A Survey on Perfectly-Secure Verifiable Secret-Sharing

Anirudh C* Ashish Choudhury† Arpita Patra‡

April 6, 2021

Abstract

Verifiable Secret-Sharing (VSS) is a fundamental primitive in secure distributed computing. It is used as an important building block in several distributed computing tasks, such as Byzantine agreement and secure multi-party computation. VSS has been widely studied in various dimensions over the last three decades and several important results have been achieved related to the fault-tolerance, round-complexity and communication efficiency of VSS schemes. In this article, we consider VSS schemes with *perfect* security, tolerating *computationally unbounded* adversaries. We comprehensively survey the existing perfectly-secure VSS schemes in three different settings, namely synchronous, asynchronous and hybrid communication settings and provide the full details of each of the existing schemes in these settings. The aim of this survey is to provide a clear knowledge and foundation to researchers who are interested in knowing and extending the state-of-the-art perfectly-secure VSS schemes.

1 Introduction

A central concept in cryptographic protocols is that of *Secret Sharing* (SS) [48, 13]. Consider a set $\mathcal{P} = \{P_1, \ldots, P_n\}$ of mutually distrusting parties, where the distrust in the system is modeled by a centralized adversary, who can control upto t parties. Then a SS scheme allows a designated party $\mathsf{D} \in \mathcal{P}$ (called *dealer*) to share a secret s among $\mathcal{P}$, by providing each party P_i a *share* of the secret s. The sharing is done in such a way that the adversary controlling any subset of at most t share-holders fails to learn anything about s, while any subset of at least $t+1$ share-holders can jointly recover s. In a SS scheme, it is assumed that *all* the parties including the ones under the adversary's control follow the protocol instructions correctly (thus, the adversary is assumed to be *passive*, who can only eavesdrop the computation and communication of the parties under its control). *Verifiable Secret Sharing* (VSS) [19] extends the notion of SS to the more powerful *malicious/active* adversarial model, where the adversary can completely dictate the behaviour of the parties under its control during the execution of a protocol. Moreover, D is allowed to be potentially corrupted. A VSS scheme consists of a sharing phase and a reconstruction phase, each implemented by a publicly-known protocol. During the sharing phase, D shares its secret in a *verifiable* fashion, which is later reconstructed during the reconstruction phase. If D is *honest*, then the privacy of its secret is maintained during the sharing phase and the shared secret is later robustly reconstructed, irrespective of the behaviour

*International Institute of Information Technology, Bangalore India. Email: anirudh.c@iiitb.ac.in.

†International Institute of Information Technology, Bangalore India. Email: ashish.choudhury@iiitb.ac.in. This research is an outcome of the R & D work undertaken in the project under the Visvesvaraya PhD Scheme of Ministry of Electronics & Information Technology, Government of India, being implemented by Digital India Corporation (formerly Media Lab Asia).

‡Indian Institute of Science, Bangalore India. Email: arpita@iisc.ac.in. Arpita Patra would like to acknowledge financial support from SERB MATRICS (Theoretical Sciences) Grant 2020 and Google India AI/ML Research Award 2020.

of the corrupt parties. The interesting property of VSS is the *verifiability* property, which guarantees that even if D is *corrupt*, it has "consistently/correctly" shared some value among the parties and the same value is later reconstructed. One can interpret VSS as a distributed commitment, where, during the sharing phase, D publicly commits to a private input (known only to D) and later during the reconstruction phase, the committed value is reconstructed publicly (even if D does not cooperate).

VSS serves as a central building block in a variety of important secure distributed-computing tasks, such as secure *multi-party computation* (MPC) [11, 47] and Byzantine agreement (BA) [28]. Due to its importance, VSS has been studied in various settings, based on the following categorizations.

- **Conditional vs Unconditional Security**: If the adversary is *computationally bounded* (where it is allowed to perform only polynomial amount of computations), then the notion of security achieved is conditional/cryptographic [46, 5, 6], whereas unconditionally-secure VSS provides security even against a *computationally unbounded* adversary. Unconditionally-secure VSS can be further categorized as *perfectly-secure* where all security guarantees are achieved in an error-free fashion [11], and *statistically-secure* where a negligible error is allowed [47, 24, 37].
- **Type of Communication**: Here we have two categories. The more popular *synchronous* model assumes that the parties are synchronized through a global clock and there are strict (publicly-known) upper bounds on the message delays [11, 47, 31, 30, 36, 3]. The other category is the *asynchronous* communication model [10, 12, 8, 1, 42, 43, 23, 7], which models the real-world networks like the Internet where the parties are not synchronized and where the messages can be arbitrarily (but finitely) delayed. A major challenge in the asynchronous model is that a receiving party cannot distinguish between a *slow* sender whose messages are delayed and a *corrupt* sender who does not send the messages at all. Due to this inherent phenomenon, asynchronous VSS (AVSS) protocols are more complicated than their synchronous counterparts. A third category of VSS protocols are designed in the *hybrid* communication setting [44], which is a mix of the synchronous and asynchronous models. Namely, the protocol starts with a few initial synchronous rounds, followed by a completely asynchronous execution. The main motivation for considering a hybrid setting is to "bridge" the feasibility and efficiency gaps between completely synchronous and completely asynchronous protocols.
- **Corruption Capacity**: Most of the works on VSS assume a *threshold* adversary, where it is assumed that the adversary can corrupt any subset of upto t parties. A *non-threshold* adversary [25, 40, 21, 22] is a more generalized adversary, where the corruption capacity of the adversary is specified by a publicly-known *adversary structure*, which is a collection of potentially corrupt subset of parties. During the protocol execution, the adversary can choose any subset from the collection for corruption. A threshold adversary is a special type of non-threshold adversary, where the adversary structure consists of all t-sized subsets of $\mathcal{P}$.

Our Contributions In this work, we provide a comprehensive survey of all the existing *perfectly-secure* VSS schemes tolerating a *threshold* adversary. We consider all the three communication settings, namely synchronous, asynchronous and hybrid. These schemes are designed over a period of three decades. The nuances, subtleties and foundational ideas involved in these works need a holistic and unified treatment, which is the focus of this work. This survey is structured to provide an easy digest of the perfectly-secure VSS schemes. Through this survey, we hope to provide a clear knowledge and foundation to researchers who are interested in knowing and extending the state-of-the-art perfectly-secure VSS schemes.

Organization The survey is divided into three parts, each dealing with a separate communication model. The first part covers the synchronous communication model, where the bulk of the work on

VSS has been done. The second part deals with the asynchronous communication model. The third part covers the hybrid communication model.

Part I : Synchronous Communication Setting

2 Preliminaries and Definitions

Throughout part I, we consider a synchronous communication setting, where there exists a set of n mutually distrusting parties $\mathcal{P} = \{P_1, \ldots, P_n\}$, connected by pair-wise private and authentic channels (also known as the *secure-channel* model). The distrust in the system is modeled by a centralized adversary Adv, who is *computationally unbounded*. The corruption capability of Adv is upper bounded by a publicly-known threshold t, such that Adv can corrupt at most t parties during the execution of a protocol in a malicious/Byzantine fashion and force them to behave in any arbitrary manner during the protocol execution. We call the parties under the control of Adv as *corrupt/malicious*, while the parties not under the control of Adv are called *honest*. For simplicity, we assume a *static* adversary, who decides the set of parties to corrupt at the beginning of a protocol. However, the protocols discussed can be proved to be secure even assuming a more powerful *adaptive* adversary, who can adaptively corrupt parties as the protocol proceeds by following [38].

Apart from the pair-wise secure channels, we assume the presence of a system-wide *broadcast* channel, which allows any designated party in $\mathcal{P}$ to send some message identically to all the parties. Any protocol in the synchronous setting operates as a sequence of *rounds*. In each round, a party can (privately) send messages to other parties and broadcast a message. Moreover, we assume that in a given round, each party can simultaneously use the broadcast channel. The messages sent or broadcast by a party is determined by its input, its random coins, and messages received from other parties in previous rounds. We assume that Adv is *rushing*, which means that in any round the corrupt parties receive the messages sent by the honest parties before deciding on their own messages for that round. The *view* of a party during a protocol execution is defined to be its inputs and random coins for the protocol, along with all the messages received by the party throughout the protocol execution. The *view* of Adv is defined to be the collection of the views of the parties under the control of Adv.

Structure of a VSS Protocol. Following the abstraction of [31], VSS protocols can be structured into two phases. A *sharing phase* executed by a protocol Sh, followed by a *reconstruction phase* executed by a protocol Rec. While the goal of the sharing phase is to share a secret held by a designated *dealer* $\mathsf{D} \in \mathcal{P}$, the aim of the reconstruction phase is to reconstruct back the shared secret. In a more detail, during the protocol Sh, the input of D is some secret $s \in \mathcal{S}$, where $\mathcal{S}$ is some publicly-known *secret-space* which is the set of all possible D's secrets. Additionally, the parties may have random inputs for the protocol. Let view_i denote the view of party P_i at the end of the protocol Sh. Based on view_i, each party P_i outputs a *share* s_i, which is some publicly-known function of view_i, as determined by the protocol Sh.

During the reconstruction phase, each P_i may reveal a subset of its view view_i, as per the protocol Rec. Each party then applies a publicly-known reconstruction function on the revealed views, as determined by the protocol Rec and reconstructs some output. Following [36], we say that *round-complexity* of protocol Sh is (r, r'), where $r' \leq r$, if Sh involves r rounds of communication and if among these r rounds, the broadcast channel is used for r' rounds. We use similar terminology for the round-complexity of Rec. By *communication complexity* of a protocol, we mean the total number of bits communicated by the honest parties in the protocol.

2.1 Definitions

We start with the definition of a *t-out-of-n secret-sharing (SS)* scheme, which consists of a share-generation function G and a recovery function R. While G is probabilistic, the function R is deterministic. The function G generates shares for the input secret, while R maps the shares back to the secret. The shares are generated in such a way that the probability distribution of any set of t shares is independent of the secret, while any set of $t+1$ shares uniquely determine the secret.

Definition 2.1 (**t-out-of-n secret-sharing [32]**)**.** It is a pair of algorithms (G, R), such that

- **Syntax**: The *share-generation function* G takes input a secret s and some randomness r and outputs a vector of n shares $(s_1, \ldots, s_n)$. The *recovery* function R takes input a set of $t+1$ shares corresponding to $t+1$ indices $\{i_1, \ldots, i_{t+1}\} \subset \{1, \ldots, n\}$ and outputs a value.
- **Correctness**: For any secret s and any vector of n shares $(s_1, \ldots, s_n)$ where $(s_1, \ldots, s_n) = \mathsf{G}(s, r)$ for some randomness r, it holds that for every subset $\{i_1, \ldots, i_{t+1}\} \subset \{1, \ldots, n\}$

$$\mathsf{R}(s_{i_1}, \ldots, s_{i_{t+1}}) = s.$$

- **Privacy**: For any subset of t indices, the probability distribution of the shares corresponding to these indices in independent of the underlying secret. That is, for any $I = \{i_1, \ldots, i_t\} \subset \{1, \ldots, n\}$, let $g_I(s) \stackrel{def}{=} (s_{i_1}, \ldots, s_{i_t})$, where $(s_1, \ldots, s_n) = \mathsf{G}(s, r)$ for some randomness r. Then we require that for every index-set I where $|I| = t$, the random variables $g_I(s)$ and $g_I(s')$ are identically distributed, for every pair (s, s') where $s \neq s'$.

Definition 2.2. Let $\Pi = (\Pi_{\mathsf{G}}, \Pi_{\mathsf{R}})$ be a t-out-of-n secret-sharing scheme. Then we say that *a value s is secret-shared among $\mathcal{P}$ as per* Π, if there exists some randomness r such that $(s_1, \ldots, s_n) = \Pi_{\mathsf{G}}(s, r)$ and each honest party $P_i \in \mathcal{P}$ holds the share s_i.

In the literature, two types of VSS protocols have been considered. The type-I VSS protocols are "weaker" in terms of the properties achieved, compared to the type-II VSS protocols. Namely, in type-II VSS, it is guaranteed that the underlying secret of the dealer is secret-shared as per some *specified* secret-sharing scheme. While the first category of VSS is sufficient to study VSS as a stand-alone primitive, the second category of VSS schemes is desirable when VSS is used as a primitive in secure MPC protocols [36, 4, 3].

Definition 2.3 (**Type-I VSS [31]**)**.** Let $(\mathsf{Sh}, \mathsf{Rec})$ be a pair of protocols for the parties in $\mathcal{P}$, where a designated *dealer* $\mathsf{D} \in \mathcal{P}$ has some private input $s \in \mathcal{S}$ for the protocol Sh. Then $(\mathsf{Sh}, \mathsf{Rec})$ is called a *Type-I perfectly-secure VSS scheme*, if the following requirements hold.

- **Privacy**: If D is *honest*, then the view of Adv during Sh is distributed independent of s.
- **Correctness**: If D is *honest*, then all honest parties output s at the end of Rec.
- **Strong Commitment**: Even if D is *corrupt*, in any execution of Sh the joint view of the honest parties defines a unique value $s^\star$ (which could be different from s), such that all honest parties output $s^\star$ at the end of Rec, irrespective of the behaviour of Adv.

Definition 2.4 (**Type-II VSS [32]**)**.** Let $(\mathsf{Sh}, \mathsf{Rec})$ be a pair of protocols for the parties in $\mathcal{P}$, where a designated *dealer* $\mathsf{D} \in \mathcal{P}$ has some private input $s \in \mathcal{S}$ for the protocol Sh. Moreover, let $\Pi = (\Pi_{\mathsf{G}}, \Pi_{\mathsf{R}})$ be a given t-out-of-n secret-sharing scheme. Then $(\mathsf{Sh}, \mathsf{Rec})$ is called a *Type-II perfectly-secure VSS scheme with respect to* Π, if the following requirements hold.

- **Privacy**: If D is *honest*, then the view of Adv during Sh is distributed independent of s.
- **Correctness**: If D is *honest*, then at the end of Sh, the value s is secret-shared among $\mathcal{P}$ as per Π. Moreover, all honest parties output s at the end of Rec.
- **Strong Commitment**: Even if D is *corrupt*, in any execution of Sh the joint view of the honest parties defines a unique value $s^\star$, such that $s^\star$ is secret-shared among $\mathcal{P}$ as per Π. Moreover, all honest parties output $s^\star$ at the end of Rec, irrespective of the behaviour of Adv.

Further stronger definition of VSS Definition 2.3-2.4 are referred as "property-based" definition of VSS, where security is defined by enumerating a list of desired security goals. However, as shown in [16, 17], the property-based security definition (for any cryptographic task) is not rigorous. For instance, the definition *does not* guarantee that all the desired security goals are indeed enumerated. More importantly, the definition does not say anything about what security guarantees are achieved, when multiple instances of a protocol are composed and executed in parallel. Motivated by these shortcomings, [16, 17] proposed the *Universal Composability* (UC)-security framework for defining and proving the security of any general cryptographic task. Loosely speaking, according to this framework, a VSS protocol is considered to be *secure* if it "emulates" an ideal-world VSS protocol [32], where the dealer D *privately* provides the secret s and an appropriate randomness r as inputs to a centralized *trusted third-party* (TTP), who then computes the shares of s as per some given secret-sharing scheme Π and privately sends the shares to the respective parties. Proving the security of VSS protocols as per the UC framework brings in additional technicalities. To the best of our knowledge, except the works of [4, 3], the security of all VSS protocols is proved according to the property-based definitions. Since the main goal of this paper is to survey the existing perfectly-secure VSS protocols, we will stick to the property-based definitions. However, we stress that the security of the protocols discussed in this article can also be proved as per the UC framework, by following [4, 3].

2.2 Properties of Polynomials Over a Finite Field

Let $\mathbb{F}$ be a finite field where $|\mathbb{F}| > n$ and let $\alpha_1, \ldots, \alpha_n$ be distinct non-zero elements of $\mathbb{F}$. A degree-d *univariate polynomial* over $\mathbb{F}$ is of the form $f(x) = a_0 + \ldots + a_d x^d$, where each $a_i \in \mathbb{F}$. A degree-(ℓ, m) *bivariate polynomial* over $\mathbb{F}$ is of the form $F(x,y) = \sum_{i=0, j=0}^{i=\ell, j=m} r_{ij} x^i y^j$, where each $r_{ij} \in \mathbb{F}$.

Let $f_i(x) \stackrel{def}{=} F(x, \alpha_i), g_i(y) \stackrel{def}{=} F(\alpha_i, y)$. We call $f_i(x)$ and $g_i(y)$ as i^{th} *row* and *column-polynomial* respectively of $F(x,y)$. This is because the distinct evaluations of the polynomials $f_i(x)$ and $g_i(y)$ at $x = \alpha_1, \ldots, \alpha_n$ and at $y = \alpha_1, \ldots, \alpha_n$ respectively, constitute an $n \times n$ matrix of distinct points on $F(x,y)$ (see Fig 1). It is easy to see that $f_i(\alpha_j) = g_j(\alpha_i)$ holds for every distinct pair of α_i, α_j, since $f_i(\alpha_j) = g_j(\alpha_i) = F(\alpha_j, \alpha_i)$. We say a degree-$m$ polynomial $F_i(x)$, where $i \in \{1, \ldots, n\}$, *lie* on a degree-(ℓ, m) bivariate polynomial $F(x,y)$, if $F(x, \alpha_i) = F_i(x)$ holds. Similarly, we say a degree-ℓ polynomial $G_i(y)$, where $i \in \{1, \ldots, n\}$, *lie* on $F(x,y)$, if $F(\alpha_i, y) = G_i(y)$ holds.

$F(x,y)$ is called *symmetric*, if $r_{ij} = r_{ji}$ holds for every i, j. This automatically implies that $F(\alpha_j, \alpha_i) = F(\alpha_i, \alpha_j)$ holds for every α_i, α_j, further implying that $F(x, \alpha_i) = F(\alpha_i, y)$ holds.

Definition 2.5 (d**-sharing** [26, 9])**.** A value $s \in \mathbb{F}$ is said to be d-shared, if there exists a degree-d polynomial, say $q(\cdot)$, with $q(0) = s$, such that each (honest) $P_i \in \mathcal{P}$ holds its *share*[1] $s_i \stackrel{def}{=} q(\alpha_i)$. The vector of shares of s corresponding to the (honest) parties in $\mathcal{P}$ is denoted as $[s]_d$. A set of values $S = (s^{(1)}, \ldots, s^{(L)}) \in \mathbb{F}^L$ is said to be d-shared, if each $s^{(i)} \in S$ is d-shared.

2.2.1 Properties of Univariate Polynomials Over $\mathbb{F}$

We next state certain standard properties of univariate polynomials, which are used extensively in VSS protocols. We start with the well-known Shamir's t-out-of-n secret-sharing scheme [48]. Most of the type-II VSS protocols are with respect to Shamir's secret-sharing scheme. The share-generation algorithm Sha_G of the Shamir's secret-sharing scheme takes input a secret $s \in \mathbb{F}$. To compute the

[1] We interchangeably use the term *shares of* s and *shares of the polynomial* $q(\cdot)$ to denote the values $q(\alpha_i)$.

shares, a polynomial $q(\cdot)$ is picked uniformly at random from the set $\mathcal{P}^{s,t}$ of all degree-t univariate polynomials over $\mathbb{F}$ whose constant term is s, where $|\mathcal{P}^{s,t}| = |\mathbb{F}|^t$. The output of the algorithm is the vector of shares $(s_1, \ldots, s_n)$, where $s_i = q(\alpha_i)$. Since the polynomial[2] is chosen randomly and has degree-t, the probability of the t shares learnt by Adv will be independent of the underlying secret. Formally:

Lemma 2.6 ([4]). *For any set of distinct non-zero elements $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$, any pair of values $s, s' \in \mathbb{F}$, any subset $I \subset \{1, \ldots, n\}$ where $|I| = \ell \leq t$, and every $\vec{y} \in \mathbb{F}^\ell$, it holds that:*

$$\underset{f(x) \in_r \mathcal{P}^{s,t}}{Pr}\Big[\vec{y} = (\{f(\alpha_i)\}_{i \in I})\Big] = \underset{g(x) \in_r \mathcal{P}^{s',t}}{Pr}\Big[\vec{y} = (\{g(\alpha_i)\}_{I \in I})\Big],$$

where $f(x)$ and $g(x)$ are chosen uniformly and independently[3] from $\mathcal{P}^{s,t}$ and $\mathcal{P}^{s',t}$, respectively.

Let $(s_1, \ldots, s_n)$ be a vector of Shamir-shares for s, generated by Sha_G. Moreover, let $I \subset \{1, \ldots, n\}$, where $|I| = t + 1$. Then the recovery function Sha_R takes input the shares $\{s_i\}_{i \in I}$ and outputs s. The standard instantiation of Sha_R is the well-known Lagrange's interpolation formula, which interpolates the unique degree-t Shamir-sharing polynomial passing through the points $\{(\alpha_i, s_i)\}_{i \in I}$.

