Secure Computation in Hybrid Network

A PROJECT REPORT
SUBMITTED IN PARTIAL FULFILMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
Master of Engineering
IN
Faculty of Engineering

BY
Divya Ravi

Computer Science and Automation
Indian Institute of Science
Bangalore – 560 012 (INDIA)

July, 2017

Declaration of Originality

I, **Divya Ravi**, with SR No. **04-04-00-10-41-14-1-11144** hereby declare that the material presented in the thesis titled

Secure Computation in Hybrid Network

represents original work carried out by me in the **Deparment of Computer Science and Automation** at **Indian Institute of Science** during the years **2015-2016**.
With my signature, I certify that:

- I have not manipulated any of the data or results.
- I have not committed any plagiarism of intellectual property. I have clearly indicated and referenced the contributions of others.
- I have explicitly acknowledged all collaborative research and discusions.
- I have understood that any false claim will result in severe disciplinary action.
- I have understood that the work may be screened for any form of academic misconduct.

Date: Student Signature

In my capacity as supervisor of the above-mentioned work, I certify that the above statements are true to the best of my knowledge, and I have carried out due diligence to ensure the originality of the report.

Advisor Name: Advisor Signature

DEDICATED TO

My family and friends

Acknowledgements

Foremost, I would like to express my sincere gratitude to my advisor Dr. Arpita Patra for her continuous guidance and motivation. I will always be grateful to her for introducing me to the world of research in Computer Science. She has been a great source of inspiration for me, not just in terms of academics but personally as well.

Besides my advisor, I am immensely thankful to all the teachers that I have come across at IISc. All of them have been instrumental in developing my knowledge and skills to be able to attempt this project. The opportunities and the encouraging environment of the institute has truly made my journey at IISc, an invaluable learning experience.

I would also like to extend my sincere thanks to Dr. Ashish Choudhury, Professor at IIIT-Bangalore for his collaboration in our project work and many fruitful discussions.

I also thank my fellow labmates: Ajith, Dheeraj and Pratik for the countless stimulating discussions and making the working environment pleasant and lively. My experience at IISc has been beautiful and memorable due to the presence of close friends. I would also like to thank my parents and sister for their unconditional support.

Abstract

Secure multi-party computation (MPC) allows a set of parties to jointly compute an agreed function over their inputs, while keeping these inputs private. Most MPC protocols are designed for synchronous networks, where every message that is sent is assumed to arrive within a constant time. However, asynchronous networks are more practical since arbitrary delays occur in real-life applications like Internet. Constructing MPC protocols in asynchronous networks has been found to be challenging and has certain limitations compared to their synchronous counterparts. To achieve the best of both, a concept of *hybrid* (partial synchronous) network has been introduced. There are well-known impossibility results in asynchronous networks which are shown to be possible in hybrid network. Hybrid networks try to overcome the limitations of fully-asynchronous networks on one hand while maintaining minimal synchronicity assumption on the other. The intent of the project was to explore the potential that hybrid networks seem to offer. Our major contribution during the project is a communication-efficient statistically-secure MPC protocol in hybrid network. This work marks the first attempt in bridging the efficiency gap between statistical MPC protocols in synchronous and asynchronous network. At the heart of our MPC protocol, lies a novel statistical verifiable secret sharing (VSS) protocol. Though the VSS has non-optimal resilience, it is the first protocol to achieve quadratic complexity over point-to-point channels in four rounds. Additionally, the VSS has a very lucrative feature of broadcast complexity being independent of the number of values shared. On the practical front, it is efficient and therefore may be of independent interest.

Publications based on this Thesis

Paper titled "VSS with a Quadratic Overhead" - (Authors: Ashish Choudhury, Arpita Patra, Divya Ravi) is currently under submission to DISC 2016 (International Symposium on DIStributed Computing) Conference.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The proliferation of the Internet has triggered tremendous opportunities for cooperative computation, where people are cooperating with each other to conduct computation tasks based on their individual inputs. These computations could occur between trusted partners, between partially trusted partners, or even between competitors. For example, customers might send queries that contain private information to a remote database, two competing financial organizations might jointly invest in a project that must satisfy both organizations private and valuable constraints, and so on. Usually, to conduct these computations, one must know inputs from all the participants; however if nobody can be trusted enough to know all the inputs, privacy will become a primary concern. Secure multi-party computation (MPC) is an effective solution in such scenarios. In secure multi-party computation (MPC), n parties wish to jointly perform a computation on their private inputs in a secure way, so that no adversary Adv actively corrupting a coalition of t parties can learn more information than their outputs (privacy), nor can they affect the outputs of the computation other than by choosing their own inputs (correctness). The MPC problem dates back to Yao [42] and the first generic solutions were presented in [30, 15]. Since then various dimensions of MPC have been explored in literature based such as the nature of the adversary, underlying communication network, circuit model of computation and so on. During this project we focused on the design of MPC protocols in different communication network settings.

1.1 Network Models in MPC

In the literature, MPC has been explored in two prominent network settings: *synchronous* and *asynchronous* networks. In the synchronous setting, it is assumed that the delay of messages in the channels of the network is bounded by a *known constant*. This allows protocols to proceed in rounds, with the strong delivery guarantee that every message sent in any given round are delivered to all recipients in the same round. In contrast, in the asynchronous setting, the channels in the network may have arbitrary delays and may deliver messages in any arbitrary order, with the only restriction that every sent message must eventually be delivered. In order to model the worst case, the adversary is allowed to control the scheduling of messages in the network. The synchronous network is well-behaved and convenient, but not realistic and inapplicable to many practical environments. Whereas, the asynchronous network can aptly model real-life networks like Internet, but is difficult to deal with and less convenient. When the channel delays are short, the protocols in asynchronous network may be faster than synchronous protocols which have to allow each round to be long enough, such that all messages can get through, even in the very worst case. On the downside, asynchronous protocols suffer from low fault-tolerance, high communication complexity and *input deprivation* where the latter refers to a property where inputs of t honest parties may be excluded from computation. All the above are supposedly caused by the following inherent and trademark difficulty in the asynchronous model.

In an asynchronous network, an honest party whose message is delayed in the network cannot be told apart from a corrupt party who did not send a message at all. So an honest party in an asynchronous protocol, unlike in a synchronous protocol, cannot wait for the messages from all the parties, as it would potentially risk him to wait infinitely. To avoid the risk, an honest party's computation in an asynchronous protocol should be carried on with the receipt of $(n-t)$ parties at any given step. Unfortunately, this may risk ignoring the values of up to t potentially honest parties at any given step. In what follows, we first define the security notions commonly used in cryptography. Next, we highlight the well-known gaps in the feasibility and efficiency results of the synchronous and asynchronous MPC protocols that corroborate with the above discussed inherent difficulty faced in asynchronous protocols.

Security Notions. The security of a cryptosystem is broadly of two types: Information-theoretic (unconditional) or cryptographic (computational). In the former, the computational power of the adversary is assumed to be unbounded while the latter assumes a polynomially-bounded adversary. There are two flavours of information-theoretic protocols: *perfect* (error-free) and *statistical* (involves some probability of error).

Synchronous and Asynchronous MPC Protocols. Unconditional perfect asynchronous MPC requires $t < n/4$ [11], whereas perfect synchronous MPC is feasible with $t < n/3$ [8]. Statistical and computational asynchronous MPC protocols require $t < n/3$ [10, 32, 33], whereas their synchronous counterparts are feasible with $t < n/2$ [41, 31]. The best known perfect MPC

Table 1.1: Current Feasibility and Efficiency Bounds of MPC in different networks

Security	Network	Resilience [1]	Comm Complexity
Perfect	Synchronous	$t < n/3$ [8]	$\mathcal{O}(\lvert C\rvert n\lvert\mathbb{F}\rvert)$ [6]
	Asynchronous	$t < n/4$ [11]	$\mathcal{O}(\lvert C\rvert n^2\lvert\mathbb{F}\rvert)$ [38]
Statistical	Synchronous	$t < n/2$ [41]	$\mathcal{O}(\lvert C\rvert n\mu)$ [12]
	Asynchronous	$t < n/3$ [10]	$\mathcal{O}(\lvert C\rvert n^5\mu)$ [40]
Cryptographic	Synchronous	$t < n/2$ [21, 31]	$\mathcal{O}(\lvert C\rvert n\kappa)$ [31]
	Asynchronous	$t < n/3$ [32, 33]	$\mathcal{O}(\lvert C\rvert n\kappa)$ [16]

protocol in the synchronous and asynchronous network achieves a communication complexity $\mathcal{O}(|C|n|\mathbb{F}|)$ [6] respectively $\mathcal{O}(|C|n^2|\mathbb{F}|)$ [38] bits. Here $|C|$ denotes the number of multiplication gates in the arithmetic circuit C representing the function to be computed and $\mathbb{F}$ denotes the underlying field. The gap is noticeably wider in the statistical case. For a statistical security parameter μ, it is $\mathcal{O}(|C|n\mu)$ bits [12] versus $\mathcal{O}(|C|n^5\mu)$ bits [40]. The situation is slightly promising in the cryptographic setting. For a security parameter denoted as κ, the best protocols in both the worlds achieve $\mathcal{O}(|C|n\kappa)$ bits of communication complexity [31, 16]. But while the synchronous protocol of [31] relies on homomorphic encryption, the protocol of [16] uses *somewhat homomorphic encryption (SHE)*. A summary of the above results is given in Table 1.1.

1.2 Related Work

Several efforts have been made to close the gaps in fault-tolerance and communication efficiency of synchronous and asynchronous MPC protocols and to regain back *input provision* where all the honest parties' input will be counted in for the computation. The literature has seen these efforts resorting to three different assumptions: (i) a few synchronous rounds with or without access to broadcast oracles in the beginning of protocol execution [32, 5, 6, 17], (ii) a synchronisation point[2] at a strategic point during the protocol execution [23, 18] and (iii) non-equivocation technique [19, 2][3]. With the goal to enforce input provision, [32] introduced a special network which they term as *hybrid network* that supports a few synchronous rounds in the start of a protocol execution before turning to asynchronous mode. Specifically, [32] used *seven* initial synchronous rounds to ensure input provision in their cryptographic MPC protocol with $t < n/3$. [5] ensured input provision in their perfect MPC protocol using *one* synchronous

[2]The synchronisation point models a certain time-out such that all messages sent by honest players before the deadline will be delivered before the deadline.

[3]Non-equivocation is a message authentication mechanism to restrict a corrupt sender from making conflicting statements to different (honest) parties.

round which is clearly optimal. However, both the above protocols contribution remain in regaining input provision in asynchronous protocols. The first attempt to bridge the feasibility gap between synchronous and asynchronous MPC is made by [7]. Their cryptographic MPC protocol not only provides input provision but also works with $t < n/2$ which is the same bound necessary and sufficient for synchronous cryptographic MPC. This is achieved at the expense of one initial synchronous round that allows access to broadcast oracle.In yet another first of its kind of work, [17] shows that the communication complexity gap between synchronous and asynchronous MPC protocols with perfect security can be closed with the help of a single synchronous round (without any access to broadcast). Namely, the protocol of [17] achieves a perfect asynchronous MPC with $\mathcal{O}(|C|n|\mathbb{F}|)$ communication complexity.

1.3 Overview of VSS

Verifiable Secret Sharing (VSS) is a fundamental building block for many distributed tasks, including MPC and Byzantine Agreement (BA) [14, 37]. Informally, VSS is a two phase protocol (Sharing and Reconstruction) carried out among n parties in the presence of an adversary who can corrupt upto t parties. The goal of VSS is to share a secret s among n parties during the sharing phase in a way that would later allow for *unique reconstruction* of this secret in the reconstruction phase, while preserving the secrecy of s until the reconstruction phase. The extensive use of VSS in the above mentioned domains of distributed cryptography makes the study of communication complexity of VSS important and necessary. It is well known that *perfectly-secure* VSS is possible if and only if $t < n/3$ [24], while *statistically-secure* VSS is possible if and only if $t < n/2$ [41]. The use of broadcast channel in VSS protocols irrespective of the settings are standard and well-known. The communication complexity of any VSS therefore has two components: communication over the point-to-point channels and communication over the broadcast channel. We use $\mathsf{PC}()$ and $\mathsf{BC}()$ respectively to denote these communication complexities. We emphasize that the use of a broadcast channel in a VSS protocol is a simplifying abstraction. The broadcast calls need to be replaced with protocols to obtain communication complexity over point-to-point channels. Quite unfortunately, the best communication complexity that can be achieved by any broadcast protocol for a single bit is $\mathsf{PC}(\Omega(n^2))$ bits [36]. A communication complexity of $\mathsf{PC}(\mathcal{O}(n\ell))$ bits can be achieved for an ℓ-bit message when ℓ is $\Omega(n^7)$ bits and $\Omega(n^3)$ bits in $t < n/2$ setting [25] and in $t < n/3$ setting [39] respectively. The above results put forth the *importance of making the broadcast communication in a VSS protocol independent of the number of shared secrets.* As cited below, the best known VSS protocols do not achieve the goal. With $t < n/3$, the best known communication efficient VSS protocol [29] has communication complexity $\mathsf{PC}(\mathcal{O}(n^2\ell))$ and $\mathsf{BC}(\mathcal{O}(n^2\ell))$ for sharing ℓ secrets.

With $t < n/2$, communication efficient VSS are presented in [35, 28] with broadcast complexity of the order $\mathsf{BC}(\Omega(n^2\ell))$. All these protocols have broadcast communication dependent on the number of secrets.

1.4 Our Contribution

Motivation. We have seen that hybrid networks seem to offer immense potential in bridging the feasibility and efficiency gap between synchronous and asynchronous MPC in various settings. Consequently, a practically motivated approach would be to improve the communication complexity of MPC by considering networks that allow partial synchrony. As we saw in table 1.1, the efficiency gap is noticeably wider in the statistical case. For a statistical security parameter μ, it is $\mathcal{O}(n\mu)$ bits [12] (synchronous) versus $\mathcal{O}(n^5\mu)$ bits [40] (asynchronous) per multiplication gate. During this project, we made the first attempt in the direction to overcome this gap using a *hybrid network*. Also, we have seen that the communication done over broadcast channels during VSS is dependent on the number of secrets to be shared and inflates proportional to the latter. This motivated us to design a statistical VSS protocol (to be used as a building block of MPC) with *broadcast communication independent of the number of shared secrets*.

Our Approach and Results. Since VSS is one of the main building blocks of MPC, we attempted to bridge the efficiency gap between statistical MPC in synchronous and asynchronous networks via VSS. Our main results during the project are:

Result 1. We designed a four round statistical VSS protocol with $t < n/3$, which shares $\Theta(n\ell)$ secrets with communication complexity $\mathsf{PC}(\mathcal{O}(n^3\ell))$ and $\mathsf{BC}(\mathcal{O}(n^3))$. So the broadcast complexity is independent of ℓ. Though our protocol has *non-optimal* resilience, it is the first protocol to achieve amortized quadratic complexity over point-to-point channels in four rounds.

Result 2. We designed a communication efficient statistically-secure MPC protocol in the *partially synchronous (hybrid)* setting. Specifically in a network that is asynchronous post four initial synchronous broadcast rounds, we give an MPC protocol with $\mathcal{O}(n^2)$ communication per multiplication gate. The MPC is constructed by plugging in our VSS in the efficient framework of [17] to get the result.

We have submitted a paper “VSS with Quadratic Overhead” with the above results, to the DISC 2016 conference.

1.5 Preliminaries

We consider a set $\mathcal{P} = \{P_1, \ldots, P_n\}$ of n parties, connected by pair-wise private and authentic channels; in addition they have access to a broadcast channel. For simplicity we assume $n = 3t + 1$, so $t = \Theta(n)$. There exists a computationally unbounded centralized adversary Adv who can maliciously corrupt any t out of the n parties and may force them to behave in any arbitrary fashion during the execution of a protocol. The adversary is static, who decides the set of corrupted parties at the beginning of the protocol execution. For simplicity, we consider a completely synchronous communication setting, where the parties are assumed to be synchronised by a global clock and where there are strict upper bounds on the message delivery. Later we will discuss the adaptation of our protocols in a partially synchronous setting. Our protocols will operate over a finite field $\mathbb{F}$, where $|\mathbb{F}| > 2n$. We assume that there exists $2n$ distinct non-zero elements $\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n$ in $\mathbb{F}$. Each element of $\mathbb{F}$ can be represented by $\mathcal{O}(\log |\mathbb{F}|)$ bits. The communication complexity of any protocol is defined to be the total number of field elements communicated by the *honest* parties in that protocol. For simplicity and without loss of generality, we assume that the parties want to securely compute the function $f : \mathbb{F}^n \rightarrow \mathbb{F}$, where $f(x_1, \ldots, x_n) = y$, such that $x_i \in \mathbb{F}$ is the input of P_i and every party is supposed to receive the output $y \in \mathbb{F}$. The function f is assumed to be represented by a publicly known arithmetic circuit C over $\mathbb{F}$. The circuit C consists of n input gates, two-input addition (linear) and multiplication (non-linear) gates, zero-input random gates (for generating random values during the computation) and one output gate. We denote by c_M and c_R the number of multiplication and random gates in C respectively. By $[X]$ and $[X, Y]$ for $Y \geq X$, we denote the sets $\{1, \ldots, X\}$ and $\{X, X+1, \ldots, Y\}$, respectively. We use $i \in [k]$ to denote that i can take a value from the set $\{1, 2 \ldots k\}$. We will also require that $|\mathbb{F}| > 4n^4(c_M + c_R)(3t+1)2^{\kappa}$ to ensure that the error-probability of our MPC protocol is at most $2^{-\kappa}$, for a given error parameter κ.

1.5.1 Definitions

Definition 1.1 (d-sharing [4, 22, 6]) *A value s is said to be d-shared if there exists a polynomial over $\mathbb{F}$, say $f(\cdot)$, of degree at most d, such that $f(0) = s$ and every (honest) party $P_i \in \mathcal{P}$ holds a share s_i of s, where $s_i = f(\alpha_i)$. We denote by $[s]_d$, the vector of shares of s corresponding to the parties in $\mathcal{P}$.*

A vector $\vec{S} = (s^{(1)}, \ldots, s^{(\ell)}) \in \mathbb{F}^{\ell}$ is said to be d-shared if *each* $s^{(i)}$ is d-shared. Note that d-sharings are *linear*: given $[a]_d$ and $[b]_d$, then $[a+b]_d = [a]_d + [b]_d$ and $[c \cdot a]_d = c \cdot [a]_d$ holds, for a public constant c. In general, given ℓ sharings $[x^{(1)}]_d, \ldots, [x^{(\ell)}]_d$ and a public linear function $g : \mathbb{F}^{\ell} \rightarrow \mathbb{F}^m$, where $g(x^{(1)}, \ldots, x^{(\ell)}) = (y^{(1)}, \ldots, y^{(m)})$, then $g([x^{(1)}]_d, \ldots,$

$[x^{(\ell)}]_d) = ([y^{(1)}]_d, \ldots, [y^{(m)}]_d)$. We say that the parties *locally compute* $([y^{(1)}]_d, \ldots, [y^{(m)}]_d) = g([x^{(1)}]_d, \ldots, [x^{(\ell)}]_d)$ to mean that every P_i (locally) computes $(y_i^{(1)}, \ldots, y_i^{(m)}) = g(x_i^{(1)}, \ldots, x_i^{(\ell)})$, where $y_i^{(l)}$ and $x_i^{(l)}$ denotes the i^{th} share of $y^{(l)}$ and $x^{(l)}$.

Definition 1.2 ((Polynomial-based) Verifiable Secret Sharing (VSS)) *Let the set of L values that a dealer $\mathsf{D} \in \mathcal{P}$ wants to t-share among $\mathcal{P}$ be denoted as $\vec{S} = (s^{(1)}, \ldots, s^{(L)}) \in \mathbb{F}^L$. Let* Sh *be a synchronous protocol for the n parties, where* D *has the input $\vec{S}$. Then* Sh *is a VSS scheme if the following holds for every possible* Adv*, on all possible inputs:* **(1) Correctness***: If* D *is honest then $\vec{S}$ is t-shared among $\mathcal{P}$ at the end of* Sh*. Moreover even if* D *is corrupted there exists a set of L values, say $(\overline{s}^{(1)}, \ldots, \overline{s}^{(L)})$, which is t-shared among $\mathcal{P}$ at the end of* Sh*.* **(2) Privacy***: If* D *is honest then* Sh *reveals no information about $\vec{S}$ to* Adv *in the information-theoretic sense; i.e.* Adv*'s view is identically distributed for all possible $\vec{S}$. If* Sh *satisfies all its properties without any error then it is called perfectly-secure. If the correctness is satisfied with probability at least $1 - \epsilon$, for a given error parameter ϵ, then it is called statistically-secure.*

Information Checking with Succinct Proof of Possession (ICPoP): An ICPoP protocol involves three entities: a designated dealer $\mathsf{D} \in \mathcal{P}$ who holds a set of L private values $\mathcal{S} = \{s^{(1)}, \ldots, s^{(L)}\}$, an intermediary $\mathsf{INT} \in \mathcal{P}$ and the set of parties $\mathcal{P}$ acting as verifiers (note that D and INT will also play the role of verifiers, apart from their designated role of dealer and intermediary respectively). The protocol proceeds in three phases, each of which is implemented by a dedicated sub-protocol: **(1) Distribution Phase**: Here D, sends $\mathcal{S}$ to INT along with some *auxiliary information*. For the purpose of verification, some *verification information* is additionally sent to each individual verifier. **(2) Authentication Phase**: This phase is initiated by INT who interacts with D and the verifiers to ensure that the information it received from D is consistent with the verification information distributed to the individual verifiers. If D wants it can publicly abort this phase, which is interpreted as if D is accusing INT of malicious behaviour. **(3) Revelation Phase**: This phase is carried out by INT and the verifiers in $\mathcal{P}$ only if D has not aborted the previous phase. Here INT reveals a proof of possession of the values received from D. The verifiers in $\mathcal{P}$ check this proof with respect to their verification information. Then based on certain criteria, each verifier either outputs `AcceptProof` (indicating that it accepts the proof) or `RejectProof` (indicating that it rejects the proof).

Definition 1.3 (Information Checking with Succinct Proof of Possession (ICPoP)) *A triplet of protocols* (Distr, AuthVal, RevealPoP) *(implementing the distribution, authentication and revelation phase respectively) is called a (1 - ϵ)-secure* ICPoP*, for an error parameter ϵ, if the following holds:* **(1) ICPoP-Correctness1***: If* D *and* INT *are honest, then each honest verifier $P_i \in \mathcal{P}$ outputs* `AcceptProof` *at the end of* RevealPoP*.* **(2) ICPoP-Correctness2***:*

If D *is corrupted and* INT *is honest and if* ICPoP *proceeds to* RevealPoP*, then except with probability at most* ϵ*, all honest verifiers output* `AcceptProof` *at the end of* RevealPoP*.* **(3) ICPoP-Correctness3***: If* D *is honest,* INT *is corrupted,* ICPoP *proceeds to* RevealPoP *and if the honest verifiers output* `AcceptProof`*, then except with probability at most* ϵ*, the proof produced by* INT *corresponds*[1] *to the values in* $\mathcal{S}$*.* **(4) ICPoP-Privacy***: If* D *and* INT *are honest, then the information obtained by* Adv *during* ICPoP *is independent of* $\mathcal{S}$*.* **(5) ICPoP-Succinctness of the Proof***: The size of the proof produced by* INT *during* RevealPoP *should be independent of* L*.*

Properties of Polynomials: A bivariate polynomial $F(x, y)$ of degree at most t is of the form $F(x, y) = \sum_{i,j=0}^{i,j=t} r_{ij}x^i y^j$, where $r_{ij} \in \mathbb{F}$. Let $f_i(x) \stackrel{\text{def}}{=} F(x, \alpha_i), g_i(y) \stackrel{\text{def}}{=} F(\alpha_i, y)$ for $i \in [n]$. We call $f_i(x)$ and $g_i(y)$ as ith *row polynomial* and *column polynomial* respectively of $F(x, y)$. We say that a row polynomial $\overline{f}_i(x)$ *lies* on a bivariate polynomial $F(x, y)$ of degree at most t if $F(x, \alpha_i) = \overline{f}_i(x)$ holds. Similarly we will say that a column polynomial $\overline{g}_i(y)$ *lies* on $F(x, y)$ if $F(\alpha_i, y) = \overline{g}_i(y)$ holds. We will use some well known standard properties of bivariate and univariate polynomials, which are stated in Appendix 2.2.

1.6 Organization

In Chapter 2, we introduce a new primitive called *information checking with succinct proof of possession* (ICPoP) that is used as a building block in our VSS. Next, we show a construction of an ICPoP protocol and give a rigorous proof of its properties.

