The Price of Active Security in Cryptographic Protocols

Carmit Hazay
Bar-Ilan University

Muthuramakrishnan Venkitasubramaniam
University of Rochester

Mor Weiss
IDC Herzliya

Abstract

We construct the first actively-secure Multi-Party Computation (MPC) protocols with an *arbitrary* number of parties in the dishonest majority setting, for an *arbitrary* field $\mathbb{F}$ with *constant communication overhead* over the "passive-GMW" protocol (Goldreich, Micali and Wigderson, STOC '87). Our protocols rely on passive implementations of Oblivious Transfer (OT) in the boolean setting and Oblivious Linear function Evaluation (OLE) in the arithmetic setting. Previously, such protocols were only known over sufficiently large fields (Genkin et al. STOC '14) or a constant number of parties (Ishai et al. CRYPTO '08).

Conceptually, our protocols are obtained via a new compiler from a passively-secure protocol for a distributed multiplication functionality $\mathcal{F}_{\text{MULT}}$, to an actively-secure protocol for general functionalities. Roughly, $\mathcal{F}_{\text{MULT}}$ is parameterized by a linear-secret sharing scheme $\mathcal{S}$, where it takes $\mathcal{S}$-shares of two secrets and returns $\mathcal{S}$-shares of their product.

We show that our compilation is concretely efficient for sufficiently large fields, resulting in an overhead of 2 when securely computing natural circuits. Our compiler has two additional benefits: (1) it can rely on *any* passive implementation of $\mathcal{F}_{\text{MULT}}$, which, besides the standard implementation based on OT (for boolean) and OLE (for arithmetic) allows us to rely on implementations based on threshold cryptosystems (Cramer et al. Eurocrypt '01); and (2) it can rely on weaker-than-passive (i.e., imperfect/leaky) implementations, which in some parameter regimes yield actively-secure protocols with overhead less than 2.

Instantiating this compiler with an "honest-majority" implementation of $\mathcal{F}_{\text{MULT}}$, we obtain the first honest-majority protocol with optimal corruption threshold for boolean circuits with constant communication overhead over the best passive protocol (Damgård and Nielsen, CRYPTO '07).

Contents

1 Introduction

The problem of Secure Multi-party Computation (MPC) considers a set of parties with private inputs that wish to jointly compute a function of their inputs while simultaneously preserving *correctness* of the outputs, and guaranteeing *privacy* of the inputs, i.e., nothing but the output is revealed. These properties are required to hold in the presence of an adversary that controls a subset of the parties, and attacks the protocol in an attempt to breach its security, e.g., learn more than it should about the honest parties' inputs.

Secure computation was first defined and explored in the mid 80s [Yao86, CCD87, GMW87, BGW88], and has been the focus of intensive study ever since. In the first two decades, research focused mainly on theoretical foundations, establishing the boundaries of feasibility and complexity. More recently, the focus has shifted to making MPC efficient and reducing its overhead over insecure implementations, both in terms of asymptotic and concrete efficiency (See [LP07, IPS08, IPS09, DPSZ12, WRK17a, WRK17b, HSS17, GLS19], and references therein.)

A basic classification in MPC considers protocols in which security is guaranteed with: (1) an *honest majority*, namely when the adversary corrupts a minority of the participants; or (2) a *dishonest majority*, where the adversary can corrupt arbitrarily many parties. The second category, which captures two-party protocols as a special case, has the advantage that any single party need not trust anyone but itself. Designing protocols from the second category is significantly more challenging, and they can only guarantee computational security, i.e., against computationally-bounded adversaries. On the other hand, the first category admits conceptually simpler solutions with statistical (or even perfect) security, namely against computationally-unbounded adversaries.

An orthogonal classification of MPC protocols is based on the adversarial behavior: (1) *passive* adversaries that follow the protocol's instructions but try to learn more than the prescribed information; and (2) *active* adversaries that may arbitrarily deviate from the protocol. A common paradigm in MPC is to design first a passively-secure protocol, and then compile it into an actively-secure one.

Hence, an important efficiency metric for MPC protocols is the *overhead* of actively-secure protocols over (the best) passively-secure ones. A primary goal in MPC today is to reduce this overhead, and specifically to design actively-secure protocols with *constant overhead* over *state-of-the-art passively-secure* protocols. That is, to design protocols whose communication and computation overheads grow only by a constant factor compared to the underlying passive protocols.

This work focuses on the most challenging MPC setting: *active security with an arbitrary number of parties*. Ideally, we would like the price of achieving active security to be minimal compared to the passively-secure counterparts.

The past decade has seen tremendous progress in the design of concretely-efficient actively-secure protocols for arbitrary functions, specified as boolean or arithmetic circuits, in either the two-party [ST04, LP07, KS08, NO09, LP12, NNOB12, SS13, HKK+14, ZRE15, RR16, LR15, GLNP15, WMK17, WRK17a, HIV17], or the multi-party setting with an arbitrary number of parties [IPS08, DPSZ12, DKL+13, LPSY15, WRK17b, HSS17, KPR18]. See Section 1.2 below for more details.

Despite this impressive progress there still remains important gaps between what is achievable with passive and active security. Indeed, no protocols for boolean computations with an arbitrary number of parties and constant communication overhead (even asymptotically) are known, both in the honest and the dishonest majority settings. For arithmetic computations with an arbitrary number of parties and over sufficiently large fields, the best concrete overhead (of 12x [GIP+14]) still seems large. In the honest majority setting an overhead of 2 has been achieved only for large fields [CGH+18].

Given this state of affairs, in this work we set out to answer the following fundamental open problem:

Can actively-secure protocols over an arbitrary field match the complexity of passively-secure protocols, in the dishonest and honest majority settings, with an arbitrary number of parties?

We resolve this open problem in terms of communication complexity in the affirmative, designing an asymptotically-efficient *actively-secure* protocol for *boolean* circuits (as well as arithmetic circuits over any field) in both the *honest majority* and *dishonest majority* settings, with *constant communication overhead* over the (best known) passively-secure counterparts.

We note that constant-overhead protocols are known based on general zero-knowledge proofs [GMW87], but these solutions rely on "heavy" tools and are *practically inefficient*. Instead, we focus on designing protocols that make *black-box* use of simpler (and lightweight) primitives such as One-Way Functions (OWFs), and parallel Oblivious-Transfer (OT) or parallel Oblivious Linear function Evaluation (OLE) in the boolean and arithmetic settings (resp.). Relying on OTs/OLEs is, in a sense, necessary since these are special cases of secure computation in their respective settings. Moreover, since our protocols make black-box use of these primitives, they will benefit from future improvements in the costs of OT/OLE implementations, which have been steadily decreasing.

Moreover, to frame a clean theoretical question, we focus on designing modular protocols in which the (relatively) computationally-expensive "cryptographic" component is separated from the rest of the protocol, and abstracted as an ideal functionality. Specifically, the "cryptographic" abstraction we consider in this work is a (constant-round) parallel protocol for computing distributed multiplication. Relying on a general multiplication functionality instead of OT/OLE allows us to simultaneously capture many settings of interest (booealn/arithmetic computations, two/multi-party, honest/dishonest majority) in a unified way. More specifically, we abstract distributed multiplication as an $\mathcal{F}_{\text{MULT}}$ functionality that is parameterized by a secret sharing scheme $\mathcal{S}$ over some field $\mathbb{F}$, takes $\mathcal{S}$-shares of two secrets, and produces $\mathcal{S}$-shares of their product. It is easy to see that one can use a general reduction from OT (resp. OLE) to a random instance $\mathcal{F}_{\text{RMULT}}$ of $\mathcal{F}_{\text{MULT}}$ (which generates additive shares of random multiplication triples in the sense of Beaver's triples [Bea91]) for boolean (resp. arithmetic) computations. In the multi-party setting, one can also realize $\mathcal{F}_{\text{MULT}}$ using more general protocols based on threshold additively-homomorphic encryption schemes [CDN01].

1.1 Our Results – A New Framework

In this work we answer the open problem stated above with respect to communication complexity on the affirmative, introducing the first actively-secure protocol with constant communication overhead over passive GMW [GMW87], for any number of parties and over any field, in the $\mathcal{F}_{\text{MULT}}$-hybrid model.

We obtain our result via a new compiler which transforms a passively-secure protocol for $\mathcal{F}_{\text{MULT}}$ into an actively-secure protocol for arbitrary functionalities, while inheriting the setting of the $\mathcal{F}_{\text{MULT}}$ protocol (i.e., boolean/arithmetic, two/multi-party, and honest/dishonest majority). Specifically, the compiler is described in the $\mathcal{F}_{\text{MULT}}$-hybrid model, and using different instantiations of $\mathcal{F}_{\text{MULT}}$ we obtain actively-secure protocols with constant communication overhead in the boolean and arithmetic, two-party and multi-party, and honest and dishonest majority settings. Moreover, for large fields and "typical" circuits, the overhead of our protocols is 2.

Working in the $\mathcal{F}_{\text{MULT}}$-hybrid model allows us to preserve a clear separation between the "passive" (alternatively, cryptographic) components of our protocol, namely the implementation of $\mathcal{F}_{\text{MULT}}$, which relies on cryptographic assumptions; and the "correctness-enforcing" (alternatively, non-cryptographic) components which involve tools from the literature of honest-majority protocols, employing consistency tests to enforce honest behavior. Besides scalability (and reduced communication complexity), we believe our ap-

Corruption Threshold	Number of Parties	Field Size	Hybrid Model	Asymptotic Overhead	Best Passive	Theorem Number
$t < n$	Arbitrary	$O(1)$	OT	Constant	[GMW87]	Theorem 4
$t < n$	Arbitrary	Arbitrary	OLE	Constant*	[GMW87]	Theorem 6
$t < n/2$	Arbitrary	Arbitrary	—	Constant	[BGW88]	Theorem 7

Table 1: Asymptotic communication overheads of our results in both the dishonest and honest majority settings for boolean and arithmetic computations. The "best passive" column refers to the passively-secure protocol over which the overhead is computed. The "theorem number" column specifies the theorem which implies the corresponding result.
* Concretely, this constant is 2 for moderately wide circuits.

proach is simple and modular.

Our compiler improves over the state-of-the-art in several settings; see Table 1 for a summary, and Section 6 for a detailed discussion.

New protocols in the dishonest majority setting. Our complier exhibits the most substantial improvements in the dishonest majority setting, yielding the *first* constant-overhead actively-secure protocol with a dishonest majority over an arbitrary number of parties for boolean circuits. The concrete constants of our compiler are yet unknown since they depend on the concrete efficiency of Algebraic Geometric (AG) secret sharing schemes over constant-size fields [CC06]. The result is summarized in the following informal theorem; see Theorem 4 for the formal statement.

Theorem 1 (Informal)**.** *Any m-party function f over a constant-size field (resp., arbitrary size field) can be securely realized by an $O(d)$-round protocol in the OT-hybrid (resp., OLE-hybrid) model against an active adversary corrupting an arbitrary number of parties with total communication $O(m^2 \cdot |\mathrm{C}|) + \mathsf{poly}(\kappa, d, m)$ field elements, where C is a depth-d circuit for f, and κ is a computational security parameter.*

For arithmetic computations, we can concretely analyze the constants introduced by our compiler, and show that they can be as small as 2 for moderately wide circuits and sufficiently large fields. This improves over [GIP+14] in two aspects. First, their work requires at least 12 invocations of an active implementation of $\mathcal{F}_{\text{MULT}}$, while ours requires only two invocation of a passive implementation. This allows us to instantiate our compiler with passive implementations of $\mathcal{F}_{\text{MULT}}$ based on threshold additively homomorphic encryption schemes [CDN01, BDOZ11]. Second, their result is only useful for computations over sufficiently large fields (where the statistical error $O\left(|\mathrm{C}| \,/\, |\mathbb{F}|\right)$ is small), whereas our result applies to fields of arbitrary size.

Building on the recent result of Hazay et al. [HIMV19], we can extend our compiler to rely on a weaker-than-passive (e.g., imperfect or leaky) implementation of $\mathcal{F}_{\text{MULT}}$. Consequently $\mathcal{F}_{\text{MULT}}$ can be instantiated with lattice-based protocols with "aggressive" (weaker) parameters, yielding actively-secure compiled protocols whose communication cost almost matches that of the best passive protocols, namely, essentially achieving active security at the cost of passive!

Additionally, we achieve an interesting corollary in the constant-round regime for boolean computations. By viewing distributed garbling [BMR90] as an arithmetic functionality over $\mathbb{GF}(2^\kappa)$, we can instantiate our compiler for arithmetic circuits to achieve constant-overhead over that passive variant of [BMR90] instantiated with $\mathcal{F}_{\text{MULT}}$ over $\mathbb{GF}(2^\kappa)$. See section 7.2 for details.

We believe our protocols can also be made to tolerate adaptive corruptions by instantiating the underlying cryptographic primitives (namely, $\mathcal{F}_{\text{MULT}}$ and $\mathcal{F}_{\text{COM}}$) with their adaptively-secure counterparts, and leave this to future work.

New protocols in the honest majority setting. In the honest majority regime for $t < n/2$, our compiler gives an actively-secure protocol for boolean circuits with constant overhead over a variant of passive-BGW [BGW88] that is instantiated using AG secret sharing schemes. This result improves over the recent protocol by Chida et al. [CGH+18], which only achieves constant overhead for large fields (introducing an extra statistical security parameter s for small fields with an overhead of $s/\log_2(|\mathbb{F}|)$), and over Ishai et al. [IKP+16] who achieve constant-overhead for arbitrary fields, but only for few parties. We note that [DI06] achieves constant-rate secure protocols, but only for suboptimal corruption thresholds. For boolean computation with an arbitrary number of parties and optimal threshold, the best protocols are due to Genkin et al. [GIW16] and achieve a $\mathsf{poly}\log(|\mathrm{C}|, s)$ overhead, where $|\mathrm{C}|$ is the circuit size.

1.2 Related Work

We give a brief overview of recent efficient protocols, summarized in Table 2.

The state-of-the-art: boolean multi-party setting. For boolean circuits, secure protocol against a dishonest majority with an (asymptotic) constant overhead over passively-secure protocols, was achieved for constant number of parties by Ishai, Prabhakaran and Sahai [IPS08] (referred to as the "IPS-compiler"). Their protocol operates in the OT-hybrid model, achieving constant overhead over passive-GMW. It also achieves constant rate, namely the communication complexity of evaluating a circuit C is $O(|\mathrm{C}|)+\mathsf{poly}(\log|\mathrm{C}|, d, m, \kappa)$, where d, m, κ are the depth of C, the number of parties, and a security parameter, respectively. For an arbitrary number of parties, the protocol of Genkin et al. [GIW16] obtains $\mathsf{poly}\log(|\mathrm{C}|, s)$ overhead over passive-GMW, where s is a statistical security parameter. This result is obtained by converting a boolean circuit C into a functionally-equivalent randomized circuit C′ that is immune against so called "additive attacks", and evaluating C′ using the semi-honest protocol of [GMW87]. (This technique was originally introduced by [GIP+14], but was essentially only useful over large fields, see discussion below.)

The state-of-the-art: arithmetic multi-party setting. In the arithmetic setting in which the computation is performed over an arbitrary field $\mathbb{F}$, Genkin et al. [GIP+14] designed MPC protocols in the OLE-hybrid model, with a statistical error of $O(|\mathrm{C}|/\mathbb{F})$, and constant communication overhead compared to an algebraic variant of passive-GMW [GMW87], for sufficiently large fields $\mathbb{F}$. As described above, their result is obtained by converting a circuit C over some field $\mathbb{F}$ into its additively-secure variant C′, and evaluating C′ using passive-GMW and actively secure implementation of OLE. In practice, the constant in the communication overhead of their protocol is 12, and moreover their protocol is only useful for circuits over large fields (for which $O(|\mathrm{C}|/\mathbb{F})$ is sufficiently small). For arbitrary fields, the work of Döttling et al. [DGN+17] give an actively secure protocol where the overhead is 22 invocations of an actively secure implementation of $\mathcal{F}_{\text{MULT}}$ per multiplication gate of the circuit. A practical implementation for arbitrary number of parties was given in [KPR18] based on "tailor-made" zero-knowledge proofs to achieve active security.

We note that in the *honest majority* setting, the recent work by Chida et al. [CGH+18] presents a new actively-secure protocol for arithmetic circuits that obtains overhead 2 over passive protocols for sufficiently large fields. Similar to our protocol, their protocol is in the $\mathcal{F}_{\text{MULT}}$-hybrid model, where $\mathcal{F}_{\text{MULT}}$ can be instantiated with any passively-secure protocol that further guarantees a notion of "security up to additive attacks" in the presence of active adversaries. It is unclear whether their paradigm extends to the dishonest majority setting, since their model of additive attacks is weaker than the standard one formulated in [GIP+14], where

Construction	Number of Parties	Hybrid Model	Asymptotic Overhead	Concrete Overhead
[IPS08]	Constant	OT	Constant*	Unexplored
[GIP+14]	Arbitrary	OLE	Constant (large fields)	12
[HIMV19]	Two	OLE	Constant	2

Table 2: Asymptotic and concrete communication overheads of state-of-the-art 2PC and MPC protocols in the dishonest majority setting. The overhead is over passive-GMW in the specified computational model (OT- or OLE-hybrid). * In terms of asymptotic complexity, we note that [IPS08] also achieves constant rate.

in all natural candidates an active attack translates into an additive attack in the latter (stronger) attack model, and is therefore not protected against by the framework of [CGH+18].

In an orthogonal vein, we note that Applebaum et al. [ADI+17] designed the first (variant of) passively-secure OLE based on LPN-style assumptions, implying secure arithmetic computation with asymptotic constant computational overhead over an insecure evaluation of the circuit.

The state-of-the-art: two-party setting. In the boolean setting, the protocols of [IPS08] and [HIV17] achieve (asymptotic) constant communication overhead over the passive protocols of [GMW87] and [Yao86], respectively. The latter has the added benefit of matching the number of OT calls in [Yao86], which (unlike [GMW87]) is sublinear in the circuit size. Practical implementations of [IPS08] have been studied in [LPO11], who identified bottlenecks in obtaining concretely-efficient protocols based on the IPS protocol due to the implementation of the so-called "watchlist channels". In the arithmetic setting, a recent work by Hazay et al. [HIMV19] instantiated the framework of [IPS08] with a concretely-efficient honest majority protocol, obtaining small multiplicative overheads (between 2-8) compared to the passive protocol of [GMW87].

2 Our Techniques

We first recall the so-called "IPS framework" of Ishai, Prabhakaran and Sahai [IPS08], that constructs actively-secure m-party protocols for a function f using the following two weaker ingredients as a black-box: (1) an actively-secure honest-majority protocol (the "outer protocol") for f with m clients and n servers, tolerating active corruption of a minority $t < n/2$ of the servers and an arbitrary number of clients; and (2) a passively secure m-party protocol (the "inner protocol") for a "simpler" functionality, tolerating an arbitrary number of corruptions.

Using appropriate instantiations of the outer and inner protocols, this framework yields a constant-overhead (in fact, constant-rate) actively-secure protocol for boolean functionalities in the dishonest majority setting with a constant number of parties m. However, it does not obtain constant overhead for a super-constant m, as we now explain.

To watch or not to watch? The high-level idea of the IPS compiler it to have the m parties "virtually" execute the outer protocol by emulating its n servers. Specifically, the parties first obtain (through some joint computation) secret shares of the initial server states, then use the inner protocol on the shared states

to generate (secret shares) of the outputs of the "next message" functions of each server. Since the outer protocol is only secure when a majority of the servers are honest, the parties must insure that most servers were correctly emulated, for which it suffices to verify that the parties behave honestly in sufficiently many of the inner protocol executions. The IPS compiler introduces a novel "watchlist" mechanism in which parties "watch" each other to enforce such honest behaviour. More precisely, every party P_i picks a random subset of t servers for which it learns the entire internal state throughout the computation. Consequently, P_i can check that all parties honestly emulated the t servers, and abort if some party misbehaves. The identity of servers watched by honest parties remains hidden from the adversary, thus even a single honest party forces the adversary to honestly emulate most (specifically, a majority) of the servers. In terms of parameters, obtaining a $2^{-\Omega(s)}$ soundness error for a statistical security parameter s requires $t, n = \Omega(s)$. Since each corrupted party can choose an arbitrary subset of t watched servers, and there could be $m-1$ corrupted parties, privacy is only preserved when $(m-1)t < n/2$. Since achieving constant-overhead requires $n = O(s)$, this is only possible for $m = O(1)$.