Relationship Between d-sharing and Reed-Solomon (RS) Codes Let s be d-shared through a polynomial $q(\cdot)$ and let $(s_1, \ldots, s_n)$ be the vector of shares. Moreover, let W be a subset of these shares, such that it is ensured that at most r shares in W are incorrect (the exact identity of the incorrect shares are not known). The goal is to error-correct the incorrect shares in W and correctly reconstruct back the polynomial $q(\cdot)$. Coding-theory [39, 41] says that this is possible if and only if $|W| \geq d + 2r + 1$ and the corresponding algorithm is denoted by $\mathsf{RS\text{-}Dec}(d, r, W)$. There are several well-known efficient instantiations of $\mathsf{RS\text{-}Dec}$, such as the Berlekamp-Welch algorithm [39].

2.2.2 Properties of Bivariate Polynomials Over $\mathbb{F}$

For univariate polynomials, there exists a unique degree-d univariate polynomial, passing through a given set of $d + 1$ distinct points. The following result can be imagined as a generalization of this result for the case of bivariate polynomials. Informally, it states that if there are "sufficiently many" univariate polynomials which are "pair-wise consistent", then together they define a unique bivariate polynomial. Formally:

Lemma 2.7 (**Pair-wise Consistency Lemma [15, 43, 4]**). *Let $f_{i_1}(x), \ldots, f_{i_q}(x)$ be degree-ℓ polynomials and let $g_{j_1}(y), \ldots, g_{j_r}(y)$ be degree-m polynomials where $q \geq m + 1, r \geq \ell + 1$ and where $i_1, \ldots, i_q, j_1, \ldots, j_r \in \{1, \ldots, n\}$. Moreover, let for every $i \in \{i_1, \ldots, i_q\}$ and every $j \in \{j_1, \ldots, j_r\}$, the condition $f_i(\alpha_j) = g_j(\alpha_i)$ holds, where $\alpha_1, \ldots, \alpha_n$ are distinct non-zero elements from $\mathbb{F}$. Then there exists a unique degree-(ℓ, m) bivariate polynomial, say $F^\star(x, y)$, such that the row polynomials $f_{i_1}(x), \ldots, f_{i_q}(x)$ and the column polynomials $g_{j_1}(y), \ldots, g_{j_r}(y)$ lie on $F^\star(x, y)$.*

In most VSS protocols, a dealer on having a secret s does the following to share it: the dealer picks a random degree-t Shamir-sharing polynomial $q(\cdot)$ where $q(0) = s$. The sharing polynomial $q(\cdot)$ is further embedded into a random degree-(t, t) bivariate polynomial $F(x, y)$ at $x = 0$. The dealer then distributes the row-polynomial $f_i(x) = F(x, \alpha_i)$ and column-polynomial $g_i(y) = F(\alpha_i, y)$ to every party P_i. Similar to the case of Shamir secret-sharing, it holds that Adv, by learning at most

[2] We often use the term Shamir-sharing polynomial to denote the degree-t polynomial used in the algorithm.

[3] Here the notation $\in_r$ denotes that the polynomials are picked uniformly at random from the respective domains.

t row and column-polynomials, does not learn any information about the underlying shared value s. Intuitively, this is because $(t+1)^2$ distinct values are required to uniquely determine $F(x,y)$, but Adv learns at most t^2+2t distinct values. In fact, it can be shown that for every two degree-t polynomials $q_1(\cdot), q_2(\cdot)$ such that $q_1(\alpha_i) = q_2(\alpha_i) = f_i(0)$ holds for every $P_i \in \mathcal{C}$ (where $\mathcal{C}$ is the set of corrupt parties), the distribution of the polynomials $\{f_i(x), g_i(y)\}_{P_i \in \mathcal{C}}$ when $F(x,y)$ is chosen based on $q_1(\cdot)$, is identical to the distribution when $F(x,y)$ is chosen based on $q_2(\cdot)$. Formally:

Lemma 2.8 ([4]). *Let $\mathcal{C} \subset \mathcal{P}$ where $|\mathcal{C}| \leq t$, and let $q_1(\cdot)$ and $q_2(\cdot)$ be two different degree-t polynomials over $\mathbb{F}$ such that $q_1(\alpha_i) = q_2(\alpha_i)$ holds for every $P_i \in \mathcal{C}$. Then,*

$$\Big\{\{F(x,\alpha_i), F(\alpha_i,y)\}_{P_i \in \mathcal{C}}\Big\} \equiv \Big\{\{F'(x,\alpha_i), F'(\alpha_i,y)\}_{P_i \in \mathcal{C}}\Big\}$$

holds, where $F(x,y)$ and $F'(x,y)$ are two different degree-(t,t) bivariate polynomials, chosen at random under the constraints that $F(0,y) = q_1(\cdot)$ and $F'(0,y) = q_2(\cdot)$ holds.

3 Lower Bounds

An obvious necessary condition for *any* perfectly-secure VSS protocol is that $n > 2t$ should hold. This follows from the following intuitive argument. In any VSS protocol, the joint view of the *honest* parties should uniquely determine the dealer's secret. Otherwise the *correctness* property of the VSS protocol will be violated and the honest parties will fail to reconstruct dealer's secret when the corrupt parties produce incorrect view during the reconstruction phase. Since in the worst case there can be $n-t$ honest parties, to satisfy the *privacy* property of VSS, the condition $n-t > t$ should hold, as otherwise the view of the adversary will not be independent of dealer's secret.

We actually need a more stricter necessary condition of $n > 3t$ to hold for *any* perfectly-secure VSS protocol, irrespective of the number of rounds. While the requirement of the necessity condition of $n > 3t$ has been argued informally in several works [11, 18, 47], a formal proof appeared in [27].

Theorem 3.1 ([27]). *Let $\Pi = (\mathsf{Sh}, \mathsf{Rec})$ be a perfectly-secure VSS scheme, where the round-complexity of Sh and Rec is $(r_{\mathsf{Sh}}, r'_{\mathsf{Sh}})$ and $(r_{\mathsf{Rec}}, r'_{\mathsf{Rec}})$ respectively. Then $n > 3t$ holds.*

Theorem 3.1 is proved by relating VSS with the problem of 1-*way perfectly-secure message transmission* (1-way PSMT) [27]. In the 1-way PSMT problem, there is a sender $\mathbf{S}$ and a receiver $\mathbf{R}$, such that there are n *disjoint uni-directional* communication channels $Ch_1, \ldots, Ch_n$ from $\mathbf{S}$ to $\mathbf{R}$ (i.e. only $\mathbf{S}$ can send messages to $\mathbf{R}$ along these channels, but the other way around communication is not possible). At most t out of these channels can be controlled by a malicious/Byzantine adversary in any arbitrary fashion. The goal is to design a protocol, which allows $\mathbf{S}$ to send some input message m *reliably* (i.e. $\mathbf{R}$ should be able to receive m without any error) and *privately* (i.e. view of the adversary should be independent of m). In [27], it is shown that a 1-way PSMT protocol exists only if $n > 3t$. Moreover, if there exists a perfectly-secure VSS scheme with $n \leq 3t$, then using it one can even design a 1-way PSMT protocol with $n \leq 3t$, which is a contradiction.

The Round Complexity of VSS In [31], the round-complexity of a VSS protocol is defined to be the number of rounds in the sharing phase. The round complexity of reconstruction phase is not counted, as all (perfectly-secure) VSS protocols adhere to a single-round reconstruction, where the parties can just reveal (a subset of) their view to every other party. While Theorem 3.1 dictates a necessary condition of $n > 3t$ for *any* VSS protocol, the interplay between the round-complexity of perfectly-secure VSS and resilience bounds (stated in Theorem 3.2) was studied in [31].

Theorem 3.2 ([31]). *Let $r \geq 1$ be a positive integer and let $r' \leq r$. Then:*

- *If $r = 1$, then there exists no perfectly-secure VSS protocol with (r, r') rounds in the sharing phase under any of the following conditions.*
  - *When $t > 1$, irrespective of the value of n.*
  - *When $t = 1$ and $n \leq 4$.*
- *If $r = 2$, then perfectly-secure VSS with (r, r') rounds in sharing phase is possible only if $n > 4t$.*
- *If $r \geq 3$, then perfectly-secure VSS with (r, r') rounds in sharing phase is possible only if $n > 3t$.*

The above bounds are obtained by relating the round complexity of VSS to the round complexity of the *secure multi-cast* (SM) problem. In the SM problem, there exists a designated sender $\mathsf{S} \in \mathcal{P}$ with some private message and a designated receiving set $\mathcal{R} \subseteq \mathcal{P}$. The goal is to design a protocol which allows S to send its message identically to all the parties *only* in the set $\mathcal{R}$, even in the presence of a computationally unbounded adversary who can control any t parties, possibly including S.

On the Usage of Broadcast Channel All perfectly-secure VSS protocols are designed under the assumption that a system-wide broadcast channel is available to the parties. However, this is just a *simplifying abstraction*, as one can "emulate" the effect of a broadcast channel by executing a perfectly-secure *reliable-broadcast* (RB) protocol [45] among the parties over the pair-wise channels, provided $n > 3t$. RB protocols with *guaranteed termination* are slow and require $\Omega(t)$ rounds of communication [29], while the fast RB protocols with *probabilistic termination* guarantees require $\mathcal{O}(1)$ expected number of rounds [28, 35] where the constants are rather high. Given the fact that the usage of broadcast channel is an "expensive resource", a natural question is whether one can design a VSS protocol with a *constant* number of rounds in the sharing phase and which *does not* require the usage of broadcast channel in any of these rounds. Unfortunately, the answer is no. This is because such a VSS protocol will imply the existence of a strict constant round RB protocol with guaranteed termination (the message to be broadcast by the sender can be shared using the VSS protocol with sender playing the role of the dealer, followed by reconstructing the shared message), which is impossible as per the result of [29]. Hence the best that one can hope for is to design VSS protocols which invokes the broadcast channel only in a fewer rounds.

4 Upper Bounds

We now discuss the optimality of the bounds presented in Theorem 3.2 by discussing VSS protocols with various round-complexities. The sharing phase of these protocols is summarized in Table 1. The round complexity of the reconstruction phase of all the protocols is $(1, 0)$. The names of the VSS schemes are prefixed with the number of rounds required in the sharing phase. In the table, $\mathbb{G}$ denotes a finite group and RSS stands for replicated secret-sharing [34] (see Section 4.1.4). The 3AKP-VSS scheme has some special properties, compared to 3FGGRS-VSS, 3KKK-VSS schemes, which are useful for designing round-optimal perfectly-secure MPC protocols (see Section 4.1.7).

4.1 VSS Protocols with $n > 3t$

We start with perfectly-secure VSS schemes with $n > 3t$. With the exception of [30], all the schemes in this category are of Type-II. The reconstruction phase of all these VSS schemes (including [30]) requires a single round. While presenting these protocols, we use the following simplifying conventions. If a party is expecting some message from a sender party in the protocol and if it either receives no message or semantically and syntactically incorrect message, then the receiving party substitutes some default value which is semantically and syntactically correct and proceeds with the steps of the protocol. Similarly, if the dealer is *publicly* identified to be cheating then the parties discard the dealer and terminate the protocol execution with a default sharing of 0.

Scheme	n	Round Complexity	Type	Sharing Semantic	Algebraic Structure	Complexity
7BGW-VSS [11]	$n > 3t$	$(7, 5)$	Type-II	Shamir	$\mathbb{F}$	polynomial(n, t)
5BGW-VSS [31]	$n > 3t$	$(5, 3)$	Type-II	Shamir	$\mathbb{F}$	polynomial(n, t)
4GIKR-VSS [31]	$n > 3t$	$(4, 3)$	Type-II	Shamir	$\mathbb{F}$	polynomial(n, t)
3GIKR-VSS [31]	$n > 3t$	$(3, 2)$	Type-II	RSS	$\mathbb{G}$	exponential(n, t)
3FGGRS-VSS [30]	$n > 3t$	$(3, 2)$	Type-I	Shamir	$\mathbb{F}$	polynomial(n, t)
3KKK-VSS [36]	$n > 3t$	$(3, 1)$	Type-II	Shamir	$\mathbb{F}$	polynomial(n, t)
3AKP-VSS [3]	$n > 3t$	$(3, 2)$	Type-II	Shamir	$\mathbb{F}$	polynomial(n, t)
2GIKR-VSS [31]	$n > 4t$	$(2, 1)$	Type-II	Shamir	$\mathbb{F}$	polynomial(n, t)
1GIKR-VSS [31]	$n = 5, t = 1$	$(1, 0)$	Type-I	Not Applicable	$\mathbb{F}$	polynomial(n, t)

Table 1: Summary of the sharing phase of the perfectly-secure VSS schemes.

4.1.1 The 7-round 7BGW-VSS Scheme

The 7BGW-VSS scheme, consisting of the protocols 7BGW-VSS-Sh and 7BGW-VSS-Rec are presented in Fig 2. The sharing phase protocol is actually a simplified version of the original protocol, taken from [33]. The approach used in the protocol 7BGW-VSS-Sh has been followed in all the followup works on perfectly-secure VSS schemes. To share a secret s, the dealer D picks a random degree-t *Shamir-sharing polynomial* $q(\cdot)$ and the goal is to ensure that D *verifiably* distributes the Shamir-shares of s as per the polynomial $q(\cdot)$ to respective parties. Later, during the protocol 7BGW-VSS-Rec, the parties exchange these Shamir-shares and correctly reconstruct $q(\cdot)$ by error-correcting upto t incorrect shares using the algorithm RS-Dec. The verifiability in 7BGW-VSS-Sh ensures that even if D is *corrupt*, the shares distributed by D to the (honest) parties during 7BGW-VSS-Sh are as per some degree-t Shamir-sharing polynomial, say $q^\star(\cdot)$, thus ensuring that $s^\star \stackrel{def}{=} q^\star(0)$ is Shamir-shared. Moreover, the same $s^\star$ gets reconstructed during 7BGW-VSS-Rec. This ensures that the 7BGW-VSS scheme is a type-II VSS scheme.

To prove that D is sharing its secret as per some degree-t Shamir-sharing polynomial $q(\cdot)$, the dealer D embeds its sharing-polynomial in a random degree-(t, t) bivariate polynomial $F(x, y)$. As shown in Fig 1, there are two approaches to do this embedding. The polynomial $q(\cdot)$ could be either embedded at $x = 0$ (the approach shown in part (a)) or it could be embedded at $y = 0$ (the approach shown in part (b)). For our description, we follow the first approach. The dealer then distributes distinct row and column-polynomials to respective parties. If D is *honest*, then this distribution of information maintains the privacy of dealer's secret (follows from Lemma 2.8). Also, if D is *honest*, then constant term of the individual row-polynomials actually constitute the Shamir-shares of s, as these values are nothing but distinct points on the sharing-polynomial $q(\cdot)$. However, a potentially *corrupt* D may distribute individual row and column-polynomials, which may not be derived from a single degree-(t, t) bivariate polynomial. Hence, the parties interact to verify if D has distributed "consistent" row and column-polynomials, without revealing any additional information about s.

Every pair of parties P_i, P_j upon receiving the polynomials $f_i(x), g_i(y)$ and $f_j(x), g_j(y)$ respectively, interact and check if $f_i(\alpha_j) = g_j(\alpha_i)$ and $f_j(\alpha_i) = g_i(\alpha_j)$ holds. If the checks pass for all the pairs of parties, then from Lemma 2.7, it follows that D has distributed consistent row (and column-polynomials) to the parties. However, if the checks do not pass, then either D has distributed inconsistent polynomials or the parties have not exchanged the correct common values. In this case, the parties interact publicly with D to resolve these inconsistencies. The details follow.

Every P_i upon receiving the supposedly common values on its column polynomial from the other parties prepares a *complaint-list* L_i, which includes all the parties P_j whose received value is inconsistent with P_i's column-polynomial (this should be interpreted as if there is a dispute between P_i and P_j). Notice that if there is a dispute between P_i and P_j, then at least one of the three parties D, P_i, P_j is *corrupt*. Each party then broadcasts its complaint-list. In response, for every

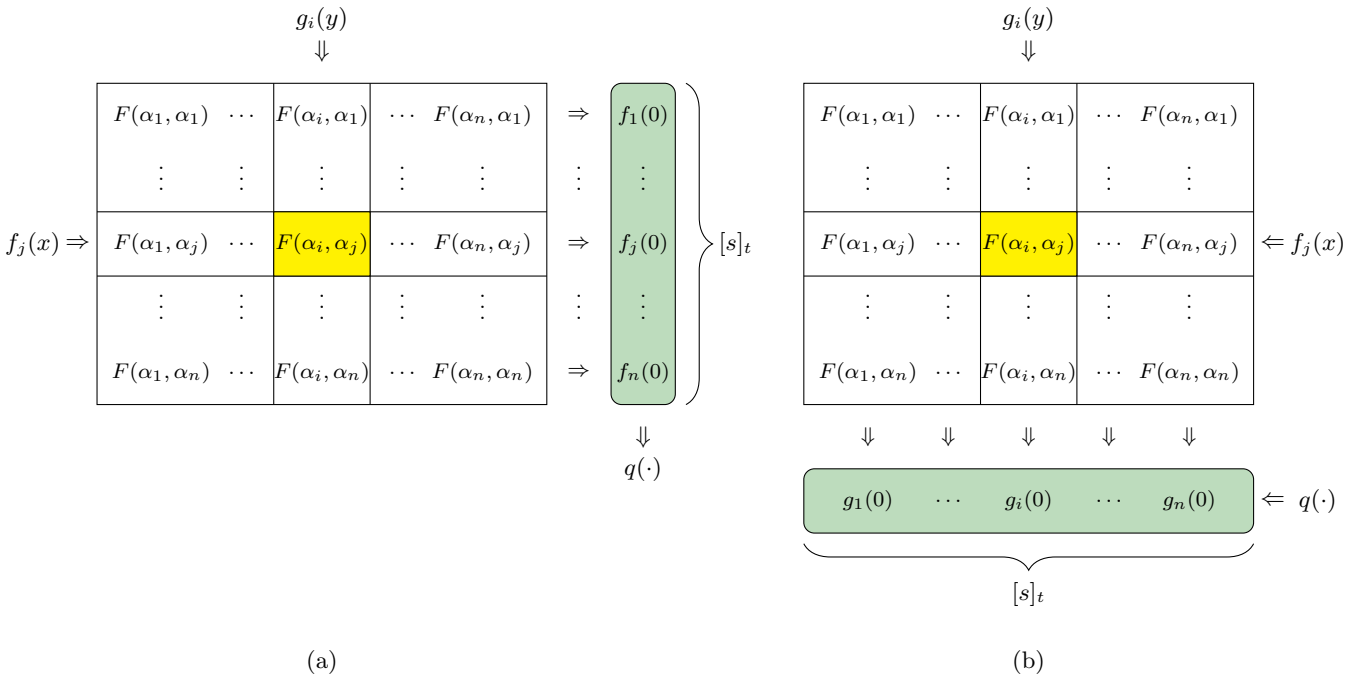

Figure 1: Pictorial depiction of the values on the degree-(t, t) polynomial $F(x, y)$ distributed by D and how they constitute $[s]_t$. The value highlighted in the yellow color denotes a common value held by every pair of parties (P_i, P_j). In the first approach, the Shamir-sharing polynomial $q(\cdot)$ is set as $F(0, y)$ and the Shamir-shares are the constant terms of the individual row-polynomials. In the second approach, $q(\cdot)$ is set as $F(x, 0)$ and the Shamir-shares are the constant terms of the individual column-polynomials.

dispute reported by a party, the dealer D makes public its version of the disputed value, namely the corresponding value on its bivariate polynomial. This is followed by the first-stage accusations against the dealer. Namely, a party publicly "accuses" D, if the party is in dispute with more than t parties or if it finds D making public a value, which is not consistent with its column-polynomial. In response, D makes public the row and column-polynomials of such accusing parties. However, care has to taken to ensure that these broadcast polynomials are consistent with the polynomials of the parties who have not yet accused D. This is done through the second-stage of public accusations against dealer, where a party (who has not yet accused D) publicly accuses D, if it finds any inconsistency between the row and column-polynomials held by the party and the polynomials which are made public by D (in response to D's response to first-stage accusations).

Throughout the protocol, if D is *honest*, then it will always respond correctly in response to any accusation or dispute raised. Moreover, there will be at most t accusations against D, as honest parties never accuse an honest D. Consequently, in the protocol if the parties find D not responding to some accusation or some dispute or if more than t parties accuse D, then D is *corrupt* and hence the parties terminate the protocol by discarding D. By making public the row and column-polynomials of the accusing parties or disputed values on the bivariate polynomial, the privacy is *not* violated. This is because if D is *honest*, then all the values which are made public correspond to corrupt parties and are already known to Adv. On the other hand, if D is *corrupt* but not discarded, then it ensures that the polynomials of all honest parties are consistent.

