Efficient Perfectly Secure Computation with Optimal Resilience

Ittai Abraham[1], Gilad Asharov[2(✉)], and Avishay Yanai[1]

[1] VMWare Research, Herzliya, Israel
{iabraham,yanaia}@vmware.com
[2] Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel
Gilad.Asharov@biu.ac.il

Abstract. Secure computation enables n mutually distrustful parties to compute a function over their private inputs jointly. In 1988 Ben-Or, Goldwasser, and Wigderson (BGW) demonstrated that any function can be computed with perfect security in the presence of a malicious adversary corrupting at most $t < n/3$ parties. After more than 30 years, protocols with perfect malicious security, with round complexity proportional to the circuit's depth, still require sharing a total of $O(n^2)$ values per multiplication. In contrast, only $O(n)$ values need to be shared per multiplication to achieve semi-honest security. Indeed sharing $\Omega(n)$ values for a single multiplication seems to be the natural barrier for polynomial secret sharing-based multiplication.

In this paper, we close this gap by constructing a new secure computation protocol with perfect, optimal resilience and malicious security that incurs sharing of only $O(n)$ values per multiplication, thus, matching the semi-honest setting for protocols with round complexity that is proportional to the circuit depth. Our protocol requires a constant number of rounds per multiplication. Like BGW, it has an overall round complexity that is proportional only to the multiplicative depth of the circuit. Our improvement is obtained by a novel construction for *weak VSS for polynomials of degree-2t*, which incurs the same communication and round complexities as the state-of-the-art constructions for *VSS for polynomials of degree-t*.

Our second contribution is a method for reducing the communication complexity for any depth-1 sub-circuit to be proportional only to the size of the input and output (rather than the size of the circuit). This implies protocols with *sublinear communication complexity* (in the size of the circuit) for perfectly secure computation for important functions like matrix multiplication.

Gilad Asharov is sponsored by the Israel Science Foundation (grant No. 2439/20), by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office, and by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 891234.

K. Nissim and B. Waters (Eds.): TCC 2021, LNCS 13043, pp. 66–96, 2021.
https://doi.org/10.1007/978-3-030-90453-1_3

1 Introduction

Secure multiparty computation is a major pillar of modern cryptography. Breakthrough results on secure multiparty computation in the late 80' prove feasibility with optimal resilience: perfect, statistical and computational security can be achieved as long as $t < n/3$ [7], $t < n/2$ (assuming broadcast) [36] and $t < n$ [27,40], respectively, where n is the number of computing parties such that at most t of them are controlled by a malicious adversary.

In this paper we focus on secure computation with perfect security, which is the strongest possible guarantee: it provides unconditional, everlasting security. Such protocols come with desirable properties. They often guarantee adaptive security [12,32] and remain secure under universal composition [11]. A central foundational result in this context is the Completeness Theorem of Ben-or, Goldwasser, and Wigderson [7] from 1988:

Theorem 1.1 (BGW with improvements [3,7,18,25]- informal)**.** *Let f be an n-ary functionality and C its arithmetic circuit representation. Given a synchronous network with pairwise private channels and a broadcast channel, there exists a protocol for computing f with perfect security in the presence of a static malicious adversary controlling up to $t < n/3$ parties, with round complexity $O(\mathsf{depth}(C))$ and communication complexity of $O(n^4 \cdot |C|)$ words in point-to-point channels and no broadcast in the optimistic case, and additional $\Omega(n^4 \cdot |C|)$ words of broadcast in the pessimistic case.*[1]

The communication complexity in the above statement (and throughout the paper) is measured in words (i.e., field elements), and we assume a word of size $O(\log n)$ bits.

In the past three decades there has been great efforts to improve the communication complexity of the BGW protocol [3,25]. Theorem 1.1 states the round and communication complexity of the protocols after these improvements. Most recently, Goyal, Liu and Song. [28], building upon Beaver [5], and Beerliová and Hirt [6], achieved $O(n|C| + n^3)$ communication words (including all broadcast costs) at the expense of increasing the round complexity to $O(n + \mathsf{depth}(C))$.

In some natural setting, e.g., secure computation of shallow circuits in high latency networks, this additive $O(n)$ term in the round complexity might render the protocol inapplicable. This state of affairs leads to the fundamental question of whether the communication complexity of perfectly secure computation can be improved *without* sacrificing the round complexity. Moreover, from theoretical perspective, the tradeoff between round complexity and communication complexity is an interesting one.

[1] In the optimistic case the adversary does not deviate from the prescribed protocol. Thus, in the pessimistic case (when it does deviate from the protocol) the adversary might only make the execution more expensive.

1.1 Our Results

We show an improvement of the communication complexity of perfectly secure protocols, without incurring any cost in round complexity. Notably, our improvement applies both to the optimistic case and to the pessimistic case:

Theorem 1.2 (Main technical result - informal)**.** *Let f be an n-ary functionality and C its arithmetic circuit representation. Given a synchronous network with pairwise private channels and a broadcast channel, there exists a protocol for computing f with perfect security in the presence of a static malicious adversary controlling up to $t < n/3$ parties, with round complexity $O(\mathsf{depth}(C))$ and communication complexity of $O(n^3 \cdot |C|)$ words on point-to-point channels and no broadcast in the optimistic case, and additional $O(n^3 \cdot |C|)$ words of broadcast in the pessimistic case.*

Our result strictly improves the state of the art and is formally incomparable to the result of Goyal et al. [28]. Our protocol will perform better in high-latency networks (e.g., the internet) on shallow circuits when $\mathsf{depth}(C) \ll n$. Whereas the protocol of [28] performs better in low-latency networks (e.g., LAN), or when $\mathsf{depth}(C) \approx \Omega(n)$.

Sub-linear perfect MPC for sub-circuits of depth-1. As our second main result, we show for the first time that for a non-trivial class of functions, there is in fact a *sub-linear* communication perfectly secure MPC (in the circuit size). Specifically, we design a perfectly secure MPC that supports all functionalities that can be computed by depth 1 circuits. The communication complexity of our protocol depends only on the input and output sizes of the function, but not on the circuit size, i.e., the number of multiplications. We prove the following:

Theorem 1.3. *Let $n > 3t$, and let $\mathbb{F}$ be a finite field with $|\mathbb{F}| > n$. For every arithmetic circuit $G : \mathbb{F}^L \to \mathbb{F}^M$ of multiplication depth 1 (i.e., degree-2 polynomial), there exists a perfect t-secure protocol that computes $(y_1, \ldots, y_M) = G(x_1, \ldots, x_L)$ in $O(1)$ rounds and $O((M + L) \cdot n^3)$ words over the point-to-point channels in the optimistic case, and additional $O((M + L) \cdot n^3)$ broadcast messages in the pessimistic case. Specifically, the communication complexity is independent of $|G|$.*

The above theorem can also be applied to compute circuits with higher depth, while paying only communication complexity that is proportional to the number of wires between the layers, and independent of the number of multiplications in each layer. Similar techniques were shown in the statistical case [14], but no protocol is known for perfect security.

Application: Secure Matrix Multiplication. As a leading example of the usefulness of our depth 1 circuit protocol, consider matrix multiplication of two $T \times T$ matrices. This operation has inputs and outputs of size $O(T^2)$, but implementing it requires $O(T^3)$ multiplications (at least when implemented naïvely). The starting point (Theorem 1.1) is $\Omega(T^3 \cdot n^4)$ point-to-point in the optimistic case

(and additional $\Omega(T^3 \cdot n^4)$ words of broadcast in the pessimistic case. Theorem 1.3 improves the communication complexity to $O(T^2 \cdot n^3)$ in the point-to-point channels with no additional broadcast in the optimistic case (and additional $O(T^2 \cdot n^3)$ words on broadcast in the pessimistic case). Our protocol also achieves $O(1)$ rounds in both the optimistic and pessimistic cases.

Secure matrix multiplication is a key building block for a variety of appealing applications. For example, anonymous communication [1] and secure collaborative learning. The latter involves multiplication of many large matrices (see [4,13,33–35,39], to name a few). For instance, the deep convolutional neural network (CNN) ResNet50 [38] requires roughly 2000 matrix multiplications, which, when computed securely, results in more than 4 billion multiplication gates. Using our protofocol of matrix multiplication, computing this task reduces by order of magnitudes, the communication to be proportional to computing only millions multiplications.

Secure Multiplication: A Natural Barrier of $\Omega(n)$ Secret Sharings
We give a very high level overview of our technical controbution, pointing to the core of our improvements. When viewed from afar, all secret-sharing based MPC protocols have a very similar flow. The starting point property is that polynomial secret sharing is additively homomorphic. This allows computing any linear combination (additional and multiplication by public constants) of secrets locally and with no interaction. The challenge is with multiplication gates: while multiplication can also be applied homomorphically (and non-interactively), it increases the degree of the underlying polynomial that hides the secret. Secure multiplication uses the fact that polynomial interpolation is just a linear combination of points on the polynomial, and hence a central part of the computation can be applied locally.

Given shares of the two inputs, every party shares a new secret which is its locally computed multiplication of its two shares. Then, all these new shares are locally combined using the linear combination of the publicly known Lagrange coefficients. This results in the desired new sharing of the multiplication of the two inputs.

This elegant framework for secure multiplication embeds a natural communication complexity barrier: each multiplication requires $\Omega(n)$ secret sharing (each party needs to secret share its local multiplication). In the malicious case, the secret sharing protocol is Verifiable Secret Sharing (VSS), hence, the total communication complexity in this framework is at least $\Omega(n \cdot \mathsf{comm}(VSS))$.

State of the art MPC for almost all settings matches this natural barrier, obtaining constant round protocols with optimal resilience using $O(n \cdot \mathsf{comm}(VSS))$ communication per multiplication complexity, where VSS is the best secret sharing for that setting.

The only exception we are aware of is the family pf BGW protocols for a malicious adversary, where all known improvements until now [3,7,25] require $\Omega(n^2 \cdot \mathsf{comm}(VSS))$ communication. This is because each party needs to share n invocations of VSSs of degree-t polynomials in order to prove that the secret

it shared for the product is indeed equal to multiplication of the already shared multiplicands.

Weak VSS and the complexity of perfect MPC. The main technical contribution of this work is a multiplication protocol that meets the natural barrier and achieves communication complexity of $O(n \cdot \mathsf{comm}(VSS))$. Since $\mathsf{comm}(VSS)$ is $O(n^2)$ words in the optimistic case (and no broadcast) and $O(n^2)$ over the point-to-point channels and additional $O(n^2)$ words of broadcast in the pessimistic case, Theorem 1.2 is obtained. The improvement can thus be described as follows:

- Semi-honest BGW requires $O(n \cdot \mathsf{comm}(SS))$ communication per multiplication.
- Malicious BGW requires $O(n^2 \cdot \mathsf{comm}(VSS))$ communication per multiplication.
- Our malicious protocol requires $O(n \cdot \mathsf{comm}(VSS))$ communication per multiplication.

Our improved efficiency is obtained by replacing n invocations of degree-t VSSs with just one invocation of a *weak* VSS for degree-$2t$, which we denote by WSS. By weak VSS, we refer to the setting in which the parties' shares define a single secret at the end of the sharing phase, and during the reconstruction phase, the parties can either recover that secret or $\perp$. We show that a single weak VSS for a degree-$2t$ polynomial (along with a constant number of strong VSS) is sufficient to prove that the secret shared for the product is equal the multiplication of its two already shared multiplicands.