Compute first, check later. To solve this problem, our first idea is to have a *single* random subset of t servers which are *simultaneously* watched by *all* parties. Of course, now that the identity of the watched servers is known to all parties, it cannot be revealed before the computation has been completed. Instead, the subset is chosen using joint coin-tossing after the circuit has been evaluated, but before the output is reconstructed from the output shares. Correctness is preserved similarly to the original IPS compiler, but checking honest behavior after-the-fact might violate privacy. Indeed, unlike the IPS compiler we can no longer "catch" the adversary as soon as it deviates from the protocol, which raises two privacy concerns. First, by actively deviating from the protocol, the adversary can potentially violate the inner protocol privacy, and learn intermediate values during the circuit evaluation. Second, the adversary can potentially violate the privacy of the outer protocol, by "corrupting" a majority of the servers in the outer protocol (i.e., by not emulating them correctly). We note that even if the inner protocol has the stronger guarantee of remaining *private* even against *active* adversaries, this does not resolve the second issue because as long as the inner protocol is *not actively-secure*, active corruptions in it might violate correctness, which corresponds to corrupting servers in the outer protocol. Thus, an active adversary might still violate privacy in the outer protocol by violating correctness in the inner protocol (thus, in effect, corrupting possibly a majority of the servers).

Which outer protocol to use? Even if we could somehow overcome these issues, the overhead for $m = \omega(1)$ parties might still be large due to the communication complexity of the outer protocol. Specifically, [IPS08] obtain constant overhead by instantiating the outer protocol with the protocol of [DI06]. In this protocol, the servers compute over secret shares, and they need to perform a global operation on their shares (specifically, degree reduction). This is achieved by having the servers execute a randomized protocol, where the randomness is provided by the clients. Thus, the communication complexity scales with the number of clients, and cannot be constant for $m = \omega(1)$.

Our approach. Due to these issues, we take a step back, and (instead of extending the IPS framework) focus on designing a new compiler that amplifies the security of a passively-secure inner protocol via a tailor-made outer protocol. Since we use different instantiates of the inner protocol, we model it more generally, assuming the parties have oracle access to an ideal multiplication functionality $\mathcal{F}_{\text{MULT}}$ that works over some agreed-upon secret sharing scheme $\mathcal{S}$. We note that in our compiler, we will not refer to "servers" (or an "outer" protocol), but rather think of these as "copies" of the circuit.

The combined protocol. To highlight the main components of our framework, we describe a basic MPC variant that will loosely rely on the passive BGW [BGW88] protocol. Though this does not yield our

asymptotic results, it will serve as a good starting point, which we build on to obtain our final framework (as described towards the end of the section).

At the onset of the computation each party P_i secret shares its input x_i using Shamir's secret sharing scheme with privacy parameter t, to obtain the shares $\left(X^1, \ldots, X^n\right)$ (as in the passive-BGW protocol). Then, P_i generates additive shares $\left(x_j^l\right)$ of each Shamir share X^l, and sends $\left(x_j^l\right)_{l\in[n]}$ to P_j. The protocol will evaluates the circuit gate-by-gate as in passive-BGW, where addition gates are locally computed. We will preserve the invariant that when parties evaluate a gate G, they collectively hold additive shares of the Shamir shares of the values of G's input wires. That is, if G's inputs are values a, b which in the passive-BGW protocol have Shamir shares $\left(A^1, \ldots, A^n\right)$, $\left(B^1, \ldots, B^n\right)$ (respectively), then for every $l \in [n]$, party P_i holds values a_i^l, b_i^l such that $\sum_i a_i^l = A^l$ and $\sum_i b_i^l = B^l$.

In passive-BGW, multiplications are performed by having each party locally multiply its Shamir shares A^l, B^l, followed by all parties jointly running a degree-reduction sub-protocol on these products. However, in our modified protocol parties can no longer locally compute the products $A^l \cdot B^l$, because no party knows A^l, B^l (parties only know additive shares of these values). To solve this issue, we use an ideal distributed-multiplication functionality $\mathcal{F}_{\text{MULT}}$ which takes as input additive shares of two values x, y, and outputs a (fresh) additive sharing of their product $x \cdot y$. (We discuss $\mathcal{F}_{\text{MULT}}$ instantiations below.) This allows parties to learn additive shares of each product $A^l \cdot B^l$.

Once (additive shares of) the products $A^l \cdot B^l$ have been computed, degree reduction should be performed. In the classical passive-BGW protocol, degree reduction requires expensive communication, which is improved by protocols such as [DN07]. We use a new approach that significantly reduces the communication complexity, leveraging the fact that degree-reduction is a linear operation over the Shamir shares.

Local degree-reduction. Each party *locally* performs degree reduction over its additive shares of the Shamir shares. Across all parties, the additive shares obtained as a result of this procedure constitute a valid Shamir sharing of the "right" value, due to the linearity properties of Shamir's secret sharing scheme. Intuitively, the second secret-sharing layer allows parties to locally perform degree reduction, because it gives each party a *global "view"* of the protocol execution, as an additive share of the global view of the protocol execution. We note that performing degree-reduction locally also has the advantage of eliminating the need for *global* randomness which should be generated from the contributions of all parties. (As discussed above, this was one of the causes of super-constant overhead in [IPS08] when $m = \omega(1)$.)

Enforcing correctness. Once the computation is completed in all copies, we ensure it was performed correctly by incorporating a "correctness-enforcing" mechanism into the protocol. Specifically, before opening the output shares obtained at the outputs of all copies, we first run some correctness tests which will check that (with high probability) all parties honestly executed the computation. The output shares are revealed (and the output is reconstructed from these shares) only if all correctness tests pass.

To explain our correctness tests, we first analyze possible malicious strategies of corrupted parties. Roughly, a corrupted party can deviate from the protocol in one of four ways. First, it can incorrectly share its input (i.e., the "sharing" isn't of the right degree t). Second, it can incorrectly perform the degree-reduction procedure, by generating a fresh sharing that either isn't of the right degree (i.e., t), or doesn't share the right value (i.e., the value shared before degree reduction). Third, when evaluating a multiplication gate (i.e., computing the product of Shamir shares as described above), it can use different values than the ones provided by $\mathcal{F}_{\text{MULT}}$. Fourth, it can incorrectly perform the local linear computations.

To handle such deviations from the protocol, we introduce three tests. The first is a *degree* test, which checks that the secrets sharings used by all parties, either to share their inputs or as input to multiplication gates, have the right degree. The second is an *equality* test, which checks that the secret sharings before

and after degree reduction share the same value. The degree and equality tests jointly guarantee that with overwhelming probability, the input sharings are valid, and the degree reduction procedure was executed correctly (in most copies). Similar degree and equality tests were used in [AHIV17, HIMV19] to check similar conditions. The last test is a *consistency* test, which verifies that (with high probability) parties correctly performed the local computations in (most) copies of the circuit. This checks that the values used by the parties when evaluating a multiplication gate are consistent with the values they obtained from $\mathcal{F}_{\text{MULT}}$, that the local linear operations were performed correctly, and will also guarantee the soundness of the degree and equality tests. For this test, a random subset of copies is chosen, each party reveals its local view of the computation in those copies, and all parties check that the views are consistent with each other. Similar tests were used in the context of MPC-in-the-head [IKOS07, IPS08].

We note that this high-level overview omits important details (see Section 4). For example, the order in which parties commit and reveal the correctness tests' values is crucial to preserving privacy even when the computations in most copies are incorrect. Using this combination of correctness tests, and proving the security of this approach is novel to our work, and requires subtle analysis.

Achieving constant communication overhead. Our basic MPC protocol does not achieve constant communication overhead since it increases the communication complexity of the underlying BGW protocol [BGW88] by $O(s)$, where s is a security parameter. We reduce this overhead to constant by replacing [BGW88] with the protocol of Franklin and Yung [FY92] that uses packed secret sharing.

Loosely speaking, packed secret sharing extends Shamir's secret sharing, allowing a block of $\mathcal{B}$ secrets to be shared within a single set of shares. To exploit the advantages of packed secret sharing, we will assume the circuit is arranged in layers that contain only one type (addition/multiplication) of gates, where each phase of the protocol evaluates the gates in one layer.

Using packed secret sharing introduces two main differences from the basic protocol. First, before evaluating a specific layer the parties need to rearrange (repack) the shared secrets corresponding to the input wire values of that layer, to align the packing in blocks with the order of gates within the layer. Then, the layer can be evaluated similarly to the basic protocol (where additions are computed locally, and multiplications involve a call to $\mathcal{F}_{\text{MULT}}$, followed by a local degree-reduction step). The second difference from the basic protocol is that to insure correctness we must now check that the parties correctly rearranged the shared secrets between layers. This is checked through an additional "permutation test" [DI06, AHIV17]. See Section 5 for further details.

This protocol reduces the amortized per-gate communication overhead to constant, because in effect the packed secret sharing allows us to evaluate many gates in one "shot". In particular, the wider the circuit to be evaluated, the larger the gains from employing packed secret sharing.

Instantiating the multiplication functionality $\mathcal{F}_{\text{MULT}}$. We instantiate $\mathcal{F}_{\text{MULT}}$ through a reduction to a simpler functionality $\mathcal{F}_{\text{RMULT}}$ which generates (unauthenticated) random triples. All prior protocols that relied on this abstraction (apart from [IPS08]), used actively-secure multiplication protocols to instantiate $\mathcal{F}_{\text{MULT}}$. Interestingly, we can greatly weaken the security of the multiplication protocol, requiring only a passively-secure instantiation, together with a coin tossing protocol to ensure correctly-sampled randomness. Moreover, our protocol can benefit from a preprocessing stage in an offline/online setting, where the triples are generated in the offline phase, and used in the online phase. The consistency test (described for our basic MPC protocol) will ensure, at the cost of a small overhead, that the triples were correctly generated with respect to the tossed coins. We note that unlike prior works, our security analysis can tolerate a small number of ill-formed triples without violating secrecy.

Related techniques. Conceptually, our consistency test can be viewed as a combination of the cut-and-

choose approach [LP07] and the watchlist mechanism of [IPS08]. Indeed, on the one hand we maintain multiple copies of the computed circuit, yet unlike the cut-and-choose technique the checked copies are not discarded, but rather used in the remainder of the computation to reconstruct the outputs. On the other hand, the purpose of our consistency test is similar to the watchlist channels, which add privacy and correctness to passively-secure protocols. The main difference between our tests and the watchlists of [IPS08] is that in IPS these channels are used to *constantly* enforce correct behaviour *throughout* the protocol execution (and consequently also cause a high overhead), whereas we perform a single consistency test *after the protocol execution has (essentially) ended*, right before the output is reconstructed. These correctness enforcement mechanisms are known to have limitations to achieving scalable MPC. Specifically, the asymptotic limit of cut-and-choose is $O(s/\log|\mathrm{C}|)$ [WRK17b], whereas the watchlists mechanism requires $O(s \cdot n)$ virtual servers for the outer protocol [LOP11]. In both cases, the communication grows with some statistical parameter, and is hence neither constant-overhead nor scalable.

3 Preliminaries

Basic notations. We denote a security parameter by κ. We say that a function $\mu : \mathbb{N} \to \mathbb{N}$ is *negligible* if for every positive polynomial $p(\cdot)$ and all sufficiently large κ's it holds that $\mu(\kappa) < \frac{1}{p(\kappa)}$. We use the abbreviation PPT to denote probabilistic polynomial-time and denote by $[n]$ the set of elements $\{1, \ldots, n\}$ for some $n \in \mathbb{N}$. We assume functions to be represented by an arithmetic circuit C (with addition and multiplication gates of fan-in 2), and denote the size of C by $|\mathrm{C}|$. By default we define the size of the circuit to include the total number of gates including input gates.

3.1 Layered Arithmetic Circuits

An arithmetic circuit defined over a finite field $\mathbb{F}$ is a directed acyclic graph, where nodes (or *gates*) are labelled either as input gates, output gates or computation gates. Input gates have no incoming edges (or *wires*), while output gates have a single incoming wire and no outgoing wires. Computation gates are labelled with a field operations (either addition or multiplication),[1] and have exactly two incoming wires, which we denote as the left and right wire. A circuit with i input gates and o output gates over a field $\mathbb{F}$ represents a function $f : \mathbb{F}^i \to \mathbb{F}^o$ whose value on input $x = (x_1, \ldots, x_i)$ can be computed by assigning a value to each wire of the circuit. Note that this abstraction captures boolean circuits as well, by setting $\mathbb{F} = \mathbb{GF}(2)$. In this work, we will exploit an additional structure of the circuit. Specifically, the gates of an arithmetic circuit can be partitioned into ordered layers $\mathcal{L}_1, \ldots, \mathcal{L}_d$, such that i) a layer only consists of gates of the same type (i.e., addition, multiplication, input or output gates belonging to the same party), and ii) the incoming wires of all gates of layer i originate from gates in layers 0 to $i-1$.

3.2 Oblivious Transfer

1-out-of-2 oblivious transfer (OT) [Rab81, EGL85] is a fundamental functionality in secure computation between a sender S and a receiver R, which allows R to learn only one of the S's inputs while S learns nothing about R's input. In this paper we consider the basic 1-out-of-2 OT functionality, and employ it to implement a product of two values using their additive shares; see Figure 1 for the formal description.

[1] Subtraction gates can be handled analogously to addition gates, and we ignore them here for simplicity.

Functionality $\mathcal{F}_{\text{OT}}$

Functionality $\mathcal{F}_{\text{OT}}$ communicates with sender S, receiver R, and adversary $\mathcal{S}$.

1. Upon receiving input $(\mathsf{sid}, \mathsf{sender}, v_1, v_2)$ from S, where each $v_i \in \{0,1\}^\ell$, record (sid, v_1, v_2).
2. Upon receiving $(\mathsf{sid}, \mathsf{receiver}, u)$ from R where $u \in \{0,1\}$, check if a $(\mathsf{sid}, \mathsf{sender}, \ldots)$ message was previously sent. If yes, send $(\mathsf{ouput}, \mathsf{sid})$ to $\mathcal{S}$. Upon an answer OK, deliver (sid, v_u) to R. Otherwise, abort.

Figure 1: The oblivious transfer functionality.

Functionality $\mathcal{F}_{\text{OLE}}$

Functionality $\mathcal{F}_{\text{OLE}}$ communicates with sender S, receiver R, and adversary $\mathcal{S}$.

1. Upon receiving the input $(\mathsf{sid}, \mathsf{sender}, (a, b))$ from S where $a, b \in \mathbb{F}$, record $(\mathsf{sid}, \mathsf{sender}, (a, b))$.
2. Upon receiving (sid, x) from R where $x \in \mathbb{F}$, check if a $(\mathsf{sid}, \mathsf{sender}, \ldots)$ message was previously sent. If yes, send $(\mathsf{ouput}, \mathsf{sid})$ to $\mathcal{S}$. Upon an answer OK, deliver $(\mathsf{sid}, a \cdot x + b)$ to R. Otherwise, abort.

Figure 2: The oblivious linear evaluation functionality.

3.3 Oblivious Linear Evaluation

An extension of the oblivious transfer functionality for larger fields is the Oblivious Linear Evaluation functionality (OLE) [NP06]. More concretely, OLE over a field $\mathbb{F}$ takes a field element $x \in \mathbb{F}$ from the receiver and a pair $(a, b) \in \mathbb{F}^2$ from the sender and delivers $ax + b$ to the receiver. Note that in the case of binary fields, OLE can be realized via a single call to the standard (bit-) 1-out-of-2 OT functionality; see Figure 2 for the formal description.

3.4 Commitment Schemes

A commitment scheme is a two-phase protocol that allows a *sender* S to commit some value v to a *receiver* R while simultaneously hiding v from R during the first "Commit" phase (the *hiding* property), and guaranteeing that the commit phase determines a unique value which can be later revealed in the second "Opening" phase (this property is called *binding*, the sender can always refuse to open the commitment, but cannot open it to a different value). The formal description of functionality $\mathcal{F}_{\text{COM}}$ is depicted in Figure 3.

3.5 Multiplication Functionalities

A core building block in our protocols is a multiplication functionality $\mathcal{F}_{\text{MULT}}$ shown in Figure 4, that takes additive shares of two secrets over some field $\mathbb{F}$ and produces additive shares of their product. In fact, we will reduce $\mathcal{F}_{\text{MULT}}$ to a random instance $\mathcal{F}_{\text{RMULT}}$, shown in Figure 5, where all shares are chosen uniformly at random from $\mathbb{F}$. The reduction, due to Beaver [Bea91], is as follows. Denote by $[a]$ the additive sharing of some value $a \in \mathbb{F}$, namely, the tuple $(a_1, \ldots, a_m)$. Then, given a random triple $[a], [b], [c]$ obtained as the output of $\mathcal{F}_{\text{RMULT}}$, and inputs $[x], [y]$ for $\mathcal{F}_{\text{MULT}}$, we can compute $[xy]$ by first reconstructing $e = [x + a]$

Functionality $\mathcal{F}_{\text{COM}}$

Functionality $\mathcal{F}_{\text{COM}}$ communicates with sender S, receiver R, and adversary $\mathcal{S}$.

1. Upon receiving input $(\mathsf{commit}, sid, m)$ from S where $m \in \{0,1\}^t$, internally record (sid, m) and send message (sid, S, R) to the adversary. Upon receiving approve from the adversary send sid to R. Ignore subsequent $(\mathsf{commit}, ., ., .)$ messages.
2. Upon receiving (reveal, sid) from S, where a tuple (sid, m) is recorded, send message m to adversary $\mathcal{S}$ and R. Otherwise, ignore.

Figure 3: The string commitment functionality.

Functionality $\mathcal{F}_{\text{MULT}}$

Functionality $\mathcal{F}_{\text{MULT}}$ communicates with parties $P_1, \ldots, P_m$ and adversary $\mathcal{S}$ corrupting a subset $I \subset [m]$ of parties.

1. Upon receiving the input (sid, a_j, b_j) from P_j record $(\mathsf{sid}, (a_j, b_j))$.
2. If a tuple is recorded from all parties continue as follows:
   (a) Compute $c = (\sum_j a_j) \cdot (\sum_j b_j)$.
   (b) Receive corrupted parties' shares $\{c_j\}_{j \in I}$.
   (c) Sample honest parties' shares $\{c_j\}_{j \notin I}$ at random from $\mathbb{F}$ condition on $c = \sum_{j=1}^{m} c_j$.
   (d) Forward c_j to party P_j.

Figure 4: The multiplication functionality.

and $d = [y + b]$. Next, the parties compute a (trivial) secret sharing $[ed]$ of ed by having P_1 set its share to ed, and the rest of the parties set their shares to 0. Finally, the parties compute the following equation (each party locally computes the equation on its own shares)

$$[xy] = [c] + e[y] + d[x] - [ed] = [ab] + (x + a)[y] + (y + b)[x] - (x + a)(y + b).$$

3.6 Secret-Sharing

A secret-sharing scheme allows a dealer to distribute a secret among n parties, where each party receives a share (or piece) of the secret during a *sharing phase*. In its simplest form, the goal of (threshold) secret-sharing is to allow only subsets of players of size at least $t + 1$ to reconstruct the secret. More formally a $t + 1$-out-of-n secret sharing scheme comes with a sharing algorithm that on input a secret s outputs n shares $s_1, \ldots, s_n$ and a reconstruction algorithm that takes as input $((s_i)_{i \in S}, S)$ where $|S| > t$ and outputs either a secret s' or $\perp$. In this work, we will use Shamir's secret sharing scheme [Sha79] with secrets in $\mathbb{F} = \mathbb{GF}(2^\kappa)$. We present the sharing and reconstruction algorithms below:

Sharing algorithm: For any input $s \in \mathbb{F}$, pick a random polynomial $p(\cdot)$ of degree t in the polynomial-field $\mathbb{F}[x]$ with the condition that $p(0) = s$ and output $p(1), \ldots, p(n)$.

Functionality $\mathcal{F}_{\text{RMULT}}$

Functionality $\mathcal{F}_{\text{RMULT}}$ communicates with parties $P_1, \ldots, P_m$ and adversary $\mathcal{S}$ corrupting the subset of parties in $I \subset [m]$.