Scheme 7BGW-VSS

Sharing Phase: Protocol 7BGW-VSS-Sh

- **Round I (sending polynomials) — the dealer does the following**:
  - On having the input $s \in \mathbb{F}$, pick a random degree-t *Shamir-sharing polynomial* $q(\cdot)$, such that $q(0) = s$ holds. Then pick a random degree-(t,t) bivariate polynomial $F(x,y)$, such that $F(0,y) = q(\cdot)$ holds.
  - For $i = 1, \ldots, n$, send the polynomials $f_i(x) \stackrel{def}{=} F(x, \alpha_i)$ and $g_i(y) \stackrel{def}{=} F(\alpha_i, y)$ to party P_i.
- **Round II (pair-wise consistency checks) — each party P_i does the following**:
  - On receiving degree-t polynomials $f_i(x), g_i(y)$ from D, send $f_{ij} \stackrel{def}{=} f_i(\alpha_j)$ to P_j, for $j = 1, \ldots, n$.
- **Round III (broadcast complaints) — each party P_i does the following**:
  - Initialize a *complaint-list* L_i to $\emptyset$. For $j = 1, \ldots, n$, include P_j to L_i, if $f_{ji} \neq g_i(\alpha_j)$. Broadcast L_i.
- **Round IV (resolving complaints) — the dealer does the following**:
  - For $i = 1, \ldots, n$, if P_i has broadcast $L_i \neq \emptyset$, then for every $P_j \in L_i$, broadcast the value $F(\alpha_i, \alpha_j)$.
- **Round V (first-stage accusations) — each party P_i does the following**:
  - Broadcast the message $(P_i, \texttt{accuse}, \mathsf{D})$, if any of the following conditions hold.
    1. $|L_i| > t$ or if $P_i \in L_i$;
    2. If $\exists k \in \{1, \ldots, n\}$, such that $P_i \in L_k$ and $F(\alpha_k, \alpha_i) \neq f_i(\alpha_k)$;
    3. If for any $P_j \in L_i$, the condition $F(\alpha_i, \alpha_j) \neq g_i(\alpha_j)$ holds.
- **Round VI (resolving first-stage accusations) — the dealer does the following**:
  - For every P_i who has broadcast $(P_i, \texttt{accuse}, \mathsf{D})$, broadcast the degree-$t$ polynomials $f_i(x)$ and $g_i(y)$.
- **Round VII (second-stage accusations) — each party P_i does the following**:
  - Broadcast the message $(P_i, \texttt{accuse}, \mathsf{D})$ if there exists any $j \in \{1, \ldots, n\}$ such that D has broadcast degree-t polynomials $f_j(x), g_j(y)$ and either $f_j(\alpha_i) \neq g_i(\alpha_j)$ or $g_j(\alpha_i) \neq f_i(\alpha_j)$ holds.
- **Output decision — each party P_i does the following**:
  - If more than t parties P_j broadcast $(P_j, \texttt{accuse}, \mathsf{D})$ throughout the protocol, then discard D.
  - Else output the *share*[a] $f_i(0)$.

Reconstruction Phase: Protocol 7BGW-VSS-Rec

Each party $P_i \in \mathcal{P}$ does the following:
- Send the share s_i to every party $P_j \in \mathcal{P}$. Let W_i be the set of shares received from the parties. Execute $\mathsf{RS\text{-}Dec}(t, t, W_i)$ to reconstruct s and terminate.

[a] If D has broadcast new polynomials $f_i(x), g_i(y)$ for P_i during Round VI, then consider these new polynomials.

Figure 2: The perfectly-secure VSS scheme of [11].

Since the idea of bivariate polynomials has been used in all the followup works on perfectly-secure VSS, we give a very high level overview of the proof of the properties of 7BGW-VSS protocol.

Theorem 4.1. *Protocols* (7BGW-VSS-Sh, 7BGW-VSS-Rec) *constitute a Type-II perfectly-secure VSS scheme with respect to Shamir's t-out-of-n secret-sharing scheme.*

Proof. If D is *honest*, then for every pair of parties (P_i, P_j), $f_i(\alpha_j) = g_j(\alpha_i)$ and $f_j(\alpha_i) = g_i(\alpha_j)$ holds. Consequently, no honest P_j will be present in the list L_i of any honest P_i. Moreover, D honestly resolves the first-stage accusations as well as second-stage accusations and consequently, no honest party accuses and discards D. Hence s will be t-shared through the degree-t polynomial $q(\cdot)$. Moreover, during 7BGW-VSS-Rec, the honest parties correctly reconstruct $q(\cdot)$ and hence s. This follows from the properties of RS-Dec and the fact that $q(\cdot)$ is a degree-t polynomial and at most t corrupt parties can send incorrect shares. Hence, the correctness property is guaranteed.

Let $\mathcal{C}$ be the set of corrupt parties. If D is *honest*, then throughout 7BGW-VSS-Sh, the view of Adv consists of $\{f_i(x), g_i(y)\}_{P_i \in \mathcal{C}}$. Moreover, no honest party accuses D and hence all the information which D makes public can be derived from $\{f_i(x), g_i(y)\}_{P_i \in \mathcal{C}}$. Now since these polynomials are derived from $F(x, y)$, which is a randomly chosen polynomial embedding $q(\cdot)$, it follows from Lemma 2.8 that the view of Adv is distributed independently of s, thus guaranteeing privacy.

For strong commitment we have to consider a *corrupt* D. If the honest parties discard D, then clearly the value $s^\star \stackrel{def}{=} 0$ will be t-shared and the same value 0 gets reconstructed during 7BGW-VSS-Rec. On the other hand, consider the case when the honest parties do not discard D during 7BGW-VSS-Sh. In this case we claim that the row and column-polynomials of all the *honest* parties are derived from a single degree-(t, t) bivariate polynomial, say $F^\star(x, y)$, which we call as D's committed bivariate polynomial. Consequently, $s^\star \stackrel{def}{=} F^\star(0, 0)$ will be t-shared through the Shamir-sharing polynomial $q^\star(\cdot) \stackrel{def}{=} F^\star(0, y)$ and $s^\star$ gets reconstructed during 7BGW-VSS-Rec.

To prove the claim, we first note that there are at least $n - t = 2t + 1$ parties who do not broadcast an `accuse` message against D (as otherwise D is discarded). Let $\mathcal{H}$ be the set of *honest* parties among these $n - t$ parties. It is easy to see that $|\mathcal{H}| \geq n - 2t = t + 1$. The parties in $\mathcal{H}$ receive degree-t row and column-polynomials from D. Moreover, for every $P_i, P_j \in \mathcal{H}$, their row and column-polynomials are pair-wise consistent (as otherwise either P_i or P_j would broadcast an `accuse` message against D). It then follows from Lemma 2.7 that the row and column-polynomials of all the parties in $\mathcal{H}$ lie on a single degree-(t, t) bivariate polynomial, say $F^\star(x, y)$. Next consider any *honest* party $P_j \notin \mathcal{H}$, who broadcasts an `accuse` message against D and corresponding to which D makes public the row and column-polynomials of P_j. To complete the proof of the claim, we need to show that these polynomials also lie on $F^\star(x, y)$. We show it for the row-polynomial of P_j and a similar argument can be used for the column-polynomial as well. So let $f_j(x)$ be the degree-t row-polynomial broadcast by D for P_j. It follows that $f_j(\alpha_i) = g_i(\alpha_j)$ holds for *every* $P_i \in \mathcal{H}$, where $g_i(y)$ is the degree-t column polynomial held by P_i (otherwise P_i would have broadcast an `accuse` message against D). Now $g_i(y) = F^\star(\alpha_i, y)$ holds. Moreover, since $|\mathcal{H}| \geq t + 1$, the distinct points $\{(\alpha_j, g_i(\alpha_j))\}_{P_i \in \mathcal{H}}$ uniquely determine the degree-t polynomial $F^\star(x, \alpha_j)$. This implies that $f_j(x) = F^\star(x, \alpha_j)$, as two different degree-t polynomials can have at most t common points. □

4.1.2 A 5-round Version of Protocol 7BGW-VSS-Sh

Protocol 7BGW-VSS-Sh follows the "share-complaint-resolve" paradigm, where D first distributes the information on its bivariate polynomial, followed by parties complaining about any "inconsistency", which is followed by D consistently resolving these inconsistencies. At the end, either all (honest) parties held consistent polynomials derived from a single degree-(t, t) bivariate polynomial or D is discarded. The "complaint" and "resolve" phases of 7BGW-VSS-Sh occupied *five* rounds. In [31], the authors proposed a *round-reducing* technique, which collapses these phases to *three* rounds, thus reducing the overall number of rounds to five. The modified protocol 5BGW-VSS-Sh is presented in Fig 3.

The high level idea of the protocol 5BGW-VSS-Sh is as follows. In 7BGW-VSS-Sh, during the third round, P_i broadcasts *only* the identity of the parties with which it has a dispute (through L_i), followed by D making the corresponding disputed values public during Round IV, which is further followed by P_i accusing D during Round V, if P_i finds D's version to mis-match with P_i's version. Let us call P_i to be *unhappy*, if it accuses D during Round V. The round-reducing technique of [31] enables to identify the set of unhappy parties $\mathcal{UH}$ by the end of Round IV as follows. During Round III, apart from broadcasting the list of disputed parties, party P_i also makes public its version of the corresponding disputed values. In response, both D and the corresponding complainee party makes public their respective version of the disputed value. Now based on whether D's version matches the

complainant's version or complainee's version, the parties can identify the set $\mathcal{UH}$.

In 7BGW-VSS-Sh, once $\mathcal{UH}$ is identified, D makes public the polynomials of the parties in $\mathcal{UH}$ during Round VI. And to verify if D made public the correct polynomials, during Round VII, the parties *not in* $\mathcal{UH}$ raise accusations against D, if they find any inconsistency between the polynomials held by them and the polynomials made public by D. The round-reducing technique of [31] collapses these two rounds into a single round. Namely, once $\mathcal{UH}$ is decided, D makes public the row-polynomials of these parties. In *parallel*, the parties not in $\mathcal{UH}$ makes public the corresponding supposedly common values on these row-polynomials. Now to check if D broadcasted correct row-polynomials, one just has to verify whether each broadcasted row-polynomial is pair-wise consistent with at least $2t+1$ corresponding values, broadcasted by the parties not in $\mathcal{UH}$.

Protocol 5BGW-VSS-Sh

- **Rounds I and II** — same as in protocol 7BGW-VSS-Sh.
- **Round III — each party P_i does the following**:
  - For every $P_j \in \mathcal{P}$ where $f_{ji} \neq g_i(\alpha_j)$, broadcast $(\texttt{complaint}, i, j, g_i(\alpha_j))$.
- **Round IV (making disputed values public) — the dealer and each party P_j does the following**:
  - If P_i broadcasts $(\texttt{complaint}, i, j, g_i(\alpha_j))$, then D and P_j broadcasts $F(\alpha_i, \alpha_j)$ and $f_j(\alpha_i)$ respectively.
- **Local computation (deciding unhappy parties) — each party P_k does the following**:
  - Initialize a set of *unhappy parties* $\mathcal{UH}$ to $\emptyset$.
  - For every pair of parties (P_i, P_j), such that P_i has broadcast $(\texttt{complaint}, i, j, g_i(\alpha_j))$, D has broadcast $F(\alpha_i, \alpha_j)$ and P_j has broadcast $f_j(\alpha_i)$, do the following.
    - If $g_i(\alpha_j) \neq F(\alpha_i, \alpha_j)$, then include P_i to $\mathcal{UH}$.
    - If $f_j(\alpha_i) \neq F(\alpha_i, \alpha_j)$, then include P_j to $\mathcal{UH}$.
  - If $|\mathcal{UH}| > t$, then discard D.
- **Round V (resolving unhappy parties) — the dealer and each $P_j \notin \mathcal{UH}$ does the following**:
  - For every $P_i \in \mathcal{UH}$, the dealer D broadcasts degree-t polynomial $f_i(x)$.
  - For every $P_i \in \mathcal{UH}$, party P_j broadcasts $g_j(\alpha_i)$.
- **Output decision — each party P_k does the following**:
  - If there exists any $P_i \in \mathcal{UH}$ for which D broadcasts $f_i(x)$ and at most $2t$ parties $P_j \notin \mathcal{UH}$ broadcast $g_j(\alpha_i)$ values where $f_i(\alpha_j) = g_j(\alpha_i)$ holds, then discard D.
  - Else output the *share* $f_k(0)$.

Figure 3: A simplified 5-round version of 7BGW-VSS-Sh due to [31].

4.1.3 The 4-round 4GIKR-VSS Scheme

In [31], the authors proposed another round-reducing technique to reduce the number of rounds of 5BGW-VSS-Sh by one. The modified protocol 4GIKR-VSS-Sh is presented in Fig 4. The idea is to ensure that the set $\mathcal{UH}$ is decided by the end of Round III, even though this might look like an impossible task. This is because the results of pair-wise consistency checks are available during Round III and only after the various versions of the disputed values are made public during Round IV, the set $\mathcal{UH}$ gets decided. The key-observation of [31] is that the parties can "initiate" the pair-wise consistency checks from Round I itself. More specifically, every P_i, P_j exchange random pads *privately* during the Round I, independently of D's distribution of the polynomials. Once P_i, P_j obtain their respective polynomials, they can broadcast a masked version of the supposedly common values on their polynomials, using the exchanged pads as the masks. If D, P_i and P_j are all *honest*, then the masked version of the common values will be the same. Moreover, nothing about the common values will be learnt, as the corresponding masks will be known only to P_i, P_j. So by comparing the masked versions of the common values, the parties will publicly come to know about the results of

pair-wise consistency checks by the end of the Round II. After this, the rest of the steps will be the same as in 5BGW-VSS-Sh.

Protocol 4GIKR-VSS-Sh

- **Round I (sending polynomials and exchanging random pads)**
  - D computes and distributes the row and column-polynomials as per round I of 7BGW-VSS-Sh.
  - Each $P_i \in \mathcal{P}$ picks a random pad $r_{ij} \in \mathbb{F}$ corresponding to every $P_j \in \mathcal{P}$ and sends r_{ij} to P_j.
- **Round II (broadcasting common values in a masked fashion) — each P_i does the following**:
  - Broadcast $a_{ij} \stackrel{def}{=} f_i(\alpha_j) + r_{ij}$ and $b_{ij} \stackrel{def}{=} g_i(\alpha_j) + r'_{ji}$, where r'_{ji} is the pad, received from P_j.
- **Round III (making disputed values public) — for all P_i, P_j where $a_{ij} \neq b_{ji}$, party P_i, P_j and D does the following**:
  - D broadcasts $F(\alpha_j, \alpha_i)$.
  - P_i broadcasts $f_i(\alpha_j)$.
  - P_j broadcasts $g_j(\alpha_i)$.
- **Remaining steps** — same as 5BGW-VSS-Sh from the local computations at the end of Round IV onwards.

Figure 4: A 4-round sharing phase protocol due to [31].

4.1.4 The 3-round 3GIKR-VSS Scheme

We next present a VSS scheme called 3GIKR-VSS from [31], which has a *round-optimal* sharing phase, namely a 3-round sharing phase. However, the protocol is *inefficient*, as it requires an exponential (in n and t) amount of computation and communication. The computations in 3GIKR-VSS scheme are performed over a finite group $(\mathbb{G}, +)$.

We first explain the notion of t-out-of-n *replicated secret-sharing* (RSS) [34]. Let $K \stackrel{def}{=} \binom{n}{t}$ and $A_1, \ldots, A_K$ denote the set of all possible subsets of $\mathcal{P}$ of size t. For $k = 1, \ldots, K$, let $\mathcal{G}_k \stackrel{def}{=} \mathcal{P} \setminus A_k$. It is easy to see that in each $\mathcal{G}_k$, the majority of the parties are *honest*. To share $s \in \mathbb{G}$, the share-generation algorithm of RSS outputs $(v^{(1)}, \ldots, v^{(K)}) \in \mathbb{G}^K$, where $v^{(1)}, \ldots, v^{(K)}$ are random elements, such that $v^{(1)} + \ldots + v^{(K)} = s$ holds. The *share* s_i for P_i is defined as $s_i \stackrel{def}{=} \{v^{(j)}\}_{P_i \in \mathcal{G}_j}$.

It is easy to see that any t-sized subset of $(s_1, \ldots, s_n)$ will have at least one "missing" element from $v^{(1)}, \ldots, v^{(K)}$, say $v^{(l)}$, whose probability distribution will be independent of s, thus ensuring privacy. On the other hand, any $(t+1)$-sized subset of $(s_1, \ldots, s_n)$ will have all values $v^{(1)}, \ldots, v^{(K)}$ which can be added to reconstruct back s, thus ensuring correctness. Following Definition 2.2, we say that $s \in \mathbb{G}$ *is RSS-shared*, if it is secret-shared as per t-out-of-n RSS. That is, if there exist $v^{(1)}, \ldots, v^{(K)}$ such that $s = v^{(1)} + \ldots + v^{(K)}$, with all parties in $\mathcal{G}_k$ holding $v^{(k)}$.

Scheme 3GIKR-VSS is presented in Fig 5. The sharing protocol 3GIKR-VSS-Sh verifiably generates a replicated secret-sharing of dealer's secret, maintaining its privacy if D is *honest*. The verifiability ensures that even if D is *corrupt*, there exists some value held by D which has been shared as per RSS by D. The reconstruction protocol 3GIKR-VSS-Rec allows the parties to reconstruct the RSS-shared value of D. During 3GIKR-VSS-Sh, D generates a vector of values $(v^{(1)}, \ldots, v^{(K)})$ as per the share-generation algorithm of RSS and sends $v^{(k)}$ to all the parties in $\mathcal{G}_k$. If D is *honest*, then this ensures the privacy of s. This is because if Adv corrupts the set of t parties in the set A_k, then it will not have access to $v^{(k)}$. To ensure that a potentially *corrupt* D has distributed the same v_k to all the (honest) parties in $\mathcal{G}_k$, each pair of parties in $\mathcal{G}_k$ privately exchange their respective copies of $v^{(k)}$ and publicly raises a complaint if they find any inconsistency. To resolve any complaint raised for $\mathcal{G}_k$, D makes public the value $v^{(k)}$ for $\mathcal{G}_k$, thus ensuring that all the parties in $\mathcal{G}_k$ have the same $v^{(k)}$. Notice that this step does not violate the privacy property. This is because if D is *honest* and if $\mathcal{G}_k$ consists of only honest parties, then there will be no pair-wise inconsistency in $\mathcal{G}_k$. So if any

inconsistency is reported for $\mathcal{G}_k$, then either D is *corrupt* or $\mathcal{G}_k$ consists of at least one corrupt party and so making $v^{(k)}$ public does not add any new information to the view of Adv.

The above process of finding and resolving inconsistencies will require *four* rounds, which can be collapsed to three, by using the round-reducing technique of [31] of publicly identifying pair-wise inconsistencies based on the idea of pre-exchanging pair-wise random pads. During 3GIKR-VSS-Rec, the goal of each P_i is to correctly obtain $v^{(1)}, \ldots, v^{(K)}$. While P_i will have all the $v^{(k)}$ values corresponding to the sets $\mathcal{G}_k$ of which it is a member, the challenge is to obtain the $v^{(k)}$ values for the sets $\mathcal{G}_k$ for which P_i is *not* a member. Consider any such $\mathcal{G}_k$ where $P_i \notin \mathcal{G}_k$ and where all (honest) parties hold the same value of $v^{(k)}$. To enable P_i obtain $v^{(k)}$, every party in $\mathcal{G}_k$ sends $v^{(k)}$ to P_i, who applies the majority rule to filter out the correct value of $v^{(k)}$.

Scheme 3GIKR-VSS

Sharing Phase: Protocol 3GIKR-VSS-Sh

- **Round I (distributing shares and exchanging random pads)**
  - D on having the input $s \in \mathbb{G}$, randomly selects $v^{(1)}, \ldots, v^{(K)} \in \mathbb{G}$ such that $s = v^{(1)} + \ldots + v^{(K)}$ holds. It then sends $v^{(k)}$ to every party $P_i \in \mathcal{G}_k$, for $k = 1, \ldots, K$.
  - For $k = 1, \ldots, K$, each $P_i \in \mathcal{G}_k$ sends a randomly chosen pad $r_{ij}^{(k)} \in \mathbb{G}$ to every $P_j \in \mathcal{G}_k$ where $i < j$.
- **Round II (pair-wise consistency check within each group)**
  - For $k = 1, \ldots, K$, each pair of parties $P_i, P_j \in \mathcal{G}_k$ with $i < j$ do the following:
    - P_i broadcasts $a_{ij}^{(k)} = v_i^{(k)} + r_{ij}^{(k)}$, where $v_i^{(k)}$ denotes the version of $v^{(k)}$ received by P_i from D.
    - P_j broadcasts $a_{ji}^{(k)} = v_j^{(k)} + r_{ij}^{\prime(k)}$, where $v_j^{(k)}$ denotes the version of $v^{(k)}$ received by P_j from D and $r_{ij}^{\prime(k)}$ denotes the pad received by P_j from P_i.
- **Round III (resolving conflicts)**
  - For $k = 1, \ldots, K$, if there exists $P_i, P_j \in \mathcal{G}_k$ such that $a_{ij}^{(k)} \neq a_{ji}^{(k)}$, then D broadcasts the value $v^{(k)}$.
- **Output determination — each party P_i does the following**:
  - If $\exists k \in \{1, \ldots, K\}$ where $P_i \in \mathcal{G}_k$ such that D broadcast $v^{(k)}$, then set $v_i^{(k)}$ to $v^{(k)}$.
  - Output the *share* $s_i \stackrel{def}{=} \{v_i^{(k)}\}_{P_i \in \mathcal{G}_k}$.

