moreover, $d + t + 1$ shares in I_r (other than the ones which are corrected by EC) will lie on $\overline{f(x)}$. Since out of these $d + t + 1$ shares, at least $d + 1$ are honest and uniquely defines the original polynomial $f(x)$ ($d+1$ honest shares are $d+1$ correct points on a degree-d polynomial and hence uniquely define a degree-d polynomial), the output polynomial $\overline{f(x)}$ is same as $f(x)$. Thus $f(x)$ will be output in $\hat{r}^{th}$ iteration and all the iterations up to iteration $\hat{r}$ will be unsuccessful, as either they will not output any degree-d polynomial or the output polynomial will not pass through $d + t + 1$ shares in I_r. □

Theorem 8 *In protocol OEC, if s is not d-shared among the parties in $\mathcal{P}$, then any of the following events may happen:*

1. $\mathbf{R}$ *may output an incorrect degree-d polynomial.*

2. $\mathbf{R}$ *may be waiting indefinitely and does not terminate.*

PROOF: The first case may occur if s is d'-shared among the parties in $\mathcal{P}$, where $d' > d$. On the other hand, second case may occur if the shares of the honest parties in $\mathcal{P}$ does not lie on a degree-d polynomial and t corrupted parties does not send their shares to $\mathbf{R}$. □

Theorem 2: *(i) Protocol Rec-Private can be used to reconstruct t-sharing as well as $2t$-sharing of s which is shared among the parties in $\mathcal{P}$, where $|\mathcal{P}| = 4t + 1$.*

(ii) If $\overline{\mathcal{P}}$ is any $3t + 1$ sized subset of $\mathcal{P}$, such that s is correctly t-shared among the parties in $\overline{\mathcal{P}}$, then Rec-Private($\overline{\mathcal{P}}, t, s, P_\alpha$) ensures successful P_α-Private-Reconstruction of s.

(iii) Let s be correctly d-shared among the parties in $\mathcal{P}$, except with probability $2^{-\Omega(\kappa)}$, where $d < n - 2t$ and $n = 4t + 1$. Then Rec-Private satisfies the TERMINATION and CORRECTNESS properties mentioned in Theorem 1, except with probability $2^{-\Omega(\kappa)}$.

PROOF: Part (i) and (ii) follows from Theorem 7. Part (iii) follows from Theorem 8 and Theorem 7. □.

APPENDIX B: Existing Protocols for Generating $2t$-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 t-sharing of $3t + 1$ 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. The protocol of [4] does not involve any error probability in CORRECTNESS and TERMINATION.

APPENDIX C: Properties of Protocol d-Share-SS

Lemma 1: *In protocol Single-Verifier-SS, if V is honest, then the points on the row polynomials, held by the honest parties in $AgreeSet^{(V,\beta)}$ (with $|AgreeSet^{(V,\beta)}| \geq 3t+1$), define n degree-t column polynomials, with very high probability. Moreover, the points on blinding polynomial $b^{(V,\beta)}(y)$ held by the honest parties in $AgreeSet^{(V,\beta)}$ will also lie on a t degree polynomial with very high probability.*

PROOF: The lemma is trivially true when D is honest. Hence we consider the case when D is corrupted. Let $H^{(V,\beta)}$ denote the set of honest parties in $AgreeSet^{(V,\beta)}$. First of all, since V is honest, he A-casts random $r^{(V,\beta)}$ only after listening `Received-From-D` signal from all the parties in $ReceivedSet^{(V,\beta)}$. Thus D has no knowledge of $r^{(V,\beta)}$, when he distributes the row polynomials and points on blinding polynomial $b^{(V,\beta)}(y)$ to the (honest) parties in $ReceivedSet^{(V,\beta)}$. Let $\overline{b^{(V,\beta)}(y)}$ denote the *minimum degree* polynomial, defined by the points on $b^{(V,\beta)}(y)$, held by the honest parties in $H^{(V,\beta)}$. Similarly, let $\overline{p_1(y)}, \ldots, \overline{p_n(y)}$ denote the *minimum degree* polynomials, defined by the row polynomials held by the parties in $H^{(V,\beta)}$. For convenience, we use an uniform notation for these $n+1$ polynomials. We denote them by $h^0(y), \ldots, h^n(y)$, respectively. Then the value $e_i^{(V,\beta)}$ A-casted by P_i is defined as $e_i^{(V,\beta)} = \sum_{j=0}^{n} \left(r^{(V,\beta)}\right)^j h^j(i)$.