1. Receive corrupted parties' shares $\{a_j, b_j, c_j\}_{j \in I}$.
2. Sample honest parties' shares $\{a_j, b_j, c_j\}_{j \notin I}$ at random from $\mathbb{F}$ condition on the following equation

$$\sum_{j=1}^{m} c_j = \left(\sum_j a_j\right) \cdot \left(\sum_j b_j\right).$$

3. Forward c_j to party P_j.

Figure 5: The random multiplication functionality.

Reconstruction algorithm: For any input $(s'_i)_{i \in S}$ where none of the s'_i are $\perp$ and $|S| > t$, compute a polynomial $g(x)$ such that $g(i) = s'_i$ for every $i \in S$. This is possible using Lagrange interpolation where g is given by

$$g(x) = \sum_{i \in S} s'_i \prod_{j \in S/\{i\}} \frac{x - j}{i - j}.$$

Finally the reconstruction algorithm outputs $g(0)$.

Packed secret-sharing. The concept of packed secret-sharing was introduced by Franking and Yung in [FY92] in order to reduce the communication complexity of secure multi-party protocols, and is an extension of standard secret-sharing. In particular, the authors considered Shamir's secret sharing with the difference that the number of secrets $s_1, \ldots, s_\ell$ is now ℓ instead of a single secret, where the secrets are represented as the evaluations of a polynomial $p(\cdot)$ at ℓ distinct points. To ensure privacy in case of t colluding corrupted parties, $p(\cdot)$ must have degree at least $t + \ell$. Packed secret sharing inherits the linearity property of Shamir's secret sharing, with the added benefit that it supports batch (block-wise) multiplications. This was used to design secure computation protocols with an honest majority and constant amortized overhead [DI06]. For this reason, we use this tool in our honest majority MPC protocol embedded within our dishonest majority protocol from Section 4, leveraging its advantages to improve the overhead of the former protocol.

3.7 Error Correcting Codes

A crucial ingredient in our construction is the use of Reed-Solomon codes as a packed secret sharing scheme [FY92] (as defined in Section 3.6). In what follows, we provide our coding notations and related definitions.

Coding notation. For a code $C \subseteq \Sigma^n$ and vector $v \in \Sigma^n$, we denote by $d(v, C)$ the minimal distance of v from C, namely the number of positions in which v differs from the closest codeword in C, and by $\Delta(v, C)$ the set of positions in which v differs from such a closest codeword (in case of a tie, take the lexicographically first closest codeword). For any $k \leq d(v, C)$, we say that v is k-close to C, and for every

$k > d(v, C)$, we say that v is k-far from C. We further denote by $d(V, C)$ the minimal distance between a vector set V and a code C, namely $d(V, C) = \min_{v \in V}\{d(v, C)\}$.

Definition 1 (Reed-Solomon code)**.** *For positive integers n, k, finite field $\mathbb{F}$, and a vector $\eta = (\eta_1, \ldots, \eta_n) \in \mathbb{F}^n$ of distinct field elements, the code $\mathsf{RS}_{\mathbb{F},n,k,\eta}$ is the $[n, k, n-k+1]$-linear code[2] over $\mathbb{F}$ that consists of all n-tuples $(p(\eta_1), \ldots, p(\eta_n))$ where p is a polynomial of degree $< k$ over $\mathbb{F}$.*

Definition 2 (Encoded message)**.** *Let $L = \mathsf{RS}_{\mathbb{F},n,k,\eta}$ be an RS code and $\zeta = (\zeta_1, \ldots, \zeta_w)$ be a sequence of distinct elements of $\mathbb{F}$ for $w \leq k$. For $u \in L$ we define the message $\mathsf{Decode}_\zeta(u)$ to be $(p_u(\zeta_1), \ldots, p_u(\zeta_w))$, where p_u is the polynomial (of degree $< k$) corresponding to u. For $U \in L^m$ with rows $u^1, \ldots, u^m \in L$, we let $\mathsf{Decode}_\zeta(U)$ be the length mw vector $x = (x_{11}, \ldots, x_{1w}, \ldots, x_{m1}, \ldots, x_{mw})$ such that $(x_{i1}, \ldots, x_{iw}) = \mathsf{Decode}_\zeta(u^i)$ for $i \in [m]$. We say that u L-encodes x (or simply encodes x) if $x = \mathsf{Decode}_\zeta(u)$.*

Moreover, we recall that $\mathsf{Decode}_\zeta(\cdot)$ is a linear operation, i.e. for any $a, b \in \mathbb{F}^n$ (even if a, b are not in L), $\mathsf{Decode}_\zeta(a + b) = \mathsf{Decode}_\zeta(a) + \mathsf{Decode}_\zeta(b)$.

It will be convenient to view m-tuples of codewords in L as codewords in an interleaved code L^m. We formally define this notion below.

Definition 3 (Interleaved code)**.** *Let $L \subset \mathbb{F}^n$ be an $[n, k, d]$ linear code over $\mathbb{F}$. We let L^m denote the $[n, mk, d]$ (interleaved) code over $\mathbb{F}^m$ whose codewords are all $m \times n$ matrices U such that every row U_i of U satisfies $U_i \in L$. For $U \in L^m$ and $j \in [n]$, we denote by $U[j]$ the j'th symbol (column) of U.*

3.8 Secure Multiparty Computation (MPC)

We use a standard stand-alone definition of secure multi-party computation protocols. In this work, we only consider static corruptions, i.e. the adversary decides which parties it corrupts before the execution begins. Following [HL10], we use two security parameters in our definition: a computational security parameter κ, and a statistical security parameter s that captures a statistical error of up to 2^{-s}. We assume that $s \leq \kappa$. We let $\mathcal{F}$ be a multi-party functionality that maps a set of n inputs to an output over some field $\mathbb{F}$ (w.l.o.g).

Let $\Pi = \langle P_1, \ldots, P_n \rangle$ denote a multi-party protocol, where each party is given an input x_i and security parameters 1^s and 1^κ. We allow honest parties to be PPT in the entire input length (this is needed to ensure correctness when no party is corrupted), but bound adversaries to time $\mathsf{poly}(\kappa)$ (this effectively means that we only require security when the input length is bounded by *some* polynomial in κ). We denote by $\mathbf{REAL}_{\Pi,\mathcal{A}(z)}(x_1, \ldots, x_n, \kappa, s)$ the output of the honest parties and the adversary $\mathcal{A}$ controlling a subset $I \subset [n]$ of parties in the real execution of Π, where z is the auxiliary input, x_i is P_i's initial input, κ is the computational security parameter, and s is the statistical security parameter. We denote by $\mathbf{IDEAL}_{\mathcal{F},\mathcal{S}(z)}(x_1, \ldots, x_n, \kappa, s)$ the output of the honest parties and the simulator $\mathcal{S}$ in the ideal model where $\mathcal{F}$ is computed by a trusted party. In some of our protocols the parties have access to ideal model implementations of certain cryptographic primitives such as ideal oblivious-transfer ($\mathcal{F}_{\mathrm{OT}}$). We denote such executions by $\mathbf{REAL}^{\mathcal{F}_{\mathrm{OT}}}_{\Pi,\mathcal{A}(z)}(x_1, \ldots, x_n, \kappa, s)$.

Definition 4. *A protocol $\Pi = \langle P_1, \ldots, P_n \rangle$ is said to* securely compute a functionality $\mathcal{F}$ in the presence of active adversaries *if the parties always have the correct output $\mathcal{F}(x_1, \ldots, x_n)$ when neither party is corrupted, and moreover the following security requirement holds. For any probabilistic $\mathsf{poly}(\kappa)$-time adversary $\mathcal{A}$ controlling a subset $I \subset [n]$ of parties in the real world, there exists a probabilistic $\mathsf{poly}(\kappa)$-time*

[2] We denote by $[n, k, d]$-linear code a linear code of length n, rank k and minimum distance d, where the minimum distance of the code is the minimal weight of a codeword in the code.

adversary (simulator) $\mathcal{S}$ controlling I in the ideal model, such that for every non-uniform $\mathsf{poly}(\kappa)$*-time distinguisher $\mathcal{D}$ there exists a negligible function $\nu(\cdot)$ such that the following ensembles are distinguished by $\mathcal{D}$ with at most $\nu(\kappa) + 2^{-s}$ advantage:*

- $\{\mathbf{REAL}_{\Pi,\mathcal{A}(z),P_i}(x_1,\ldots,x_n,\kappa,s)\}_{\kappa\in\mathbb{N},s\in\mathbb{N},x_1,\ldots,x_n,z\in\{0,1\}^*}$
- $\{\mathbf{IDEAL}_{\mathcal{F},\mathcal{S}(z),P_i}(x_1,\ldots,x_n,\kappa,s)\}_{\kappa\in\mathbb{N},s\in\mathbb{N},x_1,\ldots,x_n,z\in\{0,1\}^*}$

Secure circuit evaluation. The above definition considers $\mathcal{F}$ to be an infinite functionality, taking inputs of an arbitrary length. However, our protocols (similarly to other protocols from the literature) are formulated for a finite functionality $\mathcal{F} : \mathbb{F}^{\alpha_1} \times \cdots \times \mathbb{F}^{\alpha_n} \to \mathbb{F}$ described by an arithmetic circuit C (where the computation is performed over a finite field $\mathbb{F}$). Such protocols are formally captured by a polynomial-time *protocol compiler* that, given security parameters $1^\kappa, 1^s$ and a circuit C, outputs n circuits $(P_1,\ldots,P_n)$ that implement the next message function of n parties in the protocol (possibly using oracle calls to a cryptographic primitive or an ideal functionality oracle). Similar to Definition 4, the correctness requirement (when no party is corrupted) holds for any choice of κ, s, C, while the security requirement only considers adversaries that run in time $\mathsf{poly}(\kappa)$. That is, we require indistinguishability (in the sense of Definition 4) between

- $\{\mathbf{REAL}_{\Pi,\mathcal{A}(z)}(\mathrm{C},x_1,\ldots,x_n,\kappa,s)\}_{\kappa\in\mathbb{N},s\in\mathbb{N},\mathrm{C}\in\mathcal{C},x_1,\ldots,x_n,z\in\{0,1\}^*}$
- $\{\mathbf{IDEAL}_{\mathcal{F},\mathcal{S}(z),P_i}(\mathrm{C},x_1,\ldots,x_n,\kappa,s)\}_{\kappa\in\mathbb{N},s\in\mathbb{N},\mathrm{C}\in\mathcal{C},x_1,\ldots,x_n,z\in\{0,1\}^*}$

where $\mathcal{C}$ is the class of arithmetic circuits that take n vectors of field elements as inputs and output a field element, $x_1,\ldots,x_n$ are of lengths corresponding to the inputs of C, $\mathcal{F}$ is the functionality computed by C, and the next message functions of the parties $P_1,\ldots,P_n$ are as specified by the protocol compiler on inputs $1^\kappa, 1^s, \mathrm{C}$. We assume that C is arranged in d layers where each layer contains only multiplication gates or only addition gates over $\mathbb{F}$. We further assume that each layer takes its input only from the previous layers, and provides output to subsequent layers. The size of the circuit C, denoted by $|\mathrm{C}|$, is defined as the number of gates plus the number of wires. Its depth is the length of the longest path from an input to an output which, in the case of layered circuits, is equal to the number of layers.

4 Basic MPC Protocol

In this section we describe a simple variant of our MPC protocol, which we build on in Section 5 to achieve constant overhead.

Our starting point is a passively-secure variant of the BGW protocol [BGW88], which we amplify to the actively-secure dishonest-majority setting. Amplifying the security of this protocol requires facing three challenges: (1) high overhead due to the degree-reduction sub-protocol; (2) security holds only with a dishonest minority; and (3) security holds only against passive corruptions.

Our strategy towards addressing the first issue is to have parties *locally* perform the degree-reduction procedure which the degree-reduction sub-protocol implements, thus (almost) eliminating the interaction it requires. This is achieved by using a second layer of secret-sharing.

Concretely, our MPC protocol with m parties relies on two layers of secret sharing schemes: (1) first layer sharing: Reed-Solomon codes (which can be thought of as Shamir's secret sharing), denoted by L-encoding, where $L = \mathsf{RS}_{\mathbb{F},n,k,\eta}$ (cf. Section 3.7); and (2) second layer sharing: additive secret sharing. Throughout the execution, the parties hold additive shares of the L-encodings of the wires of the evaluated

circuit C. We note that using this two-layer secret sharing decouples the number of parties m from the length of the encoding n, since (unlike passive-BGW) parties no longer hold the symbols of the L-encodings. In fact, it will be useful to have $m \neq n$. Intuitively, this can be though of as having the parties emulate n copies of C, where the wires of the l'th copy carry the l'th symbol in the L-encodings of the wire values of C, and these symbols are additively shared among the parties. The execution maintains the invariant that when evaluating the gates in layer $\mathcal{L}$, the parties hold for each copy l additive shares of the l'th symbols in the L-encodings of the outputs of previous layers.

Our protocol is described in the $\mathcal{F}_{\text{RMULT}}$-hybrid model (cf. Section 3.5) which generates m additive shares of random triples, and is used to execute multiplications. In more detail, the parties evaluate the n copies of C layer by layer, locally performing additions, substractions and multiplications by a constant (this is possible due to the linear nature of our secret sharing schemes), whereas multiplication gates require communication.

Roughly, a multiplication gate G in the l'th copy of C is evaluated as follows. The parties hold additive shares of the l'th symbols A^l, B^l at the inputs of G, and use $\mathcal{F}_{\text{RMULT}}$ (and a reduction from $\mathcal{F}_{\text{MULT}}$ to $\mathcal{F}_{\text{RMULT}}$, described in Section 3.5) to obtain additive shares of the product $A^l B^l$. Across all copies, these products form an $\tilde{L}$-encoding of the output wire of G, where $\tilde{L} = \mathsf{RS}_{\mathbb{F},n,2k,\eta}$. To obtain a fresh L-encoding of the output wire, each party interprets its additive shares of the $\tilde{L}$-encoding (across all copies) as an encoding in $\mathsf{RS}_{\mathbb{F},n,n,\eta}$, decodes it, and then generates a fresh L-encoding of this decoded value. The additive shares obtained through this procedure reconstruct to the correct value because degree reduction is a linear operation.

Employing a second secret-sharing layer solves the second challenge (that passive-BGW is only private in the honest majority setting) since a subset of parties learn only a strict subset of additive shares. The third challenge (passive-BGW is only secure against passive corruptions) is handled by incorporating correctness-enforcing tests into the protocol, as described in Section 2.

Our detailed protocol is given in Figures 6-8. We next state the following theorem:

Theorem 1. *Protocol Φ described in Figures 6-8 securely realizes $\mathcal{F}$ in the $(\mathcal{F}_{\text{COM}}, \mathcal{F}_{\text{RMULT}}, \mathcal{F}_{\text{COIN}})$-hybrid model, tolerating $m-1$ active (static) corruptions, with statistical security error*

$$(1-e/n)^{\delta} + \frac{n-k+2}{|\mathbb{F}|} + 2^{-\Omega(e)}$$

where $k > \delta + 4e, n > 2k + 4e$ and $e \leq (n-k+1)/3$.

We first give an overview of the proof.

Proof Overview. The simulation follows by having the simulator Sim execute the protocol with the adversary, emulating the ideal functionalities for it, and emulating the honest parties on dummy 0-inputs. Before executing the output decommitment step, Sim performs several checks regarding the actions of the corrupted parties. Specifically, the simulator determines the set E of copies for which, if they were chosen during the consistency test, the test would fail. It also identifies the set E' of copies in which the $\mathcal{F}_{\text{RMULT}}$ values the corrupted parties committed to are inconsistent with the ones Sim provided to them. Then, it verifies that $|E| \leq e, |E|' \leq 3e$, and that there exist $\hat{U}, \hat{\vec{X}}_i, i \in [m]$, and $\hat{\vec{z}}$ which are valid encodings in the appropriate (interleaved) codes that agree with $\sum_{i\in[m]} U_i, \vec{X}_i, i \in [m]$, and $\sum_{i\in[m]} \vec{z}_i$ (respectively) except for the copies in E. It also verifies that there exists a $\hat{V}$ in the interleaved code over $\tilde{L}$ that agrees with $\sum_{i\in[m]} V_i$ except for the copies in $E \cup E'$. We note that Sim can perform these checks because it emulated the internal ideal functionalities for the adversary, whereas the honest parties in the protocol cannot perform

Protocol Φ.

- **Inputs.** P_i's input is x_i for all $i \in [m]$. The parties share a description of an arithmetic circuit C with fan-in 2 which contains h multiplication gates and implements functionality $\mathcal{F}$.
- **Initialization.**

  The parties invoke the $\mathcal{F}_{\text{RMULT}}$ functionality hn times. Each invocation yields additive shares $\left(r_1^1, \ldots, r_m^1\right)$, $\left(r_1^2, \ldots, r_m^2\right)$ and $\left(r_1^3, \ldots, r_m^3\right)$, with party P_i holding (r_i^1, r_i^2, r_i^3), such that $r^j = \sum_{i=1}^m r_i^j$ for $j \in \{1,2,3\}$, and $r^3 = r^1 \cdot r^2$. Each party P_i generates a random L-encoding $\vec{\gamma}_i = (\gamma_i^1, \ldots, \gamma_i^n)$ of a random value, a random L encoding $\vec{\nu}_i = (\nu_i^1, \ldots, \nu_i^n)$ of 0, and a random $\tilde{L}$ encoding $\vec{\widetilde{\gamma}}_i = (\widetilde{\gamma}_i^1, \ldots, \widetilde{\gamma}_i^n)$ of 0. P_i samples a tuple $(\vec{\psi}_i^1, \ldots, \vec{\psi}_i^m)$ such that $\vec{\psi}_i^j \in \mathbb{F}^n$ and $\sum_{j=1}^m \vec{\psi}_i^j$ is the all-$\vec{0}$ vector. P_i sends $\vec{\psi}_i^j$ to party P_j. These "blinding" encodings are used in the degree and equality tests of Figure 7.

  Then, for every copy $l \in [n]$, P_i commits using $\mathcal{F}_{\text{COM}}$ to:

  - The triples obtained from the $(l-1) \cdot h + 1, \ldots, hl$'th invocations of the $\mathcal{F}_{\text{RMULT}}$ oracle.
  - $\gamma_i^l, \nu_i^l, \widetilde{\gamma}_i^l$ and $\psi_i^{j,l}$ for every j.
- **Input sharing.** Each party P_i generates a random L-encoding $\vec{X}_i = \left(X_i^1, \ldots, X_i^n\right)$ of its input x_i (where X_i^l will be used in the evaluation of the l'th copy of C), and commits to $X_i^1, \ldots, X_i^n$ using $\mathcal{F}_{\text{COM}}$. For every $1 \le l \le n$, P_i generates an additive sharing $\left(x_{i,1}^l, \ldots, x_{i,m}^l\right)$ of X_i^l, and sends $\left(x_{i,j}^l\right)_{l \in [n]}$ to P_j. Each party P_i uses the shares $x_{j,i}^l$ ($j \in [n]$) as its inputs to the l'th copy.
- **Emulating the computation.** For every copy $l \in [n]$ of C, every layer $\mathcal{L} \in [d]$ in C, and every gate $G \in [w]$ in layer $\mathcal{L}$ (where w is the width of C), do:

  1. **Additions/subtractions.** If G is an addition or subtraction gate, each P_i performs the gate operation by applying it locally on the additive shares maintained as the inputs of that gate in the l'th copy.
  2. **Multiplications.** To compute a multiplication gate, the parties invoke the following multiplication protocol, where each party uses as inputs its l'th-copy shares of the inputs of G.
     - For every i, let a_i^l, b_i^l denote the shares of the inputs of G which P_i holds in the l'th copy of C. Then the parties compute additive shares $\left(\tilde{c}_1^l, \ldots, \tilde{c}_m^l\right)$ of $(\sum_{i=1}^m a_i^l)(\sum_{i=1}^m b_i^l)$, where P_i receives $\tilde{c}_i^l$, via the reduction from $\mathcal{F}_{\text{MULT}}$ to $\mathcal{F}_{\text{RMULT}}$ (described in Section 3.5), using the first unused triple obtained from $\mathcal{F}_{\text{RMULT}}$ in the (next unused portion of the) randomness generation phase above.
     - Then, P_i locally performs degree reduction on its shares $\tilde{c}_i^1, \ldots, \tilde{c}_i^n$ as follows: it interprets $\left(\tilde{c}_i^1, \ldots, \tilde{c}_i^n\right)$ as an encoding in $\mathsf{RS}_{\mathbb{F},n,n,\eta}$, and applies the decoding procedure to obtain a value o_i. It then generates a fresh L-encoding $\left(c_i^1, \ldots, c_i^n\right)$ of o_i, which it uses as the additive shares of the output of G across the n copies. (We note that $\tilde{c}_i^1, \ldots, \tilde{c}_i^n$ are additive shares of a purported $\tilde{L}$-encoding where $\tilde{L} = \mathsf{RS}_{\mathbb{F},n,2\cdot k,\eta}$, but as a length-$n$ encoding it is always consistent with *some* valid encoding in $\mathsf{RS}_{\mathbb{F},n,n,\eta}$.)
- **Output commitments.** For the output wire z, let $\vec{w}_i$ be the additive shares held by party P_i for the output. Then, P_i computes $\vec{z}_i = \vec{w}_i + \vec{\nu}_i$ where $\vec{\nu}_i$ is the L-encoding of 0 committed to during the initialization step. Then, P_i commits using $\mathcal{F}_{\text{COM}}$ to its shares $\vec{z}_i = \left(z_i^1, \ldots, z_i^n\right)$.