Reconstruction Phase: Protocol 3GIKR-VSS-Rec

Each party $P_i \in \mathcal{P}$ does the following:
- $\forall k \in \{1, \ldots, K\}$, such that $P_i \in \mathcal{G}_k$, send $v_i^{(k)}$ to every party in $\mathcal{P} \setminus \mathcal{G}_k$.
- $\forall k \in \{1, \ldots, K\}$ where $P_i \notin \mathcal{G}_k$, set $v^{(k)}$ to be the value $v_j^{(k)}$ received from at least $t+1$ parties $P_j \in \mathcal{G}_k$.
- Output $s = \sum_{\mathcal{G}_k : P_i \in \mathcal{G}_k} v_i^{(k)} + \sum_{\mathcal{G}_k : P_i \notin \mathcal{G}_k} v^{(k)}$.

Figure 5: The 3-round 3GIKR-VSS scheme due to [31].

4.1.5 The 3-round 3FGGRS-VSS Type-I VSS Scheme

The 4GIKR-VSS scheme comes closest in terms of number of rounds to obtain a round optimal and *efficient* VSS scheme. Round IV of 4GIKR-VSS-Sh consists of the dealer making public the polynomials of the so-called *unhappy* parties. In [30], the authors observed that the elimination of Round IV results in a primitive that satisfies a *weaker* commitment requirement, where the reconstructed value may be some predefined default value, when the dealer is *corrupt*. This primitive is called *weak* verifiable secret sharing (WSS) [47] and is used as a building block to construct a VSS scheme. WSS

is used by [30] to construct an efficient and round optimal VSS scheme. We first present the required background and the protocol for WSS of [30] before presenting the 3FGGRS-VSS scheme.

Definition 4.2 ((n,t)-WSS [47])**.** Let $(\mathsf{Sh}, \mathsf{Rec})$ be a pair of protocols for the n parties in $\mathcal{P}$, where a designated $\mathsf{D} \in \mathcal{P}$ has some private input $s \in \mathcal{S}$ for the protocol Sh. Then $(\mathsf{Sh}, \mathsf{Rec})$ is called a *perfectly-secure* (n,t)*-WSS scheme*, if the following requirements hold.

- **Privacy and Correctness**: Same as in VSS.
- **Weak Commitment**: Even if D is *corrupt*, in any execution of Sh the joint view of the honest parties defines a unique value $s^\star$ (which could be different from s), such that each honest party outputs either $s^\star$ or some default value $\perp$ at the end of Rec, irrespective of Adv.

The WSS scheme 3FGGRS-WSS of [30] is given in Fig. 6. Protocol 3FGGRS-WSS-Sh is the same as 4GIKR-VSS-Sh, except that the parties do not execute the Round IV. Consequently, upto t parties (namely the parties in $\mathcal{UH}$) will not posses their respective shares. As a result, during 3FGGRS-WSS-Rec, only the *happy* parties (namely the parties *not* in the set $\mathcal{UH}$) are allowed to participate for reconstructing the shared value. For reconstruction, the happy parties broadcast their respective polynomials. To ensure that each happy party broadcasts correct polynomials, the parties publicly verify the pair-wise consistency of these polynomials. And the polynomials which are *not* found to be pair-wise consistent with "sufficiently many" polynomials are not considered for the purpose of reconstruction. If the parties are finally left with at least $n-t$ pair-wise consistent row and column-polynomials, then these row-polynomials are used to reconstruct back D's committed Shamir-sharing polynomial and hence its secret, else the parties output $\perp$. The idea is that if the parties are finally left with $n-t$ pair-wise consistent polynomials, then they lie on the same degree-(t,t) bivariate polynomial as committed by D to *honest* happy parties during the sharing phase, as among these $n-t$ pairs of polynomials, at least $t+1$ belong to the happy honest parties.

We stress that if D is *honest*, then *no* honest party will be in the set $\mathcal{UH}$ and hence all honest parties will have their respective shares of D's Shamir-sharing polynomial. Moreover, if any potentially *corrupt* party produces an incorrect polynomial during the reconstruction phase, then it will be ignored due to the pair-wise consistency checks. Thus 3FGGRS-WSS achieves the properties of a type-II VSS, for the case of an *honest* D. However if D is *corrupt*, then during the sharing phase, upto t *honest* parties may belong to $\mathcal{UH}$. Moreover, during the reconstruction phase, even if a single corrupt party produces incorrect polynomials, then the parties end up reconstructing $\perp$. It is this latter phenomenon, which prevents 3FGGRS-WSS from being a VSS scheme.

Scheme 3FGGRS-WSS

Sharing Phase: Protocol 3FGGRS-WSS-Sh

- The parties execute the first 3 rounds of 4GIKR-VSS-Sh. Let $\mathcal{UH}$ be the set of *unhappy* parties and let $\mathcal{W} \stackrel{def}{=} \mathcal{P} \setminus \mathcal{UH}$ be the set of *happy* parties. If $|\mathcal{UH}| > t$, then discard D.

Reconstruction Phase: Protocol 3FGGRS-WSS-Rec

- **Revealing private information — each party $P_i \in \mathcal{W}$ does the following**:
  - Broadcast the polynomials $f_i(x)$ and $g_i(y)$.
- **Consistency check — each party P_i does the following**:
  - Construct a *consistency graph* G over the set of parties $\mathcal{W}$ with an edge between P_j and P_k if and only if $f_j(\alpha_k) = g_k(\alpha_j)$ and $g_j(\alpha_k) = f_k(\alpha_j)$.
  - Remove P_j from G, if it has degree less than $n-t$ in G. Repeat till no more nodes can be removed from G. Redefine $\mathcal{W}$ to be the set of parties, whose corresponding nodes remain in G.
- **Output decision — each party P_i does the following**:

– If $|\mathcal{W}| < n - t$ then output a default value $\perp$. Else interpolate a degree-t polynomial $q(\cdot)$ through the points $\{(\alpha_j, f_j(0))\}_{P_j \in \mathcal{W}}$ and output $q(\cdot)$ and $s = q(0)$.

Figure 6: The 3-round 3FGGRS-WSS scheme due to [30]

Sharing and Reconstructing Polynomial Using 3FGGRS-WSS We note that D's computation in 3FGGRS-WSS-Sh can be recasted as if D wants to share the degree-t Shamir-sharing polynomial $F^\star(0, y)$, where $F^\star(0, 0)$ is the value which D wants to share. If D is not discarded, then each (honest) $P_j \in \mathcal{W}$ receives the share $F^\star(0, \alpha_j)$ from D through its degree-t row-polynomial $F^\star(x, \alpha_j)$. Here $F^\star(x, y)$ is the degree-(t, t) bivariate polynomial committed by D to the honest parties in $\mathcal{W}$, which is the same as $F(x, y)$ for an *honest* D. If D is *honest*, then adversary learns at most t shares of the polynomial $F(0, y)$ and hence its view will be independent of $F(0, 0)$. The computations done during 3FGGRS-WSS-Rec can be similarly recasted as if the parties publicly try to reconstruct a degree-t Shamir-sharing polynomial $F^\star(0,$ y$)$, which has been shared by D during 3FGGRS-WSS-Sh. If D is *honest*, then the parties robustly reconstruct the shared polynomial. Else, the parties either reconstruct the shared polynomial or output $\perp$. Hence we propose the following notations for 3FGGRS-WSS, which later simplifies the presentation of 3FGGRS-VSS.

Notation 4.3. We use the following notations and interpretations.

- We say that party $P_j \in \mathcal{P}$ *shares a degree-t polynomial* $r(\cdot)$ held by P_j, to denote that P_j plays the role of D and invokes an instance of 3FGGRS-WSS-Sh by selecting $r(\cdot)$ as its Shamir-sharing polynomial and all the parties participate in this instance.
- We say that P_i *receives a wss-share* r_{ji} *from* P_j, to denote that in Round I of the 3FGGRS-WSS-Sh instance invoked by P_j, P_i receives a degree-t row-polynomial from P_j, whose constant term is r_{ji}. If P_j is not discarded during the 3FGGRS-WSS-Sh instance, then the wss-shares r_{ji} of all the *honest* parties in $\mathcal{W}$ lie on a unique degree-t Shamir-sharing polynomial held by P_j.
- Let $r(\cdot)$ be a degree-t polynomial shared by P_j through an instance of 3FGGRS-WSS-Sh. We say that the *parties try to reconstruct P_j's shared polynomial*, to denote that the parties execute the corresponding instance of 3FGGRS-WSS-Rec, which either outputs $r(\cdot)$ or $\perp$.

From WSS to VSS The sharing-phase protocol of the 3FGGRS-VSS scheme (see Fig. 7) is exactly the same as the first three rounds of the protocol 4GIKR-VSS-Sh with the following twist. The random pads r_{ji} used by party P_j to verify the pair-wise consistency of its row-polynomial with the other parties' column-polynomials are "tied together" by letting these pads lie on a random degree-t *blinding-polynomial* $r_j(\cdot)$, which is shared by P_j (see Notation 4.3). For pair-wise consistency, P_j makes public a degree-t polynomial $A_j(\cdot)$, which is a masked version of its row-polynomial and its blinding-polynomial, while the parties P_i make public the supposedly common value on their column-polynomials, blinded with the wss-shares of P_j's blinding-polynomial. This new way of performing pair-wise consistency checks achieves the same "goals" as performing the checks using random pads. More importantly, the privacy is still maintained in the case of an *honest* D. This is because for every *honest* P_j, the adversary obtains at most t shares of P_j's blinding-polynomial, which are randomly distributed. The set of happy parties for the VSS is identified based on the results of pair-wise consistency and conflict-resolutions, as done in 4GIKR-VSS-Sh. Moreover, the parties also ensure that for every happy party P_j for the VSS, there are at least $n - t$ happy parties, belonging to the happy set of parties for P_j's WSS-sharing instance. This latter property is crucial for ensuring the strong commitment property during the reconstruction phase.

Let $F^\star(x, y)$ be the degree-(t, t) bivariate polynomial, committed by D during the sharing phase. During the reconstruction phase, instead of simply asking the *happy* parties to make their row-

polynomials public, the parties reconstruct their blinding-polynomials (by executing instances of 3FGGRS-WSS-Rec), which are then unmasked from the corresponding $A_j(\cdot)$ polynomials to get back the row-polynomials of the happy parties. For the *honest* happy parties P_j, robust reconstruction of their blinding-polynomials is always guaranteed, thus ensuring that their corresponding row-polynomials $F^\star(x, \alpha_j)$ are robustly reconstructed. If the reconstruction of P_j's blinding-polynomial fails, then clearly P_j is *corrupt* and hence can be safely discarded from consideration during the reconstruction phase. Moreover, even if P_j is *corrupt* and the parties reconstruct a degree-t polynomial during the corresponding 3FGGRS-WSS-Rec instance, the *weak-commitment* property of the WSS guarantees that reconstructed polynomial is the same blinding-polynomial as shared by P_j. Hence unmasking this polynomial from $A_j(\cdot)$ will return back the row-polynomial $F^\star(x, \alpha_j)$. This is because there are at least $n - t$ parties, which belong to the happy set of *both* the VSS instance, as well as P_j's WSS instance. Now out of these $n-t$ common happy-parties, at least $t+1$ will be *honest*. And the wss-shares received by these honest parties P_k from P_j uniquely define P_j's blinding-polynomial $r_j(x)$, while evaluations of the column-polynomials of the same honest-parties P_k at $y = \alpha_j$ uniquely determines the degree-t row-polynomial $F^\star(x, \alpha_j)$. Moreover, during the sharing phase these honest parties P_k collectively ensured that $A_j(\cdot) = F^\star(x, \alpha_j) + r_j(\cdot)$ holds, as otherwise they do not belong to the happy set of P_j's WSS instance.

Scheme 3FGGRS-VSS

Sharing Phase: Protocol $\mathsf{3FGGRS\text{-}VSS_{Sh}}$

- **Round I (sending polynomials and exchanging random pads)**:
  - D computes and distributes the row and column-polynomials as per round I of 7BGW-VSS-Sh.
  - Each party $P_i \in \mathcal{P}$ (including D) picks a random degree-t *blinding-polynomial* $r_i(\cdot)$ and shares it through an instance $\mathsf{WSS{-}Sh}_i$ of 3FGGRS-WSS-Sh.
- **Round II (broadcasting common values in a masked fashion) — each P_i does the following**:
  - Broadcast the degree-t polynomial $A_i(\cdot) \stackrel{def}{=} f_i(x) + r_i(\cdot)$.
  - Broadcast $b_{ij} \stackrel{def}{=} g_i(\alpha_j) + r'_{ji}$, where r'_{ji} denotes the wss-share received from P_j during $\mathsf{WSS{-}Sh}_j$.
  - For every $k \in \{1, \ldots, n\}$, concurrently execute Round II of the instance $\mathsf{WSS{-}Sh}_k$.
- **Round III**:
  - **(making disputed values public)** — for all P_i, P_j where $A_i(\alpha_j) \neq b_{ji}$, dealer D broadcasts $F(\alpha_j, \alpha_i)$, party P_i broadcasts $f_i(\alpha_j)$ and party P_j broadcasts $g_j(\alpha_i)$.
  - For every $k \in \{1, \ldots, n\}$, the parties concurrently execute Round III of the instance $\mathsf{WSS{-}Sh}_k$.
- **Local computation at the end of Round III — each party P_k does the following**:
  - Initialize a set $\mathcal{UH}$ of *unhappy* parties. For every P_i, P_j where $A_i(\alpha_j) \neq b_{ji}$, do the following.
    - Include $P_i \in \mathcal{UH}$, if during Round III, $F(\alpha_j, \alpha_i) \neq f_i(\alpha_j)$ holds.
    - Include $P_j \in \mathcal{UH}$, if during Round III, $F(\alpha_j, \alpha_i) \neq g_j(\alpha_i)$ holds.

    Let $\mathcal{V} \stackrel{def}{=} \mathcal{P} \setminus \mathcal{UH}$ be the set of *happy* parties.
  - For $j \in \{1, \ldots, n\}$, let $\mathcal{W}_j$ denote the set of *happy* parties at the end of the instance $\mathsf{WSS{-}Sh}_j$. Remove P_i from $\mathcal{W}_j$, if during Round II, $A_j(\alpha_i) \neq b_{ij}$ holds.
  - For every $P_j \in \mathcal{V}$, if $|\mathcal{V} \cap \mathcal{W}_j| < n - t$, then remove P_j from $\mathcal{V}$. Repeat this step till no more parties can be removed from $\mathcal{V}$. If $|\mathcal{V}| < n - t$, then discard D.

Reconstruction Phase: Protocol $\mathsf{3FGGRS\text{-}VSS_{Rec}}$

- **Reconstructing the blinding-polynomials**:
  - For every $P_j \in \mathcal{V}$, the parties try to reconstruct P_j's blinding-polynomial by participating in an instance $\mathsf{WSS{-}Rec}_j$ of 3FGGRS-WSS-Rec. P_j is removed from $\mathcal{V}$ if $\perp$ is the output during $\mathsf{WSS{-}Rec}_j$.
- **Reconstruction and output decision — each party P_i does the following**:

– For each $P_j \in \mathcal{V}$, compute $f_j(x) = A_j(x) - r_j(\cdot)$, where $r_j(\cdot)$ is reconstructed at the end of $\mathsf{WSS{-}Rec}_j$. Interpolate a degree-t polynomial $q(\cdot)$ through $\{(\alpha_j, f_j(0))\}_{P_j \in \mathcal{V}}$. Output $s = q(0)$.

Figure 7: The 3-round 3FGGRS-VSS scheme due to [30]

4.1.6 The 3-round 3KKK-VSS Scheme

The 3FGGRS-VSS scheme is a Type-I VSS for a *corrupt* D. Moreover, it makes use of the broadcast channel during two of the rounds of the sharing phase, which is *not* optimal. The 3KKK-VSS scheme [36] rectifies both these problems.

The optimal broadcast-channel usage must first be rectified for 3FGGRS-WSS-Sh. The modified construction 3KKK-WSS-Sh (see Fig 8) is based on the observation that during Round II of 3FGGRS-WSS-Sh (which is same as Round II of 4GIKR-VSS-Sh), there is no need to *publicly* perform the pair-wise consistency checks over *masked* values. Instead, parties can first *privately* perform the pair-wise consistency checks over *unmasked* values and later *publicly* announce the results during the third round. However, we also need to add a provision for the dealer to resolve any potential conflicts during the third round itself. For this, during the second round, every P_i, P_j privately exchange their supposedly common values and also the random pads. Additionally, the pads are also "registered" with the dealer. Later during the third round, upon a disagreement between P_i and P_j, they broadcast their respective common values and their respective random pads, else they broadcast their appropriately masked common values. In parallel, dealer either broadcasts the common value in a masked fashion if the pads it received from P_i, P_j are same, else it just broadcasts the common value. The secrecy of the common values is maintained if P_i, P_j and dealer are honest. On the other hand, if dealer is corrupt and if there is a disagreement between honest P_i, P_j , then the dealer can "take side" with at most one of them during the third round.

Protocol 3KKK-WSS-Sh

- **Round I (sending polynomials and exchanging random pads)**:
  - Same as Round I of 4GIKR-VSS-Sh. Additionally, each $P_i \in \mathcal{P}$ sends the pads $\{r_{ij}\}_{P_j \in \mathcal{P}}$ to D.
- **Round II (exchanging common values and confirming pad) — each P_i does the following**:
  - Send $a_{ij} = f_i(\alpha_j)$ and $b_{ij} = g_i(\alpha_j)$ to P_j.
  - Send $\{r'_{ji}\}_{P_j \in \mathcal{P}}$ to D, where r'_{ji} denotes the pad received from P_j during Round I.
- **Round III (complaint and resolution) — each party P_i does the following**:
  - For $j \in \{1, \ldots, n\}$, let a'_{ji} and b'_{ji} be the values received from P_j during Round II.
    - If $b'_{ji} \neq f_i(\alpha_j)$, broadcast $(j, \texttt{disagree-row}, f_i(\alpha_j), r_{ij})$, else broadcast $(j, \texttt{agree-row}, f_i(\alpha_j) + r_{ij})$.
    - If $a'_{ji} \neq g_i(\alpha_j)$, broadcast $(j, \texttt{disagree-column}, g_i(\alpha_j), r'_{ji})$, else broadcast $(j, \texttt{agree-column}, g_i(\alpha_j) + r'_{ji})$.
  - If $P_i = \mathsf{D}$, then for every ordered pair of parties (P_j, P_k), *additionally* do the following.
    - Let $r^{(1)}_{jk}$ and $r^{(2)}_{jk}$ be the pads received from P_j and P_k respectively during Round I and Round II. If $r^{(1)}_{jk} \neq r^{(2)}_{jk}$, then broadcast $((j,k), \texttt{NEQ}, F(\alpha_k, \alpha_j))$, else broadcast $((j,k), \texttt{EQ}, F(\alpha_k, \alpha_j) + r^{(1)}_{jk})$.
- **Local computation (identifying unhappy parties) — each party P_k does the following**:
  - Initialize a set of *unhappy* parties $\mathcal{UH}$ to $\emptyset$. For every P_i, P_j such that P_i broadcasts $(j, \texttt{disagree-row}, f_i(\alpha_j), r_{ij})$ and P_j broadcasts $(i, \texttt{disagree-column}, g_j(\alpha_j), r'_{ij})$ where $r_{ij} = r'_{ij}$, do the following.
    - Include P_i to $\mathcal{UH}$, if one of the following holds.
      - During Round III, D broadcasts $((i,j), \texttt{NEQ}, d_{ij})$, such that $d_{ij} \neq f_i(\alpha_j)$.
      - During Round III, D broadcasts $((i,j), \texttt{EQ}, d_{ij})$, such that $d_{ij} \neq f_i(\alpha_j) + r_{ij}$.
    - Include P_j to $\mathcal{UH}$, if one of the following holds.
      - During Round III, D broadcasts $((i,j), \texttt{NEQ}, d_{ij})$, such that $d_{ij} \neq g_j(\alpha_i)$.

– During Round III, D broadcasts $((i,j), \texttt{EQ}, d_{ij})$, such that $d_{ij} \neq g_j(\alpha_i) + r'_{ij}$.