Lemma 1.4 (informal)**.** *Given $n > 3t$, there is a protocol for implementing Weak Verifiable Secret Sharing with optimal resilience, for a polynomial of degree-$2t$ with communication complexity of $O(n^2)$ words on point-to-point channels in the optimistic case, and additional $O(n^2)$ words of broadcast in the pessimistic case, and $O(1)$ rounds.*

Our new weak verifiable secret sharing of degree-$2t$ has the same asymptotic complexity as verifiable secret sharing of degree-t. In addition to improving the efficiency of the core building block in secure computation (i.e. the multiplication), we believe it also makes it simpler, which is a pedagogical benefit.

Adaptive Security and UC. Protocols that achieve perfect security have substantial advantage over protocols that are only computationally secure: It was shown [32] that perfectly secure protocols in the stand-alone setting with a black-box straight-line simulator are also secure under universal composition [11]. Moreover, it was shown [12] that perfectly secure protocols in presence of a static malicious adversary (under the security definition in [22]) enjoy also perfect security in the presence of an adaptive malicious adversary, albeit with the weaker guarantee provided by inefficient simulation. We prove security in the classic setting of a static adversary and stand-alone computation. This implies UC security. The additional requirements under the definition of [22] hold in our protocols, and thus we derive also security in the presence of adaptive adversary (with inefficient simulation).

The Broadcast Channel Model. We analyze our protocol in the broadcast model and count messages sent over private channels and over the broadcast channel separately. In our setting ($t < n/3$) the broadcast channel can also be simulated over the point-to-point channels. However, this comes with some additional cost. There are two alternatives: replace each broadcast use in the protocol requires $O(n^2)$ communication and $O(n)$ rounds [8,16], or $O(n^4 \log n)$ communication and expected constant round (even with bounded parallel composition [17,24,31]).

1.2 Related Work

Constant-Round per Multiplication. In this paper we focus on perfect security in the presence of a malicious adversary, optimal resilience and constant round per multiplication. Our protocol improves the state of the art in this line of work. As mentioned in Asharov, Lindell and Rabin [3], an additional verification protocol is needed for completing the specification of the multiplication step of BGW. In Theorem 1.1, we ignore the cost associated with those verification steps and just count the number of verifiable secret sharing needed, which is $\Omega(n^2)$ VSSs per multiplication gate. The protocol presented by Asharov, Lindell and Rabin [3] also requires $O(n^2)$ VSSs per multiplication gate. Cramer, Damgård and Maurer [18] presented a protocol that works in a different way to the BGW protocol, which also achieves constant round per multiplication. It has worst-case communication complexity of $O(n^5)$ field elements over point-to-point channels and $O(n^5)$ field elements over a broadcast channel. The optimistic cost is $O(n^4)$ field elements over point-to-point channels and $O(n^3)$ field elements over the broadcast channel.

Protocols that are Based on the Player Elimination Technique. There is a large body of work [6,19,28–30] that improves the communication complexity of information-theoretic protocols using the player elimination technique. All of these protocols have a round complexity that is linear in the number of parties. This is inherent in the player elimination technique since every time cheating is detected, two players are eliminated and some computations are repeated. In many cases player elimination would give a more efficient protocol than our approach. However, there are some cases, specifically for a low-depth circuit where n is large and over high-latency networks, in which our protocol is more efficient. Moreover, our protocol can achieve communication complexity which is sub-linear in the number of multiplication gates, depends on the circuits to be evaluated. We do not know how to achieve similar results on protocols that are based on Beaver multiplication triplets [5], such as the protocol of Goyal et al. [28]. These lines of work are therefore incomparable.

Lower Bounds. Recently, Damgård and Schwartzbach [21] showed that for any n and all large enough g, there exists a circuit C with g gates such that any perfectly secure protocol implementing C must communicate $\Omega(ng)$ bits. Note that Theorem 1.3 is sub-linear (in the circuit size) only for particular kind of circuits in which the circuit is much larger than the size of the inputs or its

outputs. It is easy to find a circuit C with g gates in which our protocol must communication $O(n^4 g)$ in the pessimistic case. A lower bound by Damgård et al. [20] shows that any perfectly-secure protocol that works in the "gate-by-gate" framework must communicate $\Omega(n)$ bits for every multiplication gate. Our protocol deviates from this framework when computing an entire multiplication layer as an atomic unit.

1.3 Open Problems

Our protocol improves the communication complexity of constant round multiplication with optimal malicious resilience from $O(n^2 \cdot \mathsf{comm}(VSS))$ to $O(n \cdot \mathsf{comm}(VSS))$, matching the number of secret-shares in the semi-honest protocol. The immediate open problem is exploring the optimal communication complexity of verifiable secret sharing protocol. To the best of our knowledge, we are not aware of any non-trivial lower bound for perfect VSS (also see survey by C, Choudhury and Patra [9]). The VSS protocol requires $O(n^2)$ words in the optimistic case over the point-to-point channel, and additional $O(n^2)$ words over the broadcast channel in the pessimistic case.

Another possible direction to generalize our work is to mitigate between the two approaches for perfect security: Design a "hybrid" protocol that computes some sub-circuits using the linear communication complexity approach, and some sub-circuits using the constant-round per multiplication approach and achieving the best of both worlds. Another interesting direction is to make sub-linear communication complexity improvement compatible with the protocols that are based on multiplication triplets.

2 Technical Overview

In this section we provide a technical overview of our results. We start with an overview of the BGW protocol in Sect. 2.1 and then overview our protocol in Sect. 2.2.

2.1 Overview of the BGW Protocol

In the following, we give a high level overview of the BGW protocol while incorporating several optimization that were given throughout the years [3,25].

Let f be the function that the parties wish to compute, mapping n inputs to n outputs. The input of party P_i is x_i and its output is y_i, where $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$. On a high level, the BGW protocol works by emulating the computation of an arithmetic circuit C that computes f and has three phases. In the first phase, the input sharing phase, each party secret shares its input with all other parties. At the end of this stage, the value of each input wire of the circuit C is secret shared among the parties, such that no subset of t parties can reconstruct the actual values on the wires. In the second phase, the circuit emulation phase, the parties emulate a computation of the circuit gate-by-gate,

computing shares on the output wire of each gate using the shares on the input wires. At the end of this stage, the output wires' values are secret shared among all parties. Finally, in the output reconstruction phase, P_i receives all the shares associated with its output wire and reconstructs its output, y_i.

The invariant maintained in the original BGW protocol is that each wire in the circuit, carrying some value a, is secret-shared among the parties using some random polynomial $A(x)$ of degree-t with a as its constant term. We follow the invariant of [3], and in our protocol, the parties hold bivariate sharing and not univariate sharing. That is, the secret is hidden using a bivariate polynomial $A(x, y)$ of degree-t in both variables in which the share of each party P_i is defined as $A(x, \alpha_i), A(\alpha_i, y)$, where α_i is the evaluation point associated with P_i. Maintaining bivariate sharing instead of univariate sharing removes one of the building blocks in the original BGW protocol, where parties sub-share their shares to verify that all the shares lie on a polynomial of degree-t. Obtaining bivariate sharing essentially comes for free. In particular, when parties share a value, they use a verifiable secret sharing protocol (VSS, see Sect. 2.2) [15,23,24], which uses bivariate sharing to verify that all the shares are consistent. However, in BGW, the parties then disregard this bivariate sharing and project it to univariate sharing. We just keep the shares in the bivariate form.

The Multiplication Protocol. In the input sharing phase, each party simply shares its input using the BGW's VSS protocol. Emulating the computation of addition gates is easy using linearity of the secret sharing scheme. The goal in the multiplication protocol is to obtain bivariate sharing of the value of the output wire of the multiplication gate using the shares on the input wires. Let a, b be the two values on the input wires, hidden with polynomials $A(x, y), B(x, y)$, respectively. The protocol proceeds as follows:

1. Each party P_i holds shares $f_i^a(x) = A(x, \alpha_i)$ and $f_i^b(x) = B(x, \alpha_i)$, each are univariate polynomials of degree-t. Each party P_i shares a bivariate polynomial $C_i(x, y)$ of degree-t such that $C_i(0, 0) = f_i^a(0) \cdot f_i^b(0)$.
2. Using a *verification protocol*, each party P_i proves in perfect zero knowledge that $C_i(0, 0) = f_i^a(0) \cdot f_i^b(0)$. We elaborate on this step below.
3. Given the shares on all (degree-t) polynomials $C_1(x, y), \ldots, C_n(x, y)$, the parties compute shares of the polynomial $C(x, y) \stackrel{\text{def}}{=} \sum_{i=1}^{n} \lambda_i \cdot C_i(x, y)$, where $\lambda_1, \ldots, \lambda_n$ are the Lagrange coefficients, by simply locally computing a linear combination of the shares they obtained in the previous step.

To see why this protocol is correct, observe that since each one of the polynomials $C_1(x, y), \ldots, C_n(x, y)$ is a polynomial of degree-t, then the resulting polynomial $C(x, y)$ is also a polynomial of degree-t. Moreover, define $h(y) \stackrel{\text{def}}{=} A(0, y) \cdot B(0, y)$ and observe that $h(y)$ is a polynomial of degree-$2t$ satisfying $h(0) = A(0, 0) \cdot B(0, 0) = ab$. It holds that $ab = \lambda_1 \cdot h(\alpha_1) + \ldots + \lambda_n \cdot h(\alpha_n)$. Thus,

$$C(0,0) \stackrel{\text{def}}{=} \sum_{i=1}^{n} \lambda_i \cdot C_i(0,0) = \sum_{i=1}^{n} \lambda_i \cdot f_i^a(0) \cdot f_i^b(0) = \sum_{i=1}^{n} \lambda_i \cdot h(\alpha_i) = ab \ ,$$

as required. Crucially, each $C_i(x,y)$ must hide $h(\alpha_i) = f_i^a(0) \cdot f_i^b(0)$ as otherwise the above linear combination would not result with the correct constant term. This explains the importance of the verification protocol.

BGW's verification protocol. In the verification protocol, the dealer holds the univariate polynomials $f_i^a(x), f_i^b(x)$ and a polynomial $C_i(x,y)$, and each party P_j holds a share on those polynomials, that is, points $f_i^a(\alpha_j), f_i^b(\alpha_j)$ and degree-t univariate polynomials $C_i(x,\alpha_j)$, $C_i(\alpha_j,y)$. The parties wish to verify that $C_i(0,0) = f_i^a(0) \cdot f_i^b(0)$.

Towards that end, the dealer defines random degree-t polynomials $D_1,\ldots,D_t$ under the constraint that

$$C_i(x,0) = f_i^a(x) \cdot f_i^b(x) - \sum_{\ell=1}^{t} x^\ell \cdot D_\ell(x,0) \ . \tag{1}$$

As shown in [3,7], the dealer can choose the polynomials $D_1,\ldots,D_t$ in a special way so as to cancel all the coefficients of degree higher than t of $f_i^a(x) \cdot f_i^b(x)$ and to ensure that $C_i(x,y)$ is of degree t. The dealer verifiably shares the polynomials $D_1,\ldots,D_t$ with all parties, and then each party P_k verifies that the shares it received satisfy Eq. (1). If not, it complaints against the dealer. Note that at this point, since all polynomials $C_i, D_1,\ldots,D_t$ are bivariate polynomial of degree-t, and $f_i^a(x), f_i^b(x)$ are univariate polynomials of degree-t, it is possible to reconstruct the shares of any party P_k without the help of the dealer. The parties can then unequivocally verify the complaint. If a complaint was resolved to be a true complaint, the dealer is dishonest, we can reconstruct its points and exclude it from the protocol. If the complaint is false, we can also eliminate the complaining party.