Figure 6: **Actively Secure MPC Φ – Part 1 (Circuit Emulation)**.

Correctness tests. The following tests are performed to verify that the parties correctly evaluated the n copies of C (including the degree reduction step executed after each multiplication gate).

- **Commit to degree test.** *This test checks that the input encodings and the shares produced by all parties at the end of every degree reduction step are valid L-encodings. This is done by checking that a random linear combination of the sum of all these shares is a valid encoding in $L = \mathsf{RS}_{\mathbb{F},n,k,\eta}$.*

  More precisely, the parties first obtain from $\mathcal{F}_{\text{COIN}}$ random vectors $\vec{r} \in \mathbb{F}^h$, $\vec{r}' \in \mathbb{F}^m$, and $r'' \in \mathbb{F}$ (recall that h is the number of multiplication gates in C, and m is the number of inputs — one from each party). Next, each party P_i constructs the matrix $U_i \in \mathbb{F}^{h \times n}$ that contains the L-encodings obtained after the degree reduction step of all multiplication gates (arranged in some arbitrary order, agreed upon by all parties). Then, P_i locally computes

$$q_i = \vec{r}^T U_i + r'_i \vec{X}_i + r'' \vec{\nu}_i + \vec{\gamma}_i,$$

  where $\vec{X}^i$ is the L-encoding of P_i's input x_i committed at the input sharing step, $\vec{\nu}_i$ is the L-encoding of 0 committed to by P_i at the initialization step and $\vec{\gamma}_i$ is the blinding L-encoding committed to at the initialization step. P_i then commits to each element of q_i, and each column of U_i, using $\mathcal{F}_{\text{COM}}$.

- **Commit to equality test.** *This test checks that the degree reduction step was performed correctly. This is done by checking that a random linear combination of the sum of differences of shares before and after the degree reduction step (performed as part of evaluating a multiplication gate) is a valid encoding of 0 in $\tilde{L} = \mathsf{RS}_{\mathbb{F},n,2k,\eta}$.*

  Specifically, the parties obtain from $\mathcal{F}_{\text{COIN}}$ a random vector $\vec{\alpha} = (\alpha_1, \ldots, \alpha_h) \in \mathbb{F}^h$ and random element $\beta \in \mathbb{F}$. P_i sets V_i to contain the additive shares which P_i obtains from the $\mathcal{F}_{\text{MULT}}$ to $\mathcal{F}_{\text{RMULT}}$ reduction computed during the evaluation of multiplication gates. Next, P_i locally computes:

$$\tilde{q}_i = \vec{\alpha}^T (V_i - U_i) + \beta \vec{\nu}_i + \vec{\tilde{\gamma}}_i + \vec{b}_i$$

  where $\vec{b}_i = \left(b_i^1, \ldots, b_i^n\right)$, $b_i^l = \sum_{j=1}^m \psi_j^{i,l}$, and $\vec{\tilde{\gamma}}_i$ is the $\tilde{L}$-encoding of 0 from the initialization step. Finally, P_i commits to each element of $\tilde{q}_i$ using $\mathcal{F}_{\text{COM}}$.

Figure 7: **Actively Secure MPC Φ – Part 2 (Correctness Tests Commitments)**.

these checks. If all checks pass then Sim can extract effective inputs for the corrupted parties, and use them to obtain the output from the trusted party. Finally, Sim "corrects" the output shares of the honest parties to share the correct outputs.

Next, we highlight some of the challenges we face when proving indistinguishability of the simulated and real views. Recall that unlike [IPS08] we run a single consistency test, right before output reconstruction. Thus, we essentially have one "shot" to catch the adversary, causing the test to be more involved. Another challenge is that parties are only committed to small portions of the execution, whereas in [IPS08] parties commit to *all their messages* via the watchlists channels. Consequently, Sim cannot verify correct behavior directly by checking the messages, and instead we need to show that the messages can be extracted from the partial information which parties commit to. Fortunately, we show that correctness can be defined based on the $\mathcal{F}_{\text{RMULT}}$ inputs, and the transcript of the reduction from $\mathcal{F}_{\text{MULT}}$ to $\mathcal{F}_{\text{RMULT}}$. Finally, correctness is guaranteed by the combination of local and global checks in our protocol. Specifically, the consistency test verifies *local* correctness of the computation within each copy, by inspecting a subset of copies; and the degree and equality tests verify that some *global* relation holds over all copies (i.e., all additive shares).

- **Consistency test.** *This test checks that the parties correctly executed the local computations in each copy.*

  $P_1, \ldots, P_m$ obtain from $\mathcal{F}_{\text{COIN}}$ a random subset $\Gamma \subset [n]$ of size δ. For every $l \in \Gamma$, each P_i opens its entire view of the execution of the l'th copy of C. Specifically, P_i decommits X_i^l, and the randomness (including all components of the commitments generated in the initialization step) it used in the execution of the l'th copy. It also opens the commitments to the degree and equality tests, and the additive shares of the final outputs of the l'th copy. Then, P_i checks (as described next) that all local computations in the copies in Γ were performed correctly, aborting if an inconsistency is detected.

  To check the l'th copy, P_i first checks that for every $j \in [m]$, $\sum_{j' \in [m]} \psi_j^{j',l} = 0$. Then, it obtains the l'th column of U_j and $\vec{z}_j$ from the decommitments of P_j, and uses the decommitments to $\mathcal{F}_{\text{RMULT}}$ values to determine the multiplication triples used by all parties for the l'th copy. Using these triples, P_i determines the inputs and outputs each party used in each multiplication gate of the l'th copy. Having determined the outputs of multiplication gates, P_j can reconstruct the l'th column of V_j. Moreover, since the final output is a linear combination of outputs of multiplication gates and parties' inputs, $\sum_j \vec{w}_j$ can be obtained by computing this linear combination over the corresponding rows in $\sum_j U_j$'s and the $\vec{X}_j$'s.

  Since addition gates are evaluated locally, correct execution of addition gates can be verified by checking that the inputs to all multiplication gates were computed correctly. Recall that an input to a multiplication gate is a linear combination of outputs of previous multiplication gates and parties' inputs. Thus, correctness can be checked by verifying that the sum of additive shares used as inputs to multiplication gates by all parties (as inferred from the $\mathcal{F}_{\text{RMULT}}$ triples, and the transcript), and the linear combination of the corresponding rows in $\sum_j U_j$ and the $\vec{X}_j$'s, are equal.

- **Degree test check.** The parties decommit the degree test commitments for all remaining copies $l \notin \Gamma$, namely each P_i opens the commitment to the value q_i computed in Figure 7. (Note that the parties do not decommit the remaining columns of U_i.) Each party computes the vector $q = (q_1 + \ldots + q_m)$ and aborts if q is not a valid L-encoding.

- **Equality test check.** The parties decommit their equality test commitments for all copies $l \notin \Gamma$, namely each P_i opens the commitment to the value $\tilde{q}_i$ computed in Figure 7. Each party computes $\tilde{q} = (\tilde{q}_1 + \ldots + \tilde{q}_m)$, and aborts if either $\tilde{q} \notin \tilde{L}$ or $\tilde{q}$ does not decode to the value 0.

- **Output decommitments.** If the consistency, degree and equality tests pass correctly, then every party P_i decommits its output commitments for all copies $l \notin \Gamma$. The parties then locally reconstruct $\vec{z} = \sum_i \vec{z}_i$, and if it is an L-encoding, decode the output of C from the encoding.

Figure 8: **Actively Secure MPC Φ – Part 3 (Correctness Tests)**.

In the proof, we show that if all the protocol tests pass then except with negligible probability, all the conditions checked by the simulator before the output reconstruction phase hold, and moreover the output is consistent with the outputs of the honest parties, and the effective outputs that Sim extracts for the corrupted parties. Thus, it suffices to prove indistinguishability of the simulated distribution and a hybrid distribution which is obtained from the real execution by performing Sim's checks, and aborting if they are violated. The difference between the hybrid and simulated distributions is that the honest parties use their real inputs in the former, and 0-inputs in the latter. We prove indistinguishability by a case analysis based on which tests pass. Intuitively, the views revealed during the consistency tests are identically distributed due to the secrecy of Shamir's secret sharing scheme (alternatively, Reed-Solomon codes). The degree test values are indistinguishable because the honest parties' values are valid L-encodings, which are uniformly random due to the masking by the $\vec{\gamma}_i$'s. The equality test values are indistinguishable because the sum of honest parties' values are valid $\tilde{L}$-encodings of $\vec{0}$, which are uniformly random subject to this constraint due to the

masking by the $\vec{\widetilde{\gamma}}_i$'s. Since the equality test values are masked by additive shares of $\vec{0}$, the values themselves are identically distributed. Finally, conditioned on all tests passing, the output shares are uniformly random L-encodings whose sum encodes the correct output, due to the masking by the $\vec{\nu}_i$'s.

We proceed to formally prove Theorem 1.

Proof of Theorem 1. We begin by defining the simulator, and then prove indistinguishability of the real and simulated views.

Simulation overview. On a high-level, the simulator Sim will emulate an execution of the protocol with the adversary $\mathcal{A}$ by honestly playing the role of the honest parties with arbitrary inputs (specifically, 0). If all the tests pass, it will extract effective inputs for the adversary, use them to obtain the output from the functionality, and manipulate the output to reveal the correct values.

More formally, let $\mathcal{A}$ be an adversary corrupting a set $\mathcal{T}$ of at most $m-1$ parties. We describe the simulator Sim for $\mathcal{A}$. Sim begins an emulation with $\mathcal{A}$, and emulates $\mathcal{F}_{\text{COM}}, \mathcal{F}_{\text{COIN}}$ and $\mathcal{F}_{\text{RMULT}}$ for the adversary.

1. In the initialization phase, Sim emulates the $\mathcal{F}_{\text{RMULT}}$ and $\mathcal{F}_{\text{COM}}$ oracles for $\mathcal{A}$ while honestly simulating the uncorrupted parties. Then, it obtains from $\mathcal{A}$ the randomness which the corrupted parties send to the commitment oracle, and records these values.

2. In the input sharing step, Sim obtains from the corrupted parties the purported L-encoding which they send to the commitment oracle, as well as the messages they send to the honest parties (i.e., the additive shares), and records these values. Additionally, Sim honestly emulates the honest parties with input 0 to generate the L-encodings of their inputs, and the additive shares of these encodings. Finally, Sim sends the corrupted parties' shares (of the input encodings of the honest parties) to the adversary.

3. In the emulation phase, Sim emulates the computation gate by gate. Notice that messages are only exchanged for multiplication gates. For multiplication gates, the simulator emulates the $\mathcal{F}_{\text{MULT}}$ to $\mathcal{F}_{\text{RMULT}}$ reduction with the adversary, honestly playing the role of the honest parties (with the values computed for them so far).

4. In the output commitment step, Sim records the output commitments sent by the corrupted parties to $\mathcal{F}_{\text{COM}}$. Then, Sim honestly emulates the honest parties and records their re-randomized output shares they would have committed to.

5. In the commit to degree and equality test steps, the simulator emulates the coin tossing oracle for the parties, and sends its output to the adversary. Then, Sim obtains and records the degree and equality test values which the corrupted parties send to the commitment oracle.

6. In the consistency test, the simulator honestly emulates the coin tossing oracle for the parties, and sends $\mathcal{F}_{\text{COIN}}$'s output $\Gamma \subseteq [n], |\Gamma| = \delta$ to the adversary. Then, for every $l \in \Gamma$ it sends to $\mathcal{A}$ the view of the honest parties in the execution of the l'th copy of C. Next, $\mathcal{A}$ decides whether to open the commitments to the corresponding views of the corrupted parties. If $\mathcal{A}$ decides not to open the views, or an inconsistency is detected, the simulator halts and outputs the view generated so far.

7. In the degree test check, Sim sends to $\mathcal{A}$ the values generated by the honest parties for the degree test. Then, $\mathcal{A}$ decides whether to open the commitments to the degree test. If $\mathcal{A}$ decides not to open, or the degree test fails (i.e., the decommitted values do not constitute a valid L-encoding) then Sim halts and outputs the view generated so far.

8. In the equality test check, Sim sends to $\mathcal{A}$ the values generated by the honest parties for the equality test. Then, $\mathcal{A}$ decides whether to open the commitments to the equality test. If $\mathcal{A}$ decides not to open, or the equality test fails (i.e., the decommitted values do not constitute a valid $\tilde{L}$-encoding of 0) then Sim halts and outputs the view generated so far.

9. Let Γ be the set of indices that were opened in the consistency test. Before proceeding to the output decommitment phase, the simulator performs the following tests, aborting with output fail if any of them fail:

   (a) **Consistency of local computations:** There exists a set E of size at most e, such that in all but the copies in E the views of all parties are locally consistent. We say that the view of P_i in a copy l is *locally consistent* if the consistency test passes for copy l when it is chosen in Γ.

   (b) **Consistency with L-encodings:** Let U_i be the matrix that P_i computed and committed to during the degree test. Let $\vec{X}_i, \vec{z}_i$ be the input and output shares committed by party P_i during the input-sharing and output steps (respectively), and let $U = \sum_{i=1}^{m} U_i$ and $\vec{z} = \sum_{i=1}^{m} \vec{z}_i$. Sim checks if $\Delta(U, L^h) \subseteq E$, $\Delta(\vec{X}_i, L) \subseteq E$ for every $i \in [m]$, and $\Delta(\vec{z}, L) \subseteq E$. (See Section 3.7 for the $\Delta(\cdot, \cdot)$ notation.)

   (c) **Consistency of $\mathcal{F}_{\text{RMULT}}$ inputs with commitments:** There exists a set E' of size at most $3e$, such that except for the copies in E', the inputs provided by $\mathcal{F}_{\text{RMULT}}$ to the corrupted parties are consistent with the commitments they made in the initialization step. (We note that the simulator can perform this check because it simulates $\mathcal{F}_{\text{RMULT}}$ for the corrupted parties, whereas the honest parties in a real execution cannot perform this check since the messages exchanged with $\mathcal{F}_{\text{RMULT}}$ cannot be monitored in the real world.)

   (d) **Input extraction:** For every $i \in \mathcal{T}$, Sim can extract an effective input for P_i. Specifically, let $\vec{X}_i = \left(X_i^1, \ldots, X_i^n\right)$ denote the purported L-encoding to which P_i committed to in Step 2 of the simulation, and notice that $\vec{X}_i$ is $|E| \leq e$ close to a valid L-encoding (because the tests in Steps 9b and 9a above passed). Since $n - 4e > k$, there exists a unique $\hat{\vec{X}}_i \in L$ that agrees with $\vec{X}_i$ on all columns except those in $E \cup E'$ (and $\hat{\vec{X}}_i$ can be efficiently computed given E, E' and $\vec{X}_i$). Sim computes $\hat{\vec{X}}_i$, and decodes it to obtain x_i^*.

   (e) **Correctness of degree-reduction:** The simulator verifies that the degree reduction step following each multiplication step was performed correctly. In more detail, the simulator reconstructs V_i for every party P_i as in the equality test (for corrupted parties P_i, V_i can be inferred from the transcript and the $\mathcal{F}_{\text{RMULT}}$ inputs committed to in the initialization step). Then, Sim decodes each row in U and $V = \sum_{i=1}^{m} V_i$ by computing the unique $\hat{U} \in L^h$ and $\hat{V} \in \tilde{L}^h$ that agree with U, V (Respectively) in all columns except those in $E, E \cup E'$ (respectively), and decoding $\hat{U}, \hat{V}$. Next, Sim verifies that the decoded vectors are equal. We now use the fact that the previous tests have passed, to argue that decoding succeeds and is unique. Specifically, since Steps 9a and 9b passed we have

$$|\Delta(U, L^h)| \leq |E| \leq e < (n - k + 1)/3.$$

In particular, $n-e > k$ so $\hat{U}$ is unique. As for V, the fact that Steps 9a and 9c passed implies that all multiplications were correctly computed by all parties in all copies excluding $E \cup E'$. Using also the fact that $n - 4e > 2k$ (so $n - 4e$ columns of V uniquely determine the encoded value), $\hat{V}$ is unique. Following a similar decoding procedure, Sim decodes $\vec{w} = \sum_j \vec{w}_j$ and $\vec{z} = \sum_j \vec{z}_j$ w.r.t L by excluding the columns in E, constructing L-encodings $\hat{\vec{w}}$ and $\hat{\vec{z}}$ and checking if the decoded values are equal.

10. In the output decommitment phase, for all honest parties but one, denoted by P_{i^*}, Sim reveals the output values computed for these parties during the simulation. For P_{i^*} it proceeds as follows. Since the simulator did not abort in Step 9, an effective input x_i^* was determined in Step 9d of the simulation for every $i \in \mathcal{T}$. Sim provides $\{x_i^*\}_{i \in \mathcal{T}}$ to the trusted party and obtains the output y. Next, it extracts the value o encoded in the output obtained in the simulation. This can be done in the same way as the rows of U are decoded (because Steps 9a and 9b passed). Sim computes an L-encoding $\vec{o}$ of the value $y - o$ in which the entries corresponding to the columns in $\Gamma \cup E \cup E'$ are 0 (this is possible because $k > \delta + 4e$), and adds $\vec{o}$ to the output shares of P_{i^*} before revealing them.

Indistinguishability. Next, we prove indistinguishability of the real and simulated views. Intuitively, the secrecy of the encoding and secret sharing scheme guarantees that the adversary learns nothing beyond the output. (Recall that the encoding we use can be thought of as Shamir's secret sharing scheme, so it guarantees secrecy.) The consistency, degree and equality tests further ensure, with overwhelming probability, the correct behaviour of the adversary.

We proceed to formally prove indistinguishability through a sequence of hybrids, where in each hybrid we output the view of the adversary.

$\mathcal{H}_0 =$ **REAL:** This is the real-world adversarial view.

We first prove that there exists a set of effective inputs for the corrupted parties such that with high probability either one of the tests in the protocol execution fails, or the output is consistent with the output of $\mathcal{F}$ on the inputs of the honest parties, and the effective inputs for the corrupted parties. Thus, if all the tests pass then with overwhelming probability the computation was correct. Formally,

Claim 4.1. *Suppose $e < (n - k + 1)/3$. Then, except with probability*

$$(1 - e/n)^{\delta} + \frac{n - k + 2}{|\mathbb{F}|} + 2^{-\Omega(e)}$$

either one of the degree, equality or consistency test fails, or all *of the following holds:*

(a) There exists a set E of size at most e such that all parties correctly performed the local computations in all copies except those in E.

(b) $\Delta(U, L^h), \Delta(\vec{X}_i, L), \Delta(\vec{\nu}, L) \subseteq E$, where $U = \sum_i U_i$ and U_i is the matrix which P_i committed to during the degree test; and $\vec{X}_i, \vec{\nu}_i$ are the input and output-masking shares committed by party P_i during the input-sharing and initialization steps (respectively).

(c) There exists a set E' of size at most $3e$ such that except for the copies in E', the initial commitments of all parties (during the initialization step) are consistent with the values they received from $\mathcal{F}_{\text{RMULT}}$.