– If $|\mathcal{UH}| > t$, then discard D. Else let $\mathcal{W} \stackrel{def}{=} \mathcal{P} \setminus \mathcal{UH}$ be the set of *happy* parties.

Figure 8: The 3-round 3KKK-WSS-Sh protocol due to [36]

From 3KKK-WSS to 3KKK-VSS We note that the notion of sharing and reconstructing degree-t polynomials (as stated in Notation 4.3) is applicable even for the 3KKK-WSS scheme. By replacing 3FGGRS-WSS with 3KKK-WSS scheme in the 3FGGRS-VSS scheme, readily provides a VSS scheme with the optimal usage of broadcast channel. However, the resultant VSS scheme need not be not be of Type-II for the case of a *corrupt* D, as the *unhappy honest* parties may not hold their respective shares. Hence the 3KKK-VSS scheme deploys an additional trick to get rid of this problem.

Let P_i be an *unhappy* party during 3FGGRS-VSS-Sh and let $F^\star(x,y)$ be the degree-(t,t) bivariate polynomial, defined by the polynomials of the (honest) *happy* parties. We note that each happy P_j broadcasts a masking $A_j(\cdot)$ of its row-polynomial $F^\star(x,\alpha_j)$. If P_i is *happy* in P_j's WSS instance WSS$-$Sh$_j$, then P_i can compute the point $F^\star(\alpha_i,\alpha_j)$ on its supposedly column-polynomial $F^\star(\alpha_i,y)$ by unmasking the wss-share r'_{ji} received during WSS$-$Sh$_j$ from $A_j(\alpha_i)$. Hence if there is a set $\mathcal{SS}_i$ of at least $t+1$ happy parties P_j, who keep P_i happy during their respective WSS$-$Sh$_j$ instances, then using the above procedure, P_i can compute $t+1$ distinct points on its column-polynomial $F^\star(\alpha_i,y)$, which are sufficient for P_i to get $F^\star(\alpha_i,y)$. However, it is not clear how to extend the above approach to enable P_i obtain its row-polynomial $F^\star(x,\alpha_i)$, which is required for P_i to obtain its share of D's Shamir-sharing polynomial $F^\star(0,y)$. The way-out is to let D embed its Shamir-sharing polynomial in a random degree-(t,t) *symmetric bivariate polynomial*, which ensures that $F^\star(x,\alpha_i) = F^\star(\alpha_i,y)$ holds, thus allowing P_i to obtain its required Shamir-share.

The above idea requires broadcast during the second as well as third round. To ensure the optimal usage of broadcast channel, the technique used in the 3KKK-WSS-Sh protocol is deployed, along with postponing the broadcast of masked row-polynomials to the third round. However, this brings additional challenges to filter out the parties from $\mathcal{W}_j$ sets for the individual WSS instances. For example, a *corrupt* D can distribute pair-wise *inconsistent* polynomials in such a way that the masked polynomial $A_j(\cdot)$ broadcast by an *honest happy* party P_j is inconsistent with the corresponding masked value broadcast by an *honest unhappy* party P_i, even though P_i belongs to $\mathcal{W}_j$ during WSS$-$Sh$_j$. Simply removing P_i from $\mathcal{W}_j$ in this case (as done in 3FGGRS-VSS-Sh) might end up resulting in P_i being removed from the $\mathcal{W}_j$ set of every honest happy party. And this may lead to $\mathcal{SS}_i$ set of size less than $t+1$. To prevent this, apart from checking the pair-wise consistency of masked row-polynomials, the parties also carefully consider the results of *private* pair-wise consistency checks performed during the second round, whose results are *public* during the third round. The details are presented in Fig. 9. We stress that even though each party obtains a single degree-t polynomial from D, it is treated both as the row as well as column-polynomial to perform the pair-wise consistency checks. Accordingly, if P_i finds a "negative" result for the *private* pair-wise consistency check with P_j, then it broadcasts a `disagree-row` as well as a `disagree-column` message against P_j. Else it broadcasts just an `agree-column` message for P_j; the `agree-row` message for P_j is assumed to be *implicitly* present in the latter case.

Scheme 3KKK-VSS

Sharing Phase: Protocol 3KKK-VSS-Sh

- **Round I (sending polynomials and exchanging random pads)**:
  - D with secret $s \in \mathbb{F}$ picks a random degree-t *Shamir-sharing polynomial* $q(\cdot)$ such that $q(0) = s$ and picks a *random symmetric* bivariate polynomial $F(x,y)$ such that $F(0,y) = q(\cdot)$. For

$i = 1, \ldots, n$, the dealer D sends only the row-polynomial $f_i(x) \stackrel{def}{=} F(x, \alpha_i)$ to party P_i.
  - Each party $P_i \in \mathcal{P}$ (including D) picks a random degree-t *blinding polynomial* $r_i(\cdot)$ and shares it through an instance $\mathsf{WSS{-}Sh}_i$ of 3KKK-WSS-Sh. In addition, P_i sends the polynomial $r_i(\cdot)$ to D.
- **Round II (exchanging common values and confirming pad) — each P_i does the following**:
  - For $j = 1, \ldots, n$, send $a_{ij} \stackrel{def}{=} f_i(\alpha_j)$ to P_j. Send $\{r'_{ji}\}_{j=1,\ldots,n}$ to D, where r'_{ji} is the wss-share received from P_j during Round I of $\mathsf{WSS{-}Sh}_j$.
  - For $j = 1, \ldots, n$ execute Round II of the instance $\mathsf{WSS{-}Sh}_j$.
- **Round III (complaint and resolution) — each party P_i does the following**:
  - Broadcast the degree-t polynomial $A_i(\cdot) \stackrel{def}{=} f_i(x) + r_i(\cdot)$.
  - For $j \in \{1, \ldots, n\}$, let a'_{ji} be the value received from P_j during Round II.
    - If $a'_{ji} \neq f_i(\alpha_j)$, then broadcast $(j, \texttt{disagree-row}, f_i(\alpha_j), r_{ij})$ and $(j, \texttt{disagree-column}, f_i(\alpha_j), r'_{ji})$.
    - Else broadcast $(j, \texttt{agree-column}, f_i(\alpha_j) + r'_{ji})$.
  - If $P_i = \mathsf{D}$, then for every ordered pair of parties (P_j, P_k), *additionally* do the following.
    - Let $r^{(1)}_{jk}$ and $r^{(2)}_{jk}$ be the pads received from P_j and P_k respectively during Round I and Round II. If $r^{(1)}_{jk} \neq r^{(2)}_{jk}$, then broadcast $((j,k), \texttt{NEQ}, F(\alpha_k, \alpha_j))$, else broadcast $((j,k), \texttt{EQ}, F(\alpha_k, \alpha_j) + r^{(1)}_{jk})$.
  - For every $j \in \{1, \ldots, n\}$, concurrently execute Round III of $\mathsf{WSS{-}Sh}_j$.
- **Local computation at the end of Round III — each party P_k does the following**:
  - Initialize a set of *unhappy* parties $\mathcal{UH}$ to $\emptyset$. For every P_i, P_j such that P_i broadcasts $(j, \texttt{disagree-row}, f_i(\alpha_j), r_{ij})$ and P_j broadcasts $(i, \texttt{disagree-column}, f_j(\alpha_i), r'_{ij})$ where $r_{ij} = r'_{ij}$, do the following.
    - Include P_i to $\mathcal{UH}$, if one of the following holds.
      - During Round III, D broadcasts $((i,j), \texttt{NEQ}, d_{ij})$, such that $d_{ij} \neq f_i(\alpha_j)$.
      - During Round III, D broadcasts $((i,j), \texttt{EQ}, d_{ij})$, such that $d_{ij} \neq f_i(\alpha_j) + r_{ij}$.
    - Include P_j to $\mathcal{UH}$, if one of the following holds.
      - During Round III, D broadcasts $((i,j), \texttt{NEQ}, d_{ij})$, such that $d_{ij} \neq f_j(\alpha_i)$.
      - During Round III, D broadcasts $((i,j), \texttt{EQ}, d_{ij})$, such that $d_{ij} \neq f_j(\alpha_i) + r'_{ij}$.
  - Let $\mathcal{V} \stackrel{def}{=} \mathcal{P} \setminus \mathcal{UH}$ be the set of *happy* parties and for every $P_j \in \mathcal{V}$, let $\mathcal{W}_j$ be the set of happy parties at the end of $\mathsf{WSS{-}Sh}_j$. Remove P_j from $\mathcal{V}$, if any of the following holds:
    - $|\mathcal{W}_j| < n - t$.
    - $\exists i \in \{1, \ldots, n\} : P_j$ broadcasts $(i, \texttt{disagree-row}, f_j(\alpha_i), r_{ji})$ where $A_j(\alpha_i) \neq f_j(\alpha_i) + r_{ji}$.
  - For every $P_j \in \mathcal{V}$, remove P_i from $\mathcal{W}_j$, if any of the following holds.
    - P_i broadcasts $(j, \texttt{agree-column}, y)$ such that $A_j(\alpha_i) \neq y$.
    - P_j broadcasts $(i, \texttt{disagree-row}, f_j(\alpha_i), r_{ji})$ and P_i broadcasts either $(j, \texttt{agree-column}, \star)$ or $(j, \texttt{disagree-column}, \star, r'_{ji})$, where $r'_{ji} \neq r_{ji}$.
  - Remove P_j from $\mathcal{V}$ if $|\mathcal{V} \cap \mathcal{W}_j| < n - t$. Repeat, till no more parties can be removed from $\mathcal{V}$.
  - If $|\mathcal{V}| < n - t$, then discard D.
- **Computing shares — each party $P_i \in \mathcal{P}$ does the following**:
  - If $P_i \in \mathcal{H}$, then output the *share* $f_i(0)$.
  - Else recompute $f_i(x)$ as follows and output the share $f_i(0)$.
    - Initialize a *support set* $\mathcal{SS}_i$ to $\emptyset$. Include P_j to $\mathcal{SS}_i$, if $P_j \in \mathcal{H}$ and $P_i \in \mathcal{H}_j$.
    - Compute a degree-t polynomial $f_i(x)$ by interpolating the points $\{(\alpha_j, A_j(\alpha_i) - r'_{ji})\}_{P_j \in \mathcal{SS}_i}$.

Reconstruction Phase: Protocol 3KKK-VSS-Rec

Same as the protocol 7BGW-VSS-Rec.

Figure 9: The 3-round 3KKK-VSS scheme due to [36]

4.1.7 The 3-round 3AKP-VSS Type-II VSS Scheme

In [3], it is shown that 4 rounds are necessary and sufficient for securely computing any n-party degree-2 functionality with perfect security and optimal resilience $t < n/3$. To design their 4-round MPC protocol, they rely on a 3-round Type-II perfectly-secure VSS which should ensure that if D is not discarded at the end of sharing phase, then one of the following holds for every *honest* party P_i at the end of Round II.

- P_i holds its *tentative Shamir-share* of the underlying secret;
- P_i holds at least $t+1$ *tentative shares* of its Shamir-share, which we call as *share-shares*.

Moreover, it is also required that at the end of Round III, either the tentative share or the tentative share-shares should turn out to be correct (which of these is going to be the case need not be known to P_i at the end of Round II). The 3FGGRS-VSS scheme fails to satisfy the above requirements, as it is not a Type-II VSS. The 3KKK-VSS scheme also fails to satisfy the above requirements. This is because if $P_i \notin \mathcal{V}$, then it obtains its share-shares through the set of parties in $\mathcal{SS}_i$ only at the end of Round III, as the parties broadcast their masked row-polynomials only during Round III. Hence in [3], a new 3-round VSS scheme called 3AKP-VSS (see Fig 10) is presented, satisfying the above requirements. The scheme is obtained by tweaking the 3FGGRS-VSS scheme and by borrowing the idea of symmetric bivariate polynomial from the 3KKK-VSS scheme.

Scheme 3AKP-VSS

Sharing Phase: Protocol 3AKP-VSS-Sh

- **Round I**: Same as Round I of 3FGGRS-VSS-Sh, except that D uses a random degree-(t,t) *symmetric* bivariate polynomial $F(x,y)$ and distributes only the row-polynomial $f_i(x) = F(x, \alpha_i)$ to every P_i.
- **Round II**: Same as Round II of 3FGGRS-VSS-Sh, except that $b_{ij} \stackrel{def}{=} f_i(\alpha_j) + r'_{ji}$. Moreover, every party P_i sets $s_i \stackrel{def}{=} f_i(0)$ as its *tentative Shamir-share* and $\{A_j(\alpha_i) - r'_{ji}\}_{P_j \in \mathcal{P}}$ as its *tentative share-shares*.
- **Round III**: Same as Round III of 3FGGRS-VSS-Sh, except that for every P_i, P_j where $A_i(\alpha_j) \neq b_{ji}$, party P_i broadcasts $(f_i(\alpha_j), r_{ij})$, party P_j broadcasts $(f_j(\alpha_i), r'_{ij})$ and D broadcasts $F(\alpha_j, \alpha_i)$.
- **Local computation at the end of Round III — each party** P_k:
  - Compute the sets $\mathcal{UH}$ and $\mathcal{V}$ as in 3FGGRS-VSS-Sh, based on every P_i, P_j for which $A_i(\alpha_j) \neq b_{ji}$.
  - Remove P_i from $\mathcal{W}_j$, if $A_j(\alpha_i) \neq b_{ij}$ during Round II and $r_{ji} \neq r'_{ji}$ during Round III.
  - Remove P_j from $\mathcal{V}$ if there exists some $i \in \{1, \ldots, n\}$, such that P_j broadcasts $(f_j(\alpha_i), r_{ji})$ during Round III and $A_j(\alpha_i) \neq f_j(\alpha_i) + r_j(\alpha_i)$.
  - Remove P_j from $\mathcal{V}$, if $|\mathcal{V} \cap \mathcal{W}_j| < n - t$. Repeat, till no more parties can be removed from $\mathcal{V}$. If $|\mathcal{V}| < n - t$, then discard D.
- **Computing shares — each party** $P_i \in \mathcal{P}$: compute the shares as in the protocol $\mathsf{3KKK\text{-}VSS_{Sh}}$.

Reconstruction Phase: Protocol 3AKP-VSS-Rec

Same as the protocol 7BGW-VSS-Rec.

Figure 10: The 3-round 3AKP-VSS scheme due to [3]

4.2 VSS Protocol with $n > 4t$

In this section, we present a perfectly-secure VSS scheme 2GIKR-VSS with $n > 4t$ due to [31], which requires 2 rounds in the sharing phase. Before proceeding further, we first recall a data structure called (n,t)-star (which we often call as just star) from [10].

Definition 4.4 ((n,t)-star [10])**.** Let G be an undirected graph over $\mathcal{P}$. Then a pair of subset of nodes $(\mathcal{C}, \mathcal{D})$ where $\mathcal{C} \subseteq \mathcal{D} \subseteq \mathcal{P}$ is called an (n,t)-star, if all the following hold.

– $|\mathcal{C}| \geq n - 2t$ and $|\mathcal{D}| \geq n - t$.
– For every $P_i \in \mathcal{C}$ and every $P_j \in \mathcal{D}$, the edge (P_i, P_j) is present in G.

In [10], an efficient algorithm is provided for checking the presence of a star in a given graph. The algorithm outputs either a star, or $\perp$ (indicating that no star is present). Whenever the input graph contains a clique of size at least $n - t$, then the algorithm outputs a star in the graph.

Protocol 2GIKR-VSS-Sh is based on a simplification of the *asynchronous* VSS scheme of [10], adapted to the synchronous setting (see Section 6.1). In 2GIKR-VSS-Sh, D hides its degree-t Shamir-sharing polynomial $q(\cdot)$ in a random degree-(t,t) bivariate polynomial $F(x,y)$ and distributes the row and column-polynomials to respective parties. The goal is then to verify if the row and column-polynomials of all (honest) parties are derived from a single degree-(t,t) bivariate polynomial. However, since $n > 4t$ (compared to $n > 3t$ in the earlier protocols), the above verification task is significantly simplified. For simplicity, we first explain an *inefficient* method for the verification tailor-made for $n > 4t$, followed by the actual *efficient* method used in the protocol.

Once the parties receive their respective polynomials, they perform the pair-wise consistency checks and publicly report any inconsistency (by using the round-reducing technique of [31] based on random pads, the results of the pair-wise consistency tests will be publicly available by the end of the second round). The parties next construct a *consistency-graph* G over $\mathcal{P}$, where there exists an edge between a pair of parties if no dispute is reported during the pair-wise consistency check involving the pair of parties. The parties next check for the presence of a *clique* of size $n - t$ in G, which is bound to exist if D is *honest*. If no clique is obtained then clearly D is *corrupt* and hence the parties discard D. If a clique $\mathcal{C}$ of size $n - t$ is obtained, then it implies that there are at least $n - 2t = 2t + 1$ *honest* parties in $\mathcal{C}$, whose polynomials are pair-wise consistent and lie on a single degree-(t,t) bivariate polynomial, say $F^\star(x,y)$, held by D. However, there could be upto t honest parties *outside* $\mathcal{C}$, whose row-polynomials may not lie on $F^\star(x,y)$. To complete the protocol, the goal will be to let each such "outsider" $P_i \notin \mathcal{C}$ obtain the consistent degree-t row-polynomial $F^\star(x,\alpha_i)$. The crucial observation here is that since we are in the setting where $n > 4t$, achieving the above goal *does not* involve any additional interaction. More specifically, each $P_i \notin \mathcal{C}$ considers the set of $n - t \geq 3t + 1$ g_{ji} values, received from the parties $P_j \in \mathcal{C}$ as part of the pair-wise consistency test. Now among these g_{ji} values, at least $2t + 1$ are sent by the *honest* parties $P_j \in \mathcal{C}$, which uniquely define the degree-t row-polynomial $F^\star(x,\alpha_i)$. This is because $F^\star(\alpha_j,\alpha_i) = g_{ji}$ holds for each of these honest parties P_j, since g_{ji} is nothing but $g_j(\alpha_i)$ where $g_j(x)$ is the degree-t column-polynomial held by P_j, which is the same as $F^\star(\alpha_j, y)$. Since $F^\star(x,\alpha_i)$ is a degree-t polynomial and there can be at most t *corrupt* parties $P_j \in \mathcal{C}$ who may provide incorrect values of g_{ji}, party P_i can error-correct these values and obtain $F^\star(x,\alpha_i)$.

The above method is *inefficient*, as finding a maximum-sized clique is an NP-complete problem. Instead, the protocol checks for the presence of a star. If D is *honest* then the set of honest parties always constitute a potential clique of size $n-t$ in G. Hence, star-finding algorithm always outputs a star. If a star $(\mathcal{C},\mathcal{D})$ is obtained in G, then there are at least $|\mathcal{C}| - t = t + 1$ *honest* parties in $\mathcal{C}$ holding degree-t row-polynomials and at least $|\mathcal{D}| - t = 2t + 1$ *honest* parties in $\mathcal{D}$ holding degree-t column-polynomials, which are pair-wise consistent and hence are derived from a single degree-(t,t) bivariate polynomial $F^\star(x,y)$ held by D. In order that every $P_i \notin \mathcal{C}$ obtains its corresponding row-polynomial $F^\star(x,\alpha_i)$, P_i applies the error-correction procedure as discussed above on the g_{ji} values received from $P_j \in \mathcal{D}$ during the pair-wise consistency check. Protocol 2GIKR-VSS-Sh is presented in Fig 11; the reconstruction protocol 2GIKR-VSS-Rec is same as 7BGW-VSS-Rec.

Protocol 2GIKR-VSS-Sh

- **Round I (sending polynomials and exchanging random pads)** — same as Round I of 4GIKR-VSS-Sh.
- **Round II (broadcasting common values in a masked fashion)** — same as Round II of 4GIKR-VSS-Sh.
- **Local computation at the end of round II — each party P_i does the following**:
  - Construct an undirected graph G over $\mathcal{P}$, where the edge (P_l, P_m) is present if $a_{lm} = b_{ml}$ and $a_{ml} = b_{lm}$ holds. Run the star-finding algorithm over G.
  - If the star-finding algorithm outputs $\perp$, then discard D. Else let $(\mathcal{C}, \mathcal{D})$ be the star obtained in G.
    - If $P_i \in \mathcal{C}$, then output the *share* $f_i(0)$.
    - Else recompute the row-polynomial $f_i(x)$ as follows and output the *share* $f_i(0)$.
      - $\forall P_j \in \mathcal{D}$, compute $g_{ji} = b_{ji} - r_{ij}$. Execute RS-Dec(t, t, S_i) to get $f_i(x)$, where $S_i \stackrel{def}{=} \{g_{ji}\}_{P_j \in \mathcal{D}}$.

Figure 11: The 2-round 2GIKR-VSS-Sh protocol with $n > 4t$ due to [31].