An honest dealer always distributes polynomials that satisfy Eq. (1). For the case of a corrupted dealer, the term $f_i^a(x) \cdot f_i^b(x) - \sum_{\ell=1}^{t} x^\ell \cdot D_\ell(x,0)$ defines a univariate polynomial of degree at most $2t$ for every choice of degree-t bivariate polynomials $D_1,\ldots,D_t$. If this polynomial agrees with the polynomial $C_i(x,0)$ for all honest parties, i.e., on $2t+1$ points, then those two polynomials are identical, and thus it must hold that $C_i(0,0) = f_i^a(0) \cdot f_i^b(0)$, as required.

2.2 Our Protocol

Simplifying the Verification Protocol. In the above verification protocol, the dealer distributes t polynomials $D_1,\ldots,D_t$ using VSS. We show how to use a more efficient technique for accomplishing the verification task. Namely, we introduce a weak secret sharing protocol, for sharing a polynomial $D(x,y)$ of degree-$2t$ in x and degree-t in y. The dealer then chooses a *single* random polynomial $D(x,y)$ under the constraint that:

$$C_i(x,0) = f_i^a(x) \cdot f_i^b(x) - D(x,0) \tag{2}$$

The dealer distributes $D(x,y)$ and the parties jointly verify that (a) Eq. (2) holds and (b) that $D(0,0) = 0$.

Our weak secret sharing protocol for distributing such $D(x, y)$ has *the same complexity as verifiable secret sharing of a degree-t polynomial*, and therefore we improve by a factor of $t = O(n)$. The secret sharing is weak in the sense that the parties cannot necessarily reconstruct the secret from the shares without the help of the dealer during the reconstruction. However, the verifiability part guarantees that there is a well-defined polynomial that can be reconstructed (or, if the dealer does not cooperate, then no polynomial would be reconstructed). Since the role of the polynomial $D(x, y)$ is just in the verification phase and requires the involvement of the dealer, to begin with, this weak verifiability suffices. If the dealer does not cooperate during the verification phase, then the parties can reconstruct its inputs and resume the computation on its behalf.

Our Weak Secret Sharing. Our weak verifiable secret sharing protocol is similar to the BGW verifiable secret sharing protocol. Introducing modifications to the protocol enables sharing of a polynomial of a higher degree, but in that case – satisfies only weak verifiability. We start with an overview of the verifiable secret sharing protocol and then describe our weak secret sharing protocol.

The Verifiable Secret Sharing Protocol. In a nutshell, the verifiable secret sharing protocol of BGW (with the simplifications of [23]) works as follows:

1. **Sharing:** The dealer wishes to distribute shares of a polynomial $D(x, y)$ of degree t in both variables. The dealer sends to each party P_i the degree-t univariate polynomials $f_i(x) = D(x, \alpha_i)$ and $g_i(y) = D(\alpha_i, y)$.
2. **Exchange sub-shares:**
   Each party P_i sends to party P_j the pair $(f_i(\alpha_j), g_i(\alpha_j))$. Note that if indeed the dealer sent correct shares, then $f_i(\alpha_j) = D(\alpha_j, \alpha_i) = g_j(\alpha_i)$ and $g_i(\alpha_j) = D(\alpha_i, \alpha_j) = f_j(\alpha_i)$. If a party does not receive from P_j the shares it expects to receive, then it broadcasts a complaint. The complaint has the form of $\mathsf{complaint}(i, j, f_i(\alpha_j), g_i(\alpha_j))$, i.e., P_i complaints that it receives from P_j wrong points, and publishes the two points that it expected to receive, corresponding to the information it had received from the dealer.
3. **Complaint resolution – the dealer:** The dealer publicly reveals all the shares of all parties that broadcast false complaints – i.e., if party P_i complaints with points different than those given in the first round, then the dealer makes the share $(f_i(x), g_i(y))$ public.
4. **Vote:** The parties vote that whatever they saw is consistent. A party is happy with its share and broadcasts good if: (a) Its share was not publicly revealed. (b) The dealer resolved all conflicts the party saw in the exchange sub-shares phase, i.e., all its complaints were resolved by the dealer by publicly opening the other parties' shares. (c) All shares that the dealer broadcasts are consistent with its shares. (d) There are no parties (j, k) that complain of each other, and the dealer did not resolve at least one of those complaints. If $2t+1$ parties broadcast good then the parties accept the shares. A party that its share was publicly revealed updates its share to be the publicly revealed one.

Note that if more than $2t+1$ parties broadcast good then more than $t+1$ honest parties are happy with their shares. Those shares determine a unique bivariate polynomial of degree-t. Moreover, any polynomial that is publicly revealed must be consistent with this bivariate polynomial, as agreeing with the points of $t+1$ honest parties uniquely determine a polynomial of degree-t.

Weak Secret Sharing. Consider this protocol when the dealer shares a polynomial $D(x,y)$ that is of degree-$2t$ in x and degree-t in y, i.e., $D(x,y) = \sum_{i=0}^{2t}\sum_{j=0}^{t} d_{i,j}x^i y^j$ for some set of coefficients $\{d_{i,j}\}_{i,j}$. Here, if $t+1$ honest parties are happy with their shares and broadcast good, their polynomials also define a unique polynomial $D(x,y)$ of degree-$2t$ in x and degree-t in y. However, if there is a complaint and the dealer opens some party's share, since $f_i(x)$ is of degree-$2t$ it is not sufficient that these $t+1$ honest parties agree with that polynomial $f_i(x)$, and $f_i(x)$ might still be "wrong". This implies that the honest parties cannot identify whether their shares are compatible with the shares of the other honest parties (that their shares were publicly revealed), and further verification is needed, which seems to trigger more rounds of complaints. Guaranteeing all honest parties obtain consistent shares is a more challenging task.

To keep the protocol constant round, we therefore take a different route and do not require the dealer to publicly open any of the $f_i(x)$ polynomials! Still, it has to publicly open only the $g_i(y)$ polynomials, as those are of degree-t. Each honest party broadcasts good only if the same conditions as in VSS are met. At the end of this protocol, some honest parties might not hold $f_i(x)$ shares on the polynomial $D(x,y)$. Those parties will not participate in the reconstruction protocol. In the reconstruction phase, since the corrupted parties might provide incorrect shares and since some honest parties do not have shares, we cannot guarantee reconstruction of the polynomial $D(x,y)$ without the help of the dealer. However, we can guarantee that only the polynomial $D(x,y)$ can be reconstructed, or no polynomial at all.

Concluding the Multiplication Protocol. Recall that in our protocol, the parties also have to jointly verify that (a) Eq. 2 holds, and that (b) that $D(0,0) = 0$. We now elaborate on those two steps.

To verify that the polynomial $D(x,y)$ satisfies $D(x,0) = f_i^a(x) \cdot f_i^b(x) - C_i(x,0)$, each party P_j simply checks that its own shares satisfy this condition, i.e., whether $D(\alpha_j,0) = f_i^a(\alpha_j) \cdot f_i^b(\alpha_j) - C_i(\alpha_j,0)$. Note that if this holds for $2t+1$ parties, then the two polynomials are identical. Each party P_j checks its own shares, and if the condition does not hold then it broadcasts complaint(j). With each complaint the dealer has to publicly reveal the shares of P_j. Since all those polynomials were shared using (weak or strong) verifiable secret sharing, the parties can easily verify whether the shares that the dealer opens are correct or not.

To check that $D(0,0) = 0$, the parties simply reconstruct the polynomial $D(0,y)$. This is a polynomial of degree-t and it can be reconstructed (with the help of the dealer, as D is shared using a weak secret sharing scheme). Moreover,

it does not reveal any information on the polynomials $f_i^a(x), f_i^b(x), C_i(x,0)$: In case of an honest dealer, the adversary already holds t shares on the polynomial $D(0,y)$ and it always holds that $D(0,0)=0$, since the dealer is honest.

2.3 Extensions

Our zero knowledge verification protocol allows the dealer to prove that its shares of a, b, c satisfy the relation $c = ab$. The cost of the protocol is proportional to a constant number of VSSs. We show an extension of the protocol allowing a dealer that its shares of $(x_1, \ldots, x_L), (y_1, \ldots, y_M)$ satisfy $(y_1, \ldots, y_M) = G(x_1, \ldots, x_L)$, where G is any circuit of multiplication depth 1 (i.e., a degree-2 polynomial). The communication complexity of the protocol is $O(L+M)$ VSSs and not $O(|G|)$ VSSs (where —G— is the number of multiplication gates in the circuit G). This allows computing the circuit in a layer-by-layer fashion and not gate-by-gate and leads to sub-linear communication complexity for circuits where $|G| \in \omega(L+M)$.

2.4 Organization

The rest of the paper is organized as follows. In Sect. 3 we provide preliminaries and definitions. In Sect. 4 we cover our weak verifiable secret sharing, strong verifiable secret sharing and some extensions. Our multiplication protocol (with a dealer) is provided in Sect. 5 and its generalization to arbitrary gates with multiplicative gate 1 is given in Sect. 6. In the full version of the paper we provide the missing proofs, as well as an overview of how the dealer is removed and how to compute a general function, following the BGW approach.

3 Preliminaries

Notations. We denote $\{1, \ldots, n\}$ by $[n]$. We denote the number of parties by n and a bound on the number of corrupted parties by t. Two random variables X and Y are identically distributed, denoted as $X \equiv Y$, if for every z it holds that $\Pr[X = z] = \Pr[Y = z]$. Two parametrized distributions $\mathcal{D}_1 = \{\mathcal{D}_1(a)\}_a$ and $\mathcal{D}_2 = \{\mathcal{D}_2(a)\}_a$ are said to be identically distributed, if for every a the two random variables $(a, \mathcal{D}_1(a)), (a, \mathcal{D}_2(a))$ are identically distributed.

3.1 Definitions of Perfect Security in the Presence of Malicious Adversaries

We follow the standard, standalone simulation-based security of multiparty computation in the perfect settings [2,10,26]. Let $f : (\{0,1\}^*)^n \to (\{0,1\}^*)^n$ be an n-party functionality and let π be an n-party protocol over ideal (i.e., authenticated and private) point-to-point channels and a broadcast channel. Let the adversary, $\mathcal{A}$, be an arbitrary machine with auxiliary input z, and let $I \subset [n]$ be the set of corrupted parties controlled by $\mathcal{A}$. We define the real and ideal executions:

- **The real execution:** In the real model, the parties run the protocol π where the adversary $\mathcal{A}$ controls the parties in I. The adversary cannot modify messages sent over the point-to-point channel. The adversary is assumed to be rushing, meaning that in every round it can see the messages sent by the honest parties before it determines the message sent by the corrupted parties. We denote by $\text{REAL}_{\pi,\mathcal{A}(z),I}(\vec{x})$ the random variable consisting of the view of the adversary $\mathcal{A}$ in the execution (consisting of all the initial inputs of the corrupted parties, their randomness and all messages they received), together with the output of all honest parties.
- **The ideal execution:** The ideal model consists of all honest parties, a trusted party and an ideal adversary $\mathcal{SIM}$, controlling the same set of corrupted parties I. The honest parties send their inputs to the trusted party. The ideal adversary $\mathcal{SIM}$ receives the auxiliary input z and sees the inputs of the corrupted parties. $\mathcal{SIM}$ can substitute any x_i with any x'_i of its choice (for the corrupted parties) under the condition that $|x'_i| = |x_i|$. Once the trusted party receives (possibly modified) inputs $(x'_1, \ldots, x'_n)$ from all parties, it computes $(y_1, \ldots, y_n) = f(x'_1, \ldots, x'_n)$ and sends y_i to P_i. The output of the ideal execution, denoted as $\text{IDEAL}_{f,\mathcal{SIM}(z),I}(\vec{x})$ is the output of all honest parties and the output of the ideal adversary $\mathcal{SIM}$.