(d) There exist unique $\hat{U} \in L^h$, $\hat{V} \in \tilde{L}^h$, $\vec{\hat{X}}_i \in L$ for all i, $\vec{\hat{w}}_i \in L$ and $\vec{\hat{\nu}}_i \in L$ that agree with $U = \sum_i U_i$, $V = \sum_i V_i$, $\vec{X}_i$, $\sum_i \vec{w}_i$ and $\vec{\nu}_i$, respectively, on all columns except $E \cup E'$. Moreover, $\hat{U}$ and $\hat{V}$ decode to the same value w.r.t L and $\tilde{L}$ respectively, and $\vec{\hat{\nu}}$ decodes to 0. Furthermore, the output reconstructed in $\mathcal{H}_0$ is $\mathcal{F}(\vec{x}^*_{\mathcal{T}}, \vec{x}_{\overline{\mathcal{T}}})$, where $\vec{x}_{\overline{\mathcal{T}}}$ denotes the inputs of the honest parties, and $\vec{x}^*_{\mathcal{T}}$ are effective inputs for the corrupted parties, encoded in the $\vec{\hat{X}}_i$'s (i.e., following the procedure described in Step 9d of the simulation.)

Proof: We identify a set of bad events, bound their probabilities, and conclude using a union bound.

- **Event $|E| > e$.** Then except with probability $(1 - e/n)^{\delta}$, the consistency test fails because it checks at least one of the inconsistent copies.

For the next event, we define the matrix

$$M = \begin{bmatrix} U \\ \vec{X}_1 \\ \vdots \\ \vec{X}_m \\ \sum_i \vec{\nu}_i \end{bmatrix}.$$

- **Event $|E| \leq e$, and $d(M, L^{h+m+1}) > e$.** Since $e < (n-k+1)/3$ then [AHIV17, Lemma 4.4][3] guarantees that

$$\Pr_r \left[d(\vec{r}^T U + \sum_{i=1}^{m} r'_i \vec{X}_i + r'' \sum_{i=1}^{m} \vec{\nu}_i, L) \leq e \right] \leq (n-k+1)/|\mathbb{F}|.$$

  Furthermore, whenever $d(\vec{r}^T U + \sum_{i=1}^{m} r'_i \vec{X}_i + r'' \sum_{i=1}^{m} \vec{\nu}_i, L) > e$ and $|E| \leq e$, then the degree test fails (because the degree test was correctly computed in all copies not in E.) Overall, the degree test fails except with probability $(n-k+1)/|\mathbb{F}|$.

- **Event $|E| \leq e$, $d(M, L^{h+m+1}) \leq e$ and $\emptyset \neq \Delta(M, L^{h+m+1}) \not\subseteq E$.** We show that in this case the degree test fails except with probability $1/|\mathbb{F}|$. Since $d(M, L^{h+m+1}) \leq e$, $|E| \leq e$ and $n - 2e > k$, then there exists a unique $\hat{M} \in L^{h+m+1}$ that agrees with M on all columns except those in $\Delta(M, L^{h+m+1}) \cup E$. Moreover, since the degree test was computed correctly in all columns excluding E (and therefore excluding $E \cup \Delta(M, L^{h+m+1})$) then for every column $l \notin E \cup \Delta(M, L^{h+m+1})$, the restriction of the linear combination computed during the degree test to column l is equal to this linear combination applied to the l'th column of $\hat{M}$. Let $\vec{v}$ denote the word constructed during the degree test (i.e., obtained through the aforementioned linear combination), and let $\vec{v}' \in L$ denote the word that would have been constructed had the degree test been applied to $\hat{M}$. Then the degree test passes only if $\vec{v} \in L$, and since $d(\vec{v}, \vec{v}') \leq 2e$ and $n - 2e > k$, then $\vec{v} = \vec{v}'$. Therefore, if there exist $j \in [h+m+1], l \in [n]$ such that the j'th row $\vec{w}$ of M has $l \in \Delta(\vec{w}, L) \setminus E$, then the degree test was performed correctly for the l'th copy (i.e., column), and moreover the linear combination it computed on the l'th column was consistent

[3] Lemma 4.4 in [AHIV17] was stated conditioned on a conjecture (Conjecture 4.1), but was later proven to hold unconditionally (i.e., irrespective of the conjecture); we are grateful to the authors of [AHIV17] for sharing their proof with us and letting us include it in Appendix B.

with this linear combination over the l'th column of $\hat{M}$ (i.e., $v_l = v'_l$). This happens only with probability $1/|\mathbb{F}|$ (because for every $\vec{r} = (r_1, \ldots, r_{j-1}, r_{j+1}, \ldots, r_n)$ chosen by $\mathcal{F}_{\text{COIN}}$ for the degree test, there is at most one value of r_j for which this happens).

- **Event $|E| \leq e$ and $|E'| > 3e$.** For every copy $l \in E'$, if $l \in \Gamma$ then the consistency test passes only if $\mathcal{A}$ successfully performed a selective attack on the $\mathcal{F}_{\text{MULT}}$ to $\mathcal{F}_{\text{RMULT}}$ reduction, which happens with probability at most $1/|\mathbb{F}|$. Moreover, an unsuccessful selective attack in copy l causes a local inconsistency in that copy, so $|E| \leq e$ implies that the selective attack failed in at most e copies. However, using a Chernoff-Hoeffding bound, if $|E'| > 3e$ then the probability that the selective attacks failed in at most e copies is at most

$$2^{-2(3e)(\frac{2e-3e/|\mathbb{F}|-1}{3e})^2} = 2^{-\Omega(e)}.$$ [4]

We now assume that $|E| \leq e, \Delta\left(M, L^{h+m+1}\right) \subseteq E$, and $|E'| \leq 3e$, and show that $U, V = \sum_i V_i, \vec{X}_i, \vec{\nu}_i$ can be uniquely "corrected" to valid encodings. More formally, Since $|E| \leq e$, $\Delta(U, L^h) \subseteq E$ and $n - e > k$, there exists a unique matrix $\hat{U} \in L^h$ that agrees with U on all columns except those in E. Similarly, there are unique L-encodings $\vec{\hat{X}}_i$ and $\vec{\hat{\nu}}_i$ that agree with $\vec{X}_i$ and $\vec{\nu}_i$ on all columns excluding those in E. Since multiplications were correctly performed in every column except those in $E \cup E'$, and $n - 4e > 2k$, then there exists a unique matrix $\hat{V} \in \tilde{L}^h$ that agrees with $V = \sum_i V_i$ on all columns except those in $E \cup E'$.

Now, since the equality test was performed correctly in all columns excluding those in $E \cup E'$ (in particular, $\sum_j \vec{\psi}_i^{j,l} = \vec{0}$ for every $l \notin E \cup E'$, so the $\vec{b}$ element in the equality test does not affect the encoded value) then for every $l \notin E \cup E'$, the restriction to the l'th column of the linear combination computed during the equality test is equal to this linear combination computed over the l'th columns of $\hat{U}, \hat{V}$, and $\sum_i \vec{\hat{\nu}}_i$. Therefore, the soundness of the equality test guarantees that except with probability $1/|\mathbb{F}|$, if the equality test passes then the values decoded in each row of $\hat{U}$ and $\hat{V}$ are equal, and the value decoded from $\vec{\hat{\nu}}_i$ is 0. Since addition gates were correctly executed on all columns except those in E, it follows that in all columns excluding those in $E \cup E'$ the entire computation was computed correctly according to the inputs decoded from $\vec{\hat{X}}_i$. Consequently, since $n - |E \cup E'| \geq n - 4e > k$, the output reconstructed in $\mathcal{H}_0$ is $\mathcal{F}(\vec{x}^*_{\mathcal{T}}, \vec{x}_{\overline{\mathcal{T}}})$, where $\vec{x}_{\overline{\mathcal{T}}}$ denotes the inputs of the honest parties, and $\vec{x}^*_{\mathcal{T}}$ are effective inputs for the corrupted parties obtained by decoding $\vec{\hat{X}}_{\mathcal{T}}$.

Using a union bound, we can now conclude that, except with probability

$$\left(1 - \frac{e}{n}\right)^{\delta} + \frac{n - k + 2}{|\mathbb{F}|} + 2^{-\Omega(e)}$$

either all the tests pass, or the four conditions in the claim hold. ■

$\mathcal{H}_1$**:** In $\mathcal{H}_1$, we check the conditions of Step 9 of the simulation above before the output decommitment phase, and abort with output fail if any of them fail. Observe that conditioned on any of the degree, equality, or consistency tests failing, the two hybrids are identically distributed as they proceed identically. Therefore, it suffices to bound the statistical distance of the hybrids when all the tests pass.

[4] Chernoff-Hoeffding bound states that for independent random variables $X_1, \ldots, X_n \in [0, 1]$ and $\overline{X} = \frac{1}{n}\sum_i X_i$,

$$\Pr[|\overline{X} - E[\overline{X}]| > t] \leq e^{-2nt^2}.$$

Conditioned on the tests passing, the two hybrids differ only if one of the tests of Step 9 fails, which we now show happens only with negligible probability. By Claim 4.1, except with probability

$$\left(1-\frac{e}{n}\right)^{\delta}+\frac{n-k+2}{|\mathbb{F}|}+2^{-\Omega(e)}$$

all the conditions of the claim hold (since the tests pass). In particular, since conditions (a)- (c) of Claim 4.1 hold then Steps 9a-9c hold (respectively). For the test of Step 9b, we also use the fact that the output encoding $\vec{z}=\sum_i \vec{z}_i$ satisfies $\vec{z}=\vec{w}+\vec{\nu}$, where $\vec{w}=\sum_i \vec{w}_i$ is the sum of rows of U and the $\vec{X}_i$'s (and therefore differs from a valid encoding only in the columns in E). Moreover, since condition (d) holds then the tests in Steps 9d and 9e pass (here, we also use the fact that conditions (a) and (c) hold, so $|E\cup E'|\leq 4e$, and that $n-4e>2k$). Therefore, the two hybrids are statistically close.

$\mathcal{H}_2$**:** In this hybrid, we switch to the **IDEAL** experiment. Notice that this is equivalent to the following experiment: first, run the protocol with $\mathcal{A}$, where the honest parties use 0 instead of their inputs. Second, before decommitting the outputs, check whether the conditions of Step 9 of the simulation hold, and abort if not. Finally, reveal the output, but manipulate the shares of one honest party as the simulator does in Step 10 of the simulation to reveal the correct output.

We prove indistinguishability of $\mathcal{H}_1$ and $\mathcal{H}_2$ by considering the following mutually disjoint cases. Let hyb_1 and hyb_2 denote the random variables representing the outputs of hybrid experiments $\mathcal{H}_1$ and $\mathcal{H}_2$, respectively.

C: Consistency test fails: The view of the adversary until (including) the consistency test contains: (1) messages from $\mathcal{F}_{\text{RMULT}}$; (2) messages exchanged during the $\mathcal{F}_{\text{MULT}}$ to $\mathcal{F}_{\text{RMULT}}$ reduction (computed for each multiplication gate across all copies); (3) input shares obtained from the honest parties; and (4) the decommitments made during the consistency test. We analyze each of these separately, and show they are identically distributed in both hybrids.

(1) and (2): messages from $\mathcal{F}_{\text{RMULT}}$, and the transcripts of the $\mathcal{F}_{\text{MULT}}$ to $\mathcal{F}_{\text{RMULT}}$ reductions, are identically distributed in both hybrids as $\mathcal{A}$ only learns additive shares of the outputs.

(3): the input shares of the honest parties are similarly uniformly random in both hybrids.

(4): the values decommitted by P_i during the consistency test are the restriction to Γ of: his input encoding $\vec{X}_i$, the matrix U_i, the values provided by $\mathcal{F}_{\text{RMULT}}$ (during the initialization step), the randomness $\vec{\gamma}_i, \vec{\tilde{\gamma}}_i, \vec{\nu}_i, \left(\vec{\psi}_i^j\right)_{j\in[m]}$, the degree and equality test values $q_i, \tilde{q}_i$, and the output shares $\vec{z}_i$. The entries of the inputs $\vec{X}_i$ of honest parties P_i revealed during the consistency test are uniformly random, because only $|\Gamma|<k$ entries are decommitted, and $\vec{X}_i$ is a random valid L-encoding. Similarly, the columns of U_i decommitted during the consistency test are also distributed uniformly at random, since each row of U_i is a fresh L-encoding sampled by P_i, and only $|\Gamma|$ columns are revealed. The values provided by $\mathcal{R}_{\text{RMULT}}$ are generated in the same way in both hybrids and are therefore identically distributed. As for the degree test equality test outcomes of honest parties P_i for copies $l\in\Gamma$, they are uniquely determined by X_i^l, the l'th columns of U_i, V_i, and the l'th entries of $\vec{\nu}_i, \vec{\gamma}_i, \vec{\tilde{\gamma}}_i, \vec{b}_i$ (and the coin tosses of $\mathcal{F}_{\text{COIN}}$). Notice that the l'th column of V_i is uniquely determined by the values decommitted by P_i during the consistency check, and b_i^l is uniquely determined by the values decommitted by all parties for the l'th copy. Moreover, $\vec{\gamma}_i, \vec{\tilde{\gamma}}_i$ are uniformly sampled L- and $\tilde{L}$-encodings (respectively) in both

hybrids, and $\vec{\nu}_i$ is a random valid L-encoding in both hybrids, so the restriction of $\vec{\gamma}_i, \vec{\tilde{\gamma}}_i$, and $\vec{\nu}_i$ to $|\Gamma|$ copies are uniformly random, and consequently the degree and equality test outcomes are identically distributed in both hybrids. Finally, the decommitted output shares of honest parties P_i for columns in Γ are uniquely determined by the restriction of $U_i, \vec{\nu}_i$ to the columns in Γ, and the input shares $\left\{x^l_{j,i}\right\}_{j\in[m],l\in\Gamma}$ (because $\vec{z}_i = \vec{w}_i + \vec{\nu}_i$, where $\vec{w}_i$ is a sum over the rows of U_i and the input shares provided to P_i by all parties). Since U_i and the input shares are identically distributed in both hybrids, and (as noted above) the restriction of $\vec{\nu}_i$ to the columns in Γ is uniformly random in both hybrids, the decommitted output shares are identically distributed in both hybrids.

Therefore, conditioned on the consistency test failing, the hybrids are identically distributed. Moreover, the probability that the consistency test fails is independent of the honest parties' inputs. Indeed, whether or not the test fails depends only on the view of the adversary, which at the end of the consistency test includes the values computed during the $\mathcal{F}_{\text{MULT}}$ to $\mathcal{F}_{\text{RMULT}}$ reductions (thorough this reduction, the adversary learns only subsets of additive shares, which are independent of the inputs of the honest parties), and the views of all honest parties in $|\Gamma|$ copies (which are independent of the inputs of honest parties due to the secrecy parameter of the L-encodings, alternatively Shamir's secret sharing scheme).

D: Degree test fails: In this case the consistency test passes, so the degree test outcomes q_i are revealed by all parties P_i. Recall from case C above that the adversarial views up to (including) the consistency test are identically distributed in both hybrids. We claim that conditioned on the consistency test passing, the degree test outcomes are L-encodings that are distributed uniformly at random subject to the constraint that they are consistent with the values revealed during the consistency test. Indeed, for every honest party P_i, $\vec{r}^T U_i + r'_i \vec{X}_i + r'' \vec{\nu}_i \in L$, and this value is blinded by a uniformly random L-encoding $\vec{\gamma}_i$.

Therefore, conditioned on the degree test failing, the hybrids are identically distributed. Again, the probability that the degree test fails is independent of the honest parties' inputs, as it only depends on the q_i supplied by the corrupted parties for the degree test outcome.

EQ: Equality test fails: First, by the analysis in cases C and D above, the two hybrids are identically distributed up to (including) the opening of the degree test commitments. Conditioned on the consistency and degree tests passing, the proof of Claim 4.1 shows that conditions (a)-(c) of the claim hold except with probability $(1 - e/n)^\delta + \frac{n-k+2}{|\mathbb{F}|} + 2^{-\Omega(e)}$. Therefore, $|E \cup E'| \leq 4e$, and there exists a unique $\tilde{L}$-encoding $\vec{\tilde{q}}$ that agrees with $\sum \tilde{q}_i$ on all columns except those in $E \cup E'$.

Consequently, to show that the two hybrids are statistically close conditioned on EQ, it suffices to argue that they are identically distributed conditioned on EQ and conditions (a)-(c) of Claim 4.1. Let W be the set of executions where both EQ and these conditions hold. Then

$$\left|\Gamma \cup E \cup E'\right| \leq \delta + 4e < k$$

for all executions in W, so similar arguments to case C above show that the adversarial view up to (including) the degree test check, together with the view of the honest parties in the executions of all copies in $\Gamma \cup E \cup E'$, are identically distributed in both hybrids.

For an execution in W, let view denote the view of the adversary up to (including) the degree test check, together with the view of the honest parties in the executions of all copies in $\Gamma \cup E \cup E'$.

Let Q_i and Q'_i be the distribution of the equality test outcome $\vec{\tilde{q}}_i$ revealed by party P_i in hybrids $\mathcal{H}_1$ and $\mathcal{H}_2$, respectively, conditioned on the partial view view. Recall that

$$\sum_i \vec{\tilde{q}}_i = \sum_i \vec{\alpha}^T(V_i - U_i) + \beta\vec{\nu}_i + \vec{\tilde{\gamma}}_i + \vec{b}_i$$

where $\vec{b}_i = \sum_{j\in[m]} \vec{\psi}^i_j$, and $\sum_{i\in[m]} \vec{\psi}^j_i = \vec{0}$. We want to prove that $(Q_i)_{i\in[m]}$ and $(Q'_i)_{i\in[m]}$ are identically distributed conditioned on view, but since the Q_i's and Q'_i's are blinded by the additive $\vec{0}$-sharings $\vec{\psi}_j$'s of all P_j's, this will follow from proving that $\sum_{i\notin\mathcal{T}} Q_i$ and $\sum_{i\notin\mathcal{T}} Q'_i$ are identically distributed conditioned on view. We proceed to prove the latter claim.

Recall that there exist unique $\hat{V} \in \tilde{L}^h, \hat{U} \in L^h, \vec{\hat{\nu}} \in L, \vec{\hat{\tilde{\gamma}}} \in L$ that agree with $\sum_i V_i, \sum_i U_i, \sum_i \vec{\nu}_i, \sum_i \vec{\tilde{\gamma}}_i$ (respectively) except for the columns in $E \cup E'$. Therefore, conditioned on view then $\sum_i Q_i - \sum_i Q'_i$ are valid $\tilde{L}$-encoding of 0 in which the entries in $\Gamma \cup E \cup E'$ are all 0. Moreover, the equality test values $\vec{\tilde{q}}_i$ of corrupted P_i's are committed to before the consistency test, so they depend on the adversarial view up to that point, which we have already shown is identically distributed. Therefore, $(Q_i)_{i\in\mathcal{T}}$ and $(Q'_i)_{i\in\mathcal{T}}$ are identically distributed point distributions (since they are conditioned on view). Therefore, conditioned on view, $\sum_{i\in\bar{\mathcal{T}}} Q_i - \sum_{i\in\bar{\mathcal{T}}} Q'_i$ are valid $\tilde{L}$-encoding of 0 in which the entries in $\Gamma \cup E \cup E'$ are all 0. Since $\vec{\tilde{\gamma}}_i$ are random valid $\tilde{L}$-encodings of 0, this implies that $\sum_{i\in\bar{\mathcal{T}}} Q_i$ and $\sum_{i\in\bar{\mathcal{T}}} Q'_i$ are identically distributed conditioned on view, as we set out to prove. We conclude that the hybrids are statistically close conditioned on the equality test failing.