In Chapter 3, we first give a high-level overview of our statistical VSS protocol. For simplicity, we first present a 5-round statistical VSS protocol Sh-Single for sharing a single secret. We then discuss the modifications to be made to reduce the number of rounds of Sh-Single from five to four. Finally we extend this four round Sh-Single protocol to present the statistical VSS protocol Sh that has amortized quadratic communication complexity and broadcast communication independent of the number of shared secrets. This is subsequently used to design the efficient statistical MPC protocol in hybrid network. We present detailed proof of security of all our constructions of VSS.

In Chapter 4, we present the first statistical MPC protocol in hybrid network that closes the efficiency gap between the two kinds of network. The key tool for our new MPC is the statistical VSS protocol presented in Chapter 3. We conclude by summarizing our results and proposing some directions for further research.

[1]The interpretation of a proof corresponding to a set of values will be clear later during the formal presentation of our ICPoP.

Chapter 2

Information Checking with Proof of Possession

In this section, we introduce a new primitive called *information checking with succinct proof of possession* (ICPoP). This is a modification of an existing primitive known as *information checking protocol* (ICP) [41, 20, 37]. ICP is traditionally used as a tool for authenticating messages and considered to be the information-theoretically secure variant of digital signatures.

An ICPoP protocol involves three entities: a designated dealer $\mathsf{D} \in \mathcal{P}$ who holds a set of L private values $\mathcal{S} = \{s^{(1)}, \ldots, s^{(L)}\}$, an intermediary $\mathsf{INT} \in \mathcal{P}$ and the set of parties $\mathcal{P}$ acting as verifiers (note that D and INT will also play the role of verifiers, apart from their designated role of dealer and intermediary respectively). The protocol proceeds in three phases, each of which is implemented by a dedicated sub-protocol:

1. **Distribution Phase**: Here D, sends $\mathcal{S}$ to INT along with some *auxiliary information*. For the purpose of verification, some *verification information* is additionally sent to each individual verifier.

2. **Authentication Phase**: This phase is initiated by INT who interacts with D and the verifiers to ensure that the information it received from D is consistent with the verification information distributed to the individual verifiers. If D wants it can publicly abort this phase, which is interpreted as if D is accusing INT of malicious behaviour.

3. **Revelation Phase**: This phase is carried out by INT and the verifiers in $\mathcal{P}$ only if D has not aborted the previous phase. Here INT reveals a proof of possession of the values received from D. The verifiers in $\mathcal{P}$ check this proof with respect to their verification information. Then based on certain criteria, each verifier either outputs `AcceptProof`

(indicating that it accepts the proof) or `RejectProof` (indicating that it rejects the proof).

2.1 The Protocol

We present a $(1-\epsilon)$-secure ICPoP protocol, where $|\mathcal{S}| = L = \ell \times \mathsf{pck}$, with $\ell \geq 1$ and $1 \leq \mathsf{pck} \leq n-t$; moreover $\epsilon = \max\{\frac{n\ell}{|\mathbb{F}|-1}, \frac{n(n-1)}{|F|-\mathsf{pck}}\}$. The protocol has communication complexity $\mathsf{PC}(\mathcal{O}(n\ell))$ and $\mathsf{BC}(\mathcal{O}(n))$. Hence the broadcast complexity is independent of ℓ. Our ICPoP is similar to the asynchronous ICP of [37], adapted to the synchronous setting with the following differences: in ICP the whole $\mathcal{S}$ is revealed during the revelation phase, as only its authenticity is required during the revelation phase. We require INT to be able to publicly prove the possession of $\mathcal{S}$ while maintaining its privacy. Hence the auxiliary information distributed in our ICPoP differs and also used differently; the details follow. Let $\mathcal{S} = \{(s^{(1,1)}, \ldots, s^{(1,\mathsf{pck})}), \cdots, (s^{(\ell,\mathsf{pck})}, \ldots, s^{(\ell,\mathsf{pck})})\}$ denote the $L = \ell \times \mathsf{pck}$ private values of the dealer D.

Distribution Phase: During the distribution phase, D embeds the values $(s^{(k,1)}, \ldots, s^{(k,\mathsf{pck})})$ for $k \in [\ell]$ in a random degree d *secret-encoding* polynomial $G^{(k)}(x)$ at $x = \beta_1, \ldots, \beta_{\mathsf{pck}}$, where $d = \mathsf{pck} + t - 1$. In addition, D picks a *masking set* M, consisting of $2 \times \mathsf{pck}$ random values $\{(m^{(1,1)}, \ldots, m^{(1,\mathsf{pck})}), (m^{(2,1)}, \ldots, m^{(2,\mathsf{pck})})\}$, which are embedded in two random degree d polynomials $H^{(1)}(x)$ and $H^{(2)}(x)$ respectively at $x = \beta_1, \ldots, \beta_{\mathsf{pck}}$; we call these polynomials as *masking polynomials*. The polynomials are sent to INT, while each verifier P_i receives the values $v_{1,i}, \ldots, v_{\ell,i}, m_{1,i}, m_{2,i}$ of these polynomials at a secret evaluation point γ_i. This achieves ICPoP-**Privacy**, as each secret-encoding polynomial has degree d and adversary may get at most t values on these polynomials; so it will lack pck values on each polynomial to uniquely interpolate them.

Revelation Phase: During revelation phase, to give a proof of possession of $\mathcal{S}$, INT produces a random linear combination of the values in $\mathcal{S} \cup \mathsf{M}$ by making public a random linear combiner, say e and a linear combination $C(x) = eH^{(1)}(x) + e^2H^{(2)}(x) + e^3G^{(1)}(x) + \ldots + e^{\ell+2}G^{(\ell)}(x)$. The values $C(\beta_1), \ldots, C(\beta_{\mathsf{pck}})$ define pck linear combinations of $\mathcal{S} \cup \mathsf{M}$ with respect to e. The pair $(e, C(x))$ is considered as a *proof of possession* of $\mathcal{S}$ (union M) and verified as follows: each verifier locally verifies if the corresponding linear combination $em_{1,i} + e^2m_{2,i} + e^3v_{1,i} + \ldots + e^{\ell+2}v_{\ell,i}$ satisfies $C(x)$ at $x = \gamma_i$ (Condition C1) and accordingly broadcast an `Accept` or a `Reject` message. If more than t verifiers broadcast `Accept` then the proof $(e, C(x))$ is said to be accepted, other wise it is rejected. The proof will be always be accepted for an *honest* D and INT, implying ICPoP-**Correctness1**. The size of the proof is $\mathcal{O}(n)$ (as $d = \mathcal{O}(n)$), which is independent of ℓ, implying ICPoP-**Succinctness of the Proof**. No additional information about the secret-encoding polynomials is revealed from $C(x)$, thanks to the masking polynomials. If D is *honest* and INT is *corrupt* then the evaluation points of the honest verifiers will be private. So if INT gives a proof of possession of $\mathcal{S}^\star \cup \mathsf{M}^\star \neq \mathcal{S} \cup \mathsf{M}$ by revealing a linear combination of $\mathcal{S}^\star \cup \mathsf{M}^\star$ through $(e, C^\star(x))$ where $C^\star(x) \neq C(x)$, then with high probability, every honest verifier will reject the proof. This is because the corresponding linear combination of the values possessed by the honest verifiers will fail to satisfy $C^\star(x)$; this implies ICPoP-**Correctness 3**.

Authentication Phase: The above mechanism, however, fails to achieve ICPoP-**Correctness 2**, as a *corrupt* D can distribute "inconsistent" polynomials and values to an *honest* INT and honest verifiers respectively; later on the proof produced by INT will be rejected by every honest verifier. To verify the consistency of the distributed information, during the authentication phase, INT "challenges" D by making public a random linear combination $A(x)$ of the received polynomials. In response, D either instructs to abort the protocol or continue, after verifying whether the $A(x)$ polynomial satisfies the corresponding random linear combination of the values held by each verifier. The idea here is that if D distributed inconsistent data, then with very high probability, any random linear combination of the distributed polynomials would fail to satisfy the corresponding linear combination of the values given to the honest verifiers. And this will be locally learned by the honest verifiers after $A(x)$ is made public. So if D still instructs to continue the protocol, then clearly D is corrupt; so later even if the proof produced in the revelation phase turns out to be inconsistent with the information held by the honest verifiers, the proof is accepted by adding an additional acceptance condition (Condition C2) to deal with this particular case. We stress that the additional acceptance condition never gets satisfied for an *honest* D and a *corrupt* INT. The privacy of the secret-encoding polynomials is still preserved during the authentication phase (for an honest INT and D), thanks to the masking polynomials. This explains the need for two masking polynomials: one is to preserve the privacy of the secret-encoding polynomials during the authentication phase while the other is used to maintain the privacy during the revelation phase. The ICPoP protocol is given in Fig. 2.1. In the protocol, if the output is `AcceptProof` then the parties additionally output pck linear combinations of the values in $\mathcal{S} \cup \mathsf{M}$ possessed by INT; this will be useful in our VSS. For the ease of understanding, in Fig. 2.2 we present a pictorial representation of the values distributed and revealed in ICPoP.

In ICPoP, the *correspondence* between a proof and a set of values is defined as follows: Let $\mathcal{S} = \{(s^{(1,1)}, \ldots, s^{(1,\mathsf{pck})}), \ldots, (s^{(\ell,1)}, \ldots, s^{(\ell,\mathsf{pck})})\}$ and $\mathsf{M} = \{(m^{(1,1)}, \ldots, m^{(1,\mathsf{pck})}), (m^{(2,1)}, \ldots, m^{(2,\mathsf{pck})})\}$. We say that a proof $(e, C(x))$ *corresponds* to $\mathcal{S} \cup \mathsf{M}$ if $C(x)$ embeds linear combination of $\mathcal{S} \cup \mathsf{M}$ with respect to e at $x = \beta_1, \ldots, \beta_{\mathsf{pck}}$; i.e. if $C(\beta_i) = em^{(1,i)} + e^2 m^{(2,i)} + e^3 s^{(1,i)} + \ldots + e^{(\ell+2)} s^{(\ell,i)}$ holds for $i \in [\mathsf{pck}]$. We shall now proceed to formally prove the properties of ICPoP according to Definition 1.3.

Figure 2.1: Efficient ICPoP protocol where $\ell \geq 1$ and $1 \leq \mathsf{pck} \leq n - t$.

$\mathsf{ICPoP}(\mathsf{D}, \mathsf{INT}, \mathcal{P}, \ell, \mathsf{pck}, \mathcal{S}) : \mathcal{S} = \{(s^{(1,1)}, \ldots, s^{(1,\mathsf{pck})}), \ldots, (s^{(\ell,1)}, \ldots, s^{(\ell,\mathsf{pck})})\}$

$\mathsf{Distr}(\mathsf{D}, \mathsf{INT}, \mathcal{P}, \ell, \mathsf{pck}, \mathcal{S} \cup \mathsf{M})$

Round 1:

- D defines a *masking set* $\mathsf{M} \stackrel{\text{def}}{=} \{(m^{(1,1)}, \ldots, m^{(1,\mathsf{pck})}), (m^{(2,1)}, \ldots, m^{(2,\mathsf{pck})})\}$ consisting of $2\mathsf{pck}$ random elements from $\mathbb{F}$. Let $d \stackrel{\text{def}}{=} \mathsf{pck} + t - 1$. Dealer D selects ℓ random *secret-encoding polynomials* $G^{(1)}(x), G^{(2)}(x), \ldots G^{(\ell)}(x)$ of degree at most d, such that $G^{(k)}(\beta_1) = s^{(k,1)}, \ldots, G^{(k)}(\beta_{\mathsf{pck}}) = s^{(k,\mathsf{pck})}$ for $k \in [\ell]$. In addition, D selects two random *masking polynomials* $H^{(1)}(x), H^{(2)}(x)$ of degree d, such that $H^{(k)}(\beta_1) = m^{(k,1)}, \ldots, H^{(k)}(\beta_{\mathsf{pck}}) = m^{(k,\mathsf{pck})}$ for $k \in [2]$. For each verifier $P_i \in \mathcal{P}$, dealer D selects a random *evaluation point* γ_i such that $\gamma_i \in \mathbb{F} \setminus \{\beta_1, \ldots, \beta_{\mathsf{pck}}\}$.
- D *gives* $\mathcal{S} \cup \mathsf{M}$ to INT by sending $G^{(1)}(x), \ldots G^{(\ell)}(x), H^{(1)}(x)$ and $H^{(2)}(x)$ to INT. To each verifier $P_i \in \mathcal{P}$, dealer D sends $(\gamma_i, v_{1,i}, v_{2,i}, \ldots v_{\ell,i}, m_{1,i}, m_{2,i})$, where $v_{k,i} \stackrel{\text{def}}{=} G^{(k)}(\gamma_i)$ for $k \in [\ell]$ and $m_{k,i} \stackrel{\text{def}}{=} H^{(k)}(\gamma_i)$ for $k \in [2]$.

Local Computation by INT: Let $\overline{G}^{(1)}(x), \ldots \overline{G}^{(\ell)}(x), \overline{H}^{(1)}(x)$ and $\overline{H}^{(2)}(x)$ be the polynomials received from D (if D is honest then these will be the same polynomials as selected by D). INT sets $\overline{\mathcal{S}} = \{(\overline{s}^{(1,1)}, \ldots, \overline{s}^{(1,\mathsf{pck})}), \ldots, (\overline{s}^{(\ell,1)}, \ldots, \overline{s}^{(\ell,\mathsf{pck})})\}$ and $\overline{\mathsf{M}} = \{(\overline{m}^{(1,1)}, \ldots, \overline{m}^{(1,\mathsf{pck})}), (\overline{m}^{(2,1)}, \ldots, \overline{m}^{(2,\mathsf{pck})})\}$, where $\overline{s}^{(k,1)} = \overline{G}^{(k)}(\beta_1), \ldots, \overline{s}^{(k,\mathsf{pck})} = \overline{G}^{(k)}(\beta_{\mathsf{pck}})$ for $k \in [\ell]$ and $\overline{m}^{(k,1)} = \overline{H}^{(k)}(\beta_1), \ldots, \overline{m}^{(k,\mathsf{pck})} = \overline{H}^{(k)}(\beta_{\mathsf{pck}})$ for $k \in [2]$; $\overline{\mathcal{S}} \cup \overline{\mathsf{M}}$ are considered to be *received* by INT from D.

Local Computation Each Verifier P_i: Let $(\overline{\gamma}_i, \overline{v}_{1,i}, \overline{v}_{2,i}, \ldots \overline{v}_{\ell,i}, \overline{m}_{1,i}, \overline{m}_{2,i})$ be the tuple received from D (if D is honest then this will be the same tuple as computed by D).

$\mathsf{AuthVal}(\mathsf{D}, \mathsf{INT}, \mathcal{P}, \ell, \mathsf{pck}, \overline{\mathcal{S}} \cup \overline{\mathsf{M}})$

Round 1: INT selects a random element $d \in \mathbb{F} \setminus \{0\}$ and broadcasts $(d, A(x))$, where $A(x) \stackrel{\text{def}}{=} d\overline{H}^{(1)}(x) + d^2\overline{H}^{(2)}(x) + d^3\overline{G}^{(1)}(x) + d^4\overline{G}^{(2)}(x) + \ldots d^{\ell+2}\overline{G}^{(\ell)}(x)$.

Round 2: Upon receiving $(d, A(x))$ from the broadcast of INT, D checks if $A(\gamma_i) = dm_{1,i} + d^2m_{2,i} + d^3v_{1,i} + d^4v_{2,i} \ldots d^{\ell+2}v_{\ell,i}$ holds for every $P_i \in \mathcal{P}$. If not then it broadcasts an `Abort` messages, else it broadcasts an `OK` message.

$\mathsf{RevealPoP}(\mathsf{D}, \mathsf{INT}, \mathcal{P}, \ell, \mathsf{pck}, \overline{\mathcal{S}} \cup \overline{\mathsf{M}})$: This protocol is executed only if D broadcasted `OK` message during $\mathsf{AuthVal}$.

Round 1: INT chooses a random element $e \in \mathbb{F} \setminus \{0\}$ and broadcasts $(e, C(x))$ as a *proof of possession* of $\overline{\mathcal{S}} \cup \overline{\mathsf{M}}$, where $C(x) \stackrel{\text{def}}{=} e\overline{H}^{(1)}(x) + e^2\overline{H}^{(2)}(x) + e^3\overline{G}^{(1)}(x) + e^4\overline{G}^{(2)}(x) \ldots e^{\ell+2}\overline{G}^{(\ell)}(x)$.

Round 2: Upon receiving the broadcast of $(e, C(x))$ from INT, every verifier $P_i \in \mathcal{P}$ locally verifies the following conditions:

- $C(\overline{\gamma}_i) \stackrel{?}{=} e\overline{m}_{i,1} + e^2\overline{m}_{i,2} + e^3\overline{v}_{1,i} + e^4\overline{v}_{2,i} + \ldots e^{\ell+2}\overline{v}_{\ell,i}$ — we call this condition as **C1**.
- $A(\gamma_i) \neq d\overline{m}_{1,i} + d^2\overline{m}_{2,i} + d^3\overline{v}_{1,i} + d^4\overline{v}_{2,i} + \ldots d^{\ell+2}\overline{v}_{\ell,i}$ holds during $\mathsf{AuthVal}$ — we call this condition as **C2**.

Verifier P_i broadcasts `Accept` if either of the conditions C1 or C2 is true for P_i, else P_i broadcasts `Reject`.

Output Determination: If more than t verifiers broadcast `Accept` then each verifier P_i outputs `AcceptProof` along with the vector $(\mathsf{comb}_1, \ldots, \mathsf{comb}_{\mathsf{pck}}) \stackrel{\text{def}}{=} (C(\beta_1), \ldots, C(\beta_{\mathsf{pck}}))$, else each verifier P_i outputs `RejectProof`.

Figure 2.2: Pictorial representation of the information generated and communicated during ICPoP protocol

(a) The values communicated during Distr. The two masking polynomials of degree d are $H^{(1)}(x)$ and $H^{(2)}(x)$ (shown in blue) which embeds the masking values $\{m^{(1,1)} \ldots m^{(1,\mathsf{pck})}\}$ and $\{m^{(2,1)} \ldots m^{(2,\mathsf{pck})}\}$ respectively. The ℓ secret-encoding polynomials of degree d are $G^{(1)}(x) \cdots G^{(\ell)}(x)$ (shown in red) where $G^{(k)}(x)$ embeds pck secrets i.e $\{s^{(k,1)} \ldots s^{(k,\mathsf{pck})}\}$. All embeddings are done at $x = \beta_1, \ldots, \beta_{\mathsf{pck}}$.

$$\begin{array}{cc|ccc|}
\cline{3-5}
H^{(1)}(x) & \Rightarrow & m^{(1,1)} & \cdots & m^{(1,\mathsf{pck})} \\
H^{(2)}(x) & \Rightarrow & m^{(2,1)} & \cdots & m^{(2,\mathsf{pck})} \\
G^{(1)}(x) & \Rightarrow & s^{(1,1)} & \cdots & s^{(1,\mathsf{pck})} \\
\vdots & & \vdots & \vdots & \vdots \\
G^{(\ell)}(x) & \Rightarrow & s^{(\ell,1)} & \cdots & s^{(\ell,\mathsf{pck})} \\
\cline{3-5}
\end{array}$$

(b) The output vector $(\mathsf{comb}_1, \ldots, \mathsf{comb}_{\mathsf{pck}}) = (C(\beta_1), \ldots, C(\beta_{\mathsf{pck}}))$ that is revealed by INT during $\mathsf{RevealPoP}$ (shown in blue color) via the proof $(e, C(x))$. We note that comb_k is a linear combination of the kth value embedded in $H^{(1)}(x), H^{(2)}(x), G^{(1)}(x), \ldots, G^{(\ell)}(x)$ with respect to the combiner e, for $k = 1, \ldots, \mathsf{pck}$. This is represented as the kth column in the matrix representation (shown in green color).

$$\begin{array}{cc|ccccc|}
\cline{3-7}
H^{(1)}(x) & \Rightarrow & m^{(1,1)} & \cdots & m^{(1,k)} & \cdots & m^{(1,\mathsf{pck})} \\
H^{(2)}(x) & \Rightarrow & m^{(2,1)} & \cdots & m^{(2,k)} & \cdots & m^{(2,\mathsf{pck})} \\
G^{(1)}(x) & \Rightarrow & s^{(1,1)} & \cdots & s^{(1,k)} & \cdots & s^{(1,\mathsf{pck})} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
G^{(\ell)}(x) & \Rightarrow & s^{(\ell,1)} & \cdots & s^{(\ell,k)} & \cdots & s^{(\ell,\mathsf{pck})} \\
\cline{3-7}
\multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \Downarrow & & \Downarrow & & \multicolumn{1}{c}{\Downarrow} \\
\multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \mathsf{comb}_1 & & \mathsf{comb}_k & & \multicolumn{1}{c}{\mathsf{comb}_{\mathsf{pck}}}
\end{array}$$

$$\begin{aligned}
\mathsf{comb}_k &= em^{(1,k)} + e^2 m^{(2,k)} + e^3 s^{(1,k)} + \cdots + e^{\ell+2} s^{(\ell,k)} \\
&= eH^{(1)}(\beta_k) + e^2 H^{(2)}(\beta_k) + \cdots + e^{\ell+2} G^{(\ell)}(\beta_k) \\
&= C(\beta_k),
\end{aligned}$$

where $C(x) = eH^{(1)}(x) + e^2 H^{(2)}(x) + e^3 G^{(1)}(x) + \ldots + e^{\ell+1} G^{(\ell)}(x)$

Lemma 2.1 (ICPoP-**Correctness1**) *If* D *and* INT *are honest then each honest verifier* $P_i \in \mathcal{P}$ *outputs* `AcceptProof` *along with* $(C(\beta_1), \ldots, C(\beta_{\mathsf{pck}}))$ *at the end of* RevealPoP*.*

Proof: If D is honest, then for each honest verifier $P_i \in \mathcal{P}$, the relationship $G^{(k)}(\gamma_i) = v_{k,i}$ will hold for each $k \in [\ell]$ and also $H^{(1)}(\gamma_i) = m_{1,i}$ and $H^{(2)}(\gamma_i) = m_{2,i}$ will hold. Moreover if INT is honest then it will correctly broadcast the $C(x)$ polynomial during RevealPoP and each honest verifier P_i will find that the condition C1 is true. Hence each honest verifier will broadcast `Accept`. As there are more than t honest verifiers who will broadcast `Accept` messages, each honest verifier will see more than t `Accept` messages and hence will output `AcceptProof`. $\square$

Lemma 2.2 (ICPoP-**Correctness2**) *If* D *is corrupt and* INT *is honest, and if* ICPoP *proceeds to* RevealPoP*, then all honest verifiers output* `AcceptProof`*, except with probability at most* $\frac{n\ell}{|\mathbb{F}|-1}$*.*

Proof: We claim that if INT is *honest* and if ICPoP proceeds to RevealPoP, then an *honest* verifier P_i will broadcast `Accept` message, except with probability at most $\frac{\ell}{|\mathbb{F}|-1}$. Assuming that the claim is true, from the union bound it follows that the probability any honest verifier fails to broadcast an `Accept` message is at most $\frac{n\ell}{|\mathbb{F}|-1}$, as the number of honest parties is upper bounded by n. This ensures that there will be more than t `Accept` messages broadcasted by honest verifiers implying that each honest verifier will output `AcceptProof` at the end of RevealPoP. We next proceed to prove our claim. For this we focus on a designated *honest* verifier P_i and consider the relationship that holds between the polynomials $\overline{G}^{(1)}(x), \ldots, \overline{G}^{(\ell)}(x), \overline{H}^{(1)}(x), \overline{H}^{(2)}(x)$ distributed by a *corrupt* D to INT and the tuple $(\overline{\gamma}_i, \overline{v}_{1,i}, \overline{v}_{2,i}, \ldots \overline{v}_{\ell,i}, \overline{m}_{1,i}, \overline{m}_{2,i})$ distributed by D to P_i. We have the following two cases:

- $\overline{v}_{k,i} = \overline{G}^{(k)}(\overline{\gamma}_i)$ for each $k \in [\ell]$ and $\overline{m}_{1,i} = \overline{H}^{(1)}(\overline{\gamma}_i), \overline{m}_{2,i} = \overline{H}^{(2)}(\overline{\gamma}_i)$: In this case, the claim is true without any error. This is because P_i will find that condition C1 is true for the $C(x)$ polynomial during RevealPoP.

- At least one of the following holds — either $\overline{v}_{k,i} \neq \overline{G}^{(k)}(\overline{\gamma}_i)$ for some $k \in [\ell]$ or $\overline{m}_{1,i} \neq \overline{H}^{(1)}(\overline{\gamma}_i)$ or $\overline{m}_{2,i} \neq \overline{H}^{(2)}(\overline{\gamma}_i)$: In this case, $A(\overline{\gamma}_i) \neq d\overline{m}_{1,i} + d^2\overline{m}_{2,i} + d^3\overline{v}_{1,i} + d^4\overline{v}_{2,i} + \ldots d^{\ell+2}\overline{v}_{\ell,i}$ will hold, except with probability at most $\frac{\ell}{|\mathbb{F}|-1}$ (follows from Claim 2.2 by substituting $L = \ell + 2$). So clearly the verifier P_i will find that condition C2 is true during RevealPoP.