Definition 3.1. *Let f and π be as above. We say that π is t-secure for f if for every adversary $\mathcal{A}$ in the real world there exists an adversary $\mathcal{SIM}$ with comparable complexity to $\mathcal{A}$ in the ideal model, such that for every $I \subset [n]$ of cardinality at most t it holds that*

$$\left\{\text{IDEAL}_{f,\mathcal{SIM}(z),I}(\vec{x})\right\}_{z,\vec{x}} \equiv \left\{\text{REAL}_{\pi,\mathcal{A}(z),I}(\vec{x})\right\}_{z,\vec{x}}$$

where $\vec{x}$ is chosen from $(\{0,1\}^*)^n$ such that $|x_1| = \ldots = |x_n|$.

Corruption-aware Functionalities. The functionalities that we consider are *corruption-aware*, namely, the functionality receives the set I of corrupted parties. We refer the reader to [2, Section 6.2] for further discussion and the necessity of this when proving security.

Reactive Functionalities, Composition and Fybrid-world. We also consider more general functionalities where the computation takes place in stages, where the trusted party can communicate with the ideal adversary (and sometimes also with the honest parties) in several stages, to obtain new inputs and send outputs in phases. See [26, Section 7.7.1.3].

The sequential modular composition theorem is an important tool for analyzing the security of a protocol in a modular way. Assume that π_f is a protocol that securely computes a function f that uses a subprotocol π_g, which in return securely computes some functionality g. Instead of showing directly that π_f securely computes f, one can consider a protocol π_f^g that does not use the subprotocol π_g but instead uses a trusted party that ideally computes g (this is called a protocol for f in the g-hybrid model). Then, by showing that (1)

π_g securely implements g, and; (2) π_f^g securely implements f, we obtain that the protocol π_f securely implements f in the plain model. See [10] for further discussion.

Remark 3.2 (Input assumption)**.** *We sometimes present functionalities in which we assume that the inputs satisfy some guarantee, for instance, that all points of the honest parties lie on the same degree-t polynomial. We remark that if the input assumption does not hold, then no security guarantees are obtained. This can be formalized as follows: In case that the condition does not hold (and the functionality can easily verify that), then it gives all the honest parties' inputs to the adversary and let the adversary singlehandedly determine all of the outputs of the honest parties. Clearly, any protocol can then be simulated. Note, however, that we always invoke the sub-protocols when the input assumptions are satisfied.*

3.2 Robust Secret Sharing

Let $\mathbb{F}$ be a finite field of order greater than n, let $\alpha_1, \ldots, \alpha_n$ be any distinct non-zero elements from $\mathbb{F}$ and denote $\vec{\alpha} = (\alpha_1, \ldots, \alpha_n)$. For a polynomial q, denote $\mathsf{Eval}_{\vec{\alpha}}(q) = (q(\alpha_1), \ldots, q(\alpha_n))$. The Shamir's $t+1$ out of n sharing scheme [37] consists of two procedure Share and $\mathsf{Reconstruct}$ as follows:

- $\mathsf{Share}(s)$. The algorithm is given $s \in \mathbb{F}$, then it chooses t independent uniformly random elements from $\mathbb{F}$, denoted $q_1, \ldots, q_t$, and defines the polynomial $q(x) = s + \sum_{i=1}^{t} q_t x^t$. Finally, it outputs $\mathsf{Eval}_{\vec{\alpha}}(q) = (q(\alpha_1), \ldots, q(\alpha_n))$. Define $s_i = q(\alpha_i)$ as the share of party P_i.
- $\mathsf{Reconstruct}(\vec{s})$. For a set $J \subseteq [n]$ of cardinality at least $t+1$, let $\vec{s} = \{s_i\}_{i \in J}$. Then, the algorithm reconstructs the secret s.

Correctness requires that every secret can be reconstructed from the shares for every subset of shares of cardinality $t+1$, and secrecy requires that every set of less than t shares is distributed uniformly in $\mathbb{F}$. We refer to [2] for a formal definition.

Reed Solomon Code. Recall that a linear $[n, k, d]$-code over a field $\mathbb{F}$ is a code of length n, dimension k and distance d. That is, each codeword is a sequence of n field elements, there are in total $|\mathbb{F}|^k$ different codewords, and the Hamming distance of any two codewords is at least d. Any possible corrupted codeword $\hat{c}$ can be corrected to the closest codeword c as long as $d(c, \hat{c}) < (d-1)/2$, where $d(x, y)$ denote the Hamming distance between the words $x, y \in \mathbb{F}^n$.

In Reed Solomon code, let $m = (m_0, \ldots, m_t)$ be the message to be encoded, where each $m_i \in \mathbb{F}$. The encoding of the message is essentially the evaluation of the degree-t polynomial $p_m(x) = m_0 + m_1 x + \ldots + m_t x^t$ on some distinct non-zero field elements $\alpha_1, \ldots, \alpha_n$. That is, $\mathsf{Encode}(m) = (p(\alpha_1), \ldots, p(\alpha_n))$. The distance of this code is $n - t$. This is because any two distinct polynomials of degree-t can agree at most t points. We have the following fact:

Fact 3.3. *The Reed-Solomon code is a linear* $[n, t+1, n-t]$ *code over* $\mathbb{F}$. *In addition, there exists an efficient decoding algorithm that corrects up to* $(n-t-1)/2$ *errors. That is, for every* $m \in \mathbb{F}^{t+1}$ *and every* $x \in \mathbb{F}^n$ *such that* $d(x, C(m)) \leq (n-t-1)/2$, *the decoding algorithm returns* m.

For the case of $t < n/3$ we get that is is possible to efficiently correct up to $(3t+1-t-1)/2 = t$ errors. Putting it differently, when sharing of a polynomial of degree-t, if during the reconstruction t errors were introduced by corrupted parties, it is still possible to recover the correct value.

3.3 Bivariate Polynomial

We call a bivariate polynomial of degree q in x and degree t in y as (q,t)-bivariate polynomial. If $q = t$ then we simply call the polynomial as degree-t bivariate polynomial. Such a polynomial can be written as follows:

$$S(x,y) = \sum_{i=0}^{q} \sum_{j=0}^{t} a_{i,j} x^i y^j \ .$$

Looking ahead, in our protocol we will consider degree-t bivariate polynomials and degree $(2t,t)$-bivariate polynomials. The proof of the following claim is given in the full version of this paper:

Claim 3.4 (Interpolation)**.** *Let* t *be a nonnegative integer, and let* $\alpha_1, \ldots, \alpha_{t+1}$ *distinct elements in* $\mathbb{F}$, *and let* $f_1(x), \ldots, f_{t+1}(x)$ *be* $t+1$ *univariate polynomials of degree at most* q. *Then, there exists a unique* (q,t) *bivariate polynomial* $S(x,y)$ *such that for every* $k = 1, \ldots, t+1$: $S(x, \alpha_k) = f_k(x)$.

Symmetrically, one can interpolate the polynomial $S(x,y)$ from a set of $q+1$ polynomials $g_i(y)$. The proof is similar to Claim 3.4.

Claim 3.5 (Interpolation)**.** *Let* t *be a nonnegative integer, and let* $\alpha_1, \ldots, \alpha_{q+1}$ *distinct elements in* $\mathbb{F}$, *and let* $g_1(y), \ldots, g_{q+1}(y)$ *be* $q+1$ *univariate polynomials of degree at most* t *each. Then, there exists a unique* (q,t) *bivariate polynomial* $S(x,y)$ *such that for every* $k = 1, \ldots, t+1$ *it holds that* $S(\alpha_k, y) = g_k(y)$.

Hiding. The following is the "hiding" claim, showing that if a dealer wishes to share some polynomial $h(x)$ of degree-q, it can choose a random (q,t)-polynomial $S(x,y)$ that satisfies $S(x,0) = h(x)$ and give each party P_i the shares $S(x, \alpha_i), S(\alpha_i, y)$. The adversary cannot learn any information about h besides $\{h(\alpha_i)\}_{i \in I}$, when it corrupts the set $I \subset [n]$. We prove the following two claims in the full version of this paper:

Claim 3.6 (Hiding)**.** *Let* $h(x)$ *be an arbitrary univariate polynomial of degree* q, *and let* $\alpha_1, \ldots, \alpha_k$ *with* $k \leq t$ *be arbitrary distinct non-zero points in* $\mathbb{F}$. *Consider the following distribution* $\mathsf{Dist}(h)$*:*

- *Choose a random* (q,t)*-bivariate polynomial* $S(x,y)$ *under the constraint that* $S(x,0) = h(x)$.

– *Output* $\{(i, S(x, \alpha_i), S(\alpha_i, y))\}_{i\in[k]}$.

Then, for every two arbitrary degree-q polynomials $h_1(x), h_2(x)$ *for which* $h_1(\alpha_i) = h_2(\alpha_i)$ *for every* $i \in [k]$ *it holds that* $\mathsf{Dist}(h_1) \equiv \mathsf{Dist}(h_2)$.

Claim 3.7 (Hiding II)**.** *Same as Claim 3.6, except that it holds that* $h_1(0) = h_1(0) = \beta$ *for some publicly known* $\beta \in \mathbb{F}$. *The output of the distribution is* $\{(i, S(x, \alpha_i), S(\alpha_i, y))\}_{i\in[k]} \cup \{S(0, y)\}$.

4 Weak Verifiable Secret Sharing and Extensions

In this section we show how to adapt the verifiable secret sharing protocol of [7] to allow weak secret sharing of a polynomial degree-t. We start with a description of the verifiable secret sharing protocol and highlight the main differences for getting a weak verifiable secret sharing protocol (sometimes we may omit the 'verifiable' and write only 'weak secret sharing'). We formally define the weak verifiable secret sharing in Sect. 4.2 and then strong VSS in Sect. 4.4. In our formalization of weak secret sharing, not all parties are happy with their shares. The set of parties that are happy with their shares is known to all parties, and is part of the output of all parties. Their shares also uniquely define the polynomial. Only parties that are happy with their shares will take part in the reconstruction. Thus, the output of WSS is a set K of all parties that are happy with their shares, where parties in $k \in K$ also output their shares (i.e., a pair $f_k(x), g_k(y)$), where parties $i \notin K$ just hold $g_i(y)$.

We remind that in BGW, after the parties verify the shares and obtain $f_i(x), g_i(y)$, they just project the bivariate shares to univariate shares by outputting $f_i(0)$. As mentioned, we will maintain bivariate sharing and the output $(f_i(x), g_i(y))$ in the strong VSS variant of the protocol.

4.1 Verifying Shares of a (q, t)-Bivariate Polynomial

Protocol 4.1: Weak/Strong Verifiable Secret Sharing of a Polynomial

– **Input:** The dealer holds a bivariate polynomial $S(x, y)$.
– **Common input:** The description of a field $\mathbb{F}$ and n non-zero distinct elements $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$.
– **The protocol:**
1. **Sharing – the dealer:**
   (a) Send to each party P_i the shares $(f_i(x), g_i(y))$ defined as $f_i(x) \stackrel{\text{def}}{=} S(x, \alpha_i)$, $g_i(y) \stackrel{\text{def}}{=} S(\alpha_i, y)$.
2. **Initial checks – each party P_i:**
   (a) If (1) $f_i(x)$ has degree greater than q; or (2) $g_i(y)$ has degree greater than t; or (3) $f_i(\alpha_i) \neq g_i(\alpha_i)$ then broadcast $\mathsf{complaint}(i)$ and proceed to step 5.
   (b) Let $R = \{k \mid P_k$ broadcast $\mathsf{complaint}(k)\}$.