O: All tests pass: By Claim 4.1, conditioned on all tests passing then except with probability $(1-e/n)^\delta + \frac{n-k+2}{|\mathbb{F}|} + 2^{-\Omega(e)}$ the final outputs $\sum_i \vec{z}_i$ are valid L-encoding of $y = \mathcal{F}(\vec{x}_{\bar{\mathcal{T}}}, \vec{x}^*_{\mathcal{T}})$ (in $\mathcal{H}_1$) and $o = \mathcal{F}(\vec{0}_{\bar{\mathcal{T}}}, \vec{x}^*_{\mathcal{T}})$ (in $\mathcal{H}_2$), where $\vec{x}^*_{\mathcal{T}}$ are the effective inputs extracted for the corrupted parties in these hybrids. Thus, it suffices to prove indistinguishability of the hybrids conditioned on O and all the conditions of Claim 4.1. The remainder of the argument is similar to that of case EQ above. Let view denote the view of the adversary up to (including) the equality test check, together with the view of the honest parties in the executions of all copies in $\Gamma \cup E \cup E'$. Let Z_i and Z'_i be the distributions of the outputs $\vec{z}_i$ of party P_i in hybrids $\mathcal{H}_1, \mathcal{H}_2$ (respectively), conditioned on view. (Notice that in $\mathcal{H}_2$ these shares are manipulated before they are revealed.) Then $\sum_i Z_i - \sum_i Z'_i$ is distributed as a random L-encoding of $y - o$ conditioned on the entries in $\Gamma \cup E \cup E'$ being 0, because the shares are blinded using the random L-encodings $\vec{\nu}_i$ of 0. Therefore, the two hybrids are identically distributed conditioned on view, because the output manipulation exactly adds a random L-encoding of $y - o$ to the output shares.

This concludes the proof of indistinguishability of the last two hybrids, and the proof of security. ■

Communication complexity of protocol Φ. Assuming the existence of a PRG, parties can commit to their $\mathcal{F}_{\text{RMULT}}$ triples by committing (during the initialization step) to a PRG seed for each copy (the other initialization-phase commitments are generated as in Figure 6). Consequently, the total communication,

assuming rate-1 commitments, is:

$$\underbrace{n \cdot m \cdot (\kappa + (3+m) \cdot \log_2 |\mathbb{F}|)}_{\text{rnd/blind com.}} + \underbrace{m \cdot n \cdot \log_2 |\mathbb{F}|}_{\text{input commitments}} + \underbrace{m^2 \cdot n \cdot \log_2 |\mathbb{F}|}_{\text{input sharing}}$$
$$+ \underbrace{n \cdot h \cdot \mathrm{CC}_{\mathrm{MULT}}}_{\text{multiplication}} + \underbrace{|\Gamma| \cdot m \cdot (\kappa + (4+m) \cdot \log_2 |\mathbb{F}|)}_{\text{consistency test}}$$
$$+ \underbrace{2 \cdot m \cdot n \cdot \log_2 |\mathbb{F}|}_{\text{degree test com. and dec.}} + \underbrace{2 \cdot m \cdot n \cdot \log_2 |\mathbb{F}|}_{\text{equality test com. and dec.}} + \underbrace{2 \cdot n \cdot m \cdot \log_2 |\mathbb{F}|}_{\text{output com. and dec.}}$$

where $\mathrm{CC}_{\mathrm{MULT}}$ is the communication complexity of the m-party multiplication protocol (implementing $\mathcal{F}_{\mathrm{RMULT}}$ and the $\mathcal{F}_{\mathrm{MULT}}$ to $\mathcal{F}_{\mathrm{RMULT}}$ reduction), and h is the number of multiplication gates in the circuit. (We note that the degree and equality test commitments revealed during the consistency test are counted as part of the degree and equality test terms, resp.) In order to get $2^{-\Omega(s)}$ soundness, we need to set $n = O(s)$. Assuming $s \leq \kappa$, the overall communication complexity can be bounded by $O(s \cdot h \cdot \mathrm{CC}_{\mathrm{MULT}}) + \mathsf{poly}(m, \kappa, \log_2 |\mathbb{F}|)$. Since h represents the size of the circuit (i.e. number of multiplication gates), the best passive protocol in the $\mathcal{F}_{\mathrm{MULT}}$-hybrid can be bounded by $O(h) \cdot \mathrm{CC}_{\mathrm{MULT}}$. Therefore, the communication overhead of our basic variant is $O(s)$.

4.1 Instantiating $\mathcal{F}_{\mathrm{RMULT}}$

Recall from Section 3.5 (Figure 5) that $\mathcal{F}_{\mathrm{RMULT}}$ is the multiplication functionality that outputs three tuples of additive shares $\vec{a}, \vec{b}, \vec{c}$ such that the "inputs" $\vec{a}, \vec{b}$ share random values a, b, and the "output" $\vec{c}$ shares the product $a \cdot b$. In this section we discuss how to realize this functionality, while identifying the minimal security properties required from it.

Our first observation is that we do not need an actively-secure implementation of the $\mathcal{F}_{\mathrm{RMULT}}$ functionality. In fact, it suffices to consider a protocol that is only "private" against active adversaries, in the sense that throughout the protocol execution, an actively corrupted party cannot violate the privacy of the honest parties' inputs. In particular, the underlying implementation does not have to retain correctness in this case, or provide a mechanism for extracting the adversary's inputs. Extraction in our protocol is achieved by requiring the adversary to commit to its input and randomness used for the $\mathcal{F}_{\mathrm{RMULT}}$-functionality. Correctness, on the other hand, is enforced through our consistency test that ensures correctness of the computations in most of the copies, by checking a random subset of δ copies.

When computing a boolean circuit, the pairwise products of the shares can be computed using Oblivious Transfer (OT) [Bea91, NNOB12]. Based on the discussion above, it suffices to use a private OT protocol [HK12]. Indeed, consistency between the different OT executions will be verified during the consistency test of our protocol, as discussed above.[5] Intuitively, privacy is guaranteed because an OT sender has no output in the execution, and the security/privacy of OT ensures that even if the sender cheats it learns nothing about the receiver's input. Moreover, though an OT receiver can use inconsistent inputs in the OT executions with different honest parties, this can only violate correctness, and not privacy, since the output of each OT execution is an additive share of the cross product (e.g., $a_i \cdot b_j$), which reveals nothing about the other party's share. Similarly, when working over large fields, $\mathcal{F}_{\mathrm{RMULT}}$ can be realized using private OLE (cf. Section 3.3), where private OLE can be defined analogously to private OT, requiring that parties do not infer any additional information (except what can be inferred from their inputs and outputs).

[5] More specifically, we use OT between pairs of parties to compute a 2-out-of-2 additive secret sharing of the product they should compute. Then. we perform the consistency check, and reconstruct the outputs of OTs only if this test passes.

Relaxing to passive implementation of the $\mathcal{F}_{\text{RMULT}}$-functionality. We can further weaken the security requirement on the $\mathcal{F}_{\text{RMULT}}$ implementation, by incorporating the reduction from defensible privacy to passive security. We first informally review the notion of *defensible privacy* which was introduced by Haitner in [Hai08, HIK+11]; see [HIK+11] for the formal definitions. Let π be a two-party protocol between P_1 and P_2, and let $\mathsf{trans} = (q_1, a_1, \ldots, q_\ell, a_\ell)$ be a transcript of an execution of π when P_1 is controlled by an adversary $\mathcal{A}$, where q_i denotes the i'th message from P_1, and a_i denotes the i'th message from P_2 (that is, a_i is the response to q_i). Informally, a *defence* of $\mathcal{A}$ relative to trans, which is provided after the protocol execution ends, is a pair (x, r) of input and random tape for P_1. We say that a defence (x, r) of $\mathcal{A}$ relative to trans is *good* if the transcript generated by running the honest P_1 with input x and random tape r against P_2's messages $a_1, \ldots, a_\ell$ results exactly in trans. Intuitively, a defense (x, r) is good relative to trans if, whenever P_i uses (x, r) in an honest execution of π, the messages sent by P_i are identical to the messages sent by the adversary in trans. Thus, in essence, a defense serves as a "proof" of honest behavior. Defensible privacy guarantees that when the adversary provides a good defence, then it learns nothing beyond what can be inferred from its input and prescribed output.[6] We note that in our protocols, the consistency tests guarantee that (if the tests pass) the adversary has a good defence, so defensible privacy implies privacy against active corruptions.

The security of a passive protocol can be amplified to defensible privacy by adding a coin tossing phase (which, in our case, samples the inputs to $\mathcal{F}_{\text{RMULT}}$), and ensuring that these coins were indeed used in the passive execution. The latter can be checked as part of our consistency test, however to guarantee privacy we cannot postpone this check until the consistency test is performed at the end of the circuit emulation, since by that time the adversary could have already violated privacy by using badly-sampled randomness. Thus, we include in our protocol two consistency tests: the first is the consistency test described in Figure 7, and the second checks consistency of $\mathcal{F}_{\text{RMULT}}$ inputs and the tossed coins, and is executed during the initialization phase. This second consistency test ensures that with overwhelming probability, all but (possibly) a small subset of random triples are correct (namely, the aggregated parties' shares correspond to $c = a \cdot b$), and consistent with the random coins. This will suffice for our security analysis, because the number of copies will be sufficiently large such that by cheating in a small number ($< k$) of copies, the adversary learns nothing. See Appendix A for further details and the concrete overhead our protocols achieve. In summary, we obtain the following from Theorem 1 and the discussion above:

Corollary 2. *Let ϕ be a passively-secure protocol realizing $\mathcal{F}_{\text{RMULT}}$. Then Protocol Φ described in Figures 6-8, when the $\mathcal{F}_{\text{RMULT}}$ oracle is replaced with ϕ, securely realizes $\mathcal{F}$ in the $(\mathcal{F}_{\text{COM}}, \mathcal{F}_{\text{COIN}})$-hybrid model, tolerating $m - 1$ active (static) corruptions, with statistical security error*

$$(1 - e/n)^{\delta} + \frac{n - k + 2}{|\mathbb{F}|} + 2^{-\Omega(e)} + (1 - e/(n + \delta'))^{\delta'}$$

where $k > \delta + 4e, n > 2k + 4e$, $e \leq (n - k + 1)/3$, and $\delta' > 0$.

Relaxing further to weaker than passive. Following ideas from [HIMV19], our protocol can, in fact, tolerate an *imperfect* passive OLE, namely one that has a non-negligible statistical privacy or correctness error. This security feature can be turned into an efficiency advantage. For example, imperfect $\mathcal{F}_{\text{RMULT}}$ can be implemented more efficiently by aggressively setting the parameters in existing LWE-based OLE constructions, see Section 7.1 for details.

[6] For instance, an OT protocol is defensibly private with respect to a corrupted sender if any adversary interacting with an honest receiver with input u, and providing a good defence at the end of the execution, does not learn u. Similarly, an OT protocol is defensibly private with respect to a corrupted receiver if for any input u, and any inputs (v_0, v_1) for the sender, any adversary interacting with the honest sender with input (v_0, v_1), that is able to provide a good defense for input u, does not learn v_{1-u}.

5 Actively Secure MPC with Constant Communication Overhead

In this section we present our main result, namely, an MPC protocol for an arbitrary number of parties that achieves constant communication overhead over the passive GMW protocol.

On a high-level, we will incorporate a variant of the protocol of Frankling and Yung [FY92] instead of [BGW88] in our basic MPC protocol. Recall that the main overhead in the basic MPC protocol is caused by the $n = O(s)$ copies of the circuit used to perform the computation, where s is a statistical security parameter. Then, similar to [FY92] we improve the communication overhead, achieving *constant* overhead, by having all copies evaluate multiple gates in parallel using packed secret sharing. Our protocol will achieve constant-overhead for moderately wide circuits (See Section 6 for a more detailed discussion.)

In more detail, given a circuit C, and block-width parameter $\mathcal{B}$, the parties agree on a divisions of the circuit evaluation into layers, where at most $\mathcal{B}$ multiplication gates are evaluated in parallel in each layer, and arbitrary linear operations are performed between layers. During the evaluation of the protocol on a specific input, we can associate with each layer of gates G a vector (block) B_O^G of $\mathcal{B}$ values whose i'th position contains the output value assigned to the i'th gate in the layer (or 0 if the block has less than $\mathcal{B}$ gates). For each layer (except blocks of input gates), we will associate two additional blocks: a "left" block B_L^G and "right" block B_R^G whose i'th position contains the value of the left input wire and right input wire of the i'th gate, respectively. In other words, the value of the i'th gate of a multiplication block can be expressed as $(B_O^G)_i = (B_L^G)_i (B_R^G)_i$. In the protocol, the parties will collectively operate on an efficient (constant-rate) Reed-Solomon encoding (equivalently, packed secret shares) of each block. The protocol parameters include a description of the Reed-Solomon code $L = \mathsf{RS}_{\mathbb{F},n,k,\eta}$, and a vector of elements $\zeta = (\zeta_1, \ldots, \zeta_{\mathcal{B}}) \in \mathbb{F}^{\mathcal{B}}$ which is used for decoding.

Next, we describe our protocol, simulation and proof by specifying the main differences from the description of the basic protocol from Section 4.

- INITIALIZATION. Recall that each party generates $\vec{\gamma}_i, \vec{\nu}_i, \vec{\tilde{\gamma}}_i$ and $(\vec{\psi}_i^1, \ldots, \vec{\psi}_i^m)$. The parties generate the same vectors except that $\vec{\gamma}_i$ is a random L-encoding of a random block of values, and $\vec{\nu}_i$ and $\vec{\tilde{\gamma}}_i$ are random L and $\tilde{L}$ encodings of the all 0's block. In addition, the parties generate a random L'-encoding $\vec{\gamma'}^i = (\gamma'^i_1, \ldots, \gamma'^i_n)$ of a block of values that are random subject to the condition that they add up to 0, where $L' = \mathsf{RS}_{\mathbb{F},n,k+\mathcal{B},\eta}$.

- INPUT SHARING. The parties share a block rather than a single element. Namely, the parties embed their input value(s) into a block of length $\mathcal{B}$, and generates a packed secret sharing L-encoding for this block (as described in Section 3.6), distributing the outcome as in the basic protocol.

- EMULATING THE COMPUTATION. The computation proceed in layers of multiplication gates, where for each layer, we maintain the invariant that the parties hold additive shares of the inputs to the (at most) $\mathcal{B}$ multiplication gates in the layer. The difference from the basic protocol is that before evaluating a layer, the parties need to repack the inputs to the layer. (See discussion below on why repacking might be needed.)

  Concretely, to evaluate a layer, each party first rearranges the left wire values and right wire values of the multiplication gates in the layer into blocks B_L and B_R, and generates an L-encoding of each block. For every i, let a_i^l, b_i^l denote P_i's shares of B_L, B_R (respectively) corresponding to the l'th copy. Then the parties compute (via the reduction from $\mathcal{F}_{\text{MULT}}$ to $\mathcal{F}_{\text{RMULT}}$) additive shares $\left(\tilde{c}_1^l, \ldots, \tilde{c}_m^l\right)$ of $(\sum_{i=1}^m a_i^l)(\sum_{i=1}^m b_i^l)$, where P_i receives $\tilde{c}_i^l$, just as in the basic MPC protocol. Then, each P_i locally performs the degree reduction procedure as in the basic MPC protocol, with the difference that P_i

decodes $(\tilde{c}_i^1, \ldots, \tilde{c}_i^n)$ to obtain a *block* of values which it uses as the additive shares of the outputs of the multiplication gates in the layer.

Why repacking is needed. To see why rearranging the values within and between blocks might be necessary, consider a circuit that has a wire connecting the 3'rd value in the 2'nd block in layer 1 with the 5'th value in the 3'rd block in layer 2; or a wire connecting the 4'th value in the 1'st block in layer 1 with the 2'nd value in the 1'st block in layer 2.

- CORRECTNESS TESTS. We will employ the equality test as before, modify the degree test to also check the repacked encodings, and add an additional *permutation test*, as described next.

  THE MODIFIED DEGREE TEST. As in the basic protocol, the degree test will compute a random linear combination of the tested encodings. These encodings include the blocks $\vec{X}_i$ encoding the parties' inputs (which were committed in the input sharing step), the block of 0s encoded in $\vec{\nu}_i$ (which was committed in the initialization step), and the matrix U_i which contains L-encodings of the blocks of additive shares that were obtained from the degree reduction step following a multiplication step (U_i was committed to during the commit to degree test step). The difference from the degree test of the basic MPC protocol is that the linear combination will now also include an additional matrix U_i'' which contains the L-encodings of the repacked blocks of additive shares that were used as inputs to multiplication gates. (We note that these values are never committed to, but as explained in the proof of Corollary 3 below, can be extracted by the simulator from the transcript of the execution.) More formally, the parties will obtain from $\mathcal{F}_{\text{COIN}}$ random vectors $\vec{r}, \vec{r}', \vec{r}''$, and a random value r'', and party P_i will compute

$$q_i = \vec{r}^T U_i + \vec{r}''^T U_i'' + r_i' \vec{X}_i + r'' \vec{\nu}_i + \vec{\gamma}_i.$$

  Permutation test: this test verifies that the parties correctly permute (i.e., rearrange) the additive shares of wire values between layers. In particular, the test verifies that the encodings of the left and right input blocks of each computation layer correctly encode the values from the previous layers (and similarly for the output blocks). Note that the set of constraints that the blocks of values have to satisfy can be expressed as a set of linear equations in at most $m\mathcal{B}$ equations and $m\mathcal{B}$ variables (where w is the width, d is the depth of the computed circuit, and $m = dw/\mathcal{B}$), where variable $x_{i,j}$ represents the j'th value in the i'th block. (For example, if the circuit has a wire between the 3'rd value of the 2'nd block and the 5'th value in the 3'rd block, the corresponding constraint would be $x_{2,3} - x_{3,5} = 0$.) These linear equations can be represented in matrix form as $Ax = 0^{m\mathcal{B}}$, where $A \in \mathbb{F}^{m\mathcal{B} \times m\mathcal{B}}$ is a public matrix which only depends on the circuit being computed. The permutation test simply picks a random vector $\vec{r} \in \mathbb{F}^{m\mathcal{B}}$ and checks that $(\vec{r}^T A)x = 0$. To check these constraints, the parties obtain from $\mathcal{F}_{\text{COIN}}$ a random vector $\vec{r} \in \mathbb{F}^{m\mathcal{B}}$ and compute

$$\vec{r}^T A = (r'_{11}, \ldots, r'_{1\mathcal{B}}, \ldots, r'_{m1}, \ldots, r'_{m\mathcal{B}}).$$

  Now, let $r_j(\cdot)$ be the unique polynomial of degree $< \mathcal{B}$ such that $r_j(\zeta_Q) = r'_{jQ}$ for every $Q \in [\mathcal{B}]$ and $j \in [m]$. Then party P_i locally computes $q_i' = (r_1(\zeta_i), \ldots, r_m(\zeta_i))^T U_i' + \gamma_i'$, where γ_i' is the blinding encoding from the initialization step (that encode in $\mathsf{RS}_{\mathbb{F},n,k+\mathcal{B},\eta}$ random blocks of values that sum to 0), and U_i' is the matrix obtained by concatenating the rows of U_i and U_i'' from the degree test. Notice that the rows of U_i' consist of the L-encodings which P_i obtained at the output of multiplication layers (after degree reduction), and the L-encodings it used as inputs to multiplication layers (after repacking). Finally, P_i commits to each element of q_i' using $\mathcal{F}_{\text{COM}}$.

- CONSISTENCY TEST CHECK. In the consistency test, we also check for all $l \in \Gamma$ that the permutation test values of copy l were computed correctly. Specifically, each party checks that for every $i \in [m]$, the l'th element of q'_i is consistent with the l'th element of γ'_i, the l'th column of U'_i, and $\vec{r}$ (the coins obtained from $\mathcal{F}_{\text{COIN}}$ for the permutation test).

- PERMUTATION TEST CHECK. The parties decommit their permutation test commitments for all copies $l \notin \Gamma$, namely each P_i opens the commitment to the value q'_i computed above. Each party computes $q'_i = (q'_1 + \ldots + q'_m)$, and aborts if $q' = (q'_1, \ldots, q'_n) \notin \mathsf{RS}_{\mathbb{F},n,k+\mathcal{B},\eta}$ or $x_1 + \cdots + x_w \neq 0$ where $x = (x_1, \ldots, x_w) = \mathsf{Decode}_\eta(q')$.

The following Theorem follows from Theorem 1 and the discussion above.

Theorem 3. *The packed variant of protocol Φ of Figures 6-8 securely realizes $\mathcal{F}$ in the ($\mathcal{F}_{\text{COM}}, \mathcal{F}_{\text{RMULT}}, \mathcal{F}_{\text{COIN}}$)-hybrid model, tolerating $m-1$ active (static) corruptions, with statistical security error*

$$(1-e/n)^\delta + ((e+k+\mathcal{B})/n)^\delta + (n-k+3)/|\mathbb{F}| + 2^{-\Omega(e)}$$

where $k > \delta + 4e + \mathcal{B}$, $n > 2k + 4e$, and $e < (n-k+1)/3$.