$\square$

Lemma 2.3 (ICPoP-**Correctness3**) *If* D *is honest,* INT *is corrupt,* ICPoP *proceeds to* RevealPoP *and if the honest verifiers output* `AcceptProof`*, then except with probability at most* $\frac{nd}{|\mathbb{F}|-\mathsf{pck}}$*, the proof produced by* INT *corresponds to the values in* $\mathcal{S} \cup \mathsf{M}$*.*

Proof: If ICPoP proceeds to $\mathsf{RevealPoP}$ then it implies that D broadcasted `OK` message during $\mathsf{AuthVal}$ which implies that INT broadcasted the correct $A(x)$ polynomial during $\mathsf{AuthVal}$. More specifically, the condition $A(\gamma_i) = dm_{1,i} + d^2 m_{2,i} + d^3 v_{1,i} + d^4 v_{2,i} \ldots d^{\ell+2} v_{\ell,i}$ will hold for every verifier $P_i \in \mathcal{P}$. This further implies that during $\mathsf{RevealPoP}$, the condition C2 will never be satisfied for any honest verifier P_i. To prove the lemma statement, we have to consider the case when a corrupt INT reveals a polynomial $C^\star(x) \neq eH^{(1)}(x) + e^2 H^{(2)}(x) + e^3 G^{(1)}(x) + e^4 G^{(2)}(x) + \ldots + e^{\ell+2} G^{(\ell)}(x)$ during $\mathsf{RevealPoP}$ (if INT produces the correct $C(x)$ polynomial then the lemma statement is true without any error probability). We claim that the probability that an honest verifier $P_i \in \mathcal{P}$ broadcasts `Accept` message corresponding to $C^\star(x)$ is at most $\frac{d}{|\mathbb{F}| - \mathsf{pck}}$. Assuming that the claim is true, it follows via the union bound that the probability that any honest verifier broadcasts `Accept` message corresponding to $C^\star(x)$ is at most $\frac{nd}{|\mathbb{F}| - \mathsf{pck}}$, as the number of honest verifiers is upper bounded by n. This implies that there can be at most t `Accept` messages corresponding to $C^\star(x)$, broadcasted by t potentially corrupt verifiers, implying that each honest verifier will output `RejectProof`. We next prove our claim. For this we focus on a designated honest verifier P_i. As discussed above, the condition C2 will never happen for P_i. So P_i will broadcast `Accept` message only if condition C1 holds for P_i. In order that C1 is satisfied for P_i, it should hold that $C^\star(\gamma_i) = C(\gamma_i)$. However since D is honest, the adversary will have no information about the secret evaluation point γ_i. So the only way a corrupt INT can ensure that $C^\star(\gamma_i) = C(\gamma_i)$ holds is by correctly guessing γ_i, which it can do with probability at most $\frac{d}{|\mathbb{F}| - \mathsf{pck}}$. This is because two different polynomials of degree at most d can have at most d common roots and $\gamma_i \in \mathbb{F} \setminus \{\beta_1, \ldots, \beta_{\mathsf{pck}}\}$. $\square$

Lemma 2.4 (ICPoP-Privacy) *If* D *and* INT *are honest, then the information obtained by* Adv *during* ICPoP *is independent of the values in* $\mathcal{S}$.

Proof: Without loss of generality, let us assume that $P_1, P_2 \ldots P_t$ are under the control of Adv. We claim that adversary learns nothing about $G^{(1)}(x), \ldots, G^{(\ell)}(x)$ beyond t distinct values on these polynomials, different from $x = \beta_1, \ldots, \beta_{\mathsf{pck}}$. As each of these polynomials are of degree at most $d = t + \mathsf{pck} - 1$, this implies that Adv learns nothing about the value of these polynomials at $\beta_1, \ldots, \beta_{\mathsf{pck}}$, which are nothing but elements of $\mathcal{S}$. We next proceed to prove our claim.

During Distr, adversary will obtain the tuple $(\gamma_i, v_{1,i}, v_{2,i}, \ldots v_{\ell,i}, m_{1,i}, m_{2,i})$ corresponding to each $P_i \in \{P_1, \ldots, P_t\}$ via which it obtains t distinct values of $G^{(1)}(x), \ldots, G^{(\ell)}(x), H^{(1)}(x), H^{(2)}(x)$. During $\mathsf{AuthVal}$, adversary will obtain $d, A(x)$. In addition, during $\mathsf{RevealPoP}$, adversary will obtain $e, C(x)$. However even after seeing $A(x)$ and $C(x)$, the privacy of $G^{(k)}(\beta_1), \ldots, G^{(k)}(\beta_{\mathsf{pck}})$ will be preserved for each $k \in [\ell]$. This is because the polynomials $G^{(1)}(x), \ldots, G^{(\ell)}(x)$ are masked with $H^{(1)}(x)$ and $H^{(2)}(x)$ in the $A(x)$ and $C(x)$ polynomials and adversary will lack

pck values of $H^{(1)}(x)$ and $H^{(2)}(x)$ to uniquely interpolate them. More specifically, from the view point of the adversary, for every choice $\overline{\mathcal{S}} = \{(\overline{s}^{(1,1)}, \ldots, \overline{s}^{(1,\mathsf{pck})}), \ldots, (\overline{s}^{(\ell,1)}, \ldots, \overline{s}^{(\ell,\mathsf{pck})})\}$ of the secret values, there exists corresponding secret-encoding polynomials $\overline{G}^{(1)}(x), \ldots, \overline{G}^{(\ell)}(x)$ of degree d, with $\overline{G}^{(k)}(\beta_1) = \overline{s}^{(k,1)}, \ldots, \overline{G}^{(k)}(\beta_{\mathsf{pck}}) = \overline{s}^{(k,\mathsf{pck})}$ for each $k \in [\ell]$, such that $\overline{G}^{(k)}(\gamma_i) = v_{k,i}$ holds corresponding to each $P_i \in \{P_1, \ldots, P_t\}$. Moreover corresponding to $\overline{G}^{(1)}(x), \ldots, \overline{G}^{(\ell)}(x)$ and the polynomials $A(x), C(x)$, there exist corresponding masking polynomials $\overline{H}^{(1)}(x), \overline{H}^{(2)}(x)$ (degree at most d) and masking set of values $\overline{\mathsf{M}} = \{(\overline{m}^{(1,1)}, \ldots, \overline{m}^{(1,\mathsf{pck})}), (\overline{m}^{(2,1)}, \ldots, \overline{m}^{(2,\mathsf{pck})})\}$, such that $A(x) = d\overline{H}^{(1)}(x) + d^2\overline{H}^{(2)}(x) + d^3\overline{G}^{(1)}(x) + d^4\overline{G}^{(2)}(x) + \ldots + d^{\ell+2}\overline{G}^{(\ell)}(x)$ and $C(x) = e\overline{H}^{(1)}(x) + e^2\overline{H}^{(2)}(x) + e^3\overline{G}^{(1)}(x) + e^4\overline{G}^{(2)}(x) + \ldots + e^{\ell+2}\overline{G}^{(\ell)}(x)$ holds, where $\overline{H}^{(1)}(\beta_1) = \overline{m}^{(1,1)}$, $\ldots, \overline{H}^{(1)}(\beta_{\mathsf{pck}}) = \overline{m}^{(1,\mathsf{pck})}$ and $\overline{H}^{(2)}(\beta_1) = \overline{m}^{(2,1)}, \ldots, \overline{H}^{(2)}(\beta_{\mathsf{pck}}) = \overline{m}^{(2,\mathsf{pck})}$ with $\overline{H}^{(1)}(\gamma_i) = m_{1,i}$, $\overline{H}^{(2)}(\gamma_i) = m_{2,i}$ holding for each $P_i \in \{P_1, \ldots, P_t\}$. $\square$

Theorem 2.1 *Protocols* ($\mathsf{Distr}, \mathsf{AuthVal}, \mathsf{RevealPoP}$) *constitute a* $(1-\epsilon)$*-secure* ICPoP *for* $L = \ell \times \mathsf{pck}$ *values with* $\ell \geq 1$ *and* $1 \leq \mathsf{pck} \leq n-t$, *where* $\epsilon = \max\{\frac{n\ell}{|\mathbb{F}|-1}, \frac{nd}{|F|-\mathsf{pck}}\}$ *and* $d = \mathsf{pck}+t-1$. *The protocol has communication complexity* $\mathsf{PC}(\mathcal{O}(n\ell))$ *and* $\mathsf{BC}(\mathcal{O}(n))$.

Proof: The properties of ICPoP follows from Lemma 2.1-2.4. We next prove the communication complexity. During Distr, D sends $\ell+2$ polynomials of degree d to INT and a tuple of $\ell+3$ values to each individual verifier. During AuthVal a polynomial of degree d is broadcasted by INT and D broadcasts either an `OK` or `Abort` message. During RevealPoP, INT broadcasts a polynomial of degree d and each individual verifier broadcasts either an `Accept` or a `Reject` message. So overall the protocol has communication complexity $\mathsf{PC}(\mathcal{O}(n\ell))$ and $\mathsf{BC}(\mathcal{O}(n))$, as $d = \mathcal{O}(n)$. This also proves the ICPoP-**Succinctness of the Proof** property, as the size of the proof is independent of ℓ. $\square$

Transferability of ICPoP. : In our VSS protocol we will use ICPoP as follows: after receiving $\mathcal{S} \cup \mathsf{M}$ from D via the secret-encoding and masking polynomials, INT will send these polynomials (and hence $\mathcal{S} \cup \mathsf{M}$) to another designated party, say $P_R \in \mathcal{P}$ (if INT is corrupt then it can send incorrect polynomials to P_R). Later on, party P_R will act as an INT and produce a proof of possession of $\mathcal{S} \cup \mathsf{M}$, which got "transferred" to P_R from INT; the proof gets verified with respect to the verification information held by the verifiers. This transfer of $\mathcal{S} \cup \mathsf{M}$ will satisfy all the properties of ICPoP, imagining P_R as the new INT. Specifically if D is *honest* and both INT and P_R are *honest*, then the privacy will hold. Moreover if P_R produces a proof of possession of incorrect sets (this can be the case if either INT or P_R is corrupt), then the proof gets rejected. If D is *corrupt* and both INT and P_R are honest then the proof given by P_R will be accepted.

2.2 Appendix: Properties of Polynomials

The following properties of bivariate polynomials are well known.

Lemma 2.5 ([13, 1, 38]) *Let $f_1(x), \ldots, f_\ell(x), g_1(y), \ldots, g_\ell(y)$ be degree t univariate polynomials with $t+1 \leq \ell \leq n$, such that $f_i(\alpha_j) = g_j(\alpha_i)$ holds for every $\alpha_i, \alpha_j \in \{\alpha_1, \ldots, \alpha_\ell\}$. Then there exists a unique bivariate polynomial $\overline{F}(x,y)$ of degree t, such that $f_i(x)$ and $g_i(y)$ lie on $\overline{F}(x,y)$, for $i \in [\ell]$.*

Lemma 2.6 ([13, 1, 38]) *Let $f_1(x), \ldots, f_\ell(x)$ be univariate polynomials of degree at most t where $t+1 \leq \ell \leq n$. Let $F(x,y)$ and $G(x,y)$ be two bivariate polynomials of degree at most t, such that $f_i(x)$ lies on both $F(x,y)$ and $G(x,y)$ for each $i \in [\ell]$. Then $F(x,y) = G(x,y)$.*

The following properties of univariate polynomials are standard.

Claim 2.2 *Let $G^{(1)}(x), \ldots G^{(L)}(x)$ be degree d polynomials and let $A(x) = eG^{(1)}(x) + \cdots + e^L G^{(L)}(x)$, where e is a random value from $\mathbb{F} \setminus \{0\}$. Let a tuple $(\gamma, v_1, v_2, \ldots v_L)$ be such that $v_i \neq G^{(i)}(\gamma)$ for some $i \in [L]$. Then except with probability at most $\frac{L-2}{|\mathbb{F}|-1}$, the condition $A(\gamma) \neq ev_1 + \ldots e^L v_L$ holds.*

Proof: Let $v_i \neq G^{(i)}(\gamma)$ for some $i \in [L]$. Then consider the two polynomials $D_1(\cdot)$ and $D_2(\cdot)$ of degree at most $L-1$ with coefficient vector $(G^{(1)}(\alpha), G^{(2)}(\alpha), \ldots, G^{(L)}(\alpha))$ and $(v_1, v_2, \ldots, v_L)$ respectively. As the coefficient vectors are different, $D_1(\cdot)$ and $D_2(\cdot)$ are two different polynomials and can have at most $L-2$ common non-zero roots. As e is randomly selected from $\mathbb{F} \setminus \{0\}$, it implies that $D_1(e) = D_2(e)$ will hold with probability at most $\frac{L-2}{|\mathbb{F}|-1}$, implying $A(\gamma) \neq ev_1 + e^2 v_2 + \ldots e^L v_L$. □

Claim 2.3 *Let $h^{(0)}(y), \ldots h^{(L)}(y)$ be $L+1$ polynomials and r be a random value from $\mathbb{F} \setminus \{0\}$ Let $h_{com}(y) \stackrel{def}{=} h^{(0)}(y) + rh^{(1)}(y) + \ldots r^L h^{(L)}(y)$. If at least one of $h^{(0)}(y), \ldots h^{(L)}(y)$ has degree more than t, then except with probability at most $\frac{L}{|\mathbb{F}|}$, the polynomial $h_{com}(y)$ will have degree more than t.*

Proof: Assume that at least one of the polynomials $h^{(0)}(y), \ldots h^{(L)}(y)$ has degree more than t. Without loss of generality, let $h^{(1)}(y)$ has the maximal degree among $h^{(0)}(y), \ldots h^{(L)}(y)$, with degree t_{max}, where $t_{max} > t$ (in our context t_{max} will be finite). Then we express every $h^{(i)}(y)$ as $h^{(i)}(y) = c_i y^{t_{max}} + \hat{h}^{(i)}(y)$, where $\hat{h}^{(i)}(y)$ has degree lower than t_{max}. Then $h_{com}(y) =$

$r^0h^{(0)}(y) + \cdots + r^Lh^{(L)}(y)$ can be written as:

$$\begin{aligned} h_{com}(y) &= r^0[c_0y^{t_{max}} + \hat{h}^{(0)}(y)] + \cdots + r^L[c_Ly^{t_{max}} + \hat{h}^{(L)}(y)] \\ &= y^{t_{max}}(r^0c_0 + \cdots + r^Lc_L) + \Sigma_{j=0}^{L} r^j\hat{h}^{(j)}(y) \\ &= y^{t_{max}}c_{com} + \Sigma_{j=0}^{L} r^j\hat{h}^{(j)}(y) \end{aligned} \tag{2.1}$$

where $c_{com} = r^0c_0 + \ldots r^Lc_L$. By our assumption $c_1 \neq 0$, as $h^{(1)}(y)$ has degree t_{max}. This implies that the vector $(c_0, \ldots c_L)$ is not a complete 0 vector. Hence $c_{com} = r^0c_0 + \ldots r^Lc_L$ will be zero withe probability at most $\frac{L}{|\mathbb{F}|}$. This is because $(c_0, \ldots c_L)$ can be considered as the set of coefficients of a polynomial, say $f(x)$ of degree atmost L and hence the value of c_{com} is the value of $f(x)$ at $x = r$. Now c_{com} will be zero if r happens to be one of the possible L roots of $f(x)$ (since $f(x)$ is of degree atmost L). So if r is a non-zero element, selected uniformly and at random from $\mathbb{F}$, then except with probability $\frac{L}{|\mathbb{F}|}$, $c_{com} \neq 0$ will hold and so $h_{com}(y)$ will have degree higher than t. $\square$

Chapter 3

Statistical Verifiable Secret Sharing

In the previous section, we saw an ICPoP protocol in which the INT publicly gives a proof of possession of the data originated from D instead of publicly revealing the data. Let us briefly discuss how the properties of *'succinctness of proof'* and *'transferability'* of ICPoP is related to the design of VSS. Recall that the proof was required to be "succinct" meaning that its size should be independent of the size of the data. Looking ahead, the succinct proof helps to get a VSS with broadcast complexity that is independent of the number of shared secrets. The transferability ensures that if D authenticates some data for an INT and if INT transfers this data to some other designated party P_R, then even P_R can publicly give a proof of possession of the data originated from D on the "behalf" of INT. We next give a high level overview of our VSS.

3.1 Overview of the Protocol

To share a secret s, we embed s in the constant term of a random bivariate polynomial $F(x, y)$ of degree t in x and y. Every party P_i then obtains a *row polynomial* $f_i(x) = F(x, \alpha_i)$. The parties then publicly verify whether the row polynomials of at least $n-t$ parties called VCORE define a unique bivariate polynomial without compromising the privacy of their row polynomials. The standard way to do this is to perform the "pair-wise checking", where every pair of parties (P_i, P_j) is asked to verify the consistency of the common values on their respective polynomials and publicly complain if there is any inconsistency, in which case D publicly resolves the complaint by making the common value public [29, 26, 34]. This approach will lead to a broadcast complexity of $\mathcal{O}(n^2)$ per secret-shared value; instead we use a statistical protocol called Poly-Check (section 3.2.1), adapted from [38], which performs the same task in parallel for ℓ secrets (and hence ℓ bivariate polynomials), but keeping the broadcast complexity independent of ℓ.

Once VCORE is found, it is ensured that D has committed a unique $F(x, y)$ and the secret $F(0, 0)$ to the parties in VCORE. To enable the parties to obtain their shares, the goal will be to enable each party P_j to compute its *column polynomial* $g_j(y) = F(\alpha_j, y)$. For this each party $P_i \in$ VCORE transfers its common value on $g_j(y)$ (namely $f_i(\alpha_j)$) to P_j. To ensure that correct values are transferred, P_j publicly gives a proof of possession of all the transferred values originated from D via the intermediary parties in VCORE. This is done in parallel for ℓ secrets (and hence ℓ bivariate polynomials); the succinctness of the proof ensures that this step has broadcast complexity, independent of ℓ. The details will be presented in the following sections. We note that our VSS for sharing multiple secrets is completely different from the notion of packed secret sharing [27], where multiple secrets are shared simultaneously by embedding them in a single polynomial. The latter works under the assumption that instead of corrupting at most t parties, the adversary will corrupt $t - k$ parties, for some parameter k. As a result, k secrets can be shared through a single polynomial. In our VSS, each secret is shared through an independent polynomial and the protocol will be resilient to t corruptions.

3.2 Statistical VSS with Quadratic Overhead

We present a 4-round VSS protocol Sh to t-share $\ell \times (n-t) = \Theta(n\ell)$ values with communication complexity $\mathsf{PC}(\mathcal{O}(n^3\ell))$ and $\mathsf{BC}(\mathcal{O}(n^3))$. So for sufficiently large ℓ, the broadcast complexity will be independent of ℓ. For simplicity, we will present a 5-round statistical VSS protocol Sh-Single for sharing a single secret. We will then explain how to reduce the number of rounds of Sh-Single from five to four. Finally we extend this four round Sh-Single to get Sh. We first discuss a protocol Poly-Check adapted from [38], used in our VSS. This approach will lead to a broadcast complexity of $\mathcal{O}(n^2)$ per secret-shared value; instead we use a statistical protocol called Poly-Check (Appendix 3.3.1), adapted from [38], which performs the same task in parallel for ℓ secrets (and hence ℓ bivariate polynomials), but keeping the broadcast complexity independent of ℓ.

Once VCORE is found, it is ensured that D has committed a unique $F(x, y)$ and the secret $F(0, 0)$ to the parties in VCORE. To enable the parties to obtain their shares, the goal will be to enable each party P_j to compute its *column polynomial* $g_j(y) = F(\alpha_j, y)$. For this each party $P_i \in$ VCORE transfers its common value on $g_j(y)$ (namely $f_i(\alpha_j)$) to P_j. To ensure that correct values are transferred, P_j publicly gives a *proof of possession of all the transferred values originated from* D via the intermediary parties in VCORE. This is done in parallel for ℓ secrets (and hence ℓ bivariate polynomials); the succinctness of the proof ensures that this step has broadcast complexity, independent of ℓ. We note that our VSS for sharing multiple secrets is

completely different from the notion of packed secret sharing [27], where multiple secrets are shared simultaneously by embedding them in a single polynomial. The latter works under the assumption that instead of corrupting at most t parties, the adversary will corrupt $t-k$ parties, for some parameter k. As a result, k secrets can be shared through a single polynomial. In our VSS, each secret is shared through an independent polynomial and the protocol will be resilient to t corruptions.

We present a 4-round VSS protocol Sh to t-share $\ell \times (n-t) = \Theta(n\ell)$ values with communication complexity $\mathsf{PC}(\mathcal{O}(n^3\ell))$ and $\mathsf{BC}(\mathcal{O}(n^3))$. So for sufficiently large ℓ, the broadcast complexity will be independent of ℓ. For simplicity, we will present a 5-round statistical VSS protocol $\mathsf{Sh\text{-}Single}$ for sharing a single secret. We will then explain how to reduce the number of rounds of $\mathsf{Sh\text{-}Single}$ from five to four. Finally we extend this four round $\mathsf{Sh\text{-}Single}$ to get Sh. We first discuss a protocol $\mathsf{Poly\text{-}Check}$ adapted from [38], used in our VSS.

3.2.1 Verifiably Distributing Values on Bivariate Polynomials of Degree at most t

In our VSS protocol we will come across the following situation: D will select L bivariate polynomials $F^{(1)}(x,y),\ldots,F^{(L)}(x,y)$, each of degree at most t and send the ith row polynomials $f_i^{(1)}(x),\ldots,f_i^{(L)}(x)$ of $F^{(1)}(x,y),\ldots,F^{(L)}(x,y)$ respectively to each P_i; we stress that the corresponding column polynomials are retained by D. The parties now want to publicly verify if there is a set of at least $t+1$ honest parties, who received row polynomials, lying on L unique bivariate polynomials of degree at most t without revealing any additional information about the polynomials. For this we use a two round protocol $\mathsf{Poly\text{-}Check}$, which is adapted from an asynchronous protocol for the same purpose, presented in [38]. In the protocol, there will be a designated *verifier* V, who challenges D to broadcast a random linear combination of the n column polynomials of all the bivariate polynomials selected by D. Specifically V will provide a challenge combiner, say r and in response D will make public a linear combination of its column polynomials with respect to r; to maintain the privacy of the column polynomials, this linear combination is blinded by a random degree t *blinding polynomial* $B(y)$, selected by D, with each party P_i having a value on this polynomial. Corresponding to the linear combination of the column polynomials produced by D, each party P_i will make public a linear combination of n values of all its row polynomials, with respect to the combiner r, which is blinded by the value of $B(y)$ possessed by it. The idea here is the following: if indeed there exists a set of $t+1$ honest parties that we are looking for, then the values of the row polynomials possessed by these parties will define degree t column polynomials. And these column and row polynomials will be "pair-wise consistent". Based on this idea we check if the blinded linear combination of the column polynomials produced by D is of degree t. Moreover it is also checked if there exists a *witness set* $\mathcal{W}^{(\mathsf{V})}$ of at least $2t+1$ parties, such that their blinded linear combination of row polynomial values satisfies the linear combination produced by D. If any one of the above conditions is not satisfied the parties output $\perp$, otherwise the parties output $\mathcal{W}^{(\mathsf{V})}$. It is ensured that if V is *honest*, then except with probability $\frac{nL}{|\mathbb{F}|}$, the honest parties in $\mathcal{W}^{(\mathsf{V})}$ constitute the desired set of row polynomial holders (see [38]).

We call this protocol as $\mathsf{Poly\text{-}Check}(\mathsf{D}, \mathsf{V}, \mathcal{P}, L, \{F^{(1)}(x,y), \ldots, F^{(L)}(x,y), B(y)\}, \{\overline{f}_i^{(1)}(x), \ldots, \overline{f}_i^{(L)}(x), \overline{b}_i\}_{i \in [n]})$, whose formal details are available in Fig. 3.2 of Appendix 3.3.1. Here $\{F^{(1)}(x,y), \ldots, F^{(L)}(x,y), B(y)\}$ are the inputs of D, while $\{\overline{f}_i^{(1)}(x), \ldots, \overline{f}_i^{(L)}(x), \overline{b}_i\}$ denote inputs for party P_i, namely the received row polynomials and the value of blinding polynomial. The properties of $\mathsf{Poly\text{-}Check}$ are stated in Lemma 3.7 of Appendix 3.3.1.