3. **Exchange subshares – each party P_i for $i \notin R$:**
   (a) Send $(f_i(\alpha_j), g_i(\alpha_j))$ to P_j for each $j \notin R$.
   (b) Let (u_j, v_j) be the values received from P_j, for $j \notin R$. If no value was received, then use $(\perp, \perp)$. If $u_j \neq g_i(\alpha_j)$ or $v_j \neq f_i(\alpha_j)$ then broadcast $\mathsf{complaint}(i, j, f_i(\alpha_j), g_i(\alpha_j))$.
   (c) If no party broadcasts $\mathsf{complaint}(i, j, \cdot, \cdot)$ and $R = \emptyset$, then[2]
   VSS: *Output* $(f_i(x), g_i(y))$ *and halt.*
   WSS: *Output* $(f_i(x), g_i(y), [n])$ *and halt.*
4. **Resolve complaints – the dealer:**
   (a) If P_i that broadcasted $\mathsf{complaint}(i)$ in Step 2a, or broadcasted $\mathsf{complaint}(i, j, u, v)$ with $u \neq S(\alpha_j, \alpha_i)$ or $v \neq S(\alpha_i, \alpha_j)$ then
   VSS: *Broadcast* $\mathsf{reveal}(i, S(x, \alpha_i), S(\alpha_i, y))$.
   WSS: *Broadcast* $\mathsf{reveal}(i, S(\alpha_i, y))$.
5. **Evaluate complaint resolutions – each party P_i:**
   (a) Add to R all indices k for which the dealer broadcasted $\mathsf{reveal}(k, \ldots)$. If $i \in R$, then replace $g_i(y)$ with the one provided in the broadcasted in $\mathsf{reveal}(i, \cdot, \cdot)$.
   VSS: *If $i \in R$, then rewrite also $f_i(x)$.*
   If $i \in R$ then proceed to Step 6.
   (b) Verify that the dealer replied to each $\mathsf{complaint}(k)$ message from Step 2a with $\mathsf{reveal}(k, \ldots)$. If not, proceed to Step 6.
   (c) Upon viewing $\mathsf{complaint}(k, j, u_1, v_1)$ and $\mathsf{complaint}(j, k, u_2, v_2)$ broadcast by P_k and P_j, respectively, with $u_1 \neq v_2$ or $v_1 \neq u_2$, mark (j, k) as a joint complaint. If the dealer did not broadcast $\mathsf{reveal}(k, \cdot)$ or $\mathsf{reveal}(j, \cdot)$, then go to Step 6.
   (d) For every $j \in R$ verify that $f_i(\alpha_j) = g_j(\alpha_i)$,
   VSS: *and that* $g_i(\alpha_j) = f_j(\alpha_i)$.
   If the verification does not hold for some $j \in R$, then go to Step 6.
   (e) Broadcast the message good.
6. **Output:** Let K be the set of of all parties that broadcast good and are not in R. If $|K| < 2t + 1$, then output $\perp$. Otherwise,
   VSS: *Output* $(f_i(x), g_i(y))$.
   WSS: *Each party P_k for $k \in K$ outputs* $(f_i(x), g_i(y), K)$. *All other parties output* $(g_i(y), K)$.

It is easy to see that in the optimistic case, when there are no cheats, the protocol ends at Step 3c and incurs a communication overhead of $O(n^2)$ point-to-point messages and no broadcast. In the pessimistic (worst) case, however, there may be $O(n)$ and $O(n^2)$ complaints (broadcasts) in Steps 2a and 3b, respectively. Then, in step 4, there are $O(n)$ messages of total size $O(n^2)$ that are broadcasted

[2] We use two rounds of silence as an optimistic early stopping agreement on no complaints. We then combine this with a standard termination protocol that uses either the fast decision or the broadcast decision. It is easy to see that there will be no conflict between the two.

by the dealer (i.e. in order to reveal the polynomials of at most t parties who placed their complaint). Finally, there are $O(n)$ broadcast of the message good if the secret sharing is successfully verified. Overall, the pessimistic case incurs a communication overhead of $O(n^2)$ point-to-point messages and $O(n^2)$ broadcast messages.

4.2 Weak Verifiable Secret Sharing

In weak verifiable secret sharing, the dealer wishes to distribute shares to all parties, and then allow reconstruction only if it takes part in the reconstruction. The result of the reconstruction can be either a unique, well-defined polynomial which was determined in the sharing phase, or $\perp$.

Functionality 4.2: F_{WSS} **– Weak Verifiable Secret Sharing Functionality**

The functionality receives a set of indices $I \subset [n]$ and works as follows:

- If the dealer is honest ($1 \notin I$):
  1. Receive a polynomial $S(x, y)$ of degree (q, t) from the dealer P_1.
  2. Send to the ideal adversary the shares $\{S(x, \alpha_i), S(\alpha_i, y)\}_{i \in I}$.
  3. Receive back from the adversary the set $I' \subseteq I$ and define $K = ([n] \backslash I) \cup I'$.
- If the dealer is corrupted ($1 \in I$):
  1. Receive a polynomial $S(x, y)$ of degree (q, t) from the dealer P_1.
  2. Receive a set $K \subseteq [n]$ of cardinality at least $2t + 1$.
  3. Verify that $S(x, y)$ is of degree (q, t). If verification fails, overwrite $K = \perp$.
- **Output:** Send K to all parties. Moreover, for every $k \in K$, send $S(x, \alpha_k), S(\alpha_k, y)$ to P_k. For every $j \notin K$, send P_j the polynomial $S(\alpha_k, y)$.

Theorem 4.3. *Let $t < n/3$. Then, Protocol 4.1: when using the* **WSS** *branch is t-secure for the f_{WSS} functionality (Functionality 4.2) in the presence of a static malicious adversary. The protocol incurs $O(n^2)$ point-to-point messages in the optimistic case and additional $O(n^2)$ broadcast messages in the pessimistic case.*

Proof. Let $\mathcal{A}$ be an adversary in the real world. We have two cases, depending on whether the dealer is corrupted or not. We note that the protocol is deterministic, as well as the functionality.

Case 1: The Dealer is Honest. In this case in the ideal execution, the honest parties always hold shares of a polynomial $S(x, y)$ that is of degree (q, t). We describe the simulator $\mathcal{SIM}$.

The simulator $\mathcal{SIM}$.

1. $\mathcal{SIM}$ invokes the adversary $\mathcal{A}$ on the auxiliary input z.
2. $\mathcal{SIM}$ receives from the trusted party the polynomials of the corrupted parties, that is, $f_i(x), g_i(y)$, and the simulates the protocol execution for $\mathcal{A}$:
(a) **Sharing:** *Simulate sending the shares $f_i(x), g_i(y)$ to each P_i, $i \in I$, as coming from the dealer P_1.*
(b) **Initial checks:** *Initialize $R = \emptyset$. An honest party never broadcasts* complaint(i). *If the adversary broadcast* complaint(i), *then add i to R.*
(c) **Exchange subshares:** *send to the adversary $\mathcal{A}$ the shares $g_i(\alpha_j), f_i(\alpha_j)$ from each honest party P_j, for each corrupted party $i \in I \backslash R$. Receive from the adversary $\mathcal{A}$ the points $(u_{i,j}, v_{i,j})$ that are supposed to be sent from P_i to P_j, for $i \in I \setminus R$ and $j \notin I$.*
(d) **Broadcast complaints:** *The simulator checks the points $(u_{i,j}, v_{i,j})$ that the adversary sent in the previous step. If $u_{i,j} \neq f_i(\alpha_j)$ or $v_{i,j} \neq g_i(\alpha_j)$ then $\mathcal{SIM}$ simulates a broadcast of* complaint$(j, i, g_i(\alpha_j), f_i(\alpha_j))$ *as coming from party P_j.*
Moreover, receive complaint(i, j, u, v) *broadcast messages from the adversary.*
If the adversary does not broadcast any reveal *message and no* complaint *message was broadcasted by any party, then send I to the trusted party, and halt.*
(e) **Resolve complaints – the dealer:** *The dealer never reveals the shares of honest parties. For every* complaint(i, j, u, v) *message received from the adversary, check that $u = f_i(\alpha_j)$ and $v = g_i(\alpha_j)$.*
If not, then broadcast reveal$(i, g_i(y))$ *as coming from the dealer, and add $i \in R$. Moreover, if there was a* complaint(i) *in the initial checks step, then broadcast* reveal$(i, g_i(y))$.
(f) **Evaluate complaint resolutions:** *Simulate all honest parties broadcast* good. *Let I' be the set of corrupted parties that broadcast* good.
3. The simulator sends $I' \setminus R$ to the trusted party.

It is easy to see by inspection of the protocol, and by inspection of the simulation, and since the two are deterministic, that the view of the adversary in the real and ideal execution is *equal*. Our next goal is to show that the output of the honest parties is the same in the real and ideal executions.

In the optimistic case, where no reveal(i) messages are broadcasted by the dealer, and there are no complaint messages by any party, then in the real execution the output of all honest parties is $[n]$ and likewise, in the simulation the simulator sends I to the trusted party, which then sends $[n]$ to all parties.
We now consider the case where there are complaints and there is a vote. An honest party P_j broadcasts good if all the following conditions are met:

1. The polynomial $f_j(x)$ has degree at most $2t$, $g_j(y)$ has degree at most t and $f_j(\alpha_j) = g_j(\alpha_j)$. An honest party P_j therefore never broadcasts complaint(j).
2. While resolving complaints, the dealer never broadcasts reveal(j).
3. Each complaint(k) message is replied by the dealer with reveal$(k, \cdot)$ message.

4. All reveal$(i, g_i(y))$ messages broadcasted by the dealer satisfy $f_j(\alpha_i) = g_i(\alpha_j)$.
5. The dealer resolves all joint complaints.

It is easy to see that all those conditions are met in the case of an honest dealer. In particular: (1) is true by the assumption on the inputs; (2) An honest party never broadcasts complaint with the values it received from the dealer; As a result, according to our input assumption, the dealer never broadcasts reveal(j); (3) True by inspection of the code of the dealer; (4) When the dealer broadcasts a polynomial it always agrees with $f_j(x)$ initially given to P_j; (5) By the dealer's code specifications, it resolves all joint complaints.

Therefore, in the real execution all honest parties broadcast good, and some additional parties $I' \subseteq I$ that the adversary controls might also broadcast good. Then, all honest parties exclude from this set the parties in R, and output it. Since the view of the adversary is equal in the ideal execution, the same parties in the simulated ideal execution broadcast good. Let $I' \subseteq I$ be the set of corrupted parties that broadcast good. The simulator sends $I' \setminus R$ to the trusted party, which then defines K to be $([n] \setminus I) \cup (I' \setminus R)$, i.e., all honest parties and all corrupted parties that broadcast good, excluding those that are in R. Thus, the outputs of the honest parties in the real and ideal are identical.

Case 2: The Dealer is Corrupted. The proof of this case is deferred to the full version of this paper.

□

4.3 Evaluation with the Help of the Dealer

We show how the parties can recover the secret polynomial using the help of the dealer. Towards that end, we show how the parties can evaluate polynomial $g_\beta(y)$ for every $\beta \in E$, where E is a set of elements in $\mathbb{F}$. By taking E to be of cardinality $q + 1$, it is possible to completely recover S (see Claim 3.5). When we are only interested in the constant term of S, we take $E = \{0\}$ to obtain $g(y) = S(0, y)$ and then output $g(0)$. The polynomial can be recovered with the help of the dealer. Looking ahead, in Protocol 5.2: in the optimistic case we will use just $E = \{0\}$. In the pessimistic case, E will contain another indices of parties that raised a complaint against the dealer.