4.3 VSS Protocol with a Single Round

In [31], the authors present a perfectly-secure VSS scheme 1GIKR-VSS (Fig 12), with $n = 5$ and $t = 1$. The sharing as well as reconstruction phase requires one round. Let $P_1, \ldots, P_5$ be the set of 5 parties and without loss of generality, let P_1 be the dealer. During the sharing phase, P_1 distributes Shamir-shares of its secret. Since no additional rounds are available, the parties cannot verify whether D has distributed consistent shares. In the reconstruction phase, the dealer is *not allowed* to participate. The remaining parties exchange their respective shares and try to error-correct one potential incorrect share. If the error-correction is successful then the parties output the constant term of the reconstructed degree-1 polynomial, else they output $\perp$. The idea here is that since there can be at most one corrupt party among $\{P_1, \ldots, P_5\}$, there are two possible cases. If P_1 is *honest*, then the privacy is ensured and the shares of P_2, P_3, P_4 and P_5 lie on a degree-1 polynomial. Hence during the reconstruction phase, even if a potentially corrupt party provides incorrect share, it can be error-corrected, thus guaranteeing the correctness property.

The case when P_1 is *corrupt* can be divided into *three* possible sub-cases. If P_1 has distributed valid Shamir-shares (all lying on degree-1 polynomial), then the underlying Shamir-shared value will be reconstructed correctly during the reconstruction phase. The second sub-case is when P_1 has distributed valid Shamir-shares to *exactly three* parties among $\{P_2, P_3, P_4, P_5\}$. For simplicity, let those three parties be P_2, P_3 and P_4 and let their shares lie on a degree-1 polynomial, say $q^\star(\cdot)$. Hence $s^\star \stackrel{def}{=} q^\star(\cdot)$ will be 1-shared among $\{P_2, P_3, P_4\}$ and during the reconstruction phase, all honest parties reconstruct this $s^\star$ by error-correcting the share provided by P_5. The remaining sub-case is when the shares of no three parties among $\{P_2, P_3, P_4, P_5\}$ lie on a degree-1 polynomial. In this case we define $s^\star \stackrel{def}{=} \perp$, where $\perp \notin \mathbb{F}$, indicating that the sharing dealt by P_1 is "invalid". During the reconstruction phase, the error-correction will fail (since the shares of no three parties lie on a degree-1 polynomial) and hence the honest parties output $\perp$.

Scheme 1GIKR-VSS

Sharing Phase: Protocol 1GIKR-VSS-Sh

The dealer P_1 on having input $s \in \mathbb{F}$, picks a random degree-1 polynomial $q(\cdot)$ over $\mathbb{F}$ such that $q(0) = s$. For $i = 2, \ldots, 5$, it sends the *share* $s_i \stackrel{def}{=} q(\alpha_i)$ to P_i.

Reconstruction Phase: Protocol 1GIKR-VSS-Rec

Each party $P_i \in \{P_2, P_3, P_4, P_5\}$ does the following:

- Send s_i to every $P_j \in \mathcal{P} \setminus \{P_1\}$. On receiving s_j from P_j, include s_j in a list W_i. Execute $\mathsf{RS\text{-}Dec}(1, 1, W_i)$.
  - If $\mathsf{RS\text{-}Dec}$ outputs a degree-1 polynomial $q(\cdot)$, then output $q(0)$. Else output $\perp$.

Figure 12: The one round perfectly-secure VSS scheme of [31].

Part III : Asynchronous Communication Setting

5 Preliminaries and Definitions

In the *synchronous* communication setting, each party knows beforehand how long it has to wait for an expected message, and if the message does not arrive within that time-bound, then the sender party is *corrupt*. Unfortunately, it is impossible to ensure such strict time-outs in real-world networks like the Internet, where the communication channels may have arbitrary delays. Motivated by this, [10, 15] introduced the *asynchronous* communication model with *eventual message delivery*.

Apart from a better modelling of real-world networks, asynchronous protocols have the advantage of running at the *actual* speed of the underlying network. More specifically, for a synchronous protocol, the participants have to pessimistically set the global delay Δ to a large value to ensure that the messages sent by every honest party at the beginning of each round reaches their destination within the Δ time frame. But if the *actual* delay δ is such that $\delta << \Delta$, then the protocol fails to take advantage of the faster network and its running time will be proportional to Δ.

In the asynchronous model, there are *no* upper bounds on message delays and the messages can be *arbitrarily, but finitely* delayed. The only guarantee is that any sent message is *eventually* delivered. Moreover, they need not be delivered in the same order in which they were sent. The sequence of message delivery is controlled by a *scheduler* and to model the worst case scenario, we assume that the scheduler is under the control of Adv. Due to the lack of any upper bound on the message delays, no party can wait to receive communication from *all* its neighbours to avoid an endless wait (as a corrupt neighbour may not send any message). As a result, in each step of an asynchronous protocol, a party can afford to wait for messages from *at most* $n - t$ parties (including itself), thus ignoring communication from t potentially *honest* neighbours. Consequently, all synchronous VSS protocols fail completely when executed in an asynchronous environment, as they depend upon the fact that the messages of *all* the honest parties are considered.

Informally, an AVSS scheme consists of *asynchronous* sharing and reconstruction phase, providing privacy, correctness and strong commitment guarantees. However, we need these properties to hold *eventually*. Additionally, we need termination guarantees. Namely, if D is *honest*, then we require that these phases eventually terminate. On the other hand, if D is *corrupt*, then the termination demands are "weaker". Namely, we require the honest parties to terminate the sharing and reconstruction phase *only if* some honest party has terminated the sharing phase. This models the fact that a potentially corrupt D may not invoke the sharing phase in the first place[4].

Definition 5.1 (Perfectly-Secure AVSS [15]). Let $(\mathsf{Sh}, \mathsf{Rec})$ be a pair of asynchronous protocols for the n parties, where a designated *dealer* $\mathsf{D} \in \mathcal{P}$ has a private input s for Sh and where each (honest) party who completes Sh, subsequently invokes Rec, with its local output of Sh. Then $(\mathsf{Sh}, \mathsf{Rec})$ constitute a *perfectly-secure AVSS scheme*, if all the following holds for every possible Adv.

- **Termination**:
  - If D is *honest*, then every honest party will eventually complete Sh.

[4]This is unlike synchronous VSS, where protocols always terminate after a "finite" number of communication rounds.

  - If some honest party has completed Sh, then all honest parties will eventually complete Sh.
  - If some honest party has completed Sh, then it will eventually complete Rec.
- **Privacy and Correctness**: Same as for synchronous VSS.
- **Strong Commitment**: If D is *corrupt* and some honest party terminates Sh, then the joint view of the honest parties at the end of Sh defines a value $s^\star$ (possibly different from s), such that all honest parties output $s^\star$ at the end of Rec.

We can have Type-I and Type-II AVSS schemes. All the existing perfectly-secure AVSS schemes are of Type-II, where the secret is *always* shared as per Shamir's secret-sharing scheme.

5.1 Asynchronous Tools

Asynchronous Reliable-Broadcast (ACast) An ACast protocol allows a designated *sender* $\mathsf{S} \in \mathcal{P}$ to identically send a message m to all the parties. If S is *honest*, then all honest parties eventually terminate with output m. While, if S is *corrupt and* some honest party terminates with output $m^\star$, then eventually every other honest party should terminate with output $m^\star$. Hence the termination guarantees are "weaker" than *synchronous* reliable-broadcast, where the protocol *always* terminates, irrespective of S. Bracha [14] presented a very elegant instantiation of ACast for any $n > 3t$. We use the term P_i *broadcasts* m to mean that P_i acts as S and invokes an instance of ACast protocol to broadcast m. Similarly, the term P_j *receives* m *from the broadcast of* P_i means that P_j (as a receiver) completes the instance of ACast protocol where P_i is S, with m as output.

Online Error-Correction (OEC) Let s be an unknown value, which is d-shared among $\mathcal{P}' \subseteq \mathcal{P}$ such that $d < (|\mathcal{P}'| - 2t)$ holds. The goal is to make some *designated* party, say P_R, reconstruct s (actually OEC allows P_R to reconstruct the entire sharing-polynomial). In the *synchronous* setting, this could be easily achieved by asking *every* party in $\mathcal{P}'$ to send its share to P_R, who can apply the algorithm RS-Dec and error-correct up to t potentially incorrect shares. Given that $d < (|\mathcal{P}'| - 2t)$, the reconstruction will be robust. In the asynchronous setting, achieving the same goal requires a bit of trick. The intuition behind OEC is that P_R keeps waiting till it receives $d + t + 1$ shares from the parties in $\mathcal{P}'$, all of which lie on a *unique* degree-d polynomial, which eventually happens for P_R as there are at least $|\mathcal{P}'| - t \geq d + t + 1$ honest parties in the set $\mathcal{P}'$. This step requires applying the RS-Dec procedure repeatedly. We denote the OEC procedure by the notation $\mathsf{OEC}(\mathcal{P}', d)$.

6 Perfectly-Secure AVSS Schemes

We discuss the various perfectly-secure AVSS schemes [10, 43, 20]. While the sharing phase requires $n > 4t$, with the exception of [43] the reconstruction phase requires $n > 3t$. The necessity of the condition $n > 4t$ follows from [12, 2], where it is shown that in any AVSS scheme designed with $n \leq 4t$, there is a *non-zero* probability in the termination property. We summarize the AVSS schemes in Table 2. While the degree of sharing d is t for [10, 20], the degree d could be more than t for [43].

	Sharing Phase					Reconstruction Phase	
Scheme	n	d	L	$\lvert\mathbb{F}\rvert$	Communication Complexity (CC)	n	CC
BCG-AVSS [10]	$n > 4t$	t	1	$\lvert\mathbb{F}\rvert > n$	$\mathcal{O}(n^2 \log \lvert\mathbb{F}\rvert) + \mathcal{BC}(n^2 \log n)$	$n > 3t$	$\mathcal{O}(n^2 \log \lvert\mathbb{F}\rvert)$
PCR-AVSS [43]	$n > 4t$	$t < d < n - 2t$	1	$\lvert\mathbb{F}\rvert > n$	$\mathcal{O}(n^2 \log \lvert\mathbb{F}\rvert) + \mathcal{BC}(n^2 \log n)$	$n > 4t$	$\mathcal{O}(n^2 \log \lvert\mathbb{F}\rvert)$
CHP-AVSS [20]	$n > 4t$	t	$\geq n - 3t$	$\lvert\mathbb{F}\rvert > 2n - 3t$	$\mathcal{O}(L \cdot n^2 \log \lvert\mathbb{F}\rvert) + \mathcal{BC}(n^2 \log n)$	$n > 3t$	$\mathcal{O}(L \cdot n^2 \log \lvert\mathbb{F}\rvert)$

Table 2: Summary of the perfectly-secure AVSS schemes. Here L denotes the number of values shared through a single AVSS instance and $\mathcal{BC}$ denotes the communication happening through ACast.

6.1 BCG-AVSS Scheme

The BCG-AVSS scheme is presented in Fig 13. The sharing phase protocol BCG-AVSS-Sh is a slightly modified and simplified version of the original protocol [10], based on the simplifications suggested in [8, 43]. Protocol BCG-AVSS-Sh is similar to 2GIKR-VSS-Sh (see Fig 11), executed in the asynchronous setting. The protocol has four stages, each of which is executed asynchronously.

During the first stage, D distributes the row and column-polynomials to the respective parties. During the second stage, each party upon receiving its polynomials performs pair-wise consistency checks by exchanging the common values on its polynomials and publicly announcing the results. Based on these results, each party builds a consistency-graph. Since the results of the consistency checks are broadcast *asynchronously*, the consistency-graph might be different for different parties (however, the edges which are present in the graph of one honest party will be *eventually* included in the graph of every other honest party). During the third stage, D keeps updating its consistency-graph till it finds a star $(\mathcal{C}, \mathcal{D})$ in its consistency-graph, which it then broadcasts as a "proof" that the row-polynomials of the (honest) parties in $\mathcal{C}$ and the column-polynomials of the (honest) parties in $\mathcal{D}$ lie a single degree-(t,t) bivariate polynomial, which is considered as D's "committed" bivariate polynomial. A party upon receiving $(\mathcal{C}, \mathcal{D})$ from D *accepts* it, when $(\mathcal{C}, \mathcal{D})$ constitutes a star in its own consistency-graph. The idea here is that if D has behaved *honestly* then the set of honest parties eventually constitute a clique in D's consistency-graph and hence it eventually finds a star. And if $(\mathcal{C}, \mathcal{D})$ constitutes a star in D's consistency-graph, then it will eventually constitute a star in every other party's consistency-graph as well and hence will be accepted.

Once a star is accepted by P_i then in the last stage, its goal is to compute its share, for which P_i should hold its degree-t row-polynomial, lying on D's committed bivariate polynomial. If $P_i \in \mathcal{C}$, then it already has this polynomial. Else, P_i waits for the common values on the required row-polynomial from the parties in $\mathcal{D}$ and error-corrects the incorrectly received values using OEC. Since P_i's desired row-polynomial has degree-t and since each P_j in $\mathcal{D}$ holds a share of this polynomial in the form of a common value on its column-polynomial, OEC eventually outputs the desired row-polynomial for P_i. This is because $|\mathcal{D}| \geq 3t+1$ and contains at most t corrupt parties[5].

During reconstruction phase, every party sends its share to every other party. The parties then reconstruct the secret by using OEC on the received shares (this step will work even if $n > 3t$).

Scheme BCG-AVSS

Sharing Phase: Protocol BCG-AVSS-Sh

- **Stage I : Distributing Polynomials — the dealer** D **does the following**
  - On having the input $s \in \mathbb{F}$, pick a random degree-t *Shamir-sharing polynomial* $q(\cdot)$, such that $q(0) = s$ holds. Then pick a random degree-(t,t) bivariate polynomial $F(x,y)$, such that $F(0,y) = q(\cdot)$ holds.
  - For $i = 1, \ldots, n$, send the polynomials $f_i(x) \stackrel{def}{=} F(x, \alpha_i)$ and $g_i(y) \stackrel{def}{=} F(\alpha_i, y)$. to P_i.
- **Stage II : Pair-wise consistency checks and building consistency-graph — each party** P_i
  - Upon receiving $f_i(x), g_i(y)$ from D, send $f_{ij} \stackrel{def}{=} f_i(\alpha_j)$ and $g_{ij} \stackrel{def}{=} g_i(\alpha_j)$ to P_j, for $j = 1, \ldots, n$.
  - Upon receiving f_{ji}, g_{ji} from P_j, broadcast $(\mathtt{OK}, i, j)$ if $f_{ji} = g_i(\alpha_j)$ and $g_{ji} = f_i(\alpha_j)$ hold.
  - Construct a graph G_i over $\mathcal{P}$. Add the edge (P_j, P_k) in G_i, if $(\mathtt{OK}, j, k)$ and $(\mathtt{OK}, k, j)$ are received from the broadcast of P_j and P_k respectively. Keep updating G_i, upon receiving new $(\mathtt{OK}, \star, \star)$ messages.
- **Stage III : Finding** star **in the consistency-graph — the dealer** D **does the following**
  - Let G_D be the consistency-graph built by D. After every update in G_D, run the star-finding algorithm to check for the presence of a star in G_D. If a star $(\mathcal{C}, \mathcal{D})$ is found in G_D, then broadcast $(\mathcal{C}, \mathcal{D})$.

[5] For this step, it is necessary that $n > 4t$. Else $|\mathcal{D}| \leq 3t$ and OEC will fail to let P_i obtain its desired row-polynomial.

- **Stage IV : share computation — each party P_i does the following**
  - If $(\mathcal{C}, \mathcal{D})$ is received from the broadcast of D, then *accept* it if $(\mathcal{C}, \mathcal{D})$ is a star in the graph G_i.
  - If $(\mathcal{C}, \mathcal{D})$ is accepted, then compute the *share* s_i as follows and terminate.
    - If $P_i \in \mathcal{C}$, then set $s_i = f_i(0)$, where $f_i(x)$ is the degree-t row-polynomial received from D.
    - Else initialize W_i to $\emptyset$. Upon receiving g_{ji} from $P_j \in \mathcal{D}$, include g_{ji} to W_i. Keep updating W_i and keep executing $\mathsf{OEC}(W_i, t)$ till a degree-t polynomial $f_i(x)$ is obtained. Then set $s_i = f_i(0)$.

Reconstruction Phase: Protocol BCG-AVSS-Rec

Each party $P_i \in \mathcal{P}$ does the following.
- Send the share s_i to every party $P_j \in \mathcal{P}$.
- Initialize a set R_i to $\emptyset$. Upon receiving s_j from P_j, include s_j to R_i. Keep updating R_i and executing $\mathsf{OEC}(R_i, t)$ till a degree-t polynomial $q(\cdot)$ is obtained. Then output $s = q(0)$ and terminate.

Figure 13: The perfectly-secure AVSS scheme of [10].

6.2 PCR-AVSS Scheme

In [43] it is observed that since we are in the setting where $n > 4t$, one could modify the BCG-AVSS scheme in a non-trivial way to generate a d-sharing of D's input in a verifiable fashion, for *any* given d in the range $t \leq d < n - 2t$. The resultant scheme PCR-AVSS is presented in Fig 14. The main motivation to design an AVSS where the degree of sharing is *greater* than t is that such a sharing leads to more efficient MPC protocols (see for instance [26, 8, 9, 43]).

During the first stage of PCR-AVSS-Sh, to d-share s, the dealer D picks a random degree-d *Shamir-sharing polynomial* $q(\cdot)$ with $q(0) = s$ and embeds $q(\cdot)$ in a random degree-(d, t) bivariate polynomial $F(x, y)$ at $y = 0$. Thus the row and column-polynomials have *varying* degrees: the row-polynomials have degree-d, while the column-polynomials have degree-t. This is in *contrast* to the VSS schemes discussed till now where $q(\cdot)$ has degree-t and where $q(\cdot)$ is embedded at $x = 0$ (instead of $y = 0$) in a random degree-(t, t) bivariate polynomial. The dealer then distributes the row and column-polynomials to respective parties. The second stage is similar to protocol BCG-AVSS-Sh where the parties perform pair-wise consistency checks, publicly announce their results and build consistency-graphs.

During the third stage, D proves that it has distributed consistent polynomials to "sufficiently many" parties, derived from a single degree-(d, t) bivariate polynomial, say $F^\star(x, y)$ (where $F^\star(x, y) = F(x, y)$ for an *honest* D). This stage is *different* from BCG-AVSS-Sh where D has to prove that it has distributed consistent polynomials, derived from a single degree-(t, t) bivariate polynomial. Hence the proof mechanism is *different* and constitutes the core of PCR-AVSS-Sh. In a more detail, during the third stage, D publicly proves that it has delivered degree-d row-polynomials lying on $F^\star(x, y)$, to at least $n - t = 3t + 1$ parties $\mathcal{E}$ and degree-t column-polynomials lying on $F^\star(x, y)$, to at least $n - t = 3t + 1$ parties $\mathcal{F}$ (the sets $\mathcal{E}$ and $\mathcal{F}$ need not be the same).

Once the existence of the sets $(\mathcal{E}, \mathcal{F})$ is confirmed (we will discuss in the sequel how such sets are identified), D's sharing is completed in the fourth stage, where the goal will be to ensure that every P_i gets its degree-d column-polynomial $g_i(y)$. Party P_i can then output $g_i(0)$ (which is the same as $F^\star(\alpha_i, 0)$) as its share and the value $s^\star \stackrel{def}{=} F^\star(0, 0)$ will be d-shared through the degree-d polynomial $F^\star(x, 0)$. While the parties $P_i \in \mathcal{F}$ will already have their respective $g_i(y)$ polynomials, the goal is to actually ensure that the "outsider" parties $P_i \notin \mathcal{F}$ get their respective $g_i(y)$ polynomial. For this we observe that *every* $P_j \in \mathcal{E}$ possesses a share on $g_i(y)$, which also constitute a point on P_j's row-polynomial and which P_j would have sent to P_i as part of pair-wise consistency test. Since there

are at least $3t+1$ such parties P_j in $\mathcal{F}$ and since $g_i(y)$ has degree-t, party P_i can reconstruct $g_i(y)$ by error-correcting upto t incorrect values sent by the parties in $\mathcal{E}$ using the OEC.

If D is *honest*, then throughout the protocol Adv learns at most t row-polynomials and t column-polynomials, derived from $F(x,y)$ which is a random degree-(d,t) bivariate polynomial where $F(x,0) = q(\cdot)$. Now similar to Lemma 2.8, one can show that the probability distribution of the row and column-polynomials learnt by Adv will be independent of $q(\cdot)$. That is, for every candidate degree-d polynomial $q(\cdot)$, there exists some degree-(d,t) bivariate polynomial, consistent with the row and column-polynomials learnt by Adv. Intuitively this is because $(d+1)(t+1)$ distinct points are required to uniquely determine D's bivariate polynomial $F(x,y)$, but through the rows and column-polynomials of t corrupt parties, Adv learns at most $t(d+1)+t$ distinct points, leaving $d+1-t$ "degree of freedom" from the view-point of Adv. This ensures the privacy of D's input. We next discuss how the desired $(\mathcal{E},\mathcal{F})$ sets are identified during the third stage of $\mathsf{PCR\text{-}AVSS\text{-}Sh}$.