We only sketch the proof, which is very similar to that of Theorem 1.

Proof sketch. The proof follows similarly to that of Theorem 1. We highlight the difference below.

First, we revise the simulator description by modifying two of the tests it performs in Step 9. First, in Step 9b Sim additionally checks that $\Delta\left(U', L^\ell\right) \subseteq E$, where $U' = \sum_{i\in[m]} U'_i$, U'_i is the matrix which P_i (should have) used during the permutation test, and ℓ denotes the number of rows in U'. (Notice that the simulator can determine U'_i even for corrupted parties: U_i was committed to, and U''_i can be extracted from the messages exchanged with the corrupted parties during the $\mathcal{F}_{\text{MULT}}$ to $\mathcal{F}_{\text{RMULT}}$ reductions.) Second, in Step 9e, Sim additionally checks that repacking was performed correctly. This is done by decoding the values of U' similarly to how the simulator decodes U, V (by constructing the unique L^ℓ-encoding $\hat{U}'$ that agrees with U' on all columns except those in E, and decoding $\hat{U}'$; such a $\hat{U}'$ exists because $\left|\Delta\left(U', L^\ell\right)\right| \leq |E| \leq e < n-k+1$), and uses the decoded values to verify that repacking was performed correctly. (That is, if the decoded values are $\vec{x}$, it verifies that $A\vec{x} = \vec{0}$.) We also note that in Step 10 the simulator is able to "correct" the output shares as in the proof of Theorem 1, even though the shares might pack $\mathcal{B}$ values, because $k > \delta + 4e + \mathcal{B}$.

Second, we revise the statement of Claim 4.1, where the soundness error increases by an additive $1/|\mathbb{F}| + ((e+k+\mathcal{B})/n)^\delta$ term,[7] and except with this probability, either one of the consistency, degree, equality *or permutation* tests fails, or all the conditions stated in the claim hold. We additionally change Condition (b) to include $\Delta\left(U', L^\ell\right) \subseteq E$, and Condition (d) to include the existence of a unique $\hat{U}' \in L^\ell$ that agrees with U' on all columns except those in E, and decodes to values $\vec{x}$ such that $A\vec{x} = \vec{0}$.

Then, we modify the proof of Claim 4.1 to account for the probability that the permutation test fails. In more detail, we change the case $|E| \leq e$, $d\left(M, L^{h+m+1}\right) > e$ in the proof of Claim 4.1 to the case $|E| \leq e$, $d\left(M', L^{\ell+m+1}\right) > e$, where the rows of M' consist of $U, U'' = \sum_{i\in[m]} U''_i, \vec{X}_1, \ldots, \vec{X}_m$ and $\sum_{i\in[m]} \vec{\nu}_i$. Since our modified degree test computes a linear combination over M' (instead of M), an analogous analysis

[7] Overall, the soundness error is now

$$(1-e/n)^\delta + ((e+k+\mathcal{B})/n)^\delta + (n-k+3)/|\mathbb{F}| + 2^{-\Omega(e)}.$$

to that of case $|E| \leq e$, $d\left(M, L^{h+m+1}\right) > e$ in the proof of Claim 4.1 guarantees that the modified degree test fails except with probability $(n-k+1)/|\mathbb{F}|$, because $e < (n-k+1)/3$.

Similarly, we replace the case $|E| \leq e$, $d\left(M, L^{h+m+1}\right) \leq e$ and $\Delta\left(M, L^{h+m+1}\right) \nsubseteq E$ in the proof of Claim 4.1 with the case $|E| \leq e$, $d\left(M', L^{\ell+m+1}\right) \leq e$, and $\Delta\left(M', L^{\ell+m+1}\right) \nsubseteq E$, and an analogous analysis to one provided in the proof of Claim 4.1 shows that in this case the degree test fails except with probability $1/|\mathbb{F}|$.

Conditioned on none of the (modified) bad events (from the proof of Claim 4.1, modified as described above) occurring, then U can be uniquely extended to a $\hat{U} \in L^h$ that agrees with U on all columns except those in E, and V can be uniquely extended to a $\hat{V} \in \tilde{L}^h$ that agrees with V on all columns except those in $E \cup E'$. This follows identically to the analysis in the proof of Claim 4.1. Moreover, a similar analysis shows that U' can be uniquely extended to a $\hat{U}' \in L^\ell$ that agrees with U' on all columns except those in E. Now, since $|E| \leq e$, then $d\left(U', \hat{U}'\right) \leq e$. If $\hat{U}'$ decodes to values $\vec{x}$ such that $A\vec{x} \neq \vec{0}$, then by [AHIV17, Lemma 4.6], either the permutation or the consistency tests fail except with probability $1/|\mathbb{F}| + ((e+k+\mathcal{B})/n)^\delta$. Therefore, except for this probability, $\hat{U}'$ encodes values $\vec{x}$ such that $A\vec{x} = \vec{0}$, i.e., repacking was computed correctly in all columns except those in E. Thus, it follows similarly to the proof of Claim 4.1 that the output reconstructed in $\mathcal{H}_0$ is $\mathcal{F}\left(x_{\overline{\mathcal{T}}}, x^*_{\mathcal{T}}\right)$.

The description and analysis of hybrid $\mathcal{H}_1$ follows identically to Theorem 1.

The description of $\mathcal{H}_2$ is identical to the proof of Theorem 1, where in the case analysis we add after case EQ the case "P: permutation test fails". The analysis of cases C, D, and O follows identically to the proof of Theorem 1. The analysis of case EQ only differs in the following point. The claim that for all executions in W the joint view of the adversary up to (including) the degree test check, together with the views of the honest parties in copies $\Gamma \cup E \cup E'$ are identically distributed in $\mathcal{H}_1, \mathcal{H}_2$, holds because $|\Gamma \cup E \cup E'| + \mathcal{B} \leq \delta + 4e + \mathcal{B} < k$.

We now analyze case P. In this case, the equality test passes, so the permutation test outcomes q'_i are revealed by all parties P_i. From the analysis of cases C, D and EQ in the proof of Theorem 1, except with probability $(1-e/n)^\delta + \frac{n-k+2}{|\mathbb{F}|} + 2^{-\Omega(e)}$, conditions (a)-(c) of Claim 4.1 hold, in which case the adversarial views up to (including) the equality test check are identically distributed in $\mathcal{H}_1, \mathcal{H}_2$. We claim that conditioned on the consistency test passing, the permutation test outcomes of honest parties are L-encodings that are distributed uniformly at random subject to the constraint that they are consistent with the values revealed during the consistency test, and the values they encode sum to 0. Indeed, for every honest party P_i, $p_i = (r_1(\zeta_i), \ldots, r_m(\zeta_i))^T U'_i \in L'$, and p_i decodes to values that sum to 0. Therefore, since p_i is blinded by an L'-encoding $\vec{\gamma}'_i$ that is uniformly random subject to the constraint that it encodes random values that sum to 0, then q'_i is uniformly random in L' subject to the constraint that the values it encodes sum to 0.

Therefore, conditioned on the permutation test failing, and on conditions (a)-(c) of Claim 4.1, the hybrids are identically distributed. Moreover, the probability that the permutation test fails is independent of the honest parties' inputs, as it only depends on the q'_i's supplied by the corrupted parties as their degree test outcomes. Therefore, the hybrids are statistically close in this case. ■

Communication Complexity. Assuming that each layer of the circuit has at least $\mathcal{B}$ parallel multiplications, the communication complexity of this variant is given by $O(n \cdot \frac{h}{\mathcal{B}} \cdot \mathrm{CC}_{\mathrm{MULT}}) + \mathsf{poly}(m, \kappa, \log_2 |\mathbb{F}|)$ since we amortize over $\mathcal{B}$ multiplications. By setting $n = O(s)$, the amortized communication overhead of this protocol becomes $O(1)$ per copy. Circuits of an arbitrary structure can be easily handled at a worst-case additional cost of $\mathsf{poly}(s, d)$. The statistical error can be improved by repeating the tests.

Extending the analysis for small fields. The analysis presented above works for fields of size larger than n, for smaller fields, we present a modified analysis, which is inspired by the analysis of [DI06, Appendix A]. Notice that the soundness error of our protocol depends on the field size due to the error probability of the degree, equality and permutation tests, where the error term is at most $(n-k+3)/|\mathbb{F}|$. These are all instances of tests that check membership in a linear (error-correcting) code, as well as checking that the encoded message satisfies certain constraints. We now show that the soundness error of these tests is $1/|\mathbb{F}|$, and thus can be reduced to 2^{-s} by standard soundness amplification. (We note that this analysis works for any field size, but will only be useful for small fields, since for large fields the analysis described above gives better parameters.)

Specifically, for the following analysis we assume that $n > 2k + 8e + 2$ and discuss a modified protocol in which we run σ independent instances of each of the degree, equality and permutation test.[8] We say that a test *passes* if all of its σ instances pass. (Each instance of a test is identical to the tests described in our protocol; notice that parties now need to commit during the initialization phase to blinding encodings for all instances.) We will show that except with probability $3n \cdot 2^{-\sigma} + ((e+k+\mathcal{B})/n)^{\delta} + (1-e/n)^{\delta} + 2^{-\Omega(e)}$ either one of the tests fails, or U, V, U' are consistent (except for the coordinates in $E \cup E'$) with $\hat{U}, \hat{V}, \hat{U}'$ in the corresponding interleaved codes, and moreover the messages encoded in these codewords satisfy the required constraints. (That is, $\hat{U}, \hat{V}$ encode the same messages, and the messages $\vec{x}$ encoded in $\hat{U}'$ satisfy the linear constraints $A\vec{x} = \vec{0}$ defined by the circuit structure.) In particular, to reduce the $3n \cdot 2^{-\sigma}$ term to soundness 2^{-s}, it suffices to choose $\sigma = (s+2)\log((h+3m)n)$. We show this bound by analyzing each test separately, conditioned on the event that $|E \cup E'| \leq 4e$. As shown in the proof of Theorem 1, if the consistency test passes then this holds except with probability $(1-e/n)^{\delta} + 2^{-\Omega(e)}$. First, we consider the degree test. Let $\mathcal{W}$ denote the linear subspace spanned by the purported codewords checked during the degree test, namely by the rows of U, as well as $(X_i, \nu_i, \gamma_i)_{i\in[m]}$. We consider two cases.

CASE I: $\mathcal{W}$ CONTAINS A VECTOR $\vec{w}$ WHICH IS $(8e+2)$-FAR FROM L. We claim that in this case, with probability at least $1/|\mathbb{F}|$ over the coefficients of a random linear combination used in an instance of the degree test, the resultant linear combination is $(4e+1)$-far from L. Notice that if this claim holds then with probability at least $1/|\mathbb{F}|$ the resultant linear combination is not a valid L-encoding (causing this instance of the degree test to fail). Indeed, a random linear combination computed as part of the degree test was computed correctly in all but the (at most) $4e$ copies in $E \cup E'$, and so the result will differ in at least one coordinate from the closest codeword. It remains, therefore, to prove the claim. For this, consider a basis for $\mathcal{W}$ in which one of the basis vectors is $\vec{w}$. The linear combination computed in an instance of the degree test is simply a random vector in $\mathcal{W}$, and so is obtained by a random linear combination of the basis vectors. Fix some possible value $\vec{w}'$ of the linear combination over all basis vectors *excluding* $\vec{w}$. If $\vec{w}'$ is $(4e+1)$-far from L then with probability $1/|\mathbb{F}|$ $\vec{w}$ isn't part of the linear combination (i.e., appears with coefficient 0) and consequently the linear combination is $(4e+1)$-far from L. Otherwise, $\vec{w}'$ is $(4e+1)$-close to L, then with probability $(|\mathbb{F}|-1)/|\mathbb{F}| \geq 1/|\mathbb{F}|$, $\vec{w}$ appears in the linear combination (i.e., with non-0 coefficient), and consequently the linear combination is $(4e+1)$-far from L (because $\vec{w}$ is $(8e+2)$-far from L).

CASE II: ALL VECTORS IN $\mathcal{W}$ ARE $(8e+2)$-CLOSE TO L. In this case, we claim that all purported encodings tested during the degree test (i.e., the rows of U, and $(X_i, \nu_i, \gamma_i)_{i\in[m]}$) are consistent with L-encodings except for the coordinated in $E \cup E'$. To prove this claim, it suffices to prove that for every such purported encoding $\vec{v}$, if $l \in \Delta(\vec{v}, L)$ then $l \in E \cup E'$ except with probability $2^{-\sigma}$, since in this case the claim holds except with probability $(h+3m) \cdot n \cdot 2^{-\sigma}$ (by the union bound). Indeed, we show that if $l \notin E \cup E'$ then each instance of the degree test fails with probability $\geq 1/|\mathbb{F}|$, and conclude that if all

[8] We do not optimize our parameters as we are interested only in establishing asymptotic bounds.

instances of the degree test pass then $l \in E \cup E'$ (with high probability). Consider some linear combination computed in an instance of the degree test, and let $\vec{v}''$ denote the linear combination over all vectors *except* $\vec{v}$. By the case assumption, $\vec{v}''$ is $(8e+2)$-close to L. We now consider two cases, based on whether or not $\vec{v}''$ can "cancel out" the l'th coordinate in $\vec{v}$. First, if $l \in \Delta(\vec{v}'', L)$ then if $\vec{v}$ isn't part of the linear combination (i.e., appears with coefficient 0), which happens with probability $1/|\mathbb{F}|$, then this instance of the degree test fails (because this instance simply checks whether $\vec{v}'' \in L$). Second, assume that $l \notin \Delta(\vec{v}'', L)$. In this case, if $\vec{v}$ does appear in the linear combination (i.e., with a non-0 coefficient), which happens with probability $(|\mathbb{F}|-1)/|\mathbb{F}| \geq 1/|\mathbb{F}|$, then this instance of the degree test fails, because the tested value, which is $\vec{v}'' + c\vec{v}$ for some constant $c \neq 0$, differs from the closest codeword in L in the l'th coordinate (in particular, it is not a valid codeword).

Next, we analyze the permutation and equality tests. These differ from the degree test since they check not only that the linear combination is a valid encoding, but also that the encoded messages satisfy certain constraints. The analysis follows similarly to the degree test analysis, where we define $\mathcal{W}$ to be the subspace spanned by the purported encodings checked during the corresponding test (either the degree or the equality test). We note that in the equality test, we disregard the b_i^l values (see bottom of Figure 7 for the definition of b_i^l), which are not valid encodings. This does not affect the analysis since these are locally computed, and so are correctly computed in all copies except those in $E \cup E'$. (Indeed, by the definition of E, the local computations in all copies not in E are correctly performed. In particular, $\sum_i b_i^l$ for $l \notin E$.) In particular, these don't affect the linear combination in copies $l \notin E \cup E'$ since in such copies these are additive shares of 0. The only other difference in the analysis is that we divide case II into two subcases depending on whether or not the encoded messages satisfy the constraints. (We note that in both subcases the message encoded by a purported codeword is well defined, because $n > 2k + 8e + 2$.)

CASE II.1: ALL VECTORS IN $\mathcal{W}$ ARE $(8e+2)$-CLOSE TO THE CODE, AND THE ENCODED MESSAGES SATISFY THE CONSTRAINTS. In this case, the analysis follows similarly to case II above.

CASE II.2: ALL VECTORS IN $\mathcal{W}$ ARE $(8e+2)$-CLOSE TO THE CODE, BUT THE ENCODED MESSAGES DON'T SATISFY THE CONSTRAINTS. In this case, by the analysis in the proof of Theorem 3, either the consistency or the permutation tests fail except with probability $(1/|\mathbb{F}|)^\sigma + ((e+k+\mathcal{B})/n)^\delta$.

6 Corollaries and Applications

In this section we consider several different instantiations of the $\mathcal{F}_{\text{RMULT}}$ functionality, thus obtaining our main results in the different settings as instances of the generic protocol of Section 5.

6.1 Constant Overhead MPC for Constant-Size Fields

Dishonest majority. Our main result is obtained by replacing the Reed-Solomon codes in our protocol with Algebraic Geometric (AG) secret sharing over fields of constant size [CC06], instantiating the $\mathcal{F}_{\text{RMULT}}$ functionality with pairwise calls to a passively-secure implementation of the $\mathcal{F}_{\text{OT}}$ functionality, and instantiating commitments using a pseudorandom generator. Formally:

Theorem 4 (Theorem 1, restated)**.** *Let κ, s denote computational and statistical security parameters (resp.), m denote the number of parties, and $\mathbb{F}$ be a constant-size field. Then there exists a protocol compiler that, given a pseudorandom generator G with seed length κ, s, a constant-round implementation of the $\mathcal{F}_{\text{OT}}$ functionality with total communication complexity CC_{OT}, and a description of an m-party functionality expressed as a depth-d circuit C with constant fan-in, outputs a UC-secure $O(d)$-round m-party protocol*

realizing f with communication complexity $O(m^2 \cdot |\mathrm{C}| \cdot \mathrm{CC}_{\mathrm{OT}}) + \mathsf{poly}(m, \kappa, d)$, where security holds against an active adversary corrupting an arbitrary number of parties.

The communication complexity of our protocol using a bit-OT protocol for the boolean setting asymptotically matches the communication complexity of the best known passively-secure protocol, namely [GMW87] using a passively-secure OT protocol. The best known previous result for active security is due to Genkin et al. [GIW16] who achieve $O(m^2 \cdot |\mathrm{C}| \cdot \mathsf{poly}\log(s))$ communication complexity, i.e., a multiplicative factor of $\mathsf{poly}\log(s)$ over GMW.

Honest majority. To obtain our main result for the honest majority setting, we need to slightly modify our protocol in two ways. First, we will rely on the passive variant of a protocol of Damgård and Nielsen [DN07], instantiated with secret-sharing based on AG codes over constant-size finite fields, to instantiate the parallel $\mathcal{F}_{\mathrm{RMULT}}$ functionality (i.e., to generate the triples in the initialization phase). To achieve this, we replace the additive secret sharing used in our protocol with secret sharing based on AG codes for constant-size fields. We note that the passively-secure honest-majority m-party protocol of [DN07] can generate $T = \Omega(m)$ random triples with total communication complexity $O(mT)$. Second, we will consider $\mathcal{F}_{\mathrm{RMULT}}$ and $\mathcal{F}_{\mathrm{MULT}}$ whose underlying secret sharing scheme is based on the same AG secret sharing scheme. Specifically, parallel $\mathcal{F}_{\mathrm{RMULT}}$ distributes secret-shares of many triples a, b and c such that $a \cdot b = c$. Then the $\mathcal{F}_{\mathrm{MULT}}$ to $\mathcal{F}_{\mathrm{RMULT}}$ reduction works essentially as described in Section 3.5, where the only difference is that the values e, d are reconstructed using the reconstruction procedure of the AG secret sharing scheme. Consequently, we obtain the following theorem.

Theorem 5. *Let κ, s denote computational and statistical security parameters (resp.), m denote the number of parties, and $\mathbb{F}$ be a constant-size field. Then there exists a protocol compiler that, given a pseudorandom generator G with seed length κ, s, and a description of an m-party functionality expressed as a depth-d circuit C with constant fan-in, outputs a UC-secure $O(d)$-round m-party protocol realizing f with $O(m|\mathrm{C}|) + \mathsf{poly}(m, \kappa, d)$ bits total communication complexity, and security against a static adversary corrupting a minority of parties.*

We remark that this improves over the result of Chida et al. [CGH+18] that achieves $O(s)$ overhead for binary fields, and generalizes the result of Ishai et al. [IKP+16] that achieves the same result, but only for a constant number of parties. We remark that the latter protocol additionally achieve constant-rate, while our protocol only achieves constant-overhead.