We now claim that with very high probability, $h^0(y), \ldots, h^n(y)$ have degree-t. On the contrary, if we assume that at least one of the polynomials has degree more than t, then we can show that the minimum degree polynomial, say $h^{min}(y)$, defined by $e_i^{(V,\beta)}$'s for $P_i \in H^{(V,\beta)}$ will be of degree more than t, with very high probability. This will clearly imply $E^{(V,\beta)}(y) \neq h^{min}(y)$ and hence $e_i^{(V,\beta)} \neq E^{(V,\beta)}(i)$ for at least one $P_i \in H^{(V,\beta)}$. This is a contradiction as $e_i^{(V,\beta)} = E^{(V,\beta)}(i)$ holds for every $P_i \in Agree^{(V,\beta)}$ and $H^{(V,\beta)} \subseteq Agree^{(V,\beta)}$. This shows that our claim is true.

So we proceed to prove that $h^{min}(y)$ will be of degree more than t with very high probability, when one of $h^0(y), \ldots, h^n(y)$ has degree more than t. For this, we show that the following two holds:

1. We first show that $h^{def}(y) = \Sigma_{j=0}^{n} \left(r^{(V,\beta)}\right)^j h^j(y)$ will of degree more than t with very high probability, if one of $h^0(y), \ldots, h^n(y)$ has degree more than t.
2. We then show that $h^{min}(y) = h^{def}(y)$, implying that $h^{min}(y)$ will be of degree more than t with very high probability

To prove the first point, assume that at least one of $h^0(y), \ldots, h^n(y)$, has degree more than t. Let m be such that $h^m(y)$ has maximal degree among $h^0(y), \ldots, h^n(y)$, and let t_m be the degree of $h^m(y)$. Then according to the condition, $t_m > t$. Note that $t_m < |H^{(V,\beta)}|$. This is because given $|H^{(V,\beta)}|$ values (recall that $h^0(y), \ldots, h^n(y)$ are defined by the the row polynomials and values held by the honest parties in $H^{(V,\beta)}$), the maximum degree polynomial that we can define using them is $|H^{(V,\beta)}| - 1$. Now each polynomial $h^i(y)$ can be written as $h^i(y) = c^i_{t_m} y^{t_m} + \widehat{h^i(y)}$ where $\widehat{h^i(y)}$ has degree lower than t_m. Thus the polynomial $h^{def}(y)$ can be written as:

$$\begin{aligned} h^{def}(y) &= [c^0_{t_m} y^{t_m} + \widehat{h^0(y)}] + r^{(V,\beta)}[c^1_{t_m} y^{t_m} + \widehat{h^1(y)}] + \ldots + \left(r^{(V,\beta)}\right)^n [c^n_{t_m} y^{t_m} + \widehat{h^n(y)}] \\ &= y^{t_m}(c^0_{t_m} + \ldots + \left(r^{(V,\beta)}\right)^n c^n_{t_m}) + \Sigma_{j=0}^{n} \left(r^{(V,\beta)}\right)^j \widehat{h^j(y)} \end{aligned}$$

By assumption $c^m_{t_m} \neq 0$. It implies that the vector $(c^0_{t_m}, \ldots, c^n_{t_m})$ is not a complete 0 vector. Hence $c_{t_m} = c^0_{t_m} + \ldots + \left(r^{(V,\beta)}\right)^n c^n_{t_m}$ will be zero with probability $\frac{n}{|\mathbb{E}|} \approx 2^{-\Omega(\kappa)}$ (which is negligible). This is because the vector $(c^0_{t_m}, \ldots, c^n_{t_m})$ may be considered as the set of coefficients of a n degree polynomial, say $\mu(x)$, and hence the value c_{t_m} is the value of $\mu(x)$ evaluated at $r^{(V,\beta)}$. Now c_{t_m} will be zero if $r^{(V,\beta)}$ happens to be one of the n roots of $\mu(x)$ (since degree of $\mu(x)$ is at most n). Now since $r^{(V,\beta)}$ is chosen randomly from $\mathbb{E}$ by V, independent of the polynomials $h^0(y), \ldots, h^n(y)$, the probability that it is a root of $\mu(x)$ is $\frac{n}{|\mathbb{E}|} \approx 2^{-\Omega(\kappa)}$. So with very high probability c_{t_m}, which is the t_m^{th} coefficient of $h^{def}(y)$ is non-zero. This implies that $h^{def}(y)$ will be of degree at least $t_m > t$. Notice that each $e_i^{(V,\beta)}$ (A-casted by P_i), corresponding to every $P_i \in H^{(V,\beta)}$ will lie on $h^{def}(y)$.