Functionality 4.4: F_{WEval}: Evaluation of a polynomial in Weak VSS

The functionality receives a set of indices $I \subseteq [n]$ and works as follows:

1. The functionality receives the sets (K, E) from all honest parties, where E is a set of elements in $\mathbb{F}$. Moreover, for every $k \in ([n] \setminus I) \cap K$ it receives the polynomial $f_k(x)$ from P_k. The dealer holds a polynomial S' of degree (q, t). When the dealer is honest, it is guaranteed that the indices of all honest parties are included in K.

2. The functionality reconstructs the unique (q, t) bivariate polynomial S that agrees with the shares of the honest parties. When the dealer is honest it always holds that $S' = S$. Note that if the shares do not define a unique polynomial, then no security is guaranteed[3].
3. If the dealer is honest ($1 \notin I$) then send $S(x, \alpha_i), S(\alpha_i, y)$ for every $i \in I$ together with the set E to the ideal adversary. Moreover, send the set of polynomials $\{S(\beta, y)\}_{\beta \in E}$ to all parties (and the ideal adversary).
4. If the dealer is corrupted ($1 \in I$) then:
   (a) Send the polynomial $S(x, y)$ to the ideal adversary together with $(K, \{S(\beta, y)\}_{\beta \in E})$.
   (b) Receive either ok or $\perp$ from the ideal adversary.
   (c) If ok, then send $\{S(\beta, y)\}_{\beta \in E}$ to all parties, and otherwise, send $\perp$ to all parties.

Protocol 4.5: Evaluation of a polynomial in Weak VSS

- **Input:** All parties hold a set $K \subseteq [n]$ and a set E of elements in $\mathbb{F}$. Each party P_k with $k \in K$ holds $f_k(x)$. The dealer holds also a polynomial $S(x, y)$.
- **Input guarantees:** When the dealer is honest, the indices of all honest parties are included in K.
- **The protocol:**
  1. The dealer broadcasts $\{S(\beta, y)\}_{\beta \in E}$.
  2. Each party P_k with $k \in K$ checks that the broadcasted polynomials are of degree at most t, and that $S(\beta, \alpha_k) = f_k(\beta)$ for every $\beta \in E$. If so, it broadcast good.
  3. **Output:** If $2t + 1$ parties in K broadcast good, then output the message broadcasted by the dealer. Otherwise, output $\perp$.

We prove the following theorem in the full version of this paper:

Theorem 4.6. *Let $t < n/3$. Protocol 4.5: is t-secure for the F_{WEval} functionality (Functionality 4.4:) in the presence of a static malicious adversary. The protocol incurs $O(n \cdot |E|)$ broadcast field elements.*

Remark 4.7. (On the optimistic case of Protocol 4.5:)**.** *In the optimistic case, we can implement Protocol 4.5: without any broadcast messages and with $O(n^2)$ field elements over the point-to-point channels. Specifically, in the optimistic case of the entire protocol (Protocol 5.2:) we have that $K = [n]$ and $E = \{0\}$. Each party P_k can send on the point-to-point channel to every other party P_j the message $f_k(0)$. Then, each party P_j uses the Reed Solomon decoding procedure to obtain the unique degree-t polynomial $g_\beta(y)$ satisfying $g_0(\alpha) = \gamma_k$, where γ_k is the point received from party P_k. Since there are $2t + 1$ honest parties in K, and since $S(0, y)$ is guaranteed to be a polynomial of degree-t, reconstruction works.*

[3] In that case, we simply give the adversary all inputs of all honest parties which makes any protocol vacuously secure as anything can be easily simulated, see Remark 3.2.

4.4 Strong Verifiable Secret Sharing

We provide the functionality for strong verifiable secret sharing, and prove its security. The main difference from [2] is that the output is the two univariate polynomials and not the projection to univariate sharing, and we therefore provide a proof for completeness in the full version of this paper.

Functionality 4.8: Strong Verifiable Secret Sharing

- **Input:** Receive $S(x, y)$ from the dealer P_1.
- **Output:** If $S(x, y)$ is of degree-t in both variables, then send $(S(x, \alpha_i), S(\alpha_i, y))$ to each party P_i. Otherwise, send $\perp$.

Theorem 4.9. *Let $t < n/3$. Then, Protocol 4.1: when using the* **VSS** *branch and with $q = t$ is t-secure for the f_{VSS} functionality (Functionality 4.8:) in the presence of a static malicious adversary. The protocol incurs $O(n^2)$ field elements in the point-to-point channels in the optimistic case and additional $O(n^2)$ field elements on the broadcast channel in the pessimistic case.*

Evaluation. Once a polynomial was shared using strong VSS, we use Functionality 4.4: to evaluate points on the polynomial with the help of the dealer. Note that in this case we have that $q = t$. Moreover, the parties use $K = [n]$. Note that K might now not be the same group of parties that broadcast `good` when the polynomial was shared, yet, since all honest parties hold shares $(f_j(x), g_j(y))$ it is safe to use $K = [n]$. Thus, to evaluate points E on a polynomial that was shared with VSS can be implemented using $O(n|E|)$ field messages broadcasted, as in Theorem 4.6.

4.5 Extending Univariate Sharing to Bivariate Sharing with a Dealer

Sometimes each party P_i holds a share $h(\alpha_i)$ of some univariate degree-t polynomial $h(x)$. The following functionality allows a dealer, who holds h, to distribute shares of a bivariate polynomial $S(x, y)$ satisfying $S(x, 0) = h(x)$. The protocol is very simple, demonstrating the advantage for working with bivariate sharing. This is the functionality $\tilde{F}_{extend}$ from [3]:

Functionality 4.10: F_{Extend}: Extending Univariate Sharing to Bivariate Sharing

The functionality receives the set of corrupted parties $I \subset [n]$ and works as follows:

- **Input:** The functionality receives the shares of the honest parties $\{u_j\}_{j \notin I}$. Let $h(x)$ be the unique degree-t polynomial determined by the points (α_j, u_j) for every $j \notin I$. If no such polynomial exists then no security is guaranteed.

– If the dealer is corrupted then send $h(x)$ to the ideal adversary.
– Receive $S(x, y)$ from the dealer. Check that $S(x, y)$ is of degree-t and that $S(x, 0) = h(x)$.
– If both conditions hold, then send to $S(x, \alpha_i), S(\alpha_i, y)$ to P_i for every i. Otherwise, send $\perp$ to everyone.

Protocol 4.11: Implementing F_{Extend} in the F_{VSS}-hybrid model

– **Input:** Each party holds u_j. The dealer holds $S(x, y)$ and $h(x)$.
– **The protocol:**
  1. The dealer uses F_{VSS} to distribute $S(x, y)$.
  2. Each party P_i receives $(f_i(x), g_i(y)) \stackrel{\text{def}}{=} (S(x, \alpha_i), S(\alpha_i, y))$. If instead $\perp$ was received, then output $\perp$ and halt.
  3. Each party P_i verifies that $g_i(0) = u_j$. If not, it broadcast $\mathsf{complaint}(i)$.
  4. **Output:** If there are more than t complaints, then output $\perp$. Otherwise, output $(f_i(x), g_i(y))$.

The communication cost of the protocol is the same as Protocol 4.1: for VSS. Note that in the optimistic case there are no complaints, and thus there are no additional broadcast messages. We provide a proof of the following theorem in the full version paper.

Theorem 4.12. *Let $t < n/3$. Then, Protocol 4.11: is t-secure for the F_{Extend} functionality (Functionality 4.10:) for in the presence of a static malicious adversary, in the F_{VSS}-hybrid model. The protocol incurs $O(n^2)$ point-to-point messages in the optimistic case and additional $O(n^2)$ broadcast messages in the pessimistic case.*

5 Multiplication with a Constant Number of VSSs and WSSs

We now turn to the multiplication protocol. The multiplication protocol is reduced to multiplication with a dealer, i.e., when one dealer holds two univariate polynomials $f^a(x), f^b(x)$, each party holds a share on those polynomials, and the dealer wishes to distribute a polynomial $C(x, y)$ of degree-t in both variables in which $C(0, 0) = f^a(0) \cdot f^b(0)$. We refer the reader to the full version of this paper to see how this functionality suffices to compute any multiplication gate (i.e., when there is no dealer). In Sect. 5.1 we show the functionality of this building block, in Sect. 5.2 we show the protocol that realizes it.

5.1 Functionality – Multiplication with a Dealer

Functionality 5.1: Functionality F_{VSS}^{mult} for sharing a product of shares

F_{VSS}^{mult} receives a set of indices $I \subseteq [n]$ and works as follows:

1. Receive a pair of points $(u_j, v_j) \in \mathbb{F}^2$ from P_j.
2. Compute the unique degree-t univariate polynomials $f^a(x)$ and $f^b(x)$ satisfying $f^a(\alpha_j) = u_j$ and $f^b(\alpha_j) = v_j$ for every $j \notin I$. (if no such polynomials f^a or f^b exist, then no security is guaranteed).
3. If the dealer P_1 is honest ($1 \notin I$), then:
   (a) choose a random degree-t bivariate polynomial $C(x, y)$ under the constraint that $C(0, 0) = f^a(0) \cdot f^b(0)$.
   (b) *Output for honest:* send $C(x, y)$ to P_1, and $C(x, \alpha_j)$, $C(\alpha_j, y)$ to P_j for every $j \notin I$.
   (c) *Output for adversary:* send $f^a(\alpha_i), f^b(\alpha_i)$, $C(x, \alpha_i), C(\alpha_i, y)$ to the (ideal) adversary, for every $i \in I$.
4. If the dealer P_1 is corrupted ($1 \in I$), then:
   (a) Send $f^a(x), f^b(x)$ to the (ideal) adversary.
   (b) Receive a bivariate polynomial C as input from the (ideal) adversary.
   (c) If either $\deg(C) > t$ or $C(0, 0) \neq f^a(0) \cdot f^b(0)$, then reset $C(x, y) = f^a(0) \cdot f^b(0)$; that is, $C(x, y)$ is a constant polynomial that equals $f^a(0) \cdot f^b(0)$ everywhere.
   (d) *Output for honest:* send $C(x, \alpha_j), C(\alpha_j, y)$ to P_j, for every $j \notin I$. (There is no more output for the adversary in this case.)

5.2 The Protocol

As mentioned in the introduction, in our protocol the dealer distributes $C(x, y)$ using verifiable secret sharing, and then also distributes a random $(2t, t)$-polynomial $D(x, y)$ under the constraint that $D(x, 0) = f^a(x) \cdot f^b(x) - C(x, 0)$ and that $D(0, 0) = 0$ by reconstructing the univariate polynomial $D(0, y)$.

To verify that $D(x, y)$ indeed satisfies this constraint, each party P_i verifies that $D(\alpha_i, 0) = f^a(\alpha_i) \cdot f^b(\alpha_i) - C(\alpha_i, 0)$ using the shares it received from P_1. If the verification fails, it broadcasts a complaint and all parties reconstruct the share of P_i. Since all polynomials are shared, it is possible to see whether the complaint is justified. Moreover, if for all honest parties the verification holds, then it must be that the two degree-$2t$ polynomials, $D(x, 0)$ and $f^a(x) \cdot f^b(x) - C(x, 0)$ are equal, as they agree on $2t + 1$ points.