3.2.2 Five Round Statistical VSS for a Single Secret

To t-share s, D selects a random *secret-carrying* bivariate polynomial $F(x,y)$ of degree at most t such that $s = F(0,0)$. The ith row polynomial $f_i(x)$ of $F(x,y)$ is given to each party P_i. We stress that *only* the row polynomials are distributed by D. The parties then verify the consistency of the distributed polynomials by publicly verifying the existence of a set VCORE of at least $2t+1$ parties, such that the row polynomials of the *honest* parties in VCORE lie on a unique bivariate polynomial, say $\overline{F}(x,y)$, of degree at most t. For this, n instances of $\mathsf{Poly\text{-}Check}$ are executed (one on the behalf of each party playing the role of the designated verifier V) and it is verified if there is common subset of at least $2t+1$ parties, present across all the generated witness sets. As there will be at least one instance of $\mathsf{Poly\text{-}Check}$ executed on the behalf of an honest verifier, clearly the common subset of $2t+1$ parties satisfies the properties of VCORE. To maintain the privacy of the row polynomials during the $\mathsf{Poly\text{-}Check}$ instances, n independent *blinding polynomials* are used by D, one for each instance. If a VCORE is found, then we say that D has "committed" the secret $\overline{s} = \overline{F}(0,0)$ to the parties in VCORE via their row polynomials and the next goal will be to ensure that each party P_j obtains its column polynomial $\overline{g}_j(y)$ of $\overline{F}(x,y)$; party P_j can then output its share $\overline{s}_j = \overline{g}_j(0)$ of $\overline{s}$ and hence $\overline{s}$ will be t-shared via $\overline{F}(x,0)$. Notice that if D is *honest* then $\overline{F}(x,y) = F(x,y)$ will hold (and hence $\overline{s} = s$), as VCORE will include all the honest parties.

To enable P_j obtain $\overline{g}_j(y)$, each $P_i \in \mathsf{VCORE}$ can send the common point $\overline{f}_i(\alpha_j)$ on $\overline{g}_j(y)$ to P_j, where $\overline{f}_i(\alpha_j)$ denotes the jth value on the ith row polynomial received by P_i (if D is honest then $\overline{f}_i(\alpha_j) = f_i(\alpha_j)$ holds). The honest parties in VCORE will always send the correct values; however the corrupted parties may send incorrect values. Due to insufficient redundancy in the received $\overline{f}_i(\alpha_j)$ values, party P_j cannot error-correct them (for this we require $|\mathsf{VCORE}|$ to be of size at least $3t + 1$). The way out is that P_j gives a *proof of possession* of the $\overline{f}_i(\alpha_j)$ values received from the parties P_i in VCORE. Namely the values on the row polynomials are initially distributed by D by executing instances of Distr. There will be n^2 such instances and instance Distr_{ij} is executed to distribute $f_i(\alpha_j)$ to P_i, considering P_i as an INT; the corresponding instances $\mathsf{AuthVal}_{ij}$ are also executed and it is ensured that the $\mathsf{AuthVal}$ instances, involving any party from VCORE as an INT, is not aborted by D. Now when a party P_i in VCORE sends $\overline{f}_i(\alpha_j)$ to P_j, party P_j acts as an INT and publicly gives a proof of possession of $\overline{f}_i(\alpha_j)$ by executing an instance $\mathsf{RevealPoP}_{ji}$ of $\mathsf{RevealPoP}$. The idea here is to use the *transferability* property of ICPoP to prevent corrupted parties in VCORE from transferring incorrect values. Namely if D is *honest* and an incorrect $\overline{f}_i(\alpha_j)$ is transferred to P_j, then the corresponding proof will get rejected during $\mathsf{RevealPoP}_{ji}$ and P_j will discard such values.

Unfortunately, if D is *corrupted* then the above mechanism alone is not sufficient for P_j to robustly reconstruct $\overline{g}_j(y)$. Because a *corrupted* P_i in VCORE can then transfer an incorrect $\overline{f}_i(\alpha_j)$ to P_j and still the proof will get accepted; this is because if *both* D and INT are corrupted, then INT will know the full auxiliary and verification information involved in ICPoP. As a result, P_j will end up not reconstructing a degree t column polynomial from the received $\overline{f}_i(\alpha_j)$ values. To deal with this particular case, we ensure that the M sets used by D in the ICPoP instances have similar "structure" as the corresponding $\mathcal{S}$ sets. Specifically, D selects two random *masking* bivariate polynomials $M^{(1)}(x, y)$ and $M^{(2)}(x, y)$ each of degree at most t. Let $m_i^{(1)}(x), m_i^{(2)}(x)$ denote the corresponding row polynomials. The instances Distr_{ij} are executed by setting $\mathcal{S}_{ij} = \{f_i(\alpha_j)\}$ and $\mathsf{M}_{ij} = \{m_i^{(1)}(\alpha_j), m_i^{(2)}(\alpha_j)\}$ (thus $\ell = 1$ and $\mathsf{pck} = 1$ in these instances). The corresponding $\mathsf{AuthVal}_{ij}$ instances are executed with $\overline{\mathcal{S}}_{ij} = \{\overline{f}_i(\alpha_j)\}$ and $\overline{\mathsf{M}}_{ij} = \{\overline{m}_i^{(1)}(\alpha_j), \overline{m}_i^{(2)}(\alpha_j)\}$, which denotes the $\mathcal{S}$ and M sets respectively received by P_i during Distr_{ij} (if D is honest then these will be the same as $\mathcal{S}_{ij}$ and M_{ij}). The existence of VCORE will now imply that D has committed a secret-carrying polynomial, say $\overline{F}(x, y)$ and two masking bivariate polynomials, say $\overline{M}^{(1)}(x, y), \overline{M}^{(2)}(x, y)$ to the parties in VCORE, where all these polynomials have degree at most t. It follows that any linear combination of the column polynomials $\overline{F}(\alpha_j, y), \overline{M}^{(1)}(\alpha_j, y)$ and $\overline{M}^{(2)}(\alpha_j, y)$ will be a degree t univariate polynomial. And this property is used by P_j to identify the correctly transferred $\overline{\mathcal{S}}_{ij} \cup \overline{\mathsf{M}}_{ij}$ sets. Namely the values in the transferred $\overline{\mathcal{S}}_{ij} \cup \overline{\mathsf{M}}_{ij}$ sets should lie on degree t univariate polynomials and hence any random linear combination of these sets should also lie on a degree t polynomial. Based on this observation, party P_j selects a *common* random combiner, say e_j, for *all* the transferred $\overline{\mathcal{S}}_{ij} \cup \overline{\mathsf{M}}_{ij}$ sets and publicly reveals a linear combination of these $\overline{\mathcal{S}}_{ij} \cup \overline{\mathsf{M}}_{ij}$ sets via the $\mathsf{RevealPoP}_{ji}$ instances. It is then publicly verified if these linearly combined values lie on a degree t polynomial. If not then it implies that D is corrupted and it is discarded; see Fig. 3.1 for the formal details.

For the ease of understanding, a pictorial representation of the information distributed during Sh-Single is given in Fig. 3.3 of Appendix 3.3.2.

The following theorem states the properties of Sh-Single.

Theorem 3.1 Sh-Single *is a five round VSS protocol for a single secret, satisfying the requirements of VSS except with probability* $\frac{n^3 t}{|\mathbb{F}|-1}$. *The protocol has communication complexity* $\mathsf{PC}(\mathcal{O}(n^3))$ *and* $\mathsf{BC}(\mathcal{O}(n^3))$.

We first present some claims useful in proving the above theorem.

Claim 3.2 *If* D *is honest then except with probability at most* $\frac{n^3 t}{|\mathbb{F}|-1}$, *it will not be discarded during* Sh-Single.

Proof: If D is honest then no honest P_i will broadcast $(\mathtt{Abort}, \star)$ message as the received row polynomials will be of degree at most t. More specifically, $f_i(x) = \overline{f}_i(x) = F(x, \alpha_i), m_i^{(1)}(x) = \overline{m}_i^{(1)}(x) = M^{(1)}(x, \alpha_i)$ and $m_i^{(2)}(x) = \overline{m}_i^{(2)}(x) = M^{(2)}(x, \alpha_i)$ will hold for P_i. So there can be at most t $(\mathtt{Abort}, \star)$ messages corresponding to t potentially corrupted parties. Since D will distribute consistent row polynomials to all the parties, it follows from Lemma 3.7 and protocol steps of Poly-Check that all honest parties will be present in $\mathcal{W}^{(P_1)}, \ldots, \mathcal{W}^{(P_n)}$ and so clearly $|\mathsf{VCORE}| \geq 2t+1$ will hold. Now consider a pair of parties P_i, P_j, with at least one of them being *corrupted*, such that in the $\mathsf{RevealPoP}_{ji}$ instance the revealed proof does not correspond to $\mathcal{S}_{ij} \cup \mathsf{M}_{ij}$[1]. It follows via Lemma 2.3 (by substituting $\mathsf{pck} = 1$ and $d = t + \mathsf{pck} - 1 = t$) that except with probability at most $\frac{nt}{|\mathbb{F}|-1}$, the proof will be rejected. As there can be at most n^2 such pairs of (P_i, P_j), from the union bound it follows that except with probability at most $\frac{n^3 t}{|\mathbb{F}|-1}$, the values which are finally considered for reconstructing the column polynomials for the parties will be correct and will lie on polynomials of degree at most t. So except with probability at most $\frac{n^3 t}{|\mathbb{F}|-1}$, the conditions which will lead to an honest D being discarded never occur. □

Lemma 3.1 (Correctness for an honest D) *If* D *is honest then except with probability at most* $\frac{n^3 t}{|\mathbb{F}|-1}$, *the value* s *will be* t*-shared at the end of* Sh-Single.

Proof: If D is honest then from Claim 3.2 it follows that except with probability at most $\frac{n^3 t}{|\mathbb{F}|-1}$, any incorrect linear combination of values revealed in any of the RevealPoP instances will be rejected. More specifically, if P_j is *honest* and $P_i \in \sup_j$, then the linear combination comb_{ji} revealed by P_j in the instance $\mathsf{RevealPoP}_{ji}$ will be correct and correspond to the values in $\mathcal{S}_{ij} \cup \mathsf{M}_{ij}$. This further implies that P_i transferred the correct $\mathcal{S}_{ij} \cup \mathsf{M}_{ij}$ to P_j. Thus the values

[1] This may happen if a corrupted P_i transfers incorrect values to an honest P_j or if a corrupted P_j purposely tries to reveal a proof corresponding to an incorrect set of values.

Figure 3.1: VSS for sharing a single secret.

$\mathsf{Sh\text{-}Single}(\mathsf{D}, \mathcal{P}, s)$

Round 1: Dealer D does the following:

- Select a random *secret-carrying bivariate polynomial* $F(x, y)$ of degree at most t with $F(0, 0) = s$. Select two random *masking bivariate polynomials* $M^{(1)}(x, y)$ and $M^{(2)}(x, y)$, each of degree at most t. In addition select n random *blinding univariate polynomials* $B^{(P_1)}(y), \ldots, B^{(P_n)}(y)$, each of degree at most t, where $B^{(P_i)}$ is *associated* with party $P_i \in \mathcal{P}$. Corresponding to each $P_i \in \mathcal{P}$, compute row polynomials $f_i(x) \stackrel{\text{def}}{=} F(x, \alpha_i), m_i^{(1)}(x) \stackrel{\text{def}}{=} M^{(1)}(x, \alpha_i), m_i^{(2)}(x) \stackrel{\text{def}}{=} M^{(2)}(x, \alpha_i)$ and share-vector $(b_i^{(P_1)}, \ldots, b_i^{(P_n)})$ of blinding polynomials, where $b_i^{(P_j)} \stackrel{\text{def}}{=} B^{(P_j)}(\alpha_i)$ for $j \in [n]$. Let $\mathcal{S}_{ij} \stackrel{\text{def}}{=} \{f_i(\alpha_j)\}$ and $\mathsf{M}_{ij} \stackrel{\text{def}}{=} \{m_i^{(1)}(\alpha_j), m_i^{(2)}(\alpha_j)\}$ for $i, j \in [n]$.

- To each $P_i \in \mathcal{P}$, send $(b_i^{(P_1)}, \ldots, b_i^{(P_n)})$. In addition, for $j \in [n]$, execute an instance $\mathsf{Distr}(\mathsf{D}, P_i, \mathcal{P}, 1, 1, \mathcal{S}_{ij} \cup \mathsf{M}_{ij})$ of Distr to give $\mathcal{S}_{ij} \cup \mathsf{M}_{ij}$ to P_i, considering P_i as an INT. Let Distr_{ij} denote the corresponding instance of Distr.

Round 2: Each $P_i \in \mathcal{P}$ (including D) does the following: let $\overline{\mathcal{S}}_{ij} = \{\overline{f}_{ij}\}$ and $\overline{\mathsf{M}}_{ij} = \{\overline{m}_{ij}^{(1)}, \overline{m}_{ij}^{(2)}\}$ be the secret and masking set respectively received from D in Distr_{ij}. In addition, let $(\overline{b}_i^{(P_1)}, \ldots, \overline{b}_i^{(P_n)})$ denote the vector received [a]from D. Let $\overline{f}_i(x), \overline{m}_i^{(1)}(x)$ and $\overline{m}_i^{(2)}(x)$ be the polynomials defined by the points $\{(\alpha_j, \overline{f}_{ij})\}_{j \in [n]}, \{(\alpha_j, \overline{m}_{ij}^{(1)})\}_{j \in [n]}$ and $\{(\alpha_j, \overline{m}_{ij}^{(2)})\}_{j \in [n]}$ respectively. If these polynomials are not of degree t then P_i broadcasts $(\texttt{Abort}, P_i)$, else it does the following:

- Transfer $\overline{\mathcal{S}}_{ij} \cup \overline{\mathsf{M}}_{ij}$ to P_j by sending all the information received from D in the instance Distr_{ij}.

- As an INT, execute the steps of Round 1 of an instance $\mathsf{AuthVal}(\mathsf{D}, P_i, \mathcal{P}, 1, 1, \overline{\mathcal{S}}_{ij} \cup \overline{\mathsf{M}}_{ij})$ of AuthVal, corresponding to the instance Distr_{ij}, for $j \in [n]$. Let this instance of AuthVal be denoted as $\mathsf{AuthVal}_{ij}$.

- As a verifier V, execute the steps of Round 1 of an instance $\mathsf{Poly\text{-}Check}(\mathsf{D}, P_i, \mathcal{P}, 3, \{M^{(1)}(x, y), M^{(2)}(x, y), F(x, y), B^{(P_i)}(y)\}, \{\overline{m}_j^{(1)}(x), \overline{m}_j^{(2)}(x), \overline{f}_j(x), \overline{b}_j^{(P_i)}\}_{j \in [n]})$ of Poly-Check; denote this instance as $\mathsf{Poly\text{-}Check}^{(P_i)}$.

Round 3: Each $P_i \in \mathcal{P}$ (including D) does the following: If $(\texttt{Abort}, \star)$ message is received from the broadcast of more than t parties then discard D and abort Sh-Single. Else P_i does the following:

- Corresponding to each $j, k \in [n]$, participate as a verifier during Round 2 of AuthVal, in the instances $\mathsf{AuthVal}_{jk}$

- Execute the steps of Round 2 of Poly-Check, corresponding to the instances $\mathsf{Poly\text{-}Check}^{(P_1)}, \ldots, \mathsf{Poly\text{-}Check}^{(P_n)}$.

[Additional steps, If $P_i = \mathsf{D}$] — In addition to the above steps, P_i executes the following steps if P_i is D: (a) As a D, execute the steps of Round 2 of AuthVal, corresponding to the instances $\mathsf{AuthVal}_{jk}$ for each $j, k \in [n]$. (b) As a D, execute the steps of Round 2 of Poly-Check, corresponding to $\mathsf{Poly\text{-}Check}^{(P_1)}, \ldots, \mathsf{Poly\text{-}Check}^{(P_n)}$.

Computation of VCORE — Every party $P_i \in \mathcal{P}$ (including D) executes the following steps: (a) If in any of the instances $\mathsf{Poly\text{-}Check}^{(P_1)}, \ldots, \mathsf{Poly\text{-}Check}^{(P_n)}$ the output is $\perp$, then discard D and abort Sh-Single. (b) Let $\mathcal{W}^{(P_1)}, \ldots, \mathcal{W}^{(P_n)}$ denote the witness sets obtained in $\mathsf{Poly\text{-}Check}^{(P_1)}, \ldots, \mathsf{Poly\text{-}Check}^{(P_n)}$ respectively. If $|\mathcal{W}^{(P_1)} \cap \mathcal{W}^{(P_2)} \cap \ldots \cap \mathcal{W}^{(P_n)}| < 2t + 1$, then discard D and abort Sh-Single. Else set $\mathsf{VCORE} \stackrel{\text{def}}{=} \mathcal{W}^{(P_1)} \cap \mathcal{W}^{(P_2)} \cap \ldots \cap \mathcal{W}^{(P_n)}$. (c)If there exists any $P_j \in \mathsf{VCORE}$, such that D broadcasted Abort message in some instance $\mathsf{AuthVal}_{jk}$ involving P_j as an INT, where $k \in [n]$, then remove P_j from VCORE. If finally $|\mathsf{VCORE}| < 2t + 1$ then discard D and abort Sh-Single.

Round 4: Each party $P_j \in \mathcal{P}$ does the following: Corresponding to each $P_i \in \mathsf{VCORE}$, act as an INT and execute the steps of Round 1 of an instance $\mathsf{RevealPoP}(\mathsf{D}, P_j, \mathcal{P}, 1, 1, \overline{\mathcal{S}}_{ij} \cup \overline{\mathsf{M}}_{ij})$ of RevealPoP, to reveal a random linear combination of the values in $\overline{\mathcal{S}}_{ij} \cup \overline{\mathsf{M}}_{ij}$, which were transferred from P_i to P_j during Round 2 of Sh-Single. In all these instances of RevealPoP, party P_j uses the same random combiner, say e_j. Let these instances of RevealPoP be denoted by $\mathsf{RevealPoP}_{ji}$.

Round 5: Every party $P_k \in \mathcal{P}$ (including D) acts as a verifier and executes the steps of Round 2 of RevealPoP, corresponding to the instances $\mathsf{RevealPoP}_{ji}$, where $j \in [n]$ and $P_i \in \mathsf{VCORE}$.

Consistency checking of the values transferred by the parties in VCORE: Each $P_k \in \mathcal{P}$ verifies the following for each $P_j \in \mathcal{P}$:

- Let $\sup_j$ denote the set of all $P_i \in \mathsf{VCORE}$, such that in the corresponding $\mathsf{RevealPoP}_{ji}$ instances, the output is AcceptProof, along with a linear combination of values, say comb_{ji}.

- Discard D and abort Sh-Single if $\{(\alpha_i, \mathsf{comb}_{ji})\}_{P_i \in \sup_j}$ lie on a polynomial of degree more than t.

Share determination — Each $P_j \in \mathcal{P}$ interpolates a polynomial $\overline{g}_j(y)$ through $\{(\alpha_i, \overline{f}_{ij})\}_{P_i \in \sup_j}$, where $\overline{\mathcal{S}}_{ij} = \{\overline{f}_{ij}\}$ denotes the secret set transferred to P_j from P_i during Round 2 of Sh-Single. Party P_j outputs $\overline{s}_j = \overline{g}_j(0)$ as its share and terminates.

used by an honest P_j to determine its column polynomial are correct (lying on $g_j(y) = F(\alpha_j, y)$). So $\overline{g}_j(y) = g_j(y)$ holds for each honest P_j, implying that s will be t-shared via the polynomial $f_0(x) \stackrel{\text{def}}{=} F(x, 0)$, with P_j holding the share $f_0(j) = g_j(0)$. $\square$

Claim 3.3 *Let $\overline{f}_i(x), \overline{m}_i^{(1)}(x)$ and $\overline{m}_i^{(2)}(x)$ be the row polynomials defined by the values in $\overline{\mathcal{S}}_{ij} \cup \overline{\mathsf{M}}_{ij}$ received by party $P_i \in \mathcal{P}$ from D for $j \in [n]$. If D is corrupted and a VCORE is formed during $\mathsf{Sh\text{-}Single}$ then except with probability at most $\frac{3n^2}{|\mathbb{F}|}$, there exist bivariate polynomials, say $\overline{F}(x, y), \overline{M}^{(1)}(x, y)$ and $\overline{M}^{(2)}(x, y)$, each of degree at most t, such that for each honest $P_i \in \mathsf{VCORE}$, the polynomials $\overline{f}_i(x), \overline{m}_i^{(1)}(x)$ and $\overline{m}_i^{(2)}(x)$ lie on $\overline{F}(x, y), \overline{M}^{(1)}(x, y)$ and $\overline{M}^{(2)}(x, y)$ respectively.*

Proof: From the definition, $\mathsf{VCORE} = \mathcal{W}^{(P_1)} \cap \mathcal{W}^{(P_2)} \cap \ldots \cap \mathcal{W}^{(P_n)}$ and $|\mathsf{VCORE}| \geq 2t+1$. This ensures that there are at least $t+1$ common honest parties in VCORE, say HVCORE. Consider an honest party $P_j \in \mathcal{P}$, playing the role of the verifier V in the instance $\mathsf{Poly\text{-}Check}^{(P_j)}$. It follows from Lemma 3.7 (by substituting $L = 3$) that for the instance $\mathsf{Poly\text{-}Check}^{(P_j)}$, except with probability at most $\frac{3n}{|\mathbb{F}|}$, the row polynomials $\overline{f}_i(x), \overline{m}_i^{(1)}(x)$ and $\overline{m}_i^{(2)}(x)$ of the parties $P_i \in \mathsf{HVCORE}$ together lie on three unique bivariate polynomials, say $\overline{F}(x, y), \overline{M}^{(1)}(x, y)$ and $\overline{M}^{(2)}(x, y)$ respectively of degree at most t. The same will be true with respect to every other instance $\mathsf{Poly\text{-}Check}^{(P_k)}$, corresponding to every other honest verifier $P_k \neq P_j$. Moreover, the set of three bivariate polynomials defined via each of these instances of $\mathsf{Poly\text{-}Check}$ will be the same, namely $\overline{F}(x, y), \overline{M}^{(1)}(x, y)$ and $\overline{M}^{(2)}(x, y)$ respectively. This follows from Lemma 2.6 (by substituting $\ell = |\mathsf{HVCORE}|$) and the fact that $|\mathsf{HVCORE}| \geq t+1$. The lemma now follows from the union bound and the fact that there are $\Theta(n)$ honest parties, playing the role of V. $\square$

Lemma 3.2 (Correctness for a corrupted D) *If D is corrupted and not discarded during $\mathsf{Sh\text{-}Single}$, then there exists some value, say $\overline{s}$, such that except with probability at most $\frac{n^3}{|\mathbb{F}|-1}$, the value $\overline{s}$ will be t-shared at the end of $\mathsf{Sh\text{-}Single}$.*

Proof: If a *corrupted* D is not discarded then it implies that a set VCORE with $|\mathsf{VCORE}| \geq 2t+1$ is constructed during $\mathsf{Sh\text{-}Single}$. Let HVCORE be the set of honest parties in VCORE; clearly $|\mathsf{HVCORE}| \geq t+1$. From Claim 3.3 it follows that except with probability at most $\frac{3n^2}{|\mathbb{F}|}$, the row polynomials $\overline{f}_i(x), \overline{m}_i^{(1)}(x)$ and $\overline{m}_i^{(2)}(x)$ of the parties in HVCORE lie on unique bivariate polynomials, say $\overline{F}(x, y), \overline{M}^{(1)}(x, y)$ and $\overline{M}^{(2)}(x, y)$ of degree at most t. We define $\overline{s} \stackrel{\text{def}}{=} \overline{F}(0, 0)$ and claim that $\overline{s}$ will be t-shared via the polynomial $\overline{f}_0(x) \stackrel{\text{def}}{=} \overline{F}(x, 0)$, with each honest P_j holding the share $\overline{s}_j \stackrel{\text{def}}{=} \overline{F}(\alpha_j, 0)$. To prove our claim, we will show that each *honest* party P_j outputs its degree t univariate polynomial $\overline{g}_j(y) \stackrel{\text{def}}{=} \overline{F}(\alpha_j, y)$ except with probability at most

$\frac{n^2}{|\mathbb{F}|-1}$; this ensures that P_j obtains the correct share, as $\overline{s}_j = \overline{g}_j(0)$. For this, we further need to show that the $\overline{\mathbb{S}}_{ij}$ set transferred by each party $P_i \in \mathsf{sup}_j$ to P_j contains the value $\overline{g}_j(\alpha_i)$.