Dealer first finds a star $(\mathcal{C},\mathcal{D})$ in its consistency-graph. The presence of a star implies that the row and column-polynomials of the honest parties in $\mathcal{C}$ and $\mathcal{D}$ respectively lie on a single degree-(d,t) bivariate polynomial $F^\star(x,y)$. This follows from Lemma 2.7 and the fact that there are at least $n-3t = t+1$ *honest* parties in $\mathcal{C}$ (holding degree-d row-polynomials) and at least $n-2t = d+1$ *honest* parties in $\mathcal{D}$ (holding degree-t column-polynomials), whose row and column-polynomials are pair-wise consistent. To find the required sets $(\mathcal{E},\mathcal{F})$, the dealer tries to find additional "supportive parties" (apart from $\mathcal{C}$ and $\mathcal{D}$) whose row and column-polynomials also lie on $F^\star(x,y)$. The idea is that if D has behaved honestly, then the rows and column-polynomials of *all* honest parties would lie on $F^\star(x,y)$ and there are $n-t$ honest parties. To hunt for these additional supportive parties, D follows the following two-stage non-intuitive approach.

- It first tries to "expand" $\mathcal{D}$ by identifying additional parties whose column-polynomials also lie on $F^\star(x,y)$. The expanded set $\mathcal{F}$, includes all the parties having edges with at least $2t+1$ parties from $\mathcal{C}$. The parties in $\mathcal{D}$ will be already satisfying this criteria and included in $\mathcal{F}$. It is easy to see that the column-polynomial of every $P_j \in \mathcal{F}$ lies on $F^\star(x,y)$. This is because P_j's column-polynomial has degree-t and since P_j has an edge with at least $2t+1$ parties from $\mathcal{C}$, this implies that its column-polynomial is pair-wise consistent with the row-polynomial of at least $t+1$ *honest* parties from $\mathcal{C}$, all of which lie on $F^\star(x,y)$.
- D then tries to "expand" $\mathcal{C}$ by searching for the parties P_j, who have an edge with at least $d+t+1$ parties from $\mathcal{F}$. The idea is that the row-polynomial of such P_j is of degree-d and out of the $d+t+1$ parties from $\mathcal{F}$ (with whom P_j has an edge), at least $d+1$ are honest. Thus, P_j's row-polynomial will be pair-wise consistent with the column-polynomials of at least $d+1$ *honest* parties from $\mathcal{F}$, all of which lie on $F^\star(x,y)$. This implies that the row-polynomial of P_j also lies on $F^\star(x,y)$. Hence D includes P_j in a set $\mathcal{E}$. Notice that all the parties in $\mathcal{C}$ will be already included in $\mathcal{E}$, as they satisfy the above criteria.

Once D finds the sets $(\mathcal{E},\mathcal{F})$ of size $n-t$, it broadcasts them and then the parties verify whether indeed they satisfy the same "conditions" in their respective consistency-graphs, as satisfied in the consistency-graph of D. However there is a subtle issue, as D may have to wait *indefinitely* for the "expansion" of $\mathcal{D}$ and $\mathcal{C}$ sets, beyond their initial cardinalities. For instance, consider the case when $n = 4t+1$, $d = 2t$ and $\mathcal{C}$ and $\mathcal{D}$ are exactly of size $2t+1$ and $3t+1$ respectively, such that they contain t corrupt parties. If the *corrupt* parties P_i in $\mathcal{C}$ choose to be inconsistent with the parties P_j *outside* $\mathcal{D}$ (by not broadcasting the $(\mathtt{OK},i,j)$ messages), then the *honest* parties P_j outside $\mathcal{D}$ will have edges with only $t+1$ parties from $\mathcal{C}$ and will not be included in the set $\mathcal{F}$. So $\mathcal{F}$ will remain the same as $\mathcal{D}$ and will not include any additional party. Similarly, if the *corrupt* parties P_i in $\mathcal{F}$ choose to be inconsistent with the parties P_j outside $\mathcal{C}$, then the *honest* parties P_j outside $\mathcal{C}$ will be never included in $\mathcal{E}$. So $\mathcal{C}$ may never expand from its initial size of $2t+1$.

To deal with the above, in [43] it is observed that if D is *honest* then eventually all *honest* parties

(at least $n-t$) will be consistent with each other and will eventually form a clique in the consistency-graph. Moreover, if the star-finding algorithm is executed on "this" instance of the consistency graph which has all the honest parties forming a clique, then the $\mathcal{C}$ component of the obtained star will have at least $2t+1$ *honest* parties. Now if $\mathcal{C}$ contains at least $2t+1$ honest parties, then eventually $\mathcal{D}$ will expand to $\mathcal{F}$, which will contain all $n-t$ honest parties and eventually $\mathcal{C}$ will expand to $\mathcal{E}$ containing $n-t = 3t+1$ parties. This crucial observation is at the heart of the protocol PCR-AVSS-Sh. However, it is difficult for D to identify an instance of its dynamic consistency-graph that contains a clique involving at least $n-t$ honest parties. The way-out is to repeatedly run the star-finding algorithm and the expansion of every instance of star $(\mathcal{C}, \mathcal{D})$ obtained in the consistency-graph. That is, after every update in its consistency-graph, D checks for the presence of a *new* star in the graph (which was not found earlier) along with the corresponding $\mathcal{F}$ and $\mathcal{E}$ sets and update the *existing* $\mathcal{F}$ and $\mathcal{E}$ sets (corresponding to all the previously generated stars).

Scheme PCR-AVSS

Sharing Phase: Protocol PCR-AVSS-Sh

- **Stage I : Distributing Polynomials** — same steps as BCG-AVSS-Sh except that the *Shamir-sharing polynomial* $q(\cdot)$ is of degree-d, the bivariate polynomial $F(x,y)$ is of degree-(d,t) and $F(x,0) = q(\cdot)$.
- **Stage II : Pair-wise consistency checks and building consistency-graph** — same as BCG-AVSS-Sh.
- **Stage III : Finding $(\mathcal{E}, \mathcal{F})$ in the consistency-graph — the dealer D does the following**
  - After every update in G_D, run the star-finding algorithm to check for the presence of a star. Let there are α number of distinct stars that are found till now in G_D, where $\alpha \geq 0$.
    - If a new star $(\mathcal{C}_{\alpha+1}, \mathcal{D}_{\alpha+1})$ is found in G_D, then do the following:
      1. Add P_j to a set $\mathcal{F}_{\alpha+1}$ if P_j has an edge with at least $2t+1$ parties from $\mathcal{C}_{\alpha+1}$ in G_D.
      2. Add P_j to a set $\mathcal{E}_{\alpha+1}$ if P_j has an edge with at least $d+t+1$ parties from $\mathcal{F}_{\alpha+1}$ in G_D.
      3. For $\beta = 1, \ldots, \alpha$, update the existing $\mathcal{F}_\beta$ and $\mathcal{E}_\beta$ sets as follows:
         - Add P_j to $\mathcal{F}_\beta$, if $P_j \notin \mathcal{F}_\beta$ and P_j has an edge with at least $2t+1$ parties from $\mathcal{C}_\beta$ in G_D.
         - Add P_j to $\mathcal{E}_\beta$, if $P_j \notin \mathcal{E}_\beta$ and P_j has an edge with at least $d+t+1$ parties from $\mathcal{F}_\beta$ in G_D.
    - If no new star is obtained, then update the existing sets $\mathcal{F}_\beta, \mathcal{E}_\beta$ by executing the step 3 as above.
  - Let $(\mathcal{E}_\gamma, \mathcal{F}_\gamma)$ be the first pair among the generated pairs $(\mathcal{E}_\beta, \mathcal{F}_\beta)$ such that $|\mathcal{E}_\gamma| \geq 3t+1$ and $|\mathcal{F}_\gamma| \geq 3t+1$. Then broadcast $((\mathcal{C}_\gamma, \mathcal{D}_\gamma), (\mathcal{E}_\gamma, \mathcal{F}_\gamma))$.
- **Stage IV : share computation — each party P_i does the following**
  - If $((\mathcal{C}_\gamma, \mathcal{D}_\gamma), (\mathcal{E}_\gamma, \mathcal{F}_\gamma))$ is received from the broadcast of D, *accept* it if all the following holds.
    - $|\mathcal{E}_\gamma| \geq 3t+1$ and $|\mathcal{F}_\gamma| \geq 3t+1$.
    - $(\mathcal{C}_\gamma, \mathcal{D}_\gamma)$ is a star in the consistency-graph G_i.
    - Every party $P_j \in \mathcal{F}_\gamma$ has an edge with at least $2t+1$ parties from $\mathcal{C}_\gamma$ in G_i.
    - Every party $P_j \in \mathcal{E}_\gamma$ has an edge with at least $d+t+1$ parties from $\mathcal{F}_\gamma$ in G_i.
  - If $((\mathcal{C}_\gamma, \mathcal{D}_\gamma), (\mathcal{E}_\gamma, \mathcal{F}_\gamma))$ is accepted, then compute the *share* s_i as follows and terminate.
    - If $P_i \in \mathcal{F}_\gamma$, then set $s_i = g_i(0)$, where $g_i(y)$ is the degree-t column-polynomial received from D.
    - Else initialize W_i to $\emptyset$. Upon receiving f_{ji} from $P_j \in \mathcal{E}$, include f_{ji} to W_i. Keep updating W_i and keep executing $\mathsf{OEC}(W_i, t)$ till a degree-t polynomial $g_i(y)$ is obtained. Then set $s_i = g_i(0)$.

Reconstruction Phase: Protocol PCR-AVSS-Rec

Each party $P_i \in \mathcal{P}$ executes the following steps.

- Send the share s_i to every party $P_j \in \mathcal{P}$.

– Initialize R_i to $\emptyset$. Upon receiving s_j from P_j, include s_j to R_i. Keep updating R_i and keep executing $\mathsf{OEC}(R_i, d)$ till a degree-d polynomial $q(\cdot)$ is obtained. Then output $s = q(0)$ and terminate.

Figure 14: The perfectly-secure AVSS scheme of [43].

6.3 CHP-AVSS Scheme

Protocol BCG-AVSS-Sh requires a communication of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits over the pair-wise channels, apart from the broadcast of $\Theta(n^2)$ OK messages and the broadcast of star. The protocol generates t-sharing of a *singe* secret. If D wants to t-share L secrets, then it can invoke L instances of BCG-AVSS-Sh. This makes the *broadcast-complexity* (namely the number of bits to be broadcast) proportional to L. Instead [20] proposed a modification of [6] PCR-AVSS-Sh called CHP-AVSS-Sh, which allows D to t-share L secrets for *any* given $L \geq n - 3t$, *without* incurring any additional communication complexity. The broadcast-complexity of CHP-AVSS-Sh will be *independent* of L, which is a significant saving. This is because each instance of the broadcast in the asynchronous setting needs to be emulated by running the costly Bracha's ACast protocol.

We explain the idea of CHP-AVSS-Sh assuming $L = n - 3t$. If $L > n - 3t$, then D can divide its inputs into multiple batches of $n - 3t$ and invoke an instance of CHP-AVSS-Sh for each batch. Recall that in protocol PCR-AVSS-Sh, if D is *honest*, then the adversary's view (which consists of t number of degree-d row and degree-t column-polynomials) leaves $d + 1 - t$ "degree of freedom" in the degree-(d, t) bivariate polynomial $F(x, y)$, where $t < d < n - 2t$. If we consider the *maximum* value d_{max} of d which is $n - 2t - 1$, this implies $n - 3t$ degree of freedom in $F(x, y)$. While PCR-AVSS-Sh uses this degree of freedom for generating a d_{max}-sharing of a *single* value by embedding a *single* degree-d_{max} sharing-polynomial in $F(x, y)$, CHP-AVSS-Sh uses it to generate t-sharing of $n - 3t$ different values by embedding $n - 3t$ independent degree-t Shamir-sharing polynomials in $F(x, y)$.

In a more detail, given $s^{(1)}, \ldots, s^{(n-3t)}$ for t-sharing, D picks $n - 3t$ random degree-t Shamir-sharing polynomials $q^{(1)}(\cdot), \ldots, q^{(n-3t)}(\cdot)$, where $q^{(k)}(0) = s^{(k)}$. These polynomials are embedded in a degree-(d_{max}, t) bivariate polynomial, which is otherwise a random polynomial, except that $F(\beta_k, y) = q^{(k)}(\cdot)$ holds. Here $\beta_1, \ldots, \beta_{n-3t}$ are distinct, publicly-known non-zero elements from $\mathbb{F}$, different from the evaluation-points[7] $\alpha_1, \ldots, \alpha_n$. Notice that the embedding and the degree of the sharing-polynomials are different in PCR-AVSS-Sh and CHP-AVSS-Sh. Accordingly, the shares of the parties are different (see Fig 15). The *shares* of P_i in CHP-AVSS-Sh will be $\{F(\beta_k, \alpha_i)\}_{k \in \{1,\ldots,n-3t\}}$. And to compute them, the goal will be to ensure that P_i gets its row-polynomial $f_i(x) = F(x, \alpha_i)$, as P_i can then compute its shares by evaluating $f_i(x)$ at $x = \beta_1, \ldots, \beta_{n-3t}$.

To achieve the above goal, we observe that if D invokes the protocol PCR-AVSS-Sh (with the above modifications) and if the protocol terminates for the honest parties, then it ensures that D has "committed" a degree-(d_{max}, t) bivariate polynomial, say $F^\star(x, y)$ (where $F^\star(x, y)$ is the same as $F(x, y)$ for an *honest* D), such that each (honest) party P_j possesses its degree-t column-polynomial $g_j(y) = F^\star(\alpha_j, y)$. We also observe that for each row-polynomial $f_i(x) = F^\star(x, \alpha_i)$, every party P_j holds a share $F^\star(\alpha_j, \alpha_i)$ in the form of $g_j(\alpha_i)$. Moreover, the degree of $f_i(x)$ is $d_{max} = n - 2t - 1$ and there are total n parties holding shares of $f_i(x)$. Hence, if every party P_j sends its share $g_j(\alpha_i)$ of $f_i(x)$ to P_i, then P_i can reconstruct its desired row-polynomial $f_i(x)$ by applying OEC on the received values. We note that in CHP-AVSS-Sh for a *corrupt* D, the values $\vec{S}' \stackrel{def}{=} (F^\star(\beta_1, 0), \ldots, F^\star(\beta_{n-3t}, 0))$ will be t-shared, where $\vec{S}' = (s^{(1)}, \ldots, s^{(n-3t)})$ for an *honest* D. As most of the protocol steps of CHP-AVSS-Sh are the same as PCR-AVSS-Sh, we do not give the formal details to avoid repetition.

Part III : Hybrid Communication Setting

[6] It is easy to see that the communication complexity of PCR-AVSS-Sh is the same as BCG-AVSS-Sh.

[7] This imposes the restriction of $|\mathbb{F}| > 2n - 3t$ to hold in the protocol CHP-AVSS-Sh.

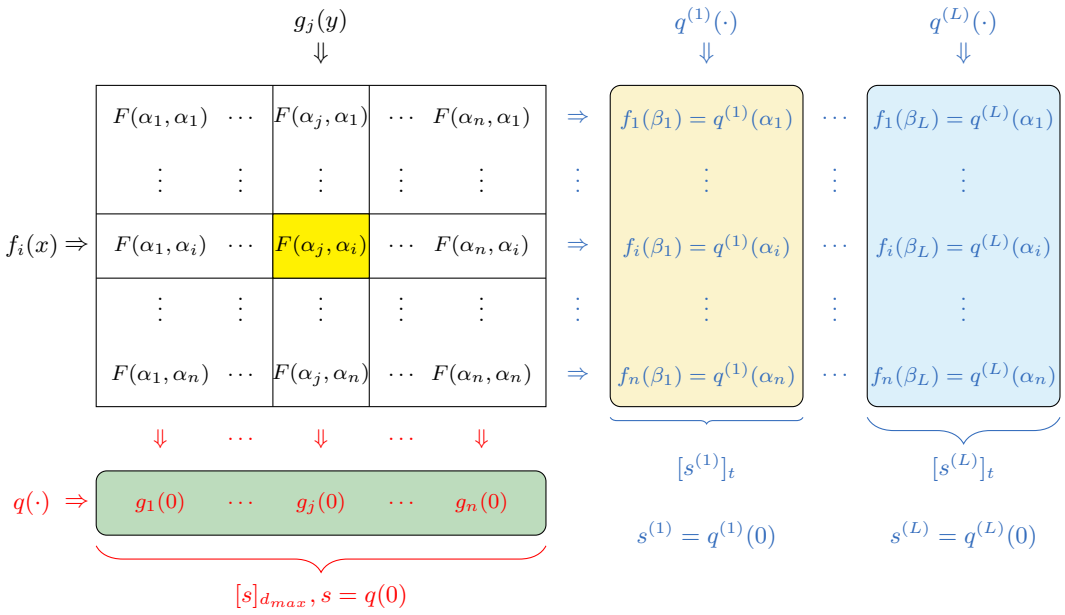

Figure 15: Pictorial representation of the values distributed by D in PCR-AVSS-Sh and CHP-AVSS-Sh. Polynomial $F(x,y)$ has degree-(d_{max},t), where $d_{max}=n-2t-1$. The row-polynomials have degree-d_{max}, while the column-polynomials have degree-t. In PCR-AVSS-Sh, the value s is d_{max}-shared through the degree-d_{max} row-polynomial $F(x,0)$ (shown in red color), while in protocol CHP-AVSS-Sh, the values $s^{(1)},\ldots,s^{(L)}$ are t-shared through degree-t column polynomials $F(\beta_1,y),\ldots,F(\beta_L,y)$ (shown in blue color), where $L=n-3t$.

7 Preliminaries and Definitions for Hybrid Communication Setting

Even though the asynchronous model is practically more relevant compared to the synchronous setting, there are some inherent downsides to the requirements of AVSS. For instance, the optimal resilience bounds for synchronous and asynchronous VSS are $t<n/4$ and $t<n/3$ respectively. Additionally, the AVSS protocols are more complex than their synchronous counterparts. A potential solution to bridge this "gap" is to devise a *hybrid* communication setting, which is a mix of synchronous and asynchronous setting. Namely, the first r rounds are assumed to be synchronous, where $r\geq 1$. After these r rounds, the entire communication happens asynchronously. In an interesting work [44], it is shown that one can design a Type-II perfectly-secure VSS satisfying Definition 5.1 with $t<n/3$ in the hybrid setting, where $r=1$. Notice that the protocol has optimal resilience as well as requires the optimal number of synchronous rounds.

We next define *weak polynomial sharing* (WPS), which is used as a building block in the hybrid VSS scheme of [44]. The primitive allows a dealer D to distribute shares of a private degree-t polynomial held by D. If D is *honest*, then every honest party should eventually terminate with its share. Moreover, even if D is *corrupt* and some honest party terminates Sh, then it is ensured that D has distributed shares of some degree-t polynomial to at least $t+1$ honest parties, with the remaining honest parties outputting $\perp$. Property-wise, WPS is a weaker primitive than WSS, as it has only a sharing phase and may not allow even a "weak reconstruction" of D's shared polynomial.

Definition 7.1 (Weak Polynomial Sharing (WPS) [44]). Let Sh be protocol for the n parties in the hybrid setting, where a designated *dealer* $\mathsf{D}\in\mathcal{P}$ has a private degree-t polynomial $f(x)$ over $\mathbb{F}$ for Sh. Then Sh constitutes a *perfectly-secure WPS*, if all the following holds.

- **Termination**: Same as AVSS.

- **Privacy**: Same as AVSS.
- **Correctness**: If some honest party terminates Sh, then there exists a *weakly committed polynomial* $f^\star(x)$ over $\mathbb{F}$ such that.
  - If D is *honest*, then $f^\star(x) = f(x)$ and each honest P_i outputs $f(\alpha_i)$ at the end of Sh.
  - If D is *corrupt*, then every honest P_i outputs either $f^\star(\alpha_i)$ or some default value $\perp$, with at least $t+1$ honest parties outputting $f^\star(\alpha_i)$.

8 Hybrid VSS Protocol with $t < n/3$

In the AVSS schemes, the parties not receiving their shares from D, deploy OEC to recompute their shares from the share-shares received from the parties, who received their shares from D. This inherently requires $n > 4t$, as OEC is used to error-correct t errors and reconstruct degree-t polynomials, for which at least $3t+1$ parties should have valid shares. On contrary, the hybrid VSS of [44] is designed only with $n > 3t$. The presence of a synchronous round at the beginning simplifies certain aspects of verifiability, that were difficult in the asynchronous model and completely avoids the need for OEC. We first start with the WPS construction of [44].