Protocol 5.2: Computing F_{VSS}^{mult} in the $(F_{VSS}, F_{WSS}, F_{\mathsf{Extend}}, F_{\mathsf{WEval}})$ -hybrid model

- **Input:**
  1. The dealer P_1 holds two degree-t polynomials $f^a(x), f^b(x)$.
  2. Each party P_i holds two points $(u_i, v_i) = (f^a(\alpha_i), f^b(\alpha_i))$.
- **Common input:** A field $\mathbb{F}$ and distinct non-zero elements $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$.
- **The protocol:**
  1. **Sharing phase**:
     (a) P_1 chooses a degree-t bivariate polynomial $C(x, y)$ under the constraint that $C(0, 0) = f^a(0) \cdot f^b(0)$.

(b) P_1 chooses a random degree $(2t, t)$-bivariate polynomial $D(x, y)$ under the constraint that $D(x, 0) = f^a(x) \cdot f^b(x) - C(x, 0)$.
(c) Invoke F_{VSS} to share $C(x, y)$, and let $(f_i^c(x), g_i^c(y))$ be the output of P_i.
(d) Invoke F_{WSS} to share $D(x, y)$. Let $K \subseteq [n]$ be the output of F_{WSS}, such that each P_k for $k \in K$ also receives $(f_k^d(x), g_k^d(y))$, and each party P_j for $j \notin K$ receives $g_j^d(y)$.
(e) If $\perp$ was received in any of the above, then proceed to Step 5b.

2. **Verifying that $D(x, 0) = f^a(x) \cdot f^b(x) - C(x, 0)$:**
(a) Each party P_i verifies that $g_i^d(0) = u_i \cdot v_i - g_i^c(0)$. If no, broadcast $\mathsf{complaint}(i)$.
(b) If no party broadcasts a complaint, then proceed to Step 4.

3. **Complaint resolution (only in pessimistic case):**
(a) Let R be the set of all parties broadcast $\mathsf{complaint}(i)$, and let $E = \{\alpha_i\}_{i \in R}$.
(b) P_1 chooses two random degree-t bivariate polynomials, A, B under the constraint that $A(x, 0) = f^a(x)$ and $B(x, 0) = f^b(x)$. The parties run the F_{Extend} functionality twice, where each party P_i inputs u_i and the dealer inputs $A(x, y)$ in the first execution, and each party P_i inputs v_i and the dealer inputs $B(x, y)$ in the second execution.
(c) The parties call to F_{WEval} where each party P_i inputs $(f_i^a(x), g_i^a(y), E, [n])$. Let $(f_j^a(x), g_j^a(y))$ be the result for every $j \in R$. Likewise, reconstruct $(f_j^b(x), g_j^b(y))$, $(f_j^c(x), g_j^c(y))$. If F_{WEval} returned $\perp$ in any one of the invocations, then proceed to Step 5b.
(d) The parties call to F_{WEval} where all parties input K, E and each party P_k for $k \in K$ inputs also $(f_k^d(x), g_k^d(y))$. The output of F_{WEval} is $g_i^d(y)$ for every $i \in R$. If F_{WEval} returned $\perp$, then proceed to Step 5b.
(e) For every $j \notin K$, all parties verify that $g_j^d(0) = g_j^a(0) \cdot g_j^b(0) - g_j^c(0)$. If not, then proceed to Step 5b.

4. **Verifying that $D(0, 0) = 0$:**
(a) The parties call to F_{WEval} where all parties input $K, \{0\}$ and each party P_k for $k \in K$ inputs also $(f_k^d(x), g_k^d(y))$. The output of F_{WEval} is $g_0^d(y) = D(0, y)$ to all parties. If F_{WEval} returned $\perp$, then proceed to Step 5b.
(b) Verify that $g_0^d(0) = 0$. If not, proceed to Step 5b.

5. **Finalization:**
(a) *Accept:* If the dealer was not rejected, then each party P_i outputs $(f_i^c(x), g_i^c(y))$.
(b) *Reject:* If the dealer is rejected, then each party P_i sends to P_j its points u_i, v_i. The parties reconstruct the polynomials $f^a(x), f^b(x)$ using Reed-Solomon decoding, and define their output shares $f_i^c(x) = g_i^c(y) = f^a(0) \cdot f^b(0)$.

The communication cost of the entire sharing phase (Step 1) is equal to the cost of a VSS/WSS, since it calls to F_{VSS} for C and F_{WSS} for D. Thus, it completes with communication overhead of $O(n^2)$ over the point-to-point channels in the optimistic case and additional overhead of $O(n^2)$ over the broadcast channel in the pessimistic case.

In Step 2, the optimistic case we have no complaints, no evaluation is required, therefore, there is no communication cost. On the other hand, the size of E may be $O(n)$ in the worst case, which leads to $O(n \cdot |E|) = O(n^2)$ broadcasted field elements.

Finally, in Step 4 there is a reconstruction of $D(0, y)$. In the optimistic case, this can be done using $O(n^2)$ words over the point-to-point channels and no broadcast (see Remark 4.5:). In the pessimistic case, this requires a broadcast of $O(n)$ field elements.

Overall, the optimistic case incurs a communication overhead of $O(n^2)$ over the point-to-point channels, and the pessimistic case incurs an additional communication overhead of $O(n^2)$ over the broadcast channel.

Theorem 5.3. *Let $t < n/3$. Then, Protocol 5.2: is t-secure for the F_{VSS}^{mult} functionality in the presence of a static malicious adversary, in the $(F_{VSS}, F_{WSS}, F_{\mathsf{Extend}}, F_{\mathsf{WEval}})$-hybrid model. The optimistic case incurs $O(n^2)$ point-to-point field elements, and the pessimistic case incurs additional $O(n^2)$ broadcast messages of field elements.*

The proof is provided in the full version of this paper.

By combining Theorems 4.9, 4.3, 4.12 and 4.6 with Theorem 5.3 we obtain the following Corollary:

Corollary 5.4. *Let $t < n/3$. Then, there exists a protocol that is t-secure for the F_{VSS}^{mult} functionality in the presence of a static malicious adversary in the plain model.*

6 Extension: Arbitrary Gates with Multiplicative Depth-1

We show how to extend the protocol in Sect. 5 to allow the dealer distributing any shares $b_1, \ldots, b_L$ given input shares $a_1, \ldots, a_M$ such that $(b_1, \ldots, b_L) = G(a_1, \ldots, a_M)$ where G is some circuit of multiplicative depth 1. Section 5 is a special case where $G(a_1, a_2) = a_1 \cdot a_2$.

Functionality 6.1: Functionality F^G_{VSS} for sharing a result of an evaluation of G

F^G_{VSS} receives a set of indices $I \subseteq [n]$ and works as follows, where P_1 is the dealer:

1. Receive a sequence of points $u_{j,1}, \ldots, u_{j,M} \in \mathbb{F}^M$ from P_j.
2. Compute the unique degree-t univariate polynomials $f^{a_1}(x), \ldots, f^{a_M}(x)$ satisfying $f^{a_m}(\alpha_j) = u_{j,m}$ for every $j \notin I$ and $m \in [M]$ (if no such polynomials $f^{a_m}(x)$ exist, then no security is guaranteed).
3. Let $(a_1, \ldots, a_m) \stackrel{\text{def}}{=} (f^{a_1}(0), \ldots, f^{a_m}(0))$. Evaluate $(b_1, \ldots, b_L) = G(a_1, \ldots, a_m)$.
4. If the dealer P_1 is honest ($1 \notin I$) then:
   (a) For every $\ell \in [L]$, choose a random degree-t bivariate polynomial C_ℓ under the constraint that $C_\ell(0,0) = b_\ell$.
   (b) *Output for honest:* send C_ℓ to P_1 and $(C_\ell(x, \alpha_j), C_\ell(\alpha_j, y))$ to P_j for every $j \notin I$ and $\ell \in [L]$.
   (c) *Output for adversary:* send to the (ideal) adversary: (1) $f^{a_1}(\alpha_i), \ldots, f^{a_m}(\alpha_i)$ for every $i \in I$; (2) $(C_\ell(x, \alpha_i), C_\ell(\alpha_i, y))$ for every $i \in I$.
5. If the dealer P_1 is corrupted ($1 \in I$), then:
   (a) Send $f^a(x), f^b(x)$ to the (ideal) adversary.
   (b) Receive bivariate polynomials $C_1, \ldots, C_L$ as input from the (ideal) adversary.
   (c) If either $\deg(C_\ell) > t$ or $C_\ell(0,0) \neq b_\ell$ for some $\ell \in [L]$, then reset $C_\ell(x,y) = b_\ell$ for every $\ell \in [L]$.
   (d) *Output for honest:* send $C_\ell(x, \alpha_j), C_\ell(\alpha_j, y)$ to P_j, for every $j \notin I$ and $\ell \in [L]$. (There is no more output for the adversary in this case.)

The protocol is similar to Protocol 5.2:. Given such a circuit G with L outputs, we let $G_1, \ldots, G_L$ be the circuits that define each outputs. That is, for $(b_1, \ldots, b_L) = G(a_1, \ldots, a_m)$ we let $b_\ell = G_\ell(a_1, \ldots, a_m)$ for every $\ell \in [L]$. In the protocol, the dealer distributes polynomials $C_1(x,y), \ldots, C_L(x,y)$ using VSS that are supposed to hide $b_1, \ldots, b_L$. Then, it defines L bivariate polynomials of degree$(2t,t)$, $D_1, \ldots, D_L$ such that for every $\ell \in [L]$ it holds that $D_\ell(x,0) = G(f^{a_1}(x), \ldots, f^{a_m}(x)) - C_\ell(x,0)$. The dealer distributes them using F_{WSS}. The parties then check from the shares they received that each one of the polynomials $C_1, \ldots, C_L$ is correct, and that $D_\ell(0,0)$ for every $\ell \in [L]$. When a party P_i complains the parties open the shares of P_i and publicly verify the complaint.