Consider an honest P_j. Notice that $\mathsf{sup}_j \subseteq \mathsf{VCORE}$. We first argue that all $P_i \in \mathsf{HVCORE}$ will be present in sup_j, except with probability at most $\frac{n^2}{|\mathbb{F}|-1}$. This is because there are $\Theta(n)$ such parties P_i and in each corresponding $\mathsf{RevealPoP}_{ji}$ instance, the output will be $\mathtt{AcceptProof}$, which follows from Lemma 2.2 (by substituting $\ell = 1$). Now consider the set of values $\overline{\mathbb{S}}_{ij} = \{\overline{f}_{ij}\}$ and $\overline{\mathsf{M}}_{ij} = \{\overline{m}_{ij}^{(1)}, \overline{m}_{ij}^{(2)}\}$ transferred by the parties $P_i \in \mathsf{HVCORE}$ to P_j. Since $\overline{f}_{ij} = \overline{f}_i(\alpha_j) = \overline{g}_j(\alpha_i)$ holds, it follows that the values $\{\overline{f}_{ij}\}_{P_i \in \mathsf{HVCORE}}$ define the degree t univariate polynomial $\overline{g}_j(y)$. Similarly the values $\{\overline{m}_{ij}^{(1)}\}_{P_i \in \mathsf{HVCORE}}$ and $\{\overline{m}_{ij}^{(2)}\}_{P_i \in \mathsf{HVCORE}}$ define degree t univariate polynomials $\overline{M}^{(1)}(y, \alpha_j)$ and $\overline{M}^{(2)}(y, \alpha_j)$ respectively. To complete the proof, we argue that except with probability at most $\frac{2}{|\mathbb{F}|}$, the values in the $\overline{\mathbb{S}}_{ij}$ and $\overline{\mathsf{M}}_{ij}$ set transferred by a *corrupted* party $P_i \in \mathsf{sup}_j$ lie on $\overline{g}_j(y), \overline{M}^{(1)}(y, \alpha_j)$ and $\overline{M}^{(2)}(y, \alpha_j)$ respectively. This is because the combiner e_j selected by the honest P_j in the $\mathsf{RevealPoP}_{ji}$ instances corresponding to the parties in sup_j is truly random and unknown to the adversary in advance, when the $\overline{\mathbb{S}}_{ij}$ and $\overline{\mathsf{M}}_{ij}$ sets were transferred to P_j. The rest follows from Claim 2.3 (by substituting $L = 2$) and the fact that the values $\{\mathsf{comb}_{ji}\}_{P_i \in \mathsf{sup}_j}$ lie on a polynomial of degree at most t (otherwise D would have been discarded), say $\mathsf{comb}_j(y)$, where $\mathsf{comb}_j(y) \stackrel{\text{def}}{=} e_j\overline{M}^{(1)}(y, \alpha_j) + e_j^2\overline{M}^{(2)}(y, \alpha_j) + e_j^3\overline{g}_j(y)$. As there can be n^2 pair of parties involving a corrupted party, it follows by the union bound that except with probability at most $\frac{2n^2}{|\mathbb{F}|}$, the corrupted parties in VCORE transfer the correct values to the honest parties.

As each honest P_j correctly obtains its column polynomial except with probability at most $\frac{n^2}{|\mathbb{F}|-1}$ and as there are $\Theta(n)$ such honest parties, it follows that except with probability at most $\frac{n^3}{|\mathbb{F}|-1}$, the value $\overline{s}$ will be t-shared. $\square$

Lemma 3.3 (Privacy) *In protocol* $\mathsf{Sh\text{-}Single}$*, the value* s *remains information-theoretically secure.*

Proof: For the privacy property, we have to consider an honest D. Without loss of generality, let $P_1, \ldots, P_t$ be under the control of Adv. We argue that throughout the protocol $\mathsf{Sh\text{-}Single}$, the adversary learns nothing about $F(x, y)$, beyond the row polynomials $f_1(x), \ldots, f_t(x)$ and the column polynomials $g_1(y), \ldots, g_t(y)$. Through these polynomials, the adversary will learn $t^2 + 2t$ distinct values of $F(x, y)$. As the degree of $F(x, y)$ is t, the adversary will lack one additional value on $F(x, y)$ to uniquely interpolate $F(x, y)$, implying information-theoretic security for s.

Through the instances Distr_{ij} where $i \in [t]$ and $j \in [n]$, the adversary Adv learns the row polynomials $f_1(x), \ldots, f_t(x), m_1^{(1)}(x), \ldots, m_t^{(1)}(x), m_1^{(2)}(x), \ldots, m_t^{(2)}(x)$ on the bivariate polynomials $F(x, y), M^{(1)}(x, y)$ and $M^{(2)}(x, y)$ respectively. From Lemma 3.7, during $\mathsf{Poly\text{-}Check}^{(P_1)}, \ldots,$

$\mathsf{Poly\text{-}Check}^{(P_n)}$, no additional information about $F(x,y), M^{(1)}(x,y)$ and $M^{(2)}(x,y)$ is revealed to the adversary, because in each instance $\mathsf{Poly\text{-}Check}^{(P_i)}$, a random blinding univariate polynomial $B^{(P_i)}(y)$ is used. Now consider a pair of *honest* parties $P_i, P_j \in \mathcal{P}$. In the protocol, party P_i executes an instance $\mathsf{AuthVal}_{ij}$ involving $\mathcal{S}_{ij} = \{f_i(\alpha_j)\}$ and $\mathsf{M}_{ij} = \{m_i^{(1)}(\alpha_j), m_i^{(2)}(\alpha_j)\}$. Moreover, the set $\mathcal{S}_{ij} \cup \mathsf{M}_{ij}$ is privately transferred to P_j by P_i and later on during Round 4 and 5, an instance $\mathsf{RevealPoP}_{ji}$ is instantiated again involving $\mathcal{S}_{ij} \cup \mathsf{M}_{ij}$. We claim that during $\mathsf{AuthVal}_{ij}$ and $\mathsf{RevealPoP}_{ji}$, the privacy of $\mathcal{S}_{ij}$ is preserved. This follows from the privacy property of ICPoP (Lemma 2.4) and the fact that the corresponding masking set M_{ij} used in these instances are private. Thus for every pair of honest parties P_i, P_j, no additional information about the $f_i(\alpha_j)$ values (which are the same as the $g_j(\alpha_i)$ values) are revealed during the instances $\mathsf{AuthVal}_{ij}$ and $\mathsf{RevealPoP}_{ji}$. The adversary will be able to compute the column polynomials $g_1(y), \ldots, g_t(y)$ through the common values on these column polynomials which are transferred to $P_1, \ldots, P_t$ by the honest parties. Hence throughout the protocol, the adversary learns t row and column polynomials, proving the privacy. □

Proof of Theorem 3.1 :

The properties of VSS follows from Lemma 3.1-3.3. In the protocol n^2 instances of ICPoP (with $\ell = 1, \mathsf{pck} = 1$) and n instances of Poly-Check (each with $L = 3$) are executed. The rest follows from the communication complexity of ICPoP (Thorem 2.1) and Poly-Check (Lemma 3.7).

From Five Rounds to Four Rounds: In Sh-Single, the instances of RevealPoP which start getting executed during Round 4 can be instead instantiated during Round 3 itself. Namely irrespective of the formation of VCORE, each party P_j starts executing the instance $\mathsf{RevealPoP}_{ji}$ corresponding to *each* party $P_i \in \mathcal{P}$, based on the set of values in $\overline{\mathcal{S}}_{ij} \cup \overline{\mathsf{M}}_{ij}$ which were transferred to P_j by P_i during Round 2. Next VCORE is computed and if P_i is found not to be present in VCORE, then the instance $\mathsf{RevealPoP}_{ji}$ can be halted; otherwise the remaining steps of the $\mathsf{RevealPoP}_{ji}$ instance will be executed during Round 4. Based on this modification, Sh-Single now requires four rounds, while rest of the properties remain the same.

3.2.3 VSS for multiple secrets

We now discuss the modifications to be made to Sh-Single to get a four round VSS protocol Sh, which allows D to t-share $\ell \times (n-t) = \Theta(n\ell)$ secrets with communication complexity $\mathsf{PC}(\mathcal{O}(n^3\ell))$ and $\mathsf{BC}(\mathcal{O}(n^3))$. For simplicity, we first discuss how to t-share $n - t = \Theta(n)$

secrets with communication complexity $\mathsf{PC}(\mathcal{O}(n^3))$ and $\mathsf{BC}(\mathcal{O}(n^3))$. The modifications to share $\ell \times (n-t)$ secrets follow in a straight forward fashion.

Sharing $n-t$ Secrets: The idea behind efficiently sharing $n-t$ secrets is to invoke the underlying instances of $\mathsf{Distr}, \mathsf{AuthVal}$ and $\mathsf{RevealPoP}$ in $\mathsf{Sh\text{-}Single}$ with the maximum possible value of pck, which is $n-t$ (for the moment we will restrict to $\ell = 1$). The rest of the protocol steps remain the same, with a slight modification in the steps for consistency checking of the values transferred by the parties in VCORE. More specifically, let $\vec{S} = (s^{(1)}, \ldots, s^{(n-t)})$ be the set of values, which need to be t-shared. To do so D selects $n-t$ random degree t secret-carrying bivariate polynomials $F^{(1)}(x,y), \ldots, F^{(n-t)}(x,y)$, embedding the secrets $s^{(1)}, \ldots, s^{(n-t)}$ respectively in their constant terms. In addition, D picks $2(n-t)$ random masking bivariate polynomials $M^{(1,1)}(x,y), \ldots, M^{(1,n-t)}(x,y), M^{(2,1)}(x,y), \ldots, M^{(2,n-t)}(x,y)$ polynomials. The reason for picking so many masking polynomials will be clear in the sequel. Let $f_i^{(1)}(x), \ldots, f_i^{(n-t)}(x)$ and $g^{(1)}(y), \ldots, g^{(n-t)}(y)$ denote the ith row and column polynomials of $F^{(1)}(x,y), \ldots, F^{(n-t)}(x,y)$ respectively. Similarly, let $m_i^{(1,1)}(x), \ldots, m_i^{(1,n-t)}(x), m_i^{(2,1)}(x), \ldots, m_i^{(2,n-t)}(x)$ denote the ith row polynomials of the masking bivariate polynomials. Corresponding to each party P_i, the dealer D sets $\mathcal{S}_{ij} = \{f_i^{(1)}(\alpha_j), \ldots, f_i^{(n-t)}(\alpha_j)\}$ and $\mathsf{M}_{ij} = \{(m_i^{(1,1)}(\alpha_j), \ldots, m_i^{(1,n-t)}(\alpha_j)), (m_i^{(2,1)}(\alpha_j), \ldots, m_i^{(2,n-t)}(\alpha_j))\}$. An instance Distr_{ij} is executed, considering P_i as an INT to give $\mathcal{S}_{ij} \cup \mathsf{M}_{ij}$ to P_i, for $j = 1, \ldots, n$. The instances of Distr are executed by setting $\ell = 1$ and $\mathsf{pck} = n-t$ (hence d will be $n-1$ in these instances). Let $\overline{f}_i^{(1)}(x), \ldots, \overline{f}_i^{(n-t)}(x), \overline{m}_i^{(1,1)}(x), \ldots, \overline{m}_i^{(1,n-t)}(x), \overline{m}_i^{(2,1)}(x), \ldots, \overline{m}_i^{(2,n-t)}(x)$ denote the row polynomials received by P_i via the instances Distr_{ij}. The parties check for the existence of VCORE as in $\mathsf{Sh\text{-}Single}$ by executing n instances of $\mathsf{Poly\text{-}Check}$, where P_i plays the role of the designated verifier in the ith instance. For each instance, one independent blinding polynomial will be used, which will be shared by D during the first round. If a VCORE is obtained, then it implies that the row polynomials of the honest parties P_i in VCORE lie on $n-t$ secret-carrying bivariate polynomials of degree t, say $\overline{F}^{(1)}(x,y), \ldots, \overline{F}^{(n-t)}(x,y)$ and $2(n-t)$ masking bivariate polynomials, say $\overline{M}^{(1,1)}(x,y), \ldots, \overline{M}^{(1,n-t)}(x,y), \overline{M}^{(2,1)}(x,y), \ldots, \overline{M}^{(2,n-t)}(x,y)$ respectively We define $(\overline{F}^{(1)}(0,0), \ldots, \overline{F}^{(n-t)}(0,0))$ to be the $n-t$ secrets "committed" by D (if D is honest then these will be the same as $\vec{S}$) and proceed to complete t-sharing of these values by ensuring that each P_j gets its degree t column polynomials $\overline{F}^{(1)}(\alpha_j, y), \ldots, \overline{F}^{(n-t)}(\alpha_j, y)$ and outputs their constant terms as its shares. This is done as follows.

Let $\overline{\mathcal{S}}_{ij} = \{\overline{f}_i^{(1)}(\alpha_j), \ldots, \overline{f}_i^{(n-t)}(\alpha_j)\}$ and $\overline{\mathsf{M}}_{ij} = \{(\overline{m}_i^{(1,1)}(\alpha_j), \ldots, \overline{m}_i^{(1,n-t)}(\alpha_j)), (\overline{m}_i^{(2,1)}(\alpha_j), \ldots, \overline{m}_i^{(2,n-t)}(\alpha_j))\}$ denote the sets received by P_i at the end of Distr_{ij}. By the properties of VCORE, each honest $P_i \in \mathsf{VCORE}$ will be able to give a proof of possession of $\overline{\mathcal{S}}_{ij} \cup \overline{\mathsf{M}}_{ij}$, as the corresponding $\mathsf{AuthVal}_{ij}$ instance would not be aborted by D. Hence if P_i transfers these sets to P_j, then even P_j can give a proof of possession of these sets. So Each P_i (in VCORE) [1] sends the set $\overline{\mathsf{S}}_{ij} \cup \overline{\mathsf{M}}_{ij}$ to P_j, who then publicly verifies these values by executing an instance $\mathsf{RevealPoP}_{ji}$ of $\mathsf{RevealPoP}$ and giving a proof of possession of these sets of values. Party P_j ensures that the *same* randomness e_j is used in all the $\mathsf{RevealPoP}_{ji}$ instances. Let sup_j denote the set of parties P_i from VCORE, such that in the corresponding $\mathsf{RevealPoP}_{ji}$ instance the output is $\mathtt{AcceptProof}$, along with a set of $n-t$ linearly combined values, say $(\mathsf{comb}_{ji}^{(1)}, \ldots, \mathsf{comb}_{ji}^{(n-t)})$ (recall that now the instances of $\mathsf{RevealPoP}$ are executed with $\mathsf{pck} = n-t$ and so $n-t$ linearly combined values will be produced in these instances). If D is *honest* then with high probability, only the parties sending the correct $\overline{\mathsf{S}}_{ij} \cup \overline{\mathsf{M}}_{ij}$ sets will be present in sup_j. However if D is *corrupted* then a corrupted P_i can send incorrect sets and still be present in sup_j. To check this, it is publicly verified if the sets of values $\{(\alpha_i, \mathsf{comb}_{ji}^{(1)})\}_{P_i \in \mathrm{sup}_j}, \ldots, \{(\alpha_i, \mathsf{comb}_{ji}^{(n-t)})\}_{P_i \in \mathrm{sup}_j}$ lie on $n-t$ univariate polynomials of degree at most t. If so then it ensures that with high probability, the parties in sup_j sent the correct sets to P_j. This is because the values in $\overline{\mathsf{S}}_{ij} \cup \overline{M}_{ij}$ corresponding to the *honest* parties in sup_j clearly define degree t column polynomials $\overline{F}^{(1)}(\alpha_j, y), \ldots, \overline{F}^{(n-t)}(\alpha_j, y), \overline{M}^{(1,1)}(\alpha_j, y), \ldots, \overline{M}^{(1,n-t)}(\alpha_j, y), \overline{M}^{(2,1)}(\alpha_j, y), \ldots, \overline{M}^{(2,n-t)}(\alpha_j, y)$. Since P_j uses the same combiner e_j to produce the linear combination of the values in $\overline{\mathsf{S}}_{ij} \cup \overline{M}_{ij}$ in all the $\mathsf{RevealPoP}_{ji}$ instances, it follows that the linear combinations $\mathsf{comb}_{ji}^{(1)}, \ldots, \mathsf{comb}_{ji}^{(n-t)}$ of these $\overline{\mathsf{S}}_{ij} \cup \overline{M}_{ij}$ sets also lie on a degree t univariate polynomial; specifically the set of values $\{(\alpha_i, \mathsf{comb}_{ji}^{(k)})\}$ corresponding to the *honest* parties P_i in sup_j will define a degree t univariate polynomial $e_j\overline{M}^{(1,k)}(\alpha_j, y) + e_j^2\overline{M}^{(2,k)}(\alpha_j, y) + e_j^3\overline{F}^{(k)}(\alpha_j, y)$ for $k = 1, \ldots, n-t$. Now if a *corrupted* P_i in sup_j sent an incorrect set to P_j, then with high probability, the corresponding $\mathsf{comb}_{ji}^{(k)}$ values will not lie on the degree t univariate polynomial $e_j\overline{M}^{(1,k)}(\alpha_j, y) + e_j^2\overline{M}^{(2,k)}(\alpha_j, y) + e_j^3\overline{F}^{(k)}(\alpha_j, y)$, in which case D will be discarded. For the ease of understanding, a pictorial representation of the values distributed during Sh to share $n-t$ secrets is shown in Fig. 3.4 of Appendix 3.3.2.

Sharing $\ell \times (n-t)$ Secrets Simultaneously: The principle behind sharing $\ell \times (n-t)$ secrets $\vec{S} = (s^{(1,1)}, \ldots, s^{(1,n-t)}, \ldots, s^{(\ell,1)}, \ldots, s^{(\ell,n-t)})$ will be similar to that of sharing $n-t$ secrets as discussed above. The only difference will be that the S_{ij} sets in the underlying Distr_{ij}, $\mathsf{AuthVal}$ and $\mathsf{RevealPoP}_{ji}$ instances will be of size $\ell \times (n-t)$, instead of $n-t$; the M_{ij} sets will remain the same as above. More specifically, D will now select $\ell \times (n-t)$ secret-carrying random bivariate polynomials of degree t, say $F^{(l,k)}(x, y)$ for $l \in [\ell]$ and $k \in [n-t]$, each embedding a secret from $\vec{S}$ in its constant term; the number of masking polynomials remain $2(n-t)$. Now the $\{(\alpha_i, \mathsf{comb}_{ji}^{(k)})\}$ values corresponding to the *honest* parties P_i in sup_j will define

[1] Even though each P_i sends the corresponding $\overline{\mathsf{S}}_{ij} \cup \overline{\mathsf{M}}_{ij}$ to P_j, party P_j will focus only on the P_is in VCORE

a linear combination of $\ell + 2$ column polynomials $M^{(1,k)}(\alpha_j, y), M^{(2,k)}(\alpha_j, y), F^{(1,k)}(\alpha_j, y), \ldots, F^{(\ell,k)}(\alpha_j, y)$ for $k \in [n-t]$. The rest of the protocol steps remain the same as above. For the ease of understanding, a pictorial representation of the values distributed and communicated during Sh to share $\ell \times (n-t)$ secrets is shown in Fig. 3.5 of Appendix 3.3.2.

The properties of Sh are stated in Theorem 3.4.

Theorem 3.4 *Sh is a four round VSS for $\ell \times (n-t)$ values, with an error probability of $\max\{\frac{n^3(n-1)}{|\mathbb{F}|-(n-t)}, \frac{n^3\ell}{|\mathbb{F}|-1}\}$. The protocol has communication complexity $\mathsf{PC}(\mathcal{O}(n^3\ell))$ and $\mathsf{BC}(\mathcal{O}(n^3))$.*

To avoid repetition, we do not present the complete formal steps of Sh and the detailed proof of its properties. Instead we state the formal properties of Sh which follow in a straight forward fashion from the corresponding properties of $\mathsf{Sh\text{-}Single}$, taking into account that the underlying instances of ICPoP that are executed deal with $\ell \times (n-t)$ values.

Claim 3.5 *If D is honest then except with probability at most $\frac{n^3(n-1)}{|\mathbb{F}|-(n-t)}$, it will not be discarded during Sh.*

Proof: Similar to Claim 3.2, except that now each instance of ICPoP satisfies the ICPoP-**Correctness3** property except with probability at most $\frac{nd}{|\mathbb{F}|-\mathsf{pck}}$, where $\mathsf{pck} = n - t$ and $d = t + \mathsf{pck} - 1 = n - 1$. This ensures that if a corrupted $P_i \in \mathsf{VCORE}$ transfers incorrect values to an honest P_j, then it will be caught in the corresponding $\mathsf{RevealPoP}_{ji}$ instance. And there will be n^2 such instances, involving a corrupted P_i and an honest P_j. □

Lemma 3.4 (Correctness for an honest D) *If D is honest then except with probability at most $\frac{n^3(n-1)}{|\mathbb{F}|-(n-t)}$, the $\ell \times (n-t)$ values $(s^{(1,1)}, \ldots, s^{(1,n-t)}, \ldots, s^{(\ell,1)}, \ldots, s^{(\ell,n-t)})$ will be t-shared at the end of Sh.*

Proof: Similar to Lemma 3.1, except that now we rely on Claim 3.5. □

Claim 3.6 *Let $\overline{f}_i^{(1,1)}(x), \ldots, \overline{f}_i^{(1,n-t)}(x), \ldots, \overline{f}_i^{(\ell,1)}(x), \ldots, \overline{f}_i^{(\ell,n-t)}(x), \overline{m}_i^{(1,1)}(x), \ldots, \overline{m}_i^{(1,n-t)}(x)$ and $\overline{m}_i^{(2,1)}(x), \ldots, \overline{m}_i^{(2,n-t)}(x)$ be the row polynomials defined by the values in $\mathsf{S}_{ij} \cup \mathsf{M}_{ij}$ received by party $P_i \in \mathcal{P}$ from D for $j \in [n]$. If D is corrupted and a VCORE is formed during Sh then except with probability at most $\frac{n^2(\ell+2)(n-t)}{|\mathbb{F}|}$, there exist $(\ell+2)(n-t)$ bivariate polynomials, say $\overline{F}^{(1,1)}(x,y), \ldots, \overline{F}^{(1,n-t)}(x,y), \ldots, \overline{F}^{(\ell,1)}(x,y), \ldots, \overline{F}^{(\ell,n-t)}(x,y), \overline{M}^{(1,1)}(x,y), \ldots, \overline{M}^{(1,n-t)}(x,y)$, $\overline{M}^{(2,1)}(x,y), \ldots, \overline{M}^{(2,n-t)}(x,y)$, each of degree at most t, such that for each honest $P_i \in \mathsf{VCORE}$, the polynomial $\overline{f}_i^{(l,k)}(x)$ lie on $\overline{F}^{(l,k)}(x,y)$ for $l \in [\ell], k \in [n-t]$, the polynomial $\overline{m}_i^{(1,k)}(x)$ lie on $\overline{M}^{(1,k)}(x,y)$ for $k \in [n-t]$ and the polynomial $\overline{m}_i^{(2,k)}(x)$ lie on $\overline{M}^{(2,k)}(x,y)$ for $k \in [n-t]$.*

Proof: Similar to Claim 3.3, except that now we rely on Lemma 3.7 with $L = (\ell + 2)(n - t)$. $\square$

Lemma 3.5 (Correctness for a corrupted D) *If D is corrupted and not discarded during* $\mathsf{Sh\text{-}Single}$*, then there exists $\ell \times (n-t)$ values, say $(\overline{s}^{(1,1)}, \ldots, \overline{s}^{(1,n-t)}, \ldots, \overline{s}^{(\ell,1)}, \ldots, \overline{s}^{(\ell,n-t)})$, such that then except with probability at most $\frac{n^3\ell}{|\mathbb{F}|-1}$, the values $\overline{s}^{(l,k)}$ will be t-shared at the end of* Sh *for $l \in [\ell]$ and $k \in [n-t]$.*

Proof: Similar to Lemma 3.2, except that we now use Claim 3.6. Moreover, for every pair of *honest* parties (P_i, P_j), where $P_i \in \mathsf{VCORE}$, it will be ensured that except with probability at most $\frac{n\ell}{|\mathbb{F}|-1}$, party P_i will be present in $\sup_j$; this follows from Lemma 2.2. As there are $\Theta(n^2)$ such pairs, from the union bound it is ensured that except with probability at most $\frac{n^3\ell}{|\mathbb{F}|-1}$, every honest party from VCORE will be present in the $\sup_j$ set of every honest P_j Furthermore it will be ensured that except with probability at most $\frac{(\ell+1)}{|\mathbb{F}|}$, no *corrupted* party $P_i \in \mathsf{VCORE}$ will be present in $\sup_j$ set of an *honest* P_j; this will follow from Claim 2.3 (by substituting $L = \ell + 1$). As there can be $\mathcal{O}(n^2)$ pairs of parties, from the union bound it follows that except with probability at most $\frac{n^2(\ell+1)}{|\mathbb{F}|}$, the values transferred by the corrupted parties in VCORE to the honest parties will be correct. So overall the error probability will be at most $\frac{n^3\ell}{|\mathbb{F}|-1}$. $\square$

Lemma 3.6 (Privacy) *In protocol* Sh*, the values $(s^{(1,1)}, \ldots, s^{(1,n-t)}, \ldots, s^{(\ell,1)}, \ldots, s^{(\ell,n-t)})$ remain information-theoretically secure.*

Proof of Theorem 3.4 :

The properties of VSS follows from Lemma 3.4-3.6. In the protocol n^2 instances of ICPoP (with $\mathsf{pck} = n - t$) and n instances of $\mathsf{Poly\text{-}Check}$ (each with $L = (\ell + 2)(n - t)$) are executed. The rest follows from the communication complexity of ICPoP (Thorem 2.1) and $\mathsf{Poly\text{-}Check}$ (Lemma 3.7).