The WPS construction (fig 16) is similar to the 3KKK-WSS-Sh protocol. The dealer embeds its degree-t polynomial in a random *symmetric* degree-(t,t) bivariate polynomial and distributes its row-polynomials. In parallel, the parties pair-wise exchange random pads. Since the pads are exchanged during the synchronous round, it is ensured that each party P_i receives the pad selected for it for every other party by the end of the synchronous round. The sent and received pads are also "registered" with D for comparison purpose (while the sent pads are communicated to D during the synchronous round, the received pads are communicated asynchronously). Based on this, D sends to P_i its list of conflicting-parties $\mathcal{C}_i$, who did not concur on the pads. Based on $\mathcal{C}_i$, party P_i broadcasts its common values in a masked fashion for the parties who are not in $\mathcal{C}_i$ and in an unmasked fashion for the parties who are in $\mathcal{C}_i$. The dealer then checks whether P_i's public value are consistent with the bivariate polynomial and the pads registered with D and accordingly includes P_i to a set $\mathcal{W}$. Once $\mathcal{W}$ achieves the size of $2t+1$ (which eventually happens if D behaves honestly), D broadcasts $\mathcal{W}$. It is then publicly verified that no pair of parties in $\mathcal{W}$ publicly conflicts over their supposedly common values that are either in padded or in clear form.

If $\mathcal{W}$ is successfully verified, then the row-polynomials of the (honest) parties in $\mathcal{W}$ lie on a single degree-(t,t) symmetric bivariate polynomial $F^\star(x,y)$ held by D. While every $P_i \in \mathcal{W}$ can directly output the constant term of its row-polynomial as its share, any party $P_i \notin \mathcal{W}$ tries to compute its row-polynomial by interpolating the common values with the parties in $\mathcal{W}$. If P_i has a conflict with any party in $\mathcal{W}$, then the common values is publicly available. Else, it computes the common value by subtracting the pad it sent to that party in the synchronous round from padded value made public by that party. If the interpolation does not give a degree-t polynomial (which can happen only for a *corrupt* D), then P_i sets $\perp$ as its share.

Protocol WPS

Synchronous Phase

- **Sending polynomials and exchanging random pads**:
  - D with input $f(\cdot)$ chooses a random symmetric degree-(t,t) bivariate $F(x,y)$ such that $F(0,y) = f(\cdot)$ and sends $f_i(x) = F(x,\alpha_i)$ to each party $P_i \in \mathcal{P}$
  - Each party $P_i \in \mathcal{P}$ picks a random pad m_{ij} for every $P_j \in \mathcal{P}$ and sends m_{ij}.
  - Each P_i sends $\{m_{ij}\}_{P_j \in \mathcal{P}}$ to D. Let $\{m^s_{ij}\}_{P_j \in \mathcal{P}}$ be the list of *sent-pads* received by D from P_i.

Asynchronous Phase

- **Verifying masks**:
  - Each P_i sends the pads $\{m_{ji}\}_{P_j \in \mathcal{P}}$ received from various parties to D. Let $\{m^r_{ji}\}_{P_j \in \mathcal{P}}$ be the list of *received-pads* which D receives from P_i.
  - Upon receiving $\{m^r_{ji}\}_{P_j \in \mathcal{P}}$ from P_i, dealer D sends to P_i a set $\mathcal{C}_i = \{P_j : m^r_{ji} \neq m^s_{ij}\}$.
- **Broadcasting masked/unmasked common values — each party** P_i: Broadcasts $(\mathcal{A}_i, \mathcal{B}_i, \mathcal{C}_i)$, where:
  - $\mathcal{A}_i = \{a_{ij} = f_i(\alpha_j) + m_{ij}\}_{P_j \in \mathcal{P}}$.
  - $\mathcal{B}_i = \{b_{ij}\}_{P_j \in \mathcal{P}}$, where $b_{ij} = f_i(\alpha_j)$ if $P_j \in \mathcal{C}_i$ and $b_{ij} = f_i(\alpha_j) + m_{ji}$ otherwise.
- **Constructing and broadcasting** $\mathcal{W}$ **— Dealer** D **does the following**:
  - Upon receiving $(\mathcal{A}_i, \mathcal{B}_i, \mathcal{C}_i)$ from P_i, mark P_i as *correct* and include in $\mathcal{W}$, if all the following hold.
    1. $a_{ij} - m^s_{ij} = F(\alpha_j, \alpha_i)$
    2. $b_{ij} = F(\alpha_j, \alpha_i)$ for all $P_j \in \mathcal{C}_i$ and $b_{ij} - m^r_{ji} = F(\alpha_j, \alpha_i)$ otherwise
    3. $\mathcal{C}_i$ is the same set sent by D to P_i.
  - Wait until $|\mathcal{W}| \geq 2t + 1$ and then broadcast $\mathcal{W}$.
- **Verifying** $\mathcal{W}$ **— each party** P_i: *Accept* $\mathcal{W}$ received from the broadcast of D if all the following holds.
  - $|\mathcal{W}| \geq 2t + 1$ and $(\mathcal{A}_j, \mathcal{B}_j, \mathcal{C}_j)$ is received from the broadcast of each $P_j \in \mathcal{W}$.
  - Every $P_j, P_k \in \mathcal{W}$ are *pair-wise* consistent, as per the following conditions.
    1. if $P_j \in \mathcal{C}_k$ and $P_k \in \mathcal{C}_j$ then $b_{jk} = b_{kj}$
    2. if $P_j \in \mathcal{C}_k$ and $P_k \notin \mathcal{C}_j$ then $a_{kj} = b_{jk}$
    3. if $P_j \notin \mathcal{C}_k$ and $P_k \in \mathcal{C}_j$ then $a_{jk} = b_{kj}$
    4. Else $a_{jk} = b_{kj}$ and $a_{kj} = b_{jk}$
- **Output stage — each party** P_i: If $\mathcal{W}$ is accepted, then terminate with output s_i, computed as follows.
  - If $P_i \in \mathcal{W}$ then set $s_i = f_i(0)$.
  - Else interpolate the points $\{(\alpha_j, s_{ij})\}_{P_j \in \mathcal{W}}$ where $s_{ij} = b_{ji}$ if $P_i \in \mathcal{C}_j$ and $s_{ij} = b_{ji} - m_{ij}$ otherwise. If the interpolation outputs a degree-t polynomial $f_i(x)$ then set $s_i = f_i(0)$, else set $s_i = \perp$.

Figure 16: Weak Polynomial Sharing protocol due to [44]

From WPS to VSS WPS fails to serve as a VSS because if D is *corrupt*, then the parties outside $\mathcal{W}$ may get $\perp$. Protocol $\mathsf{PR\text{-}Sh}$ (see Fig 17) fixes this shortcoming. The protocol has two "layers" of communication. The goal of the first layer is to identify a set $\mathcal{V}$ of $2t + 1$ parties whose row-polynomials are pair-wise consistent and lie on a single degree-(t, t) symmetric bivariate polynomial $F^\star(x, y)$. The structure of this layer is same as WPS. The second layer is executed in parallel, enabling the parties not in $\mathcal{V}$ to obtain their respective row-polynomials lying on $F^\star(x, y)$. In a more detail, every P_j picks a random *blinding-polynomial* $p_j(\cdot)$ and shares it by invoking an instance WPS_j of WPS. Additionally, it makes public the polynomial $p_j(\cdot) + f_j(x)$ during the asynchronous phase. The idea is that if later P_j is declared to be a part of $\mathcal{V}$, then any $P_i \notin \mathcal{V}$ can compute the point $f_j(\alpha_i)$ (which is the same as $f_i(\alpha_j)$) on P_i's row-polynomial, if P_i obtains the output $p_j(\alpha_i)$ during WPS_j. While an *honest* $P_j \in \mathcal{V}$ makes public the correct $p_j(\cdot) + f_j(x)$, care has to be taken to ensure that even a *corrupt* $P_j \in \mathcal{V}$ has made public the correct $p_j(\cdot) + f_j(x)$. This is done as follows. First, each party P_k participates *conditionally* during WPS_j depending upon whether the blinded polynomial of P_k is consistent with respect to its received point on $p_k(\cdot)$ during WPS_k and the supposedly common share $f_k(j)$. Second, P_j is included in $\mathcal{V}$ only when during WPS_j the generated $\mathcal{W}$ set $\mathcal{W}_j$ is *accepted* and which has an overlap of $2t + 1$ with $\mathcal{V}$.

For a *corrupt* $P_j \in \mathcal{V}$, an *honest* $P_i \notin \mathcal{V}$ may end up obtaining $\perp$ during WPS_j. However there will be at least $t + 1$ *honest* $P_j \in \mathcal{V}$, corresponding to whom P_i eventually obtains $p_j(\alpha_i)$ during WPS_j, using which P_i obtains $t + 1$ points on $f_i(x)$, which are sufficient to compute $f_i(x)$.

Protocol PR-Sh

Synchronous Phase

D and parties execute the same steps as in the synchronous phase of WPS. Additionally, each P_i picks a random degree-t *blinding-polynomial* $p_i(\cdot)$ and as a dealer invokes an instance WPS_i of WPS to share $p_i(\cdot)$. Moreover, P_i also participates in the synchronous phase of the instance WPS_j for every $P_j \in \mathcal{P}$.

Asynchronous Phase

- **Verifying masks**: Parties and D execute the same steps as in WPS.
- **Broadcasting values — each party** P_i: Broadcasts $(\mathcal{A}_i, \mathcal{B}_i, \mathcal{C}_i, d_i(x))$, where $\mathcal{A}_i, \mathcal{B}_i, \mathcal{C}_i$ are same as in WPS and $d_i(x) \stackrel{def}{=} p_i(x) + f_i(x)$.
- **Participating in WPS instances — each party** P_i: For $j = 1, \ldots, n$, participates in WPS_j if a degree-t polynomial $d_j(x)$ is received from the broadcast of P_j and if $d_j(\alpha_i) = p_j(\alpha_i) + f_i(\alpha_j)$ holds.
- **Computing and Broadcasting** $\mathcal{V}$ **— the** D: computes and broadcasts $(\mathcal{V}, \{\mathcal{W}_i\}_{P_i \in \mathcal{V}})$, such that:
  - Every $P_i \in \mathcal{V}$ is marked as *correct* (by satisfying the same conditions as in WPS).
  - For every $P_i \in \mathcal{V}$, the set $\mathcal{W}_i$ is *accepted* during the instance WPS_i.
  - $|\mathcal{V}| \geq 2t + 1$ and $|\mathcal{V} \cap \mathcal{W}_i| \geq 2t + 1$ for every $P_i \in \mathcal{V}$.
- **Verifying** $\mathcal{V}$ **— each party** P_i: *Accept* $(\mathcal{V}, \{\mathcal{W}_i\}_{P_i \in \mathcal{V}})$ received from the broadcast of D, if:
  1. Every $P_j, P_k \in \mathcal{V}$ are *pair-wise consistent* (using the same criteria as in WPS).
  2. For all $P_j \in \mathcal{V}$, the set $\mathcal{W}_j$ is *accepted* during the instance WPS_j.
  3. $|\mathcal{V}| \geq 2t + 1$ and for every $P_j \in \mathcal{V}$, the condition $|\mathcal{V} \cap \mathcal{W}_j| \geq 2t + 1$ holds.
- **Output stage — each party** P_i: If $(\mathcal{V}, \{\mathcal{W}_i\}_{P_i \in \mathcal{V}})$ is accepted then terminate with output s_i, where:
  - If $P_i \in \mathcal{V}$ then set $s_i = f_i(0)$.
  - Else compute the output p_{ji} in WPS_j for every $P_j \in \mathcal{V}$, interpolate degree-t polynomial $f_i(x)$ through the points $\{(\alpha_i, s_{ij} = d_j(\alpha_i) - p_{ji})\}_{P_j \in \mathcal{V} \wedge p_{ji} \neq \perp}$ and set $s_i = f_i(0)$.

Figure 17: The hybrid VSS protocol due to [44]

9 Conclusion

In this article, we surveyed the existing perfectly-secure VSS schemes. We considered three different communication settings, the synchronous, asynchronous and the hybrid communication setting. We identify the following open problems in the domain of perfectly-secure VSS.

- All VSS schemes in the *synchronous* setting with $n > 3t$ and *three* round sharing phase requires a communication of $\mathcal{O}(n^3 \log |\mathbb{F}|)$ bits, both over the pair-wise channels as well as over the broadcast channel. This is in contrast to the VSS schemes which allows four or more rounds in the sharing phase and which requires a communication of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits. The main reason for this is that all the 3-round VSS schemes deploy n invocations of the WSS, each with a communication of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits. It is interesting to see if one can design a 3-round perfectly-secure VSS scheme with a communication complexity of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits.
- To the best of our knowledge, we are unaware of any non-trivial lower bound on the communication complexity of *any* perfectly-secure VSS scheme. One could explore to derive a non-trivial lower bound on the communication complexity of perfectly-secure VSS schemes.
- The *broadcast complexity* (namely the total number of bits broadcasted by the honest parties) of all the VSS schemes with $n > 3t$ is proportional to the number of values shared by the VSS instance. This is unlike the *asynchronous* VSS scheme of [20], whose broadcast complexity is *independent* of the number of values shared by the VSS instance. It is interesting to see if

one can design a perfectly-secure VSS scheme with $n > 3t$ where the broadcast complexity is independent of the number of shared values.

- The round complexity of the 3AKP-VSS scheme [3] is *not* optimal in terms of the usage of broadcast channel. One could explore to achieve the same properties as the 3AKP-VSS scheme, with the broadcast channel being used only during the second round.

References

[1] I. Abraham, D. Dolev, and J. Y. Halpern. An Almost-Surely Terminating Polynomial Protocol for Asynchronous Byzantine Agreement with Optimal Resilience. In *PODC*, pages 405–414. ACM, 2008.

[2] I. Abraham, D. Dolev, and G. Stern. Revisiting Asynchronous Fault Tolerant Computation with Optimal Resilience. In *PODC*, pages 139–148. ACM, 2020.

[3] B. Applebaum, E. Kachlon, and A. Patra. The Round Complexity of Perfect MPC with Active Security and Optimal Resiliency. In *FOCS*, pages 1277–1284. IEEE, 2020.

[4] G. Asharov and Y. Lindell. A Full Proof of the BGW Protocol for Perfectly Secure Multiparty Computation. *J. Cryptology*, 30(1):58–151, 2017.

[5] M. Backes, A. Kate, and A. Patra. Computational Verifiable Secret Sharing Revisited. In *ASIACRYPT*, volume 7073 of *Lecture Notes in Computer Science*, pages 590–609. Springer, 2011.

[6] L. Bangalore, A. Choudhury, and G. Garimella. Round Efficient Computationally Secure Multi-Party Computation Revisited. In *ICDCN*, pages 292–301. ACM, 2019.

[7] L. Bangalore, A. Choudhury, and A. Patra. The Power of Shunning: Efficient Asynchronous Byzantine Agreement Revisited. *J. ACM*, 67(3):14:1–14:59, 2020.

[8] Z. Beerliová-Trubíniová and M. Hirt. Simple and Efficient Perfectly-Secure Asynchronous MPC. In *ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 376–392. Springer, 2007.

[9] Z. Beerliová-Trubíniová and M. Hirt. Perfectly-Secure MPC with Linear Communication Complexity. In *TCC*, volume 4948 of *Lecture Notes in Computer Science*, pages 213–230. Springer, 2008.

[10] M. Ben-Or, R. Canetti, and O. Goldreich. Asynchronous Secure Computation. In *STOC*, pages 52–61. ACM, 1993.

[11] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract). In *STOC*, pages 1–10. ACM, 1988.

[12] M. Ben-Or, B. Kelmer, and T. Rabin. Asynchronous Secure Computations with Optimal Resilience (Extended Abstract). In *PODC*, pages 183–192. ACM, 1994.

[13] G. R. Blakley. Safeguarding Cryptographic Keys. In *AFIPS National Computer Conference*, pages 313–317. IEEE, 1979.

[14] G. Bracha. An Asynchronous [(n-1)/3]-Resilient Consensus Protocol. In *PODC*, pages 154–162. ACM, 1984.

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

[16] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *FOCS*, pages 136–145. IEEE Computer Society, 2001.

[17] R. Canetti. Universally Composable Security. *J. ACM*, 67(5):28:1–28:94, 2020.

[18] D. Chaum, C. Crépeau, and I. Damgård. Multiparty Unconditionally Secure Protocols (Extended Abstract). In *STOC*, pages 11–19. ACM, 1988.

[19] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults (Extended Abstract). In *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*, pages 383–395. IEEE Computer Society, 1985.

[20] A. Choudhury, M. Hirt, and A. Patra. Asynchronous Multiparty Computation with Linear Communication Complexity. In *DISC*, volume 8205 of *Lecture Notes in Computer Science*, pages 388–402. Springer, 2013.

[21] A. Choudhury, K. Kurosawa, and A. Patra. The Round Complexity of Perfectly Secure General VSS. In *ICITS*, volume 6673 of *Lecture Notes in Computer Science*, pages 143–162. Springer, 2011.

[22] A. Choudhury and N. Pappu. Perfectly-Secure Asynchronous MPC for General Adversaries (Extended Abstract). In *INDOCRYPT*, volume 12578 of *Lecture Notes in Computer Science*, pages 786–809. Springer, 2020.

[23] A. Choudhury and A. Patra. An Efficient Framework for Unconditionally Secure Multiparty Computation. *IEEE Trans. Inf. Theory*, 63(1):428–468, 2017.

[24] 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.

[25] R. Cramer, I. Damgård, and U. M. Maurer. General Secure Multi-party Computation from any Linear Secret-Sharing Scheme. In B. Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, volume 1807 of *Lecture Notes in Computer Science*, pages 316–334. Springer Verlag, 2000.

[26] I. Damgård and J. B. Nielsen. Scalable and Unconditionally Secure Multiparty Computation. In *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 572–590. Springer, 2007.

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

[28] P. Feldman and S. Micali. An Optimal Probabilistic Protocol for Synchronous Byzantine Agreement. *SIAM J. Comput.*, 26(4):873–933, 1997.

[29] M. J. Fischer and N. A. Lynch. A Lower Bound for the Time to Assure Interactive Consistency. *Inf. Process. Lett.*, 14(4):183–186, 1982.

[30] M. Fitzi, J. A. Garay, S. Gollakota, C. Pandu Rangan, and K. Srinathan. Round-Optimal and Efficient Verifiable Secret Sharing. In *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 329–342. Springer, 2006.

[31] R. Gennaro, Y. Ishai, E. Kushilevitz, and T. Rabin. The Round Complexity of Verifiable Secret Sharing and Secure Multicast. In *STOC*, pages 580–589. ACM, 2001.

[32] O. Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.

[33] M. Hirt. *Multi Party Computation: Efficient Protocols, General Adversaries, and Voting*. PhD thesis, ETH Zurich, Zürich, Switzerland, 2001.

[34] M. Ito, A. Saito, and T. Nishizeki. Secret Sharing Schemes Realizing General Access Structures). In *Global Telecommunication Conference, Globecom*, pages 99–102. IEEE Computer Society, 1987.

[35] J. Katz and C. Y. Koo. On Expected Constant-Round Protocols for Byzantine Agreement. In *CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 445–462. Springer, 2006.

[36] 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.

[37] 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.

[38] E. Kushilevitz, Y. Lindell, and T. Rabin. Information-Theoretically Secure Protocols and Security under Composition. *SIAM J. Comput.*, 39(5):2090–2112, 2010.

[39] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error Correcting Codes*. North-Holland Publishing Company, 1978.

[40] U. M. Maurer. Secure Multi-Party Computation Made Simple. *Discret. Appl. Math.*, 154(2):370–381, 2006.

[41] R. J. McEliece and D. V. Sarwate. On Sharing Secrets and Reed-Solomon Codes. *Commun. ACM*, 24(9):583–584, 1981.

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

[43] A. Patra, A. Choudhury, and C. Pandu Rangan. Efficient Asynchronous Verifiable Secret Sharing and Multiparty Computation. *J. Cryptology*, 28(1):49–109, 2015.

[44] A. Patra and D. Ravi. On the power of hybrid networks in multi-party computation. *IEEE Transactions on Information Theory*, 64(6):4207–4227, 2018.

[45] M. C. Pease, R. E. Shostak, and L. Lamport. Reaching Agreement in the Presence of Faults. *J. ACM*, 27(2):228–234, 1980.

[46] T. P. Pedersen. A Threshold Cryptosystem without a Trusted Party (Extended Abstract). In *EUROCRYPT*, volume 547 of *Lecture Notes in Computer Science*, pages 522–526. Springer, 1991.

[47] T. Rabin and M. Ben-Or. Verifiable Secret Sharing and Multiparty Protocols with Honest Majority (Extended Abstract). In *STOC*, pages 73–85. ACM, 1989.

[48] A. Shamir. How to Share a Secret. *Commun. ACM*, 22(11):612–613, 1979.