Protocol 6.2: Computing F_{VSS}^{G} in the $(F_{VSS}, F_{WSS}, F_{\mathsf{Extend}}, F_{\mathsf{WEval}})$-hybrid model

- **Input:**
  1. The dealer P_1 holds M degree-t polynomials $\{f^{a_m}(x)\}_{m\in[M]}$.
  2. Each party P_i holds a point $u_{i,m}$ for every $m \in [M]$ (where $u_{i,m} = f^{a_m}(\alpha_i)$).
- **Common input:** A field $\mathbb{F}$ and distinct non-zero elements $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$.
- **The protocol:**
  1. **Sharing phase**:
     (a) P_1 computes $(b_1, \ldots, b_L) = G(f^{a_1}(0), \ldots, f^{a_M}(0))$.
     (b) For every $\ell \in [L]$, P_1 chooses a random degree-t bivariate polynomials, $C_\ell(x, y)$ such that $C_\ell(0, 0) = b_\ell$.
     (c) For every $\ell \in [L]$, P_1 chooses a random degree $(2t, t)$-bivariate polynomial $D_\ell(x, y)$ under the constraint that $D_\ell(x, 0) = G_\ell(f^{a_1}(x), \ldots, f^{a_M}(x)) - C_\ell(x, 0)$.
     (d) For every $\ell \in [L]$, invoke F_{VSS} to share $C_\ell(x, y)$ and let $(f_i^{b_\ell}(x), g_i^{b_\ell}(y))$ be the resulting share of P_i.
     (e) For every $\ell \in [L]$, invoke F_{WSS} to share $D_\ell(x, y)$. Let $K_\ell \subseteq [n]$ be the output of F_{WSS}, such that each P_j for $k \in K_\ell$ also receives $(f_k^{d_\ell}(x), g_k^{d_\ell}(y))$, and each party P_j for $j \notin K_\ell$ receives $g_j^{d_\ell}(y)$.
     (f) If $\perp$ was received in any of the above F_{VSS} or F_{WSS} invocations, then proceed to Step 5b.
  2. **Verifying that $D_\ell(x, 0) = G_\ell((f^{a_1}(x), \ldots, f^{a_M}(x))) - C_\ell(x, 0)$ for all $\ell \in [L]$**:[4]
     (a) For every $\ell \in [L]$, each party P_i verifies that $g_i^{d_\ell}(0) = G_\ell(u_{i,1}, \ldots, u_{i,M}) - g_i^{c_\ell}(0)$. If not, broadcast $\mathsf{complaint}(i)$
     (b) If no party broadcast a complaint, proceed to Step 4.
  3. **Complaint resolution (only in pessimistic case):**
     (a) Let R be the set of all parties broadcast $\mathsf{complaint}(i)$, and let $E = \{\alpha_i\}_{i\in R}$.
     (b) For every $m \in [M]$, the dealer chooses a random bivariate polynomial of degree-t polynomial A_m such that $A_m(x, 0) = f^{a_m}(x)$. The parties run F_{Extend} where each party P_i inputs $u_{i,m}$ and P_1 inputs A_m. Let $(f_i^{a_m}(x), g_i^{a_m}(y))$ be the output share of P_i.
     (c) For every $m \in [M]$, the parties call to F_{WEval} where each party P_i inputs $(f_i^{a_m}(x), g_i^{a_m}(y), E, [n])$ and the dealer inputs A_m. Let $(f_j^{a_m}(x), g_j^{a_m}(y))$ be the result for every $j \in R$. Likewise, reconstruct $(f_j^{b_\ell}(x), g_j^{b_\ell}(y))$ for every $\ell \in [L]$. If F_{WEval} returned $\perp$ in any of those invocations, then proceed to Step 5b.

[4] We abuse notation and write $G_\ell((f^{a_1}(x), \ldots, f^{a_M}(x)))$ to denote a univariate polynomial in the variable x. Specifically, we take all polynomials $f^{a_1}(x), \ldots, f^{a_M}(x)$ and perform the same arithmetic operations as in G_ℓ on those input polynomials to receive a univariate polynomial in x.

(d) For every $\ell \in [L]$, the parties call to F_{WEval} where all parties input K_ℓ, E and each party P_k for $k \in K_\ell$ inputs also $(f_k^{d_\ell}(x), g_k^{d_\ell}(y))$. The output of F_{WEval} is $g_j^{d_\ell}(y)$ for every $j \in R$. If F_{WEval} returned $\perp$, then proceed to Step 5b.

(e) For every $j \in R$, $\ell \in [L]$, all parties verify that $g_j^{d_\ell}(0) = G(g_j^{a_1}(0), \ldots, g_j^{a_M}(0)) - g_j^{c_\ell}(0)$. If not, then proceed to Step 5b.

4. **Verifying that $D_\ell(0,0) = 0$ for all $\ell \in [L]$:**
   (a) For every $\ell \in [L]$, the parties call to F_{WEval} where all parties input $K_\ell, \{0\}$ and each party P_j for $j \in K_\ell$ inputs also $(f_j^{d_\ell}(x), g_j^{d_\ell}(y))$. The output of F_{WEval} is $g_0^{d_\ell}(y) = D_\ell(0, y)$ to all parties. If F_{WEval} returned $\perp$, then proceed to Step 5b.
   (b) Verify that $g_0^{d_\ell}(0) = 0$. If not, proceed to Step 5b.
5. **Finalization:**
   (a) *Accept:* If the dealer was not rejected, then each party P_i outputs $(f_i^{c_\ell}(x), g_i^{c_\ell}(y))$ for every $\ell \in [L]$.
   (b) *Reject:* If the dealer is rejected, then each party P_i sends to P_j its points $u_{i,m}$ for every $m \in [M]$. The parties reconstruct the polynomials $f^{a_m}(x)$ using Reed-Solomon decoding, and output $G(f^{a_1}(0), \ldots, f^{a_M}(0))$.

Theorem 6.3. *Let $t < n/3$. Then, Protocol 6.2: is t-secure for the F_{VSS}^{G} functionality in the presence of a static malicious adversary, in the $(F_{VSS}, F_{WSS}, F_{\mathsf{Extend}}, F_{\mathsf{WEval}})$-hybrid model. The communication complexity of the protocol is just $O(L)$ VSSs in the optimistic case. In the pessimistic case, it corresponds to $O(L + M)$ VSSs.*

The proof is provided in the full version of this paper.

Acknowledgments. Gilad Asharov would like to thank Ilan Komargodski and Ariel Nof for helpful discussions.

References

1. Abraham, I., Pinkas, B., Yanai, A.: Blinder: MPC based scalable and robust anonymous committed broadcast (2020)
2. Asharov, G., Lindell, Y.: A full proof of the BGW protocol for perfectly secure multiparty computation. J. Cryptol. **30**(1), 58–151 (2017)
3. Asharov, G., Lindell, Y., Rabin, T.: Perfectly-secure multiplication for any t¡n/3. In: Rogaway, P. (ed.) Advances in Cryptology - CRYPTO 2011–31st Annual Cryptology Conference, Santa Barbara, CA, USA, 14–18, August 2011. Proceedings. Lecture Notes in Computer Science, vol. 6841, pp. 240–258. Springer, Berlin (2011). https://doi.org/10.1007/978-3-642-22792-9_14
4. Barak, A., Escudero, D., Dalskov, A.P.K., Keller, M.: Secure evaluation of quantized neural networks. IACR Cryptol. ePrint Arch. **2019**, 131 (2019)

5. Beaver, D.: Efficient multiparty protocols using circuit randomization. In: CRYPTO, pp. 420–432 (1991)
6. Beerliová-Trubíniová, Z., Hirt, M.: Perfectly-Secure MPC with Linear Communication Complexity. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 213–230. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78524-8_13
7. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: Simon, J. (ed.) STOC, pp. 1–10. ACM (1988)
8. Berman, P., Garay, J.A., Perry, K.J.: Bit optimal distributed consensus, In: Baeza-Yates, R., Manber, U. (eds) Computer Science. Springer, Boston (1992). https://doi.org/10.1007/978-1-4615-3422-8_27
9. Anirudh, C., Choudhury, A., Patra, A.: A survey on perfectly-secure verifiable secret-sharing. IACR Cryptol. ePrint Arch. **2021**, 445 (2021). https://eprint.iacr.org/2021/445
10. Canetti, R.: Security and composition of multiparty cryptographic protocols. J. Cryptol. **13**(1), 143–202 (2000)
11. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: FOCS, pp. 136–145. IEEE Computer Society (2001)
12. Canetti, R., Damgård, I., Dziembowski, S., Ishai, Y., Malkin, T.: Adaptive versus non-adaptive security of multi-party protocols. J. Cryptol. **17**(3), 153–207 (2004)
13. Chen, H., Kim, M., Razenshteyn, I.P., Rotaru, D., Song, Y., Wagh, S.: Maliciously secure matrix multiplication with applications to private deep learning. IACR Cryptol. ePrint Arch. **2020**, 451 (2020)
14. Chida, K., et al.: Fast large-scale honest-majority MPC for malicious adversaries. In: CRYPTO, pp. 34–64 (2018)
15. Chor, B., Goldwasser, S., Micali, S., Awerbuch, B.: Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In: FOCS, pp. 383–395. IEEE Computer Society (1985)
16. Coan, B.A., Welch, J.L.: Modular construction of a byzantine agreement protocol with optimal message bit complexity. Inf. Comput. **97**(1), 61–85 (1992)
17. Cohen, R., Coretti, S., Garay, J.A., Zikas, V.: Probabilistic termination and composability of cryptographic protocols. J. Cryptol. **32**(3), 690–741 (2019)
18. Cramer, R., Damgård, I., Maurer, U.M.: General secure multi-party computation from any linear secret-sharing scheme. In: EUROCRYPT, pp. 316–334 (2000)
19. Damgård, I., Nielsen, J.B.: Scalable and Unconditionally Secure Multiparty Computation. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 572–590. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74143-5_32
20. Damgård, I., Nielsen, J.B., Polychroniadou, A., Raskin, M.: On the Communication Required for Unconditionally Secure Multiplication. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9815, pp. 459–488. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53008-5_16
21. Damgård, I., Schwartzbach, N.I.: Communication lower bounds for perfect maliciously secure MPC. IACR Cryptol. ePrint Arch. **2020**, 251 (2020). https://eprint.iacr.org/2020/251
22. Dodis, Y., Micali, S.: Parallel Reducibility for Information-Theoretically Secure Computation. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 74–92. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44598-6_5
23. Feldman, P.: Optimal algorithms for byzantine agreement (1988)
24. Feldman, P., Micali, S.: An optimal probabilistic protocol for synchronous byzantine agreement. SIAM J. Comput. **26**(4), 873–933 (1997)

25. Gennaro, R., Rabin, M.O., Rabin, T.: Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In: Coan, B.A., Afek, Y. (eds.) PODC, pp. 101–111. ACM (1998)
26. Goldreich, O.: The Foundations of Cryptography. Basic Applications, vol. 2. Cambridge University Press, Cambridge (2004)
27. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: Aho, A.V. (ed.) STOC, pp. 218–229. ACM (1987)
28. Goyal, V., Liu, Y., Song, Y.: Communication-Efficient Unconditional MPC with Guaranteed Output Delivery. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11693, pp. 85–114. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26951-7_4
29. Hirt, M., Maurer, U., Przydatek, B.: Efficient Secure Multi-party Computation. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 143–161. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44448-3_12
30. Hirt, M., Nielsen, J.B.: Robust Multiparty Computation with Linear Communication Complexity. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 463–482. Springer, Heidelberg (2006). https://doi.org/10.1007/11818175_28
31. Katz, J., Koo, C.: On expected constant-round protocols for byzantine agreement. J. Comput. Syst. Sci. **75**(2), 91–112 (2009)
32. Kushilevitz, E., Lindell, Y., Rabin, T.: Information-theoretically secure protocols and security under composition. SIAM J. Comput. **39**(5), 2090–2112 (2010)
33. Liu, J., Juuti, M., Lu, Y., Asokan, N.: Oblivious neural network predictions via minionn transformations. In: ACM CCS, pp. 619–631 (2017)
34. Mohassel, P., Rindal, P.: Aby3: a mixed protocol framework for machine learning. In: CCS, pp. 35–52 (2018)
35. Mohassel, P., Zhang, Y.: Secureml: a system for scalable privacy-preserving machine learning. In: SP, pp. 19–38 (2017)
36. Rabin, T., Ben-Or, M.: Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In: Johnson, D.S. (ed.) Proceedings of the 21st Annual ACM Symposium on Theory of Computing, 14–17, May 1989, Seattle, Washigton, USA, pp. 73–85. ACM (1989)
37. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979). https://doi.org/10.1145/359168.359176
38. Verma, A., Qassim, H., Feinzimer, D.: Residual squeeze CNDS deep learning CNN model for very large scale places image recognition. In: UEMCON, pp. 463–469 (2017)
39. Wagh, S., Gupta, D., Chandran, N.: Securenn: 3-party secure computation for neural network training. Proc. Priv. Enhancing Technol. **2019**(3), 26–49 (2019)
40. Yao, A.C.: How to generate and exchange secrets (extended abstract). In: FOCS, pp. 162–167. IEEE Computer Society (1986)