3.3 Appendix

3.3.1 Protocol Poly-Check

Protocol $\mathsf{Poly\text{-}Check}$ for the consistency checking of bivariate polynomials is given in Fig. 3.2. The figure shows how the consistency of row polynomials distributed by D is checked under the supervision of a designated verifier V. The inputs for (an honest) D are L secret bivariate polynomials $F^{(1)}(x, y), \ldots, F^{(L)}(x, y)$ of degree at most t and a secret blinding polynomial $B(y)$ of degree at most t. The inputs for (an honest) party P_i are L row polynomials $\overline{f}_i^{(1)}(x), \ldots, \overline{f}_i^{(L)}(x)$ of degree at most t and a share $\overline{b}_i$ of blinding polynomial. If D and P_i are honest then these values are private and $\overline{f}_i^{(k)}(x) = F^{(k)}(x, \alpha_i)$ and $\overline{b}_i = B(\alpha_i)$ will hold for each $k \in [L]$. The properties of Poly-Check are stated in Lemma 3.7; for the proof we refer to [38].

Lemma 3.7 (Properties of Protocol Poly-Check) *In protocol* Poly-Check, *the following holds:*

- *If* D *is honest then every honest party outputs a* $\mathcal{W}^{(\mathsf{V})}$ *set which includes all the honest parties. Moreover the row polynomials of the honest parties in* $\mathcal{W}^{(\mathsf{V})}$ *will lie on* $F^{(1)}(x, y), \ldots, F^{(L)}(x, y)$ *Furthermore* Adv *gets no additional information about* $F^{(1)}(x, y), \ldots, F^{(L)}(x, y)$ *in the protocol.*

- *If* D *is corrupted and* V *is honest and if the parties output a* $\mathcal{W}^{(\mathsf{V})}$*, then except with probability at most* $\frac{nL}{|\mathbb{F}|}$*, there exists* L *bivariate polynomials, say* $\overline{F}^{(1)}(x, y), \ldots, \overline{F}^{(L)}(x, y)$*, of degree at most* t*, such that row polynomials of the honest parties in* $\mathcal{W}^{(\mathsf{V})}$ *lie on* $\overline{F}^{(1)}(x, y), \ldots, \overline{F}^{(L)}(x, y)$.

- *The protocol requires two rounds and has communication complexity* $\mathsf{BC}(\mathcal{O}(n))$.

Figure 3.2: Polycheck Protocol

$\mathsf{Poly\text{-}Check}(\mathsf{D}, \mathsf{V}, \mathcal{P}, L, \{F^{(1)}(x, y), \ldots, F^{(L)}(x, y), B(y)\}, \{\overline{f}_i^{(1)}(x), \ldots, \overline{f}_i^{(L)}(x), \overline{b}_i\}_{i \in [n]})$

Round 1: Verifier V selects a random combiner $r \in \mathbb{F} \setminus \{0\}$ and broadcasts r.

Round 2: The parties on receiving r from the broadcast of V do the following:

- D broadcasts the polynomial $E(y) \stackrel{\text{def}}{=} B(y) + rg_1^{(1)}(y) + r^2 g_2^{(1)}(y) + \ldots + r^n g_n^{(1)}(y) + r^{(n+1)} g_1^{(2)}(y) + r^{(n+2)} g_2^{(2)}(y) + \ldots + r^{2n} g_n^{(2)}(y) + \ldots + r^{(L-1)n+1} g_1^{(L)}(y) + r^{(L-1)n+2} g_2^{(L)}(y) + \ldots + r^{Ln} g_n^{(L)}(y)$. Here $g_i^{(k)}(y) = F^{(k)}(\alpha_i, y)$ for $k \in [L]$ and $i \in [n]$.

- Each party $P_i \in \mathcal{P}$ (including D) broadcasts the linear combination $e_i \stackrel{\text{def}}{=} \overline{b}_i + r\overline{f}_i^{(1)}(\alpha_1) + r^2 \overline{f}_i^{(1)}(\alpha_2) + \ldots + r^n \overline{f}_i^{(1)}(\alpha_n) + r^{(n+1)} \overline{f}_i^{(2)}(\alpha_1) + r^{(n+2)} \overline{f}_i^{(2)}(\alpha_2) + \ldots + r^{2n} \overline{f}_i^{(2)}(\alpha_n) + \ldots + r^{(L-1)n+1} \overline{f}_i^{(L)}(\alpha_1) + r^{(L-1)n+2} \overline{f}_i^{(L)}(\alpha_2) + \ldots + r^{Ln} \overline{f}_i^{(L)}(\alpha_n)$

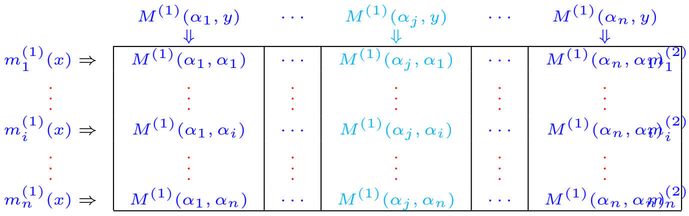
Output determination: If $E(y)$ has degree more than t then each party $P_j \in \mathcal{P}$ outputs $\perp$ and terminate. Else each party $P_j \in \mathcal{P}$ creates a *witness set* $\mathcal{W}^{(\mathsf{V})}$, initialized to $\emptyset$ and then does the following:

- Include party P_i to $\mathcal{W}^{(\mathsf{V})}$ if the relation $E(\alpha_i) \stackrel{?}{=} e_i$ is true.
- If $|\mathcal{W}^{(\mathsf{V})}| \geq 2t + 1$ then P_j outputs $\mathcal{W}^{(\mathsf{V})}$, else P_j outputs $\perp$.

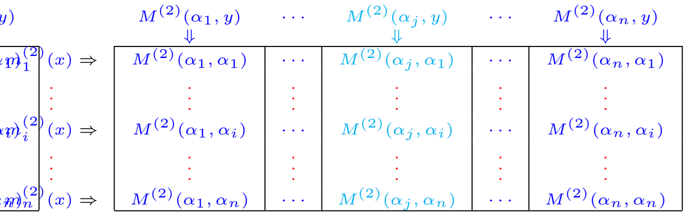
3.3.2 Pictorial Representation of the Protocols

Figure 3.3: Pictorial representation of Sh-Single protocol

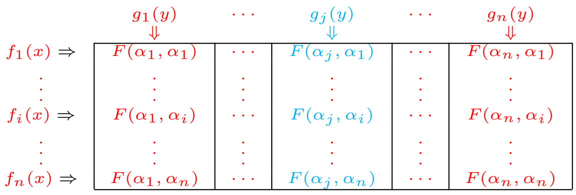
(a) $M^{(1)}(x, y)$ with ith row being possessed by P_i



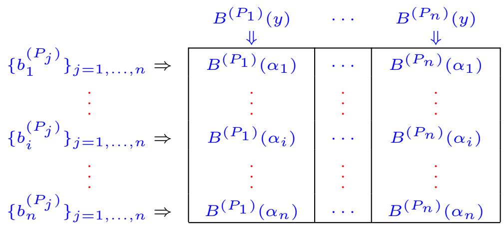
(b) $M^{(2)}(x, y)$ with ith row being possessed by P_i



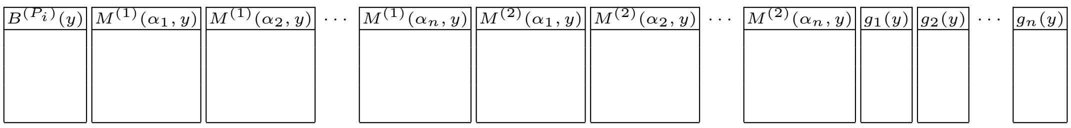
(c) $F(x, y)$ with the ith row being possessed by P_i



(d) Blinding polynomials with ith row being possessed by P_i



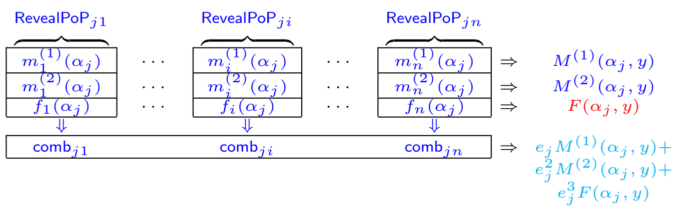
(e) Linear combination of the polynomials that are revealed during $\mathsf{Poly\text{-}Check}^{(P_i)}$



(f) $\mathsf{Distr}_{ij} = \mathsf{Distr}(\mathsf{D}, P_i, \mathcal{P}, 1, 1, \mathcal{S}_{ij} \cup \mathsf{M}_{ij})$ where $\mathcal{S}_{ij} = \{f_i(\alpha_j)\}$ and $\mathsf{M}_{ij} = \{m_i^{(1)}(\alpha_j), m_i^{(2)}(\alpha_j)\}$ for $i, j \in [n]$. Refer to the corresponding figure 2.2a which shows the distribution of values during Distr. We observe that for Distr_{ij}, $\ell = 1$ $\mathsf{pck} = 1$

$$H^{(1)}(x) \Rightarrow \boxed{m_i^{(1)}(\alpha_j)}$$
$$H^{(2)}(x) \Rightarrow \boxed{m_i^{(2)}(\alpha_j)}$$
$$G^{(1)}(x) \Rightarrow \boxed{f_i(\alpha_j)}$$

(g) $\mathsf{RevealPoP}_{ji}$ instances executed by Party P_j corresponding to the parties $P_i \in \mathsf{VCORE}$. The same random combiner e_j is used in all these instances. comb_{ji} denotes the linear combination of values output during $\mathsf{RevealPoP}_{ji}$. This is analogous to figure 2.2b with $\ell = 1, \mathsf{pck} = 1$.



$$\mathsf{comb}_{j1} = e_j m_1^{(1)}(\alpha_j) + e_j^2 m_1^{(2)}(\alpha_j) + e_j^3 f_1(\alpha_j)$$
$$\cdots$$
$$\mathsf{comb}_{ji} = e_j m_i^{(1)}(\alpha_j) + e_j^2 m_i^{(2)}(\alpha_j) + e_j^3 f_i(\alpha_j)$$
$$\cdots$$
$$\mathsf{comb}_{jn} = e_j m_n^{(1)}(\alpha_j) + e_j^2 m_n^{(2)}(\alpha_j) + e_j^3 f_n(\alpha_j)$$

$\{m_1^{(1)}(\alpha_j) \cdots m_n^{(1)}(\alpha_j)\}$ define $M^{(1)}(\alpha_j, y)$ (refer fig 3.3a). Similarly $\{m_1^{(2)}(\alpha_j) \cdots m_n^{(2)}(\alpha_j)\}$ define $M^{(2)}(\alpha_j, y)$ (refer fig 3.3b). $\{f_1(\alpha_j), f_2(\alpha_j) \cdots f_n(\alpha_j)\}$ define $F(\alpha_j, y)$ (refer fig 3.3c). Therefore $e_j M^{(1)}(\alpha_j, y) + e_j^2 M^{(2)}(\alpha_j, y) + e_j^3 F(\alpha_j, y)$ is a t degree polynomial defined by the comb_{ji} values

Figure 3.4: Pictorial representation of Sh protocol that shares $n-t$ secrets

(a) Polynomials $M^{(1,1)}(x,y), \cdots M^{(1,n-t)}(x,y)$ and $M^{(2,1)}(x,y), \cdots M^{(2,n-t)}(x,y)$.

$M^{(1,1)}(x,y)$ $\Downarrow$	$\cdots$	$M^{(1,n-t)}(x,y)$ $\Downarrow$	$M^{(2,1)}(x,y)$ $\Downarrow$	$\cdots$	$M^{(2,n-t)}(x,y)$ $\Downarrow$
$m_1^{(1,1)}(x)$	$\cdots$	$m_1^{(1,n-t)}(x)$	$m_1^{(2,1)}(x)$	$\cdots$	$m_1^{(2,n-t)}(x)$
$\vdots$		$\vdots$	$\vdots$		$\vdots$
$m_i^{(1,1)}(x)$	$\cdots$	$m_i^{(1,n-t)}(x)$	$m_i^{(2,1)}(x)$	$\cdots$	$m_i^{(2,n-t)}(x)$
$\vdots$		$\vdots$	$\vdots$		$\vdots$
$m_n^{(1,1)}(x)$	$\cdots$	$m_n^{(1,n-t)}(x)$	$m_n^{(2,1)}(x)$	$\cdots$	$m_n^{(2,n-t)}(x)$

(b) Polynomial $F^{(k)}(x,y)$ where $k \in [n-t]$.

$F^{(1)}(x,y)$ $\Downarrow$	$\cdots$	$F^{(n-t)}(x,y)$ $\Downarrow$
$f_1^{(1)}(x)$	$\cdots$	$f_1^{(n-t)}(x)$
$\vdots$		$\vdots$
$f_i^{(1)}(x)$	$\cdots$	$f_i^{(n-t)}(x)$
$\vdots$		$\vdots$
$f_n^{(1)}(x)$	$\cdots$	$f_n^{(n-t)}(x)$

(c) Closer look at $M^{(1,k)}(x,y)$ with party P_i holding the ith row

	$M^{(1,k)}(\alpha_1,y)$ $\Downarrow$	$\cdots$	$M^{(1,k)}(\alpha_j,y)$ $\Downarrow$	$\cdots$	$M^{(1,k)}(\alpha_n,y)$ $\Downarrow$
$m_1^{(1,k)}(x) \Rightarrow$	$M^{(1,k)}(\alpha_1,\alpha_1)$	$\cdots$	$M^{(1,k)}(\alpha_j,\alpha_1)$	$\cdots$	$M^{(1,k)}(\alpha_n,\alpha_1)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$m_i^{(1,k)}(x) \Rightarrow$	$M^{(1,k)}(\alpha_1,\alpha_i)$	$\cdots$	$M^{(1,k)}(\alpha_j,\alpha_i)$	$\cdots$	$M^{(1,k)}(\alpha_n,\alpha_i)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$m_n^{(1,k)}(x) \Rightarrow$	$M^{(1,k)}(\alpha_1,\alpha_n)$	$\cdots$	$M^{(1,k)}(\alpha_j,\alpha_n)$	$\cdots$	$M^{(1,k)}(\alpha_n,\alpha_n)$

(d) Closer look at $M^{(2,k)}(x,y)$ P_i holding the ith row

	$M^{(2,k)}(\alpha_1,y)$ $\Downarrow$	$\cdots$	$M^{(2,k)}(\alpha_j,y)$ $\Downarrow$	$\cdots$	$M^{(2,k)}(\alpha_n,y)$ $\Downarrow$
$m_1^{(2,k)}(x) \Rightarrow$	$M^{(2,k)}(\alpha_1,\alpha_1)$	$\cdots$	$M^{(2,k)}(\alpha_j,\alpha_1)$	$\cdots$	$M^{(2,k)}(\alpha_n,\alpha_1)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$m_i^{(2,k)}(x) \Rightarrow$	$M^{(2,k)}(\alpha_1,\alpha_i)$	$\cdots$	$M^{(2,k)}(\alpha_j,\alpha_i)$	$\cdots$	$M^{(2,k)}(\alpha_n,\alpha_i)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$m_n^{(2,k)}(x) \Rightarrow$	$M^{(2,k)}(\alpha_1,\alpha_n)$	$\cdots$	$M^{(2,k)}(\alpha_j,\alpha_n)$	$\cdots$	$M^{(2,k)}(\alpha_n,\alpha_n)$

(e) Closer look at $F^{(k)}(x,y)$ with party P_i holding the ith row

	$g_1^{(k)}(y)$ $\Downarrow$	$\cdots$	$g_j^{(k)}(y)$ $\Downarrow$	$\cdots$	$g_n^{(k)}(y)$ $\Downarrow$
$f_1^{(k)}(x) \Rightarrow$	$F^{(k)}(\alpha_1,\alpha_1)$	$\cdots$	$F^{(k)}(\alpha_j,\alpha_1)$	$\cdots$	$F^{(k)}(\alpha_n,\alpha_1)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$f_i^{(k)}(x) \Rightarrow$	$F^{(k)}(\alpha_1,\alpha_i)$	$\cdots$	$F^{(k)}(\alpha_j,\alpha_i)$	$\cdots$	$F^{(k)}(\alpha_n,\alpha_i)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$f_n^{(k)}(x) \Rightarrow$	$F^{(k)}(\alpha_1,\alpha_n)$	$\cdots$	$F^{(k)}(\alpha_j,\alpha_n)$	$\cdots$	$F^{(k)}(\alpha_n,\alpha_n)$

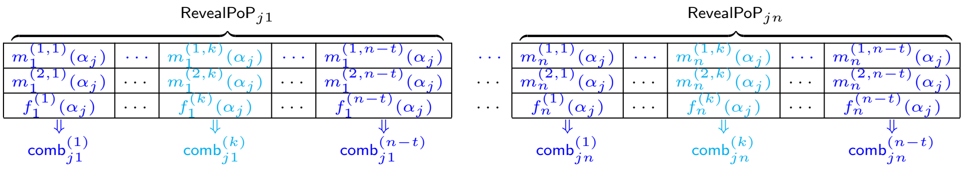
(f) Blinding polynomials with party P_i holding the ith row

	$B^{(P_1)}(y)$ $\Downarrow$	$\cdots$	$B^{(P_n)}(y)$ $\Downarrow$
$\{b_1^{(P_j)}\}_{j=1,\ldots,n} \Rightarrow$	$B^{(P_1)}(\alpha_1)$	$\cdots$	$B^{(P_n)}(\alpha_1)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\{b_i^{(P_j)}\}_{j=1,\ldots,n} \Rightarrow$	$B^{(P_1)}(\alpha_i)$	$\cdots$	$B^{(P_n)}(\alpha_i)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\{b_n^{(P_j)}\}_{j=1,\ldots,n} \Rightarrow$	$B^{(P_1)}(\alpha_n)$	$\cdots$	$B^{(P_n)}(\alpha_n)$

(g) $\mathsf{Distr}_{ij} = \mathsf{Distr}(\mathsf{D}, P_i, \mathcal{P}, 1, (n-t), \mathcal{S}_{ij} \cup \mathsf{M}_{ij})$ where $\mathcal{S}_{ij} = \{f_i^{(1)}(\alpha_j), \ldots, f_i^{(n-t)}(\alpha_j)\}$ and $\mathsf{M}_{ij} = \{(m_i^{(1,1)}(\alpha_j), \ldots, m_i^{(1,n-t)}(\alpha_j)), (m_i^{(2,1)}(\alpha_j), \ldots, m_i^{(2,n-t)}(\alpha_j))\}$ for $i, j \in [n]$.This is similar to the figure 3.3f with $\mathsf{pck} = (n-t)$.

$H^{(1)}(x) \Rightarrow$	$m_i^{(1,1)}(\alpha_j)$	$m_i^{(1,2)}(\alpha_j)$	$\cdots$	$m_i^{(1,n-t)}(\alpha_j)$
$H^{(2)}(x) \Rightarrow$	$m_i^{(2,1)}(\alpha_j)$	$m_i^{(2,2)}(\alpha_j)$	$\cdots$	$m_i^{(2,n-t)}(\alpha_j)$
$G^{(1)}(x) \Rightarrow$	$f_i^{(1)}(\alpha_j)$	$f_i^{(2)}(\alpha_j)$	$\cdots$	$f_i^{(n-t)}(\alpha_j)$

(h) $\mathsf{RevealPoP}_{ji}$ instances executed by Party P_j corresponding to the parties $P_i \in \mathsf{VCORE}$. The same random combiner e_j is used in all these instances. $\mathsf{comb}_{ji}^{(k)}$ denotes the linear combination of values revealed in these instances for $k \in [n-t]$. This is analogous to figure 3.3g with $\ell = 1, \mathsf{pck} = n - t$.

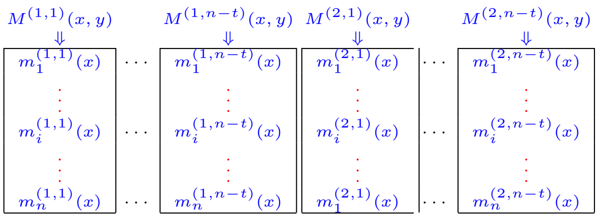


$$\mathsf{comb}_{ji}^{(k)} = e_j m_1^{(1,k)}(\alpha_j) + e_j^2 m_1^{(2,k)}(\alpha_j) + e_j^3 f_1^{(k)}(\alpha_j)$$
$$\mathsf{comb}_{j2}^{(k)} = e_j m_2^{(1,k)}(\alpha_j) + e_j^2 m_2^{(2,k)}(\alpha_j) + e_j^3 f_2^{(k)}(\alpha_j)$$
$$\cdots$$
$$\mathsf{comb}_{jn}^{(k)} = e_j m_n^{(1,k)}(\alpha_j) + e_j^2 m_n^{(2,k)}(\alpha_j) + e_j^3 f_n^{(k)}(\alpha_j)$$

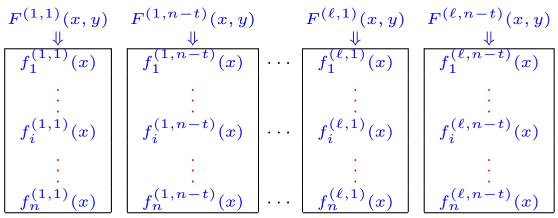
Note from figure 3.5c that $\{m_1^{(1,k)}(\alpha_j), m_2^{(1,k)}(\alpha_j) \cdots m_n^{(1,k)}(\alpha_j)\}$ define $M^{(1,k)}(\alpha_j, y)$. From figure 3.5d, $\{m_1^{(2,k)}(\alpha_j), \cdots m_n^{(2,k)}(\alpha_j)\}$ define $M^{(2,k)}(\alpha_j, y)$. Also from figure 3.5e, $\{f_1^{(k)}(\alpha_j), f_2^{(k)}(\alpha_j) \cdots f_n^{(k)}(\alpha_j)\}$ define $F^{(k)}(\alpha_j, y)$ where $k \in [n-t]$. Hence, the combination i.e $e_j M^{(1.k)}(\alpha_j, y) + e_j^2 M^{(2,k)}(\alpha_j, y) + e_j^3 F^{(k)}(\alpha_j, y)$ is a univariate t-degree polynomial defined by the $\mathsf{comb}_{ji}^{(k)}$ values

Figure 3.5: Pictorial representation of Sh protocol that shares $\ell \times (n-t)$ secrets

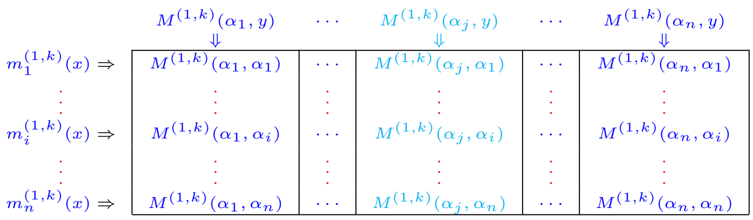
(a) Polynomials $M^{(1,1)}(x,y), \cdots M^{(1,n-t)}(x,y)$ and $M^{(2,1)}(x,y), \cdots M^{(2,n-t)}(x,y)$



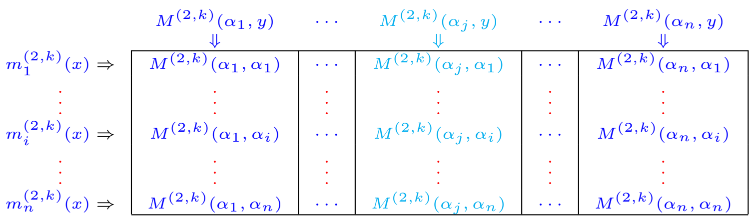
(b) $\ell \times (n-t)$ secret-carrying polynomials $F^{(l,k)}(x,y)$ where $l \in [\ell], k \in [n-t]$. This is analogous to figure 3.4b



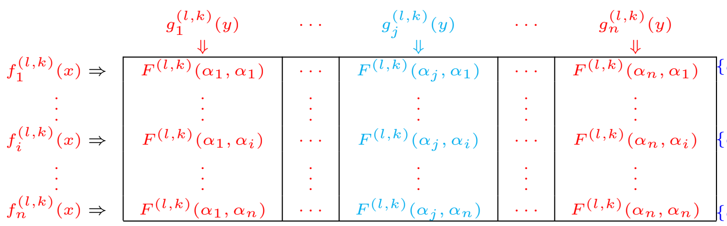
(c) Closer look at $M^{(1,k)}(x,y)$ with party P_i holding the ith row



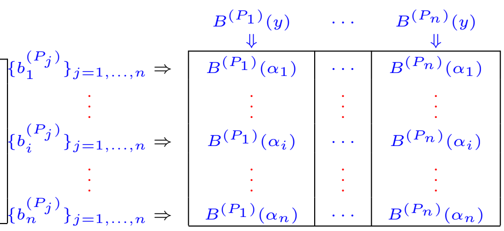
(d) Closer look at $M^{(2,k)}(x,y)$ with party P_i holding the ith row



(e) Closer look at $F^{(l,k)}(x,y)$ with party P_i holding the ith row



(f) Blinding polynomials with party P_i holding the ith row



(g) $\mathsf{Distr}_{ij} = \mathsf{Distr}(\mathsf{D}, P_i, \mathcal{P}, \ell, (n-t), \mathcal{S}_{ij} \cup \mathsf{M}_{ij})$ where $\mathcal{S}_{ij} = \{(f_i^{(1,1)}(\alpha_j), \ldots, f_i^{(1,n-t)}(\alpha_j)), \ldots, (f_i^{(\ell,1)}(\alpha_j), \ldots, f_i^{(\ell,n-t)}(\alpha_j))\}$ and $\mathsf{M}_{ij} = \{(m_i^{(1,1)}(\alpha_j), \ldots, m_i^{(1,n-t)}(\alpha_j)), (m_i^{(2,1)}(\alpha_j), \ldots, m_i^{(2,n-t)}(\alpha_j))\}$ for $i,j \in [n]$.This is similar to the figure 3.4g.