Now we will show that $h^{min}(y) = h^{def}(y)$ and thus $h^{min}(y)$ has degree at least t_m which is greater than t. So consider the difference polynomial $dp(y) = h^{def}(y) - h^{min}(y)$. Clearly, $dp(y) = 0$, for all $y = i$, where $P_i \in H^{(V,\beta)}$. Thus $dp(y)$ will have at least $|H^{(V,\beta)}|$ roots. On the other hand, maximum degree of $dp(y)$ could be t_m, which is at most $|H^{(V,\beta)}| - 1$. These two facts together imply that $dp(y)$ is the zero polynomial, implying that $h^{def}(y) = h^{min}(y)$ and thus $h^{min}(y)$ has degree $t_m > t$. □

APPENDIX D: Protocol d-Share-MS

Protocol Distr-MS($D, \mathcal{P}, S = (s^1, \ldots, s^\ell), d$)

CODE FOR D:

1. Select ℓ random bivariate polynomial $F^1(x,y), \ldots, F^\ell(x,y)$ of degree-(d,t) such that $F^l(0,0) = s^l$ for $l = 1, \ldots, \ell$. Let $f_i^l(x) = F^l(x,i)$, $p_i^l(y) = F^(i,y)$ for $0 \leq i \leq n$ and $l = 1, \ldots, \ell$.
2. Select $(t+1)n$ degree-t random distinct *blinding polynomials* over $\mathbb{F}$, denoted by $b^{(P_i,1)}(y), \ldots, b^{(P_i,t+1)}(y)$ for $i = 1, \ldots, n$.
3. Send the following to party P_i: (i) $f_i^l(x)$ for $l = 1, \ldots, \ell$; (ii) $b^{(P_j,1)}(i), \ldots, b^{(P_j,t+1)}(i)$ for $j = 1, \ldots, n$.

Protocol Single-Verifier-MS($V, \mathcal{P}, S, d$)

i. CODE FOR P_i:

1. Wait to receive (a) $f_i^l(x)$ for $l = 1, \ldots, \ell$ (b) $b^{(P_j,1)}(i), \ldots, b^{(P_j,t+1)}(i)$ for $j = 1, \ldots, n$ from D.
2. After receiving, check whether $f_i^l(x)$ is a degree-d polynomial for all $l = 1, \ldots, \ell$. If yes, then send a `Received-From-D` signal to V.

ii. CODE FOR V:

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

iii. CODE FOR D:

1. If $(r^{(V,\beta)}, ReceivedSet^{(V,\beta)})$ is listened from the A-cast of V, then A-cast the polynomial $E^{(V,\beta)}(y) = \mathsf{LinCombPoly}(\mathcal{E}, R)$, where $\mathcal{E} = \{b^{(V,\beta)}(y), p_1^1(y), \ldots, p_n^1(y), \ldots, p^\ell(y), \ldots, p^\ell(y)\}$ and $R = (1, r^{(V,\beta)}, \left(r^{(V,\beta)}\right)^2, \ldots, \left(r^{(V,\beta)}\right)^{\ell n})$.

iv. CODE FOR P_i:

1. If $(r^{(V,\beta)}, ReceivedSet^{(V,\beta)})$ is received from the A-cast of V, then do the following:
   (a) Check if $P_i \in ReceivedSet^{(V,\beta)}$. If yes, then A-cast $e_i^{(V,\beta)} = \mathsf{LinCombValue}(\Delta_i, R)$, where $\Delta_i = \{b^{(V,\beta)}(i), f_i^1(1), \ldots, f_i^1(n), \ldots, f^\ell(1), \ldots, f^\ell(i)\}$ and $R = (1, r^{(V,\beta)}, \left(r^{(V,\beta)}\right)^2, \ldots, \left(r^{(V,\beta)}\right)^{\ell n})$.
2. Say that **party P_j agrees with D with respect to** $(r^{(V,\beta)}, ReceivedSet^{(V,\beta)})$ if the following holds:
   (a) $E^{(V,\beta)}(y)$ is t degree polynomial, (b) $P_j \in ReceivedSet^{(V,\beta)}$ and (c) $e_j^{(V,\beta)} = E^{(V,\beta)}(j)$
   where $e_j^{(V,\beta)}$, $E^{(V,\beta)}(y)$ and $(r^{(V,\beta)}, ReceivedSet^{(V,\beta)})$ are obtained from the A-casts of P_j, D and V respectively.
3. With respect to $(r^{(V,\beta)}, ReceivedSet^{(V,\beta)})$, when there are $3t + 1$ P_j's who agree with D, add all of them in a set $AgreeSet^{(V,\beta)}$.

Protocol Ver-Agree-on-CORE-MS($D, \mathcal{P}, S, d$)

Here, in step I(1), P_i invokes Single-Verifier-MS($P_\alpha, \mathcal{P}, S, d$) instead of Single-Verifier-SS. The rest of the protocol is same as in Protocol Ver-Agree-on-CORE-SS.

Protocol Gen-d-Share-MS($D, \mathcal{P}, S = (s^1, \ldots, s^\ell), d$)

FOR $j = 1, \ldots, n$, P_j-PRIVATE-RECONSTRUCTION OF $p_j^1(0), \ldots, p_j^\ell(0)$: CODE FOR P_i:

1. If $P_i \in CORE$, participate in Rec-Private($CORE, t, p_j^l(0), P_j$) for P_j-Private-Reconstruction of $p_j^l(0)$ for $l = 1, \ldots, \ell$ and $j = 1, \ldots, n$.
2. As a receiver participate in Rec-Private($CORE, t, p_i^l(0), P_i$) for P_i-Private-Reconstruction of $p_i^l(0)$ for $l = 1, \ldots, \ell$.
3. Output $f_0^l(i) = p_i^l(0)$ as the i^{th} share of secret s^l and terminate. s^l is now d-shared using polynomial $f_0^l(x)$ for $l = 1, \ldots, \ell$.

Protocol d-Share-MS($D, \mathcal{P}, S = \{s^1, \ldots, s^\ell\}, d$)

I. CODE FOR D:

1. Execute Distr-MS($D, \mathcal{P}, S, d$).

II. CODE FOR P_i:

1. Participate in Ver-Agree-on-CORE-MS($D, \mathcal{P}, S, d$).
2. Upon termination of Ver-Agree-on-CORE-MS($D, \mathcal{P}, S, d$), participate in Gen-d-Share-MS($D, \mathcal{P}, S, d$).

Theorem 4: *Protocol d-Share-MS satisfies the* TERMINATION, CORRECTNESS *and* SECRECY *property of Theorem 3. The protocol privately communicates $\mathcal{O}((\ell n^2 + n^3)\log|\mathbb{F}|)$ bits and A-cast $\mathcal{O}(n^3\kappa)$ bits where $\kappa = \log|\mathbb{E}|$.*

PROOF: The proof of TERMINATION, CORRECTNESS and SECRECY follows using similar arguments as in Theorem 3. We now do the communication complexity analysis of d-Share-MS.

During **Distribution by D** phase, D privately communicates $\mathcal{O}((\ell nd + n^3)\log(\mathbb{F}))$ bits. Since $t \leq d < n - 2t$, in our context $d = \mathcal{O}(n)$. During **Verification & Agreement on CORE** phase, the parties perform A-Cast of $\mathcal{O}(n^3\log(|\mathbb{F}|) + n^2\kappa)$ bits. During **Generation of d-Sharing of Secret** phase, the parties perform private communication of $\mathcal{O}(\ell n^2\log|\mathbb{F}|)$ bits. As $\log(|\mathbb{F}|) \leq \kappa$, overall the protocol involves a private communication of $\mathcal{O}((\ell n^2 + n^3)\log|\mathbb{F}|)$ bits and A-cast $\mathcal{O}(n^3\kappa)$ bits. □