$H^{(1)}(x) \Rightarrow$	$m_i^{(1,1)}(\alpha_j)$	$m_i^{(1,2)}(\alpha_j)$	$\cdots$	$m_i^{(1,n-t)}(\alpha_j)$
$H^{(2)}(x) \Rightarrow$	$m_i^{(2,1)}(\alpha_j)$	$m_i^{(2,2)}(\alpha_j)$	$\cdots$	$m_i^{(2,n-t)}(\alpha_j)$
$G^{(1)}(x) \Rightarrow$	$f_i^{(1,1)}(\alpha_j)$	$f_i^{(1,2)}(\alpha_j)$	$\cdots$	$f_i^{(1,n-t)}(\alpha_j)$
$G^{(2)}(x) \Rightarrow$	$f_i^{(2,1)}(\alpha_j)$	$f_i^{(2,2)}(\alpha_j)$	$\cdots$	$f_i^{(2,n-t)}(\alpha_j)$
$\vdots$	$\vdots$	$\cdots$	$\vdots$	
$G^{(\ell)}(x) \Rightarrow$	$f_i^{(\ell,1)}(\alpha_j)$	$f_i^{(\ell,2)}(\alpha_j)$	$\cdots$	$f_i^{(\ell,n-t)}(\alpha_j)$

(h) $\mathsf{RevealPoP}_{ji}$ instances executed by Party P_j corresponding to the parties $P_i \in \mathsf{VCORE}$. The same random combiner e_j is used for all these instances. $\mathsf{comb}_{ji}^{(k)}$ denotes the linear combination of values revealed in the instance $\mathsf{RevealPoP}_{ji}$ for $k \in [n-t]$.

$\overbrace{\qquad}^{\mathsf{RevealPoP}_{j1}}$								$\overbrace{\qquad}^{\mathsf{RevealPoP}_{jn}}$			
$m_1^{(1,1)}(\alpha_j)$	$\cdots$	$m_1^{(1,k)}(\alpha_j)$	$\cdots$	$m_1^{(1,n-t)}(\alpha_j)$	$\cdots$	$m_n^{(1,1)}(\alpha_j)$	$\cdots$	$m_n^{(1,k)}(\alpha_j)$	$\cdots$	$m_n^{(1,n-t)}(\alpha_j)$
$m_1^{(2,1)}(\alpha_j)$	$\cdots$	$m_1^{(2,k)}(\alpha_j)$	$\cdots$	$m_1^{(2,n-t)}(\alpha_j)$	$\cdots$	$m_n^{(2,1)}(\alpha_j)$	$\cdots$	$m_n^{(2,k)}(\alpha_j)$	$\cdots$	$m_n^{(2,n-t)}(\alpha_j)$
$f_1^{(1,1)}(\alpha_j)$	$\cdots$	$f_1^{(1,k)}(\alpha_j)$	$\cdots$	$f_1^{(1,n-t)}(\alpha_j)$	$\cdots$	$f_n^{(1,1)}(\alpha_j)$	$\cdots$	$f_n^{(1,k)}(\alpha_j)$	$\cdots$	$f_n^{(1,n-t)}(\alpha_j)$
$f_1^{(2,1)}(\alpha_j)$	$\cdots$	$f_1^{(2,k)}(\alpha_j)$	$\cdots$	$f_1^{(2,n-t)}(\alpha_j)$	$\cdots$	$f_n^{(2,1)}(\alpha_j)$	$\cdots$	$f_n^{(2,k)}(\alpha_j)$	$\cdots$	$f_n^{(2,n-t)}(\alpha_j)$
$\vdots$		$\vdots$		$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\cdots$	$\vdots$
$f_1^{(\ell,1)}(\alpha_j)$	$\cdots$	$f_1^{(\ell,k)}(\alpha_j)$	$\cdots$	$f_1^{(\ell,n-t)}(\alpha_j)$	$\cdots$	$f_n^{(\ell,1)}(\alpha_j)$	$\cdots$	$f_n^{(\ell,k)}(\alpha_j)$	$\cdots$	$f_n^{(\ell,n-t)}(\alpha_j)$
$\Downarrow$ $\mathsf{comb}_{j1}^{(1)}$		$\Downarrow$ $\mathsf{comb}_{j1}^{(k)}$		$\Downarrow$ $\mathsf{comb}_{j1}^{(n-t)}$		$\Downarrow$ $\mathsf{comb}_{jn}^{(1)}$		$\Downarrow$ $\mathsf{comb}_{jn}^{(k)}$		$\Downarrow$ $\mathsf{comb}_{jn}^{(n-t)}$

$$\begin{aligned}
\mathsf{comb}_{ji}^{(k)} &= e_j m_1^{(1,k)}(\alpha_j) + e_j^2 m_1^{(2,k)}(\alpha_j) + e_j^3 f_1^{(1,k)}(\alpha_j) \cdots e_j^{\ell+2} f_1^{(\ell,k)}(\alpha_j) \\
\mathsf{comb}_{j2}^{(k)} &= e_j m_2^{(1,k)}(\alpha_j) + e_j^2 m_2^{(2,k)}(\alpha_j) + e_j^3 f_2^{(1,k)}(\alpha_j) \cdots e_j^{\ell+2} f_2^{(\ell,k)}(\alpha_j) \\
&\cdots \\
\mathsf{comb}_{jn}^{(k)} &= e_j m_n^{(1,k)}(\alpha_j) + e_j^2 m_n^{(2,k)}(\alpha_j) + e_j^3 f_n^{(1,k)}(\alpha_j) + \cdots e_j^{\ell+2} f_n^{(\ell,k)}(\alpha_j)
\end{aligned}$$

Note from figure 3.5c that $\{m_1^{(1,k)}(\alpha_j), m_2^{(1,k)}(\alpha_j) \cdots m_n^{(1,k)}(\alpha_j)\}$ define $M^{(1,k)}(\alpha_j, y)$. Also from figure 3.5d, the values $\{m_1^{(2,k)}(\alpha_j), \cdots m_n^{(2,k)}(\alpha_j)\}$ define $M^{(2,k)}(\alpha_j, y)$. Note from figure 3.5e that for $l \in [\ell]$, $\{f_1^{(l,k)}(\alpha_j), f_2^{(l,k)}(\alpha_j) \cdots f_n^{(l,k)}(\alpha_j)\}$ define $F^{(l,k)}(\alpha_j, y)$. Hence, the combination $e_j M^{(1,k)}(\alpha_j, y) + e_j^2 M^{(2,k)}(\alpha_j, y) + e_j^3 F^{(1,k)}(\alpha_j, y) + \cdots + e_j^{\ell+2} F^{(\ell,k)}(\alpha_j, y)$ is a univariate t-degree polynomial defined by the $\mathsf{comb}_{ji}^{(k)}$ values.

Chapter 4

Statistical Multiparty Computation in Hybrid Network

4.1 Design of MPC Protocol

Using Sh, we design a statistical MPC protocol in the partially synchronous setting using the efficient framework of [17] by executing the following two modules: **(1) Module I (Verifiably sharing multiplication triples)**: This module allows a dealer D to verifiably t-share multiplication triples of the form (a, b, c), where $c = ab$. Specifically using Sh, D t-shares "several" triples. To verify whether the shared triples are indeed multiplication triples, we execute additional sub-protocols presented in [17], which can be executed asynchronously. If D is *honest*, then the shared triples remain private during the verification process. **(2) Module II (Extracting multiplication triples)**: The module takes input a set of multiplication triples shared by the individual parties, where the triples shared by the honest parties are random and private. It then executes an asynchronous protocol and outputs a set of t-shared random and private multiplication triples. Combining the above two modules, we get a partially synchronous offline phase protocol to generate t-sharing of $c_M + c_R$ random and private multiplication triples. The inputs of the parties for the computation are shared in parallel by executing instances of Sh. After this the circuit C is securely evaluated asynchronously in a t-shared fashion using the standard Beaver multiplication triple based technique [3, 4, 6, 17]. So overall we get Theorem 4.1.

Theorem 4.1 *Assuming that the first four communication rounds are synchronous broadcast rounds after which the entire communication is asynchronous, there exists a statistical MPC protocol to securely compute f, provided $|\mathbb{F}| \geq 4n^4(c_M + c_R)(3t+1)2^\kappa$ for a given error parameter*

κ. The protocol has communication complexity $\mathsf{PC}(\mathcal{O}(n^2(c_M + c_R) + n^4))$ *and* $\mathsf{BC}(\mathcal{O}(n^4))$.

4.2 Tools used in constructing the MPC

Before proving the Theorem 4.1, we look at some known concepts.

Asynchronous Communication Setting: We first briefly recall the asynchronous communication setting from [13, 17]. In the asynchronous model, the channels are asynchronous and messages can be arbitrarily (but finitely) delayed. The only guarantee here is that the messages sent by the honest parties will eventually reach to their destinations. The order of the message delivery is decided by a *scheduler*. To model the worst case scenario, we assume that the scheduler is under the control of the adversary. The scheduler can only schedule the messages exchanged between the honest parties, without having access to the "contents" of these messages. Designing protocol in the asynchronous setting is complicated and this stems from the fact that we cannot distinguish between a corrupted sender (who does not send any messages) and a slow but honest sender (whose messages are arbitrarily delayed). Due to this at any stage of an asynchronous protocol, no (honest) party can afford to receive communication from *all* the n parties, as this may turn out to require an endless wait. So as soon as a party listens from $n-t$ parties, it has to proceed to the next stage; but in this process, communication from t potentially honest parties may be ignored.

4.2.1 Existing Asynchronous Primitives

The following asynchronous primitives are well known.

Private Reconstruction of t-shared Values: Let $[v]_t$ be a t-sharing of v, shared through a polynomial $p(\cdot)$ of degree at most t. The goal is to make some designated party $P_R \in \mathcal{P}$ to reconstruct v in an asynchronous fashion. The well-know *online error correction* (OEC) algorithm [9, 13] allows P_R to reconstruct $p(\cdot)$ and thus v, as $p(0) = v$. We denote the protocol as $\mathsf{OEC}(P_R, [v])$, whose properties are stated in Lemma 4.1.

Lemma 4.1 ([13, 5, 38, 17]) *Let v be a value which is t-shared among the parties through a polynomial $p(\cdot)$ of degree at most t. Then for every possible* Adv *and for every possible scheduler, protocol* OEC *achieves the following in the asynchronous setting:*

(1) Termination: *Every honest party eventually terminates the protocol.* **(2) Correctness**: *Party P_R outputs $p(\cdot)$ and v.* **(3) Privacy**: *If P_R is honest then* Adv *obtains no additional information about v.* **(4) Communication Complexity**: *The protocol has communication complexity* $\mathsf{PC}(\mathcal{O}(n))$

Multiplication of Pairs of t-shared Values using Beaver's Technique: Beaver's circuit randomization method [3] is a well known method for securely computing $[x \cdot y]_t$ from $[x]_t$ and $[y]_t$, at the expense of two *public reconstructions*, using a *pre-computed* t-shared random multiplication triple (from the offline phase), say $([a]_t, [b]_t, [c]_t)$. For this, the parties first (locally) compute $[e]_t$ and $[d]_t$, where $[e]_t \stackrel{\text{def}}{=} [x]_t - [a]_t = [x - a]_t$ and $[d]_t \stackrel{\text{def}}{=} [y]_t - [b]_t = [y - b]_t$, followed by the public reconstruction of $e = (x - a)$ and $d = (y - b)$; to do the public reconstruction $2n$ instances of OEC are executed, two on the behalf of each party. Since the relation $xy = ((x-a)+a)((y-b)+b) = de + eb + da + c$ holds, the parties can locally compute $[xy]_t = de + e[b]_t + d[a]_t + [c]_t$, once d and e are publicly known. The above computation leaks no information about x and y if a and b are random and unknown to Adv. We call the protocol as $\mathsf{Beaver}(([x]_t, [y]_t, [a]_t, [b]_t, [c]_t))$ and state its properties in Lemma 4.2.

Lemma 4.2 ([17]) *Let $([x]_t, [y]_t)$ be a pair of t-sharing and $([a]_t, [b]_t, [c]_t)$ be the t-sharing of multiplication triples unknown to* Adv. *Then for every possible* Adv *and for every possible scheduler, protocol* Beaver *achieves the following in the asynchronous setting:*

(1) Termination: *All honest parties eventually terminate.* **(2) Correctness**: *The parties output $[xy]_t$.* **(3) Privacy**: *The view of* Adv *is distributed independently of x and y.* **(4) Communication Complexity**: *The protocol has communication complexity $\mathsf{PC}(\mathcal{O}(n^2))$.*

4.2.2 The Asynchronous Triple Transformation Protocol

The heart of the efficient framework of [17] for the offline phase is the asynchronous triple transformation protocol $\mathsf{TripTrans}$. The protocol takes as input a set of $(3t+1)$ independent t-shared triples, say $\{([x^{(i)}]_t, [y^{(i)}]_t, [z^{(i)}]_t\}_{i \in [3t+1]}$ and outputs a set of $(3t+1)$ "co-related" t-shared triples, say $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{z}^{(i)})\}_{i \in [3t+1]}$, such that the following holds:

- There exist polynomials, say $X(\cdot), Y(\cdot)$ and $Z(\cdot)$ of degree $\frac{3t}{2}$, $\frac{3t}{2}$ and $3t$ respectively, such that $X(\alpha_i) = \mathbf{x}^{(i)}, Y(\alpha_i) = \mathbf{y}^{(i)}$ and $Z(\alpha_i) = \mathbf{z}^{(i)}$ holds for $i \in [3t+1]$.
- $Z(\cdot) = X(\cdot)Y(\cdot)$ holds if and only if all the input triples are multiplication triples. This further implies that $Z(\cdot) = X(\cdot)Y(\cdot)$ is true if and only if all the $(3t+1)$ input triples are multiplication triples.
- If Adv knows $t' < \frac{3t}{2}$ input triples then Adv learns t' values on $X(\cdot), Y(\cdot)$ and $Z(\cdot)$, implying $\frac{3t}{2} + 1 - t'$ "degree of freedom" on $X(\cdot), Y(\cdot)$ and $Z(\cdot)$. If $t' > \frac{3t}{2}$, then Adv will completely learn $X(\cdot), Y(\cdot)$ and $Z(\cdot)$.

The protocol is inherited from the protocol for the batch verification of the multiplication triples proposed in [12]. The idea is as follows: we assume that the polynomials $X(\cdot)$ and $Y(\cdot)$ are defined by the first and second component of the first $\frac{3t}{2}+1$ input triples. Next we linearly compute $\frac{3t}{2}$ "new" points on the $X(\cdot)$ and $Y(\cdot)$ polynomials. Finally we compute the product of the $\frac{3t}{2}$ new points using Beaver's technique, making use of the remaining $\frac{3t}{2}$ input triples. The polynomial $Z(\cdot)$ is then defined by the $\frac{3t}{2}$ computed products and the third component of the first $\frac{3t}{2}+1$ input triples. To be more specific, we define the polynomial $X(\cdot)$ of degree at most $\frac{3t}{2}$ by setting $X(\alpha_i) = x^{(i)}$ for $i \in [\frac{3t}{2}+1]$ and get $[\mathbf{x}^{(i)}]_t = [X(\alpha_i)]_t = [x^{(i)}]_t$ for $i \in [\frac{3t}{2}+1]$. Following the same logic, we define $Y(\alpha_i) = y^{(i)}$ for $i \in [\frac{3t}{2}+1]$ and get $[\mathbf{y}^{(i)}]_t = [Y(\alpha_i)]_t = [y^{(i)}]_t$ for $i \in [\frac{3t}{2}+1]$. Moreover, we set $Z(\alpha_i) = z^{(i)}$ for $i \in [\frac{3t}{2}+1]$ and get $[\mathbf{z}^{(i)}]_t = [Z(\alpha_i)]_t = [z^{(i)}]_t$ for $i \in [\frac{3t}{2}+1]$. Now for $i \in [\frac{3t}{2}+1, 3t+1]$, we compute $[\mathbf{x}^{(i)}]_t = [X(\alpha_i)]_t$ and $[\mathbf{y}^{(i)}]_t = [Y(\alpha_i)]_t$ which requires only local computation on the t-sharings $\{[\mathbf{x}^{(i)}]_t, [\mathbf{y}^{(i)}]_t\}_{i \in [\frac{3t}{2}+1]}$, as this is computing a linear function. For $i \in [\frac{3t}{2}+1, 3t+1]$, fixing $\mathbf{z}^{(i)}$ to be the same as $z^{(i)}$ will, however, violate the requirement that $Z(\cdot) = X(\cdot)Y(\cdot)$ holds when all the input triples are multiplication triples; this is because for $i \in [\frac{3t}{2}+1, 3t+1]$, $\mathbf{x}^{(i)} = X(\alpha_i) \neq x^{(i)}$ and $\mathbf{y}^{(i)} = Y(\alpha_i) \neq y^{(i)}$ and thus $z^{(i)} = x^{(i)}y^{(i)} \neq \mathbf{x}^{(i)}\mathbf{y}^{(i)}$. Here we resort to the Beaver's technique to find $[\mathbf{z}^{(i)}]_t = [\mathbf{x}^{(i)}\mathbf{y}^{(i)}]_t$ from $[\mathbf{x}^{(i)}]_t$ and $[\mathbf{y}^{(i)}]_t$, using the t-shared triples $\{([x^{(i)}]_t, [y^{(i)}]_t, [z^{(i)}]_t\}_{i \in [\frac{3t}{2}+1, 3t+1]}$. We note that these triples used for the Beaver's technique are never touched before for any computation. The protocol involves $\frac{3t}{2}$ instances of Beaver and has communication complexity $\mathsf{PC}(\mathcal{O}(n^3))$. The protocol can be executed in a completely asynchronous fashion and it will be ensured that every honest party eventually terminates the protocol. This is because the only steps which require interaction among the parties are during the instances of Beaver, which eventually terminate for each honest party. We refer to [17] for the complete formal details of $\mathsf{TripTrans}$.

4.3 The Framework for the Offline Phase

In [17] an efficient framework for the offline phase for generating t-shared random multiplication triples is presented. On a very high level, the framework consists of the following two modules:

Module I — Multiplication Triple Sharing: This module allows a designated dealer D to *verifiably* t-share multiplication triples. By verifiability, it means that the triples are guaranteed to be multiplication triples. Moreover, the triples remain private if D is honest. To achieve this task, the module takes any polynomial based VSS scheme and plug it with the triple transformation protocol $\mathsf{TripTrans}$. In our context, we will use our VSS protocol Sh. The module is executed as follows.

D invokes our four round VSS protocol Sh to verifiably t-share $\mathfrak{l}(3t+1)$ values. So we require that the first four rounds are synchronous broadcast rounds, which ensures that at the end of the fourth round, $\mathfrak{l}(3t+1)$ values are shared by D. After this, the rest of the steps are executed in a completely asynchronous fashion[1]. The values shared by D can be viewed as $\mathfrak{l}$ batches of $3t+1$ triples. Consider a single batch $\{(x^{(i)}, y^{(i)}, z^{(i)})\}_{i\in[3t+1]}$. The correctness property of Sh ensures that the triples will be t-shared among $\mathcal{P}$ at the end of Sh. To check whether the triples are indeed multiplication triples, an instance of the triple transformation protocol $\mathsf{TripTrans}$ is invoked with this set of $(3t+1)$ t-shared triples as input. Let $X(\cdot), Y(\cdot)$ and $Z(\cdot)$ denote the polynomials of degree at most $\frac{3t}{2}, \frac{3t}{2}$ and $3t$ respectively, which guaranteed to exist at the end of the instance of $\mathsf{TripTrans}$. We next use a probabilistic check to verify whether the relation $Z(\cdot) = X(\cdot)Y(\cdot)$ holds by public checking of $Z(\alpha) \stackrel{?}{=} X(\alpha)Y(\alpha)$ for a *random* $\alpha \in \mathbb{F}$; the random α can be generated by any standard technique[2] and we do not bother about the communication complexity of this procedure as it will be invoked only a constant number of times. It is trivial to see that the check will pass for an honest D. For a corrupted D, if the input triples $\{([x^{(i)}]_t, [y^{(i)}]_t, [z^{(i)}]_t\}_{i\in[3t+1]}$ are not multiplication triples, then $Z(\alpha) \neq X(\alpha)Y(\alpha)$ (by the property of $\mathsf{TripTrans}$). Therefore, the probability of a corrupt D passing the check in this scenario can be computed as the probability that $Z(\alpha) = X(\alpha)Y(\alpha)$ holds, even though $Z(\cdot) \neq X(\cdot)Y(\cdot)$. This probability is atmost $\frac{3t}{|\mathbb{F}|}$ for a random α since $Z(\cdot)$ has degree at most $3t$. If D is honest, then through the above check, Adv will learn one point on $X(\cdot), Y(\cdot)$ $Z(\cdot)$ i.e the value of the polynomials at α. However, this still leaves $\frac{3t}{2}$ degree of freedom in these polynomials. So if the verification passes, the parties output $\frac{3t}{2}$ shared triples $\{([a^{(i)}]_t, [b^{(i)}]_t, [c^{(i)}]_t)\}_{i\in[\frac{3t}{2}]}$ on the "behalf" of D, where $a^{(i)} = X(\beta_i), b^{(i)} = Y(\beta_i)$ and $c^{(i)} = Z(\beta_i)$ for $\frac{3t}{2}$ distinct β_i values, distinct from the random α. Thus the multiplication triples $\{([a^{(i)}]_t, [b^{(i)}]_t, [c^{(i)}]_t)\}_{i\in[\frac{3t}{2}]}$ are finally considered to be shared on the "behalf" of D.

The above idea is applied in parallel on all the $\mathfrak{l}$ batches of $3t+1$ t-shared triples and a single random α is used for the probabilistic verification in all the $\mathfrak{l}$ batches. Through each batch $\frac{3t}{2}$ multiplication triples are considered to be shared by D and so overall the parties will get $(\mathfrak{l}\cdot\frac{3t}{2})$ t-shared multiplication triples at the end of the protocol. If D is caught cheating in any of the batches, then the parties discard D and some default $\mathfrak{l}\cdot\frac{3t}{2}$ multiplications triples are considered to be shared on the behalf of D. We call the resultant protocol $\mathsf{TripleSh}$. In $\mathsf{TripleSh}$, D needs to invoke Sh by setting $\ell = \frac{\mathfrak{l}(3t+1)}{n-t}$. This will ensure that D shares $\ell \times (n-t) = \mathfrak{l}(3t+1)$ triples, which when underwent through $\mathsf{TripTrans}$ and probabilistic check result in $(\mathfrak{l}\cdot\frac{3t}{2})$ multiplication triples being shared on the behalf of D.

The communication complexity of $\mathsf{TripleSh}$ will be $\mathsf{PC}(\mathcal{O}(n^3\mathfrak{l}))$ and $\mathsf{BC}(\mathcal{O}(n^3))$, which is

[1]We note that in [17] this module is designed to work in a completely asynchronous fashion, but with $t < n/4$. Since we are in the $t < n/3$ setting and want to use our VSS protocol Sh, we require the first four rounds to be synchronous broadcast rounds.

[2]For example, each party P_i can t-share a random $r^{(i)}$ and then we can set $[\alpha]_t \stackrel{\text{def}}{=} [r^{(1)}]_t + \ldots + [r^{(n)}]_t$. This will be followed by publicly reconstructing α using OEC. We call this protocol as $\mathsf{Rand}()$.

computed as follows: the instance of Sh will have communication complexity $\mathsf{PC}(\mathcal{O}(n^3\ell))$ and $\mathsf{BC}(\mathcal{O}(n^3))$ (see Theorem 3.4). Substituting $\ell = \frac{\mathfrak{l}(3t+1)}{n-t}$ and $n - t = 2t + 1 = \Theta(n)$, this gives $\mathsf{PC}(\mathcal{O}(n^3\mathfrak{l}))$ and $\mathsf{BC}(\mathcal{O}(n^3))$. There will be $\mathfrak{l}$ instances of $\mathsf{TripTrans}$, each having communication complexity $\mathsf{PC}(\mathcal{O}(n^3))$, so contributing $\mathsf{PC}(\mathcal{O}(n^3\mathfrak{l}))$ to the communication complexity. The error probability of $\mathsf{TripleSh}$ is computed as follows. By setting $\ell = \frac{\mathfrak{l}(3t+1)}{n-t}$ in Theorem 3.4 we find that except with probability at most $\max\{\frac{n^3(n-1)}{|\mathbb{F}|-(n-t)}, \frac{n^3\mathfrak{l}(3t+1)}{|\mathbb{F}|(n-t)}\}$, the values shared by D will be t-shared. Given that the values shared by D are t-shared, the probabilistic check ensures that except with probability at most $\mathfrak{l} \cdot \frac{3t}{2}$, the outputs values obtained on the behalf of D are indeed multiplication triples (there are $\mathfrak{l}$ batches and each batch can pass the probabilistic check with probability at most $\frac{3t}{2}$). So it follows that except with probability $\mathfrak{l} \cdot \frac{3t}{2} + \max\{\frac{n^3(n-1)}{|\mathbb{F}|-(n-t)}, \frac{n^3\mathfrak{l}(3t+1)}{|\mathbb{F}|(n-t)}\} \approx \max\{\frac{n^3(n-1)}{|\mathbb{F}|-(n-t)}, \frac{n^3\mathfrak{l}(3t+1)}{|\mathbb{F}|(n-t)}\}$, the parties output t-shared multiplication triples. The protocol will eventually terminate for each honest party: the instance of Sh will terminate, assuming that the first four communication rounds are synchronous broadcast rounds. Once Sh terminates, the instances of $\mathsf{TripTrans}$ which are executed asynchronously eventually terminate for each honest party We refer to [17] for the formal details of $\mathsf{TripleSh}$. For completeness, we state the properties of $\mathsf{TripleSh}$ in Lemma 4.3, whose proof follows from the above discussion; for a detailed proof see [17].

Lemma 4.3 *Given a partially synchronous communication setting where the first four rounds are synchronous broadcast rounds, protocol* $\mathsf{TripleSh}$ *achieves the following for every possible* Adv *and for every possible scheduler*

(1) Termination: *Irrespective of* D, *every honest party eventually terminates the protocol.* **(2) Correctness**: *If* D *is honest then* $\mathfrak{l} \cdot \frac{3t}{2}$ *multiplication triples will be t-shared. If* D *is corrupted then* $\mathfrak{l} \cdot \frac{3t}{2}$ *triples will be t-shared; moreover except with probability at most* $\max\{\frac{n^3(n-1)}{|\mathbb{F}|-(n-t)}, \frac{n^3\mathfrak{l}(3t+1)}{|\mathbb{F}|(n-t)}\}$, *the triples will be multiplication triples.* **(3) Privacy**: *If* D *is honest, then the view of* Adv *in the protocol is distributed independently of the output multiplication triples.* **(4) Communication Complexity**: *The protocol has communication complexity* $\mathsf{PC}(\mathcal{O}(n^3\mathfrak{l}))$ *and* $\mathsf{BC}(\mathcal{O}(n^3))$. *Additionally one invocation to* Rand *is required.*

Module II : Multiplication Triple Extraction. The second module of the efficient framework of [17] is an asynchronous protocol $\mathsf{TripExt}$. The input to the protocol is a set of $3t + 1$ t-shared multiplication triples, where the ith triple is selected by the party P_i. It will be ensured that if P_i is *honest*, then the ith triple is random and will be private. The protocol outputs a set of $\frac{t}{2} = \Theta(n)$ t-shared multiplications triples, each of which is random and unknown to Adv. The high level idea of $\mathsf{TripExt}$ is as follows: the input triples are first transformed using $\mathsf{TripTrans}$ to obtain a new set of t-shared $3t + 1$ triples. Let $X(\cdot), Y(\cdot)$ and $Z(\cdot)$ be the underlying polynomials associated with the transformed triples. It follows from the correctness of $\mathsf{TripTrans}$ that $Z(\cdot) = X(\cdot)Y(\cdot)$ holds, since the input triples are guaranteed to be multiplication triples. Also, since Adv may know at most t input triples, by the property of $\mathsf{TripTrans}$, it learns at most t points on $X(\cdot), Y(\cdot)$ and $Z(\cdot)$, leaving $\frac{3t}{2} - t = \frac{t}{2}$ degree of freedom on these polynomials. So the parties output $\{([X(\beta_i)]_t, [Y(\beta_i)]_t, [Z(\beta_i)]_t)\}_{i \in [\frac{t}{2}]}$, which can be computed as a linear function of the transformed triples. These triples are considered to be securely "extracted" from the set of input triples. The protocol will eventually terminate for each honest party, as interaction among the parties is required only during the instance of $\mathsf{TripTrans}$, which eventually terminates for each honest party. As one instance of $\mathsf{TripTrans}$ is involved, protocol $\mathsf{TripExt}$ has communication complexity $\mathsf{PC}(\mathcal{O}(n^3))$. For completeness the properties of $\mathsf{TripExt}$ are stated in Lemma 4.4, which follows from the above discussion; for a detailed proof we refer to [17].

Lemma 4.4 *Let $\{(x^{(i)}, y^{(i)}, z^{(i)})\}_{i \in [3t+1]}$ be a set of multiplication triples, where party $P_i \in \mathcal{P}$ has verifiably t-shared the triple $(x^{(i)}, y^{(i)}, z^{(i)})$. Then for every possible* Adv *and for every possible scheduler, protocol* $\mathsf{TripExt}$ *achieves the following in a completely asynchronous setting:*

(1) Termination: *All honest parties eventually terminate the protocol.* **(2) Correctness**: *Each of the $\frac{t}{2}$ output triples is a multiplication triple and will be t-shared.* **(3) Privacy**: *The view of* Adv *in the protocol is distributed independently of the output multiplication triples.* **(4) Communication Complexity**: *The protocol has communication complexity* $\mathsf{PC}(\mathcal{O}(n^3))$.

Module I + Module II $\Rightarrow$ Offline phase protocol in the partial synchronous setting. By combining $\mathsf{TripleSh}$ and $\mathsf{TripExt}$, we get an offline phase protocol $\mathsf{Offline}$ in the partial synchronous setting as follows. The goal of $\mathsf{Offline}$ is to generate t-sharing of $c_M + c_R$ random and private multiplication triples.

- Each party P_i acts a D and ensures that $\frac{2(c_M+c_R)}{t}$ random multiplication triples are shared on its behalf. For this, it invokes an instance $\mathsf{TripleSh}_i$ of $\mathsf{TripleSh}$ by setting $\mathfrak{l} = \frac{4(c_M+c_R)}{3t^2}$; this ensures that at the end of $\mathsf{TripleSh}_i$, $\mathfrak{l} \cdot \frac{3t}{2} = \frac{2(c_M+c_R)}{t}$ t-shared multiplication triples are available on the behalf of P_i. This step is executed in a partially synchronous setting, where it is assumed that the first four communication rounds are synchronous broadcast rounds. This is to ensure that all the $\mathsf{TripleSh}$ instances are terminated. From Lemma 4.3, by substituting the value of $\mathfrak{l}$, this step will have total communication complexity $\mathsf{PC}(\mathcal{O}(n^2(c_M + c_R)))$ and $\mathsf{BC}(\mathcal{O}(n^4))$. Additionally there will be one instance of Rand and its output can be used as a challenge across all the n instances of $\mathsf{TripleSh}$ for the verification of the shared triples. By substituting the value of $\mathfrak{l}$ and from the union bound (there are n instances of $\mathsf{TripleSh}$) it follows that at the end of this step, except with probability at most $\frac{4n^4(c_M+c_R)(3t+1)}{3t^2(n-t)|\mathbb{F}|}$, the triples available on the behalf of all the parties are indeed multiplication triples.

- The parties then execute the protocol $\mathsf{TripExt}$ on the multiplication triples obtained at the end of the previous step and securely extract $c_M + c_R$ random and private t-shared multiplication triples. More specifically, the $\frac{2(c_M+c_R)}{t}$ shared triples available on the behalf of each party are considered as $\frac{2(c_M+c_R)}{t}$ batches of $3t + 1$ triples, where the ith batch consists of the ith triple available on the behalf of all $3t + 1$ parties. So each batch is of size $3t + 1$. For every batch, the triples contributed by the honest parties will be random and private. So by applying an instance of $\mathsf{TripExt}$, the parties can extract $\frac{t}{2}$ random and private t-shared multiplication triples. For each batch an instance of $\mathsf{TripExt}$ is executed and so from $\frac{2(c_M+c_R)}{t}$ batches, the parties will get total $c_M + c_R$ random and private t-shared multiplication triples. This step is executed in a completely asynchronous fashion and it will eventually terminate for each honest party, as the underlying instances of $\mathsf{TripExt}$ will eventually terminate. As there will be $\frac{2(c_M+c_R)}{t}$ instances of $\mathsf{TripExt}$ involved, from Lemma 4.4, this step will have total communication complexity $\mathsf{PC}(\mathcal{O}(n^2(c_M+c_R)))$.

For completeness the properties of $\mathsf{Offline}$ are stated in Lemma 4.5, which follows from the above discussion; for a detailed proof we refer to [17].

Lemma 4.5 *Assuming that the first four communication rounds are synchronous broadcast rounds, protocol* $\mathsf{Offline}$ *achieves the following for every possible* Adv *and every possible scheduler:* **(1) Termination***: All honest parties eventually terminate the protocol.* **(2) Correctness***: The c_M+c_R output triples will be t-shared among the parties. Moreover, the output triples will be multiplication triples, except with probability at most* $\frac{4n^4(c_M+c_R)(3t+1)}{3t^2(n-t)|\mathbb{F}|}$. **(3) Privacy***: The view of* Adv *in the protocol is independent of the output multiplication triples.* **(4) Communication Complexity***: The protocol has communication complexity* $\mathsf{PC}(\mathcal{O}(n^2(c_M + c_R)))$ *and* $\mathsf{BC}(\mathcal{O}(n^4))$. *In addition, one invocation to* Rand *is required.*

4.4 Statistical MPC Protocol in the Partially Synchronous Setting

Our statistical MPC protocol MPC in the partially synchronous setting is straight forward and based on the standard idea of evaluating the circuit in a shared fashion, using the multiplication triplets produced in an offline phase. Specifically in MPC, the parties first execute the protocol $\mathsf{Offline}$ and generate t-sharing of $c_M + c_R$ random and private multiplication triples. For this we assume that the network is partially synchronous and the first four communication rounds are synchronous broadcast round. In parallel, each party P_i t-shares its input x_i for the computation by acting as a dealer D and invoking an instance Sh_i of Sh. These instances of Sh also utilise the first four synchronous broadcast rounds, which are utilized by $\mathsf{Offline}$. Once $\mathsf{Offline}$ is over, the parties will have $c_M + c_R$ t-shared random and private multiplication triplets. In addition, the inputs of all the parties would be available in a t-shared fashion. Next the parties start securely evaluating the circuit *asynchronously* on a gate by gate basis by maintaining the following *invariant* for each gate of the circuit: given t-sharing of the input(s) of a gate, the parties securely compute a t-sharing of the output of the gate. A gate is said to be $\mathsf{evaluated}$ if a t-sharing of the output of the gate is computed. This is achieved as follows for various gates: the linearity of the t-sharing ensures that the linear gates can be evaluated locally. For a multiplication gate, the parties associate a multiplication triple from the set of preprocessed multiplication triples and then evaluate the gate by applying the Beaver's circuit randomization technique, namely by invoking an instance of Beaver. For every random gate in the circuit for generating a random value, the parties associate a multiplication triple from the set of preprocessed multiplication triples and the first component of the triple is considered as the outcome of the random gate. This explains the need for generating $c_M + c_R$ random t-shared multiplication triples in the offline phase (c_M triples corresponding to c_M multiplication gates and c_R triples corresponding to c_R random gates). Once all the gates are evaluated, the t-sharing of the output gate is publicly reconstructed. As this approach for circuit evaluation is standard and used in al most all the recent MPC protocols, we avoid giving the complete formal details of MPC.

If it is ensured that the triples from the offline phase are indeed t-shared and multiplication triples then protocol MPC correctly computes the function f. The probability that the offline phase protocol $\mathsf{Offline}$ fails to generate t-shared multiplication triples is at most $\frac{4n^4(c_M+c_R)(3t+1)}{3t^2(n-t)|\mathbb{F}|}$. So if we ensure that $|\mathbb{F}| \geq 4n^4(c_M + c_R)(3t+1)2^{\kappa}$, then the function will be correctly computed except with an error probability of at most $2^{-\kappa}$. The protocol will achieve the privacy property, intuitively due to the following reason: the inputs of the honest parties remain private as they are t-shared. The intermediate gate outputs remain as private as possible, as they are also t-shared. This intuition can be easily formalized by giving a simulation based security proof using standard arguments (see for example [1]). The offline phase will have communication complexity $\mathsf{PC}(\mathcal{O}(n^2(c_M + c_R)))$ and $\mathsf{BC}(\mathcal{O}(n^4))$. In addition, sharing the inputs of the parties will cost $\mathsf{PC}(\mathcal{O}(n^4))$ and $\mathsf{BC}(\mathcal{O}(n^4))$. The circuit evaluation will have communication complexity $\mathsf{PC}(\mathcal{O}(c_M n^2))$, as there will be c_M instances of Beaver, while publicly reconstructing the circuit output will cost $\mathsf{PC}(\mathcal{O}(n^2))$. This completes the proof of Theorem 4.1.

Chapter 5

Conclusion

The work in this project marks the first attempt in closing the efficiency gap between statistical MPC protocols in synchronous and asynchronous networks. This MPC having communication complexity of $\mathcal{O}(|C|n^2\mu)$ succeeds in bridging the wide efficiency gap between statistical synchronous ($\mathcal{O}(|C|n\mu)$) and asynchronous ($\mathcal{O}(|C|n^5\mu)$) MPC. Here, $|C|$ and μ refer to the circuit size (primarily the number of multiplication gates) and statistical security parameter respectively. Another major contribution during the project is a novel statistical VSS protocol with $t < n/3$. Though the VSS has non-optimal resilience, it is the first protocol to achieve quadratic complexity over point-to-point channels in four rounds. Additionally, the VSS has a very lucrative feature of broadcast complexity being independent of the number of values shared. On the practical front, it is efficient and therefore may be of independent interest. Future work includes leveraging the power of hybrid network design to close the fault-tolerance and efficiency gap between synchronous and asynchronous protocols in different settings.

Bibliography

[1] G. Asharov and Y. Lindell. A Full Proof of the BGW Protocol for Perfectly-Secure Multiparty Computation. *IACR Cryptology ePrint Archive*, 2011:136, 2011. 17, 47

[2] M. Backes, F. Bendun, A. Choudhury, and A. Kate. Asynchronous MPC with a Strict Honest Majority Using Non-equivocation. In M. M. Halldórsson and S. Dolev, editors, *PODC*, pages 10–19. ACM, 2014. 3

[3] D. Beaver. Efficient Multiparty Protocols Using Circuit Randomization. In J. Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15*, volume 576 of *Lecture Notes in Computer Science*, pages 420–432. Springer Verlag, 1991. 39, 41

[4] Z. Beerliová-Trubíniová and M. Hirt. Efficient multi-party computation with dispute control. In S. Halevi and T. Rabin, editors, *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, volume 3876 of *Lecture Notes in Computer Science*, pages 305–328. Springer Verlag, 2006. 6, 39

[5] Z. Beerliová-Trubíniová and M. Hirt. Simple and Efficient Perfectly-Secure Asynchronous MPC. In K. Kurosawa, editor, *Advances in Cryptology - ASIACRYPT 2007, 13th International Conference on the Theory and Application of Cryptology and Information Security, Kuching, Malaysia, December 2-6, 2007, Proceedings*, volume 4833 of *Lecture Notes in Computer Science*, pages 376–392. Springer Verlag, 2007. 3, 40

[6] Z. Beerliová-Trubíniová and M. Hirt. Perfectly-Secure MPC with Linear Communication Complexity. In R. Canetti, editor, *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008*, volume 4948 of *Lecture Notes in Computer Science*, pages 213–230. Springer Verlag, 2008. 3, 6, 39

[7] Zuzana Beerliová-Trubíniová, Martin Hirt, and Jesper Buus Nielsen. Almost-asynchronous MPC with faulty minority. *IACR Cryptology ePrint Archive*, 2008:416, 2008. 4

[8] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract). In J. Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 1–10. ACM, 1988. 2, 3

[9] M. Ben-Or, R. Canetti, and O. Goldreich. Asynchronous Secure Computation. In S. R. Kosaraju, D. S. Johnson, and A. Aggarwal, editors, *STOC*, pages 52–61. ACM, 1993. 40

[10] M. Ben-Or, B. Kelmer, and T. Rabin. Asynchronous Secure Computations with Optimal Resilience (Extended Abstract). In J. H. Anderson, D. Peleg, and E. Borowsky, editors, *Proceedings of the Thirteenth Annual ACM Symposium on Principles of Distributed Computing, Los Angeles, California, USA, August 14-17, 1994*, pages 183–192. ACM, 1994. 2, 3

[11] Michael Ben-Or, Ran Canetti, and Oded Goldreich. Asynchronous secure computation. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 52–61, 1993. 2, 3

[12] E. Ben-Sasson, S. Fehr, and R. Ostrovsky. Near-Linear Unconditionally-Secure Multiparty Computation with a Dishonest Minority. In R. Safavi-Naini and R. Canetti, editors, *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*, pages 663–680. Springer, 2012. 3, 5, 41

[13] R. Canetti. *Studies in Secure Multiparty Computation and Applications.* PhD thesis, Weizmann Institute, Israel, 1995. 17, 40

[14] R. Canetti and T. Rabin. Fast Asynchronous Byzantine Agreement with Optimal Resilience. In *STOC*, pages 42–51, 1993. 4

[15] David Chaum, Ivan Damgård, and Jeroen van de Graaf. Multiparty computations ensuring privacy of each party's input and correctness of the result. In *Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings*, pages 87–119, 1987. 1

[16] A. Choudhury and A. Patra. Optimally Resilient Asynchronous MPC with Linear Communication Complexity. In S. K. Das, D. Krishnaswamy, S. Karkar, A. Korman, M. Kumar, M. Portmann, and S. Sastry, editors, *ICDCN*, pages 5:1–5:10. ACM, 2015. 3

[17] A. Choudhury, M. Hirt, and A. Patra. Asynchronous Multiparty Computation with Linear Communication Complexity. In Y. Afek, editor, *Distributed Computing - 27th International Symposium, DISC 2013, Jerusalem, Israel, October 14-18, 2013. Proceedings*, volume 8205 of *Lecture Notes in Computer Science*, pages 388–402. Springer, 2013. 3, 4, 5, 39, 40, 41, 42, 43, 44, 45, 46

[18] Ashish Choudhury, Emmanuela Orsini, Arpita Patra, and Nigel P. Smart. Linear overhead robust MPC with honest majority using preprocessing. *IACR Cryptology ePrint Archive*, 2015:705, 2015. 3

[19] A. Clement, F. Junqueira, A. Kate, and R. Rodrigues. On the (Limited) Power of Non-equivocation. In D. Kowalski and A. Panconesi, editors, *PODC*, pages 301–308. ACM, 2012. 3

[20] R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin. Efficient Multiparty Computations Secure Against an Adaptive Adversary. In J. Stern, editor, *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 311–326. Springer, 1999. 9

[21] R. Cramer, I. Damgård, and J. B. Nielsen. Multiparty Computation from Threshold Homomorphic Encryption. In B. Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 280–299. Springer, 2001. 3

[22] I. Damgård and J. B. Nielsen. Scalable and Unconditionally Secure Multiparty Computation. In A. Menezes, editor, *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, volume 4622 of *Lecture Notes in Computer Science*, pages 572–590. Springer Verlag, 2007. 6

[23] I. Damgård, M. Geisler, M. Krøigaard, and J. B. Nielsen. Asynchronous Multiparty Computation: Theory and Implementation. In S. Jarecki and G. Tsudik, editors, *PKC*, pages 160–179, 2009. 3

[24] D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly Secure Message Transmission. *J. ACM*, 40(1):17–47, 1993. 4

[25] M. Fitzi and M. Hirt. Optimally Efficient Multi-valued Byzantine Agreement. In E. Ruppert and D. Malkhi, editors, *PODC*, pages 163–168. ACM Press, 2006. 4

[26] M. Fitzi, J. A. Garay, S. Gollakota, C. Pandu Rangan, and K. Srinathan. Round-Optimal and Efficient Verifiable Secret Sharing. In S. Halevi and T. Rabin, editors, *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, volume 3876 of *Lecture Notes in Computer Science*, pages 329–342. Springer, 2006. 19

[27] M. K. Franklin and M. Yung. Communication Complexity of Secure Computation (Extended Abstract). In S. R. Kosaraju, M. Fellows, A. Wigderson, and J. A. Ellis, editors, *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada*, pages 699–710. ACM, 1992. 20, 21

[28] J. A. Garay, C. Givens, R. Ostrovsky, and P. Raykov. Broadcast (and Round) Efficient Verifiable Secret Sharing. In C. Padró, editor, *Information Theoretic Security - 7th International Conference, ICITS 2013, Singapore, November 28-30, 2013, Proceedings*, volume 8317 of *Lecture Notes in Computer Science*, pages 200–219. Springer, 2013. 5

[29] R. Gennaro, Y. Ishai, E. Kushilevitz, and T. Rabin. The Round Complexity of Verifiable Secret Sharing and Secure Multicast. In J. S. Vitter, P. G. Spirakis, and M. Yannakakis, editors, *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 580–589. ACM, 2001. 4, 19

[30] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 218–229, 1987. 1

[31] M. Hirt and J. B. Nielsen. Robust Multiparty Computation with Linear Communication Complexity. In C. Dwork, editor, *CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 463–482. Springer, 2006. 2, 3

[32] M. Hirt, J. B. Nielsen, and B. Przydatek. Cryptographic Asynchronous Multi-party Computation with Optimal Resilience (Extended Abstract). In R. Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*, pages 322–340. Springer, 2005. 2, 3

[33] M. Hirt, J. B. Nielsen, and B. Przydatek. Asynchronous Multi-Party Computation with Quadratic Communication. In *ICALP*, LNCS 5126, pages 473–485. Springer Verlag, 2008. 2, 3

[34] J. Katz, C. Y. Koo, and R. Kumaresan. Improving the Round Complexity of VSS in Point-to-point Networks. *Inf. Comput.*, 207(8):889–899, 2009. 19

[35] R. Kumaresan, A. Patra, and C. Pandu Rangan. The Round Complexity of Verifiable Secret Sharing: The Statistical Case. In *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 431–447. Springer, 2010. 5

[36] N. A. Lynch. *Distributed Algorithms.* Morgan Kaufmann, 1996. 4

[37] A. Patra, A. Choudhury, and C. Pandu Rangan. Asynchronous Byzantine Agreement with Optimal Resilience. *Distributed Computing*, 27(2):111–146, 2014. 4, 9, 10

[38] A. Patra, A. Choudhury, and C. Pandu Rangan. Efficient Asynchronous Verifiable Secret Sharing and Multiparty Computation. *J. Cryptology*, 28(1):49–109, 2015. 3, 17, 19, 20, 21, 22, 33, 40

[39] Arpita Patra. Error-free multi-valued broadcast and byzantine agreement with optimal communication complexity. In *Principles of Distributed Systems - 15th International Conference, OPODIS 2011, Toulouse, France, December 13-16, 2011. Proceedings*, pages 34–49, 2011. 4

[40] Arpita Patra, Ashish Choudhary, and C. Pandu Rangan. Communication efficient statistical asynchronous multiparty computation with optimal resilience. In *Information Security and Cryptology - 5th International Conference, Inscrypt 2009, Beijing, China, December 12-15, 2009. Revised Selected Papers*, pages 179–197, 2009. 3, 5

[41] T. Rabin and M. Ben-Or. Verifiable Secret Sharing and Multiparty Protocols with Honest Majority (Extended Abstract). In D. S. Johnson, editor, *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, pages 73–85. ACM, 1989. 2, 3, 4, 9

[42] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 160–164, 1982. 1