6.2 Constant Overhead MPC over Fields of Arbitrary Size

Dishonest majority. To obtain our result for fields of arbitrary size, we realize the $\mathcal{F}_{\mathrm{RMULT}}$ functionality using a passively-secure OLE protocol. For fields of size $\leq s$ we rely on AG sharing, whereas for fields of size $\Omega(s)$ we use Reed-Solomon codes. Thus, we can re-derive a result of Genkin et al. [GIP+14] (Theorem 5.7 in the full version), who construct an actively-secure m-party protocol for arbitrary functionalities (represented by an arithmetic circuit C), in the dishonest majority setting, using $O(m^2 \cdot |\mathrm{C}|)$ calls to an OLE oracle. More precisely, we have the following theorem:

Theorem 6. *Let κ, s denote computational and statistical security parameters (resp.), m denote the number of parties, and $\mathbb{F}$ be a field. Then there exists a protocol compiler that, given a pseudorandom generator G with seed length κ, s, a constant-round implementation of the $\mathcal{F}_{\mathrm{OLE}}$ functionality over $\mathbb{F}$ with total communication complexity $\mathrm{CC}_{\mathrm{OLE}}$, and a description of an m-party functionality expressed as a depth-d arithmetic*

circuit C *over* $\mathbb{F}$ *with constant fan-in, outputs a UC-secure* $O(d)$*-round* m*-party protocol realizing* f *with communication complexity* $O(m^2 \cdot |\mathrm{C}| \cdot \mathrm{CC}_{\mathrm{OLE}}) + \mathsf{poly}(m, \kappa, d)$ *field elements, with security against an active adversary corrupting an arbitrary number of parties.*

This result asymptotically matches the communication complexity of the best known passively-secure protocol [GMW87] using a passively-secure OLE protocol. Furthermore, for sufficiently wide circuits, we can show that the overhead of our protocols is 2. We present the concrete parameters in Appendix A.

Honest majority. Just as in Section 6.1, we can obtain constant overhead over the best passively-secure protocol in the honest majority setting:

Theorem 7. *Let* κ, s *denote computational and statistical security parameters (resp.),* m *denote the number of parties, and* $\mathbb{F}$ *be a field. Then there exists a protocol compiler that, given a pseudorandom generator* G *with seed length* κ*,* s*, and a description of an* m*-party functionality expressed as a depth-*d *arithmetic circuit* C *over* $\mathbb{F}$ *with constant fan-in, outputs a UC-secure* $O(d)$*-round* m*-party protocol realizing* f *with total communication complexity* $O(m|\mathrm{C}|) + \mathsf{poly}(m, \kappa, d)$ *bits, where security holds against a static adversary corrupting a minority of parties.*

Applying the concrete analysis from Appendix A we re-derive the result of Chida et al. [CGH+18] who show an overhead-2 actively-secure honest-majority protocol. Their result applies to arbitrary circuits over sufficiently large fields, whereas ours achieves overhead of 2 for sufficiently wide circuits.

7 Extensions and Further Corollaries

In this section, we discuss various generalizations and extensions of our compiler.

7.1 Imperfect $\mathcal{F}_{\mathrm{MULT}}$

As discussed in the introduction we can instantiate the $\mathcal{F}_{\mathrm{MULT}}$-functionality via a protocol that only guarantees "weaker" than passive security.

In this section, we describe how to instantiate our compiler with an *imperfect* passive implementation of $\mathcal{F}_{\mathrm{MULT}}$, which may have a non-negligible simulation error. This is particularly relevant to lattice-based implementations for which one can obtain better efficiency, at the cost of causing a simulation error, by setting the parameters more "aggressively". Hazay et al. [HIMV19] show how to compile an imperfect OLE protocol into a fully-secure OLE protocol. In their work, "imperfectness" was modeled via a natural "exclusion set" model described below.

We will restrict our attention to $\mathcal{F}_{\mathrm{MULT}}$ implementations based on OLEs, leaving more general discussions to future work. We also consider only the case of employing OLE over random inputs, since this suffices for our compiler. Furthermore, we will focus on the case of an OLE that is (fully) computationally secure against passive corruption of the sender, and is ϵ-*statistically* secure against passive corruption of the receiver on *random inputs* (see below). This will be sufficient to capture our OLE instantiations based on lattice assumptions. We start by defining the notion of ϵ-secure OLE over uniformly random inputs. Let $\Pi = \langle P_0, P_1 \rangle$ denote a two-party protocol, where each party is given an input (x for P_0 and y for P_1). Denote by $\mathbf{View}_{P_i}(P_0(x), P_1(y))$ the view of the party P_i in the real execution of Π where x is P_0's initial input, and y is P_1's initial input.

Definition 5 (ϵ-secure OLE)**.** *We say that a two party protocol $\Pi = (\mathcal{S}, \mathcal{R})$ is an ϵ-secure OLE implementation over $\mathbb{F}_p$ w.r.t the uniform distribution, if for every $x \in \mathbb{F}_p$ the statistical distance between the following two distributions is bounded by ϵ:*

- $\{(a, b, \mathbf{View}_{\mathcal{R}}(\mathcal{S}(a, b), \mathcal{R}(x)))\}$
- $\{(a', b', \mathbf{View}_{\mathcal{R}}(\mathcal{S}(a, b), \mathcal{R}(x)))\}$

over a, b sampled uniformly from $\mathbb{F}_p$, and a', b' sampled uniformly from $\mathbb{F}_p$ subject to $a'x + b' = ax + b$ over $\mathbb{F}_p$.

We conjecture that given an ϵ-secure OLE, our compiler from Section 5 can compile it to a fully-secure OLE. We leave the question of proving/disproving this conjecture as an interesting open problem. For evidence in support of this conjecture, we refer to [HIMV19] who analyzed a similar compiler with an imperfect OLE that is an instance of an ϵ-secure OLE. More precisely, they consider an ideal OLE functionality that asks the adversary to specify *an exclusion set* A, and leaks to the adversary the single information bit of whether the honest party's input belongs to A. They prove that if the exclusion set is relatively small compared to the field size, then security can be amplifies via the IPS compiler. On a high-level, they argue that in the IPS compiler, even if the adversary learns all the shares of servers on its watchlist, and a small amount of leakage (via exclusion sets) on each of the remaining shares, the actual secret remains hidden. More formally, they quantify leakage following the framework of Benhamouda et al. [BDIR18]. The bottom line in their analysis shows that an ϵ-imperfect OLE (in the exclusion set model) additionally adds a statistical error of $p^{-k+1} \cdot O((\epsilon \cdot p)^{n-t-e})$ where $p = |\mathbb{F}|$.

Concrete LWE parameters based on our conjecture. We repeat the crude analysis verbatim from [HIMV19] which suggests a choice of parameters for *imperfect* LWE-based OLE that the compiler can tolerate. To understand the leakage in standard Ring-LWE based schemes [LPR10], we recall some relevant parameters. Denote the plaintext modulus by p, and the ciphertext modulus by q. Then the magnitude of the statistical error is bounded by $\log_2(q) - \log_2(O(c \cdot p^2 \cdot \Phi))$ where $c \equiv q \bmod p$ (typically made small by choosing an appropriate q), and Φ is the packing factor. In other words, the statistical distance between encryptions of different inputs is roughly $c \cdot p^2 \cdot \Phi/q$, which could be large for aggressive parameter choices. Applying the standard implementation of (passive) OLE based on additively homomorphic encryption, instantiated with Ring-LWE encryption with these parameters, will result in an imperfect OLE where the amount of entropy from the sender's inputs leaked to the receiver (on a random input) is roughly $c \cdot p^2 \cdot \Phi/q$ bits. Our analysis only considers a simple leakage where the leakage functions are exclusion sets. We conjecture that this model is "complete" in the sense that (in the context of the IPS-style compiler of [HIMV19]) it captures a general leakage with the same amount of entropy, namely, by setting $\log_2(\epsilon)$ as $c \cdot p^2 \cdot \Phi/q$.

In order to get a passive OLE, one needs the magnitude of this error to be at least the statistical parameter (eg, 40, 80 or 128). For example, if p is a 20-bit prime and $\Phi = 2^{13}$, then a 127-bit modulus q gives a passive OLE with 64-bit security (where typically c and the constant behind $O(\cdot)$ are roughly 2^5). However, if we use a 88-bit modulus q, the statistical error will be roughly 2^{-25}. For these parameters, the error is roughly $1/\sqrt{p}$ and allows us to get negligible statistical error. Therefore, the

compiler of [HIMV19] can amplify this to a fully-secure OLE. Since their compiler requires twice as many passive/imperfect OLEs, the communication overhead of an actively secure OLE protocol against the passive OLE protocol can be estimated by $2 \cdot 88/127 = 1.38 < 2$.

For some parameter regimes (e.g., larger statistical security parameters), their construction of actively-secure OLE is actually *more communication efficient* than a naive construction of passively-secure OLE with a larger security parameter. If we used the parameters described above but demanded 128-bit security, $\log_2(q)$ will be 184 and 85 respectively for passive and active and the overhead will be $2 \cdot 85/184 = 0.924 < 1$.

7.2 Constant-Round Protocols in the Boolean Setting

We remark that our techniques also yields an interesting corollary for securely computing boolean functionalities in the constant-round regime. This follows from two observations:

1. Securely evaluating a boolean circuit can be reduced to securely computing a distributed garbling of the underlying circuit [BMR90].
2. The BMR distributed garbling scheme can be expressed as an arithmetic circuit over the field $GF(2^\kappa)$ with $O(|\mathrm{C}|)$ multiplication gates, and depth-2, where C is the circuit being garbled.

Now, we can rely on our protocol for arithmetic circuits to first securely compute a distributed garbling of the underlying functionality, and then follow the standard protocol based on BMR garbling.

We highlight that distributed garbling would be an instance of a "very-wide" circuit, as the width of the garbling circuit is $O(|\mathrm{C}|)$.

In the regime of constant-round protocols, the state-of-the-art is due to Wang et al. [WRK17b] who show how to achieve actively-secure multiparty computation for boolean computations with $s/\log|\mathrm{C}|$ overhead over the best passive protocol, namely [BMR90]. Their work is incomparable to ours, as theirs is in the OT-hybrid model, whereas ours requires OLE.

Acknowledgments

We thank Yuval Ishai and Ignacio Cascudo Pueyo for helpful discussions.

The first author was supported 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 ISF grant No. 1316/18. The second author was supported by a Google Faculty Research Grant and NSF Award CNS-1618884. The third author was supported by ISF grants 1861/16 and 1399/17, and AFOSR Award FA9550-17-1-0069.

References

[ADI+17] Benny Applebaum, Ivan Damgård, Yuval Ishai, Michael Nielsen, and Lior Zichron. Secure arithmetic computation with constant computational overhead. In *CRYPTO*, pages 223–254, 2017.

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *CCS*, pages 2087–2104, 2017.

[BDIR18] Fabrice Benhamouda, Akshay Degwekar, Yuval Ishai, and Tal Rabin. On the local leakage resilience of linear secret sharing schemes. In *Advances CRYPTO*, pages 531–561, 2018.

[BDOZ11] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In *EUROCRYPT*, pages 169–188, 2011.

[Bea91] Donald Beaver. Efficient multiparty protocols using circuit randomization. In *CRYPTO*, pages 420–432, 1991.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, pages 1–10, 1988.

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *STOC*, pages 503–513, 1990.

[CC06] Hao Chen and Ronald Cramer. Algebraic geometric secret sharing schemes and secure multi-party computations over small fields. In *CRYPTO*, pages 521–536, 2006.

[CCD87] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (abstract). In *CRYPTO*, page 462, 1987.

[CDN01] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. Multiparty computation from threshold homomorphic encryption. In *EUROCRYPT*, pages 280–299, 2001.

[CGH+18] Koji Chida, Daniel Genkin, Koki Hamada, Dai Ikarashi, Ryo Kikuchi, Yehuda Lindell, and Ariel Nof. Fast large-scale honest-majority MPC for malicious adversaries. In *CRYPTO*, pages 34–64, 2018.

[DGN+17] Nico Döttling, Satrajit Ghosh, Jesper Buus Nielsen, Tobias Nilges, and Roberto Trifiletti. TinyOLE: Efficient actively secure two-party computation from oblivious linear function evaluation. In *CCS*, pages 2263–2276, 2017.

[DI06] Ivan Damgård and Yuval Ishai. Scalable secure multiparty computation. In *CRYPTO*, pages 501–520, 2006.

[DKL+13] Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P. Smart. Practical covertly secure MPC for dishonest majority - or: Breaking the SPDZ limits. In *ESORICS*, pages 1–18, 2013.

[DN07] Ivan Damgård and Jesper Buus Nielsen. Scalable and unconditionally secure multiparty computation. In *CRYPTO*, pages 572–590, 2007.

[DPSZ12] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *CRYPTO*, pages 643–662, 2012.

[EGL85] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, 1985.

[FY92] Matthew K. Franklin and Moti Yung. Communication complexity of secure computation (extended abstract). In *STOC*, pages 699–710, 1992.

[GIP+14] Daniel Genkin, Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and Eran Tromer. Circuits resilient to additive attacks with applications to secure computation. In *STOC*, pages 495–504, 2014.

[GIW16] Daniel Genkin, Yuval Ishai, and Mor Weiss. Binary amd circuits from secure multiparty computation. In *TCC-B*, 2016.

[GLNP15] Shay Gueron, Yehuda Lindell, Ariel Nof, and Benny Pinkas. Fast garbling of circuits under standard assumptions. In *CCS*, pages 567–578, 2015.

[GLS19] Vipul Goyal, Yanyi Liu, and Yifan Song. Communication-efficient unconditional MPC with guaranteed output delivery. In *CRYPTO*, pages 85–114, 2019.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.

[Hai08] Iftach Haitner. Semi-honest to malicious oblivious transfer - the black-box way. In *TCC*, pages 412–426, 2008.

[HIK+11] Iftach Haitner, Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. Black-box constructions of protocols for secure computation. *SIAM J. Comput.*, 40(2):225–266, 2011.

[HIMV19] Carmit Hazay, Yuval Ishai, Antonio Marcedone, and Muthuramakrishnan Venkitasubramaniam. LevioSA: Lightweight secure arithmetic computation. In *To appear CCS*, 2019.

[HIV17] Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Actively secure garbled circuits with constant communication overhead in the plain model. In *TCC*, pages 3–39, 2017.

[HK12] Shai Halevi and Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. *J. Cryptology*, 25(1):158–193, 2012.

[HKK+14] Yan Huang, Jonathan Katz, Vladimir Kolesnikov, Ranjit Kumaresan, and Alex J. Malozemoff. Amortizing garbled circuits. In *CRYPTO*, pages 458–475, 2014.

[HL10] Carmit Hazay and Yehuda Lindell. *Efficient Secure Two-Party Protocols - Techniques and Constructions*. Information Security and Cryptography. Springer, 2010.

[HSS17] Carmit Hazay, Peter Scholl, and Eduardo Soria-Vazquez. Low cost constant round MPC combining BMR and oblivious transfer. In *ASIACRYPT*, pages 598–628, 2017.

[IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In *STOC*, pages 21–30, 2007.

[IKP+16] Yuval Ishai, Eyal Kushilevitz, Manoj Prabhakaran, Amit Sahai, and Ching-Hua Yu. Secure protocol transformations. In *CRYPTO*, pages 430–458, 2016.

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In *CRYPTO*, pages 572–591, 2008.

[IPS09] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Secure arithmetic computation with no honest majority. In *TCC*, pages 294–314, 2009.

[KPR18] Marcel Keller, Valerio Pastro, and Dragos Rotaru. Overdrive: Making SPDZ great again. In *EUROCRYPT*, pages 158–189, 2018.

[KS08] Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In *ICALP*, pages 486–498, 2008.

[LOP11] Yehuda Lindell, Eli Oxman, and Benny Pinkas. The IPS compiler: Optimizations, variants and concrete efficiency. In *CRYPTO*, pages 259–276, 2011.

[LP07] Yehuda Lindell and Benny Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In *EUROCRYPT*, pages 52–78, 2007.

[LP12] Yehuda Lindell and Benny Pinkas. Secure two-party computation via cut-and-choose oblivious transfer. *J. Cryptology*, 25(4):680–722, 2012.

[LPO11] Yehuda Lindell, Benny Pinkas, and Eli Oxman. The IPS compiler: Optimizations, variants and concrete efficiency. *IACR Cryptology ePrint Archive*, 2011:435, 2011.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, pages 1–23, 2010.

[LPSY15] Yehuda Lindell, Benny Pinkas, Nigel P. Smart, and Avishay Yanai. Efficient constant round multi-party computation combining BMR and SPDZ. In *CRYPTO*, pages 319–338, 2015.

[LR15] Yehuda Lindell and Ben Riva. Blazing fast 2pc in the offline/online setting with security for malicious adversaries. In *CCS*, pages 579–590, 2015.

[NNOB12] Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. In *CRYPTO*, pages 681–700, 2012.

[NO09] Jesper Buus Nielsen and Claudio Orlandi. LEGO for two-party secure computation. In *TCC*, pages 368–386, 2009.

[NP06] Moni Naor and Benny Pinkas. Oblivious polynomial evaluation. *SIAM J. Comput.*, 35(5):1254–1281, 2006.

[Rab81] M. Rabin. How to exchange secrets by oblivious transfer. Tech. Memo TR-81, Aiken Computation Laboratory, Harvard U., 1981.

[RR16] Peter Rindal and Mike Rosulek. Faster malicious 2-party secure computation with online/offline dual execution. In *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, pages 297–314, 2016.

[RZ17] Ronny Roth and Gilles Zémor. Personal communication, 2017.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[SS13] Abhi Shelat and Chih-Hao Shen. Fast two-party secure computation with minimal assumptions. In *CCS*, pages 523–534, 2013.

[ST04] Berry Schoenmakers and Pim Tuyls. Practical two-party computation based on the conditional gate. In *ASIACRYPT*, pages 119–136, 2004.

[WMK17] Xiao Wang, Alex J. Malozemoff, and Jonathan Katz. Faster secure two-party computation in the single-execution setting. In *EUROCRYPT*, pages 399–424, 2017.

[WRK17a] Xiao Wang, Samuel Ranellucci, and Jonathan Katz. Authenticated garbling and efficient maliciously secure two-party computation. In *CCS*, pages 21–37, 2017.

[WRK17b] Xiao Wang, Samuel Ranellucci, and Jonathan Katz. Global-scale secure multiparty computation. In *CCS*, pages 39–56, 2017.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167, 1986.

[ZRE15] Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole - reducing data transfer in garbled circuits using half gates. In *EUROCRYPT*, pages 220–250, 2015.

A Concrete Analysis for Large Fields

In this section we analyze the concrete constants achieved by our protocols over large fields. The first part analyzes the constants when the underling OLEs are defensibly private, and is taken almost verbatim from [HIMV19], where we extend their analysis for tolerating passively secure OLEs in our protocol.

Analysis for defensibly private OLEs. Our protocol (Section 4) depends on different parameters. Below, we discuss the different constraints on these parameters and the choices that optimize concrete performance. We remark that when $\mathcal{F}_{\text{MULT}}$ is instantiated with a concrete protocol, we can further prevent the adversary from launching selective attacks by requiring the parties to also commit to the randomness used in the protocol. Consequently, the $2^{-\Omega(e)}$ term in the soundness error can be dropped. Let s be the statistical security parameter. The set of parameters includes the number of servers n, the block width $\mathcal{B}$, the size δ of the subset of servers chosen during the consistency test, the number of corrupted copies e, and the parameters k, d of the Reed Solomon encodings. These parameters are subject to the following constraints:

$$(1-e/n)^{\delta} + (n-k+2)/\left|\mathbb{F}\right| < 2^{-s},\ \ k \geq \delta + e + \mathcal{B},\ \ 2k+e<n,\ \text{ and }\ e<(n-k+1)/3.$$

The dominating costs in our protocol are computing the OLE functionality, and (to a minor extent) also computing the Reed Solomon encodings. To optimize the concrete efficiency, we first wish to minimize the number of OLE calls. Note that for every block of $\mathcal{B}$ multiplication gates in the circuit, the parties engage in a single multiplication for each of the n copies. Therefore, if all the blocks of $\mathcal{B}$ gates are "full", i.e. they all contain exactly $\mathcal{B}$ gates, then the protocol requires $n/\mathcal{B}$ calls to $\mathcal{F}_{\text{MULT}}$. Another useful optimization is setting k to be a power of two, as this greatly increases the encoding efficiency since finite-field FFT algorithms can be used.

We provide a few examples of different sets of parameters in Table 3, where we consider 40 bits statistical security, and additionally set $k = \delta + \mathcal{B} + e$. It can be inferred from the table that as $\mathcal{B}$ grows, n/w approaches 2.

Analysis for passively-secure OLEs.[9] When the $\mathcal{F}_{\text{MULT}}$ functionality is instantiated using passively-secure OTs/OLEs, as described in Section 4.1, we need to perform a cut-and-choose step to insure that $\mathcal{F}_{\text{MULT}}$ was computed correctly. Recall that our protocol employs n copies and each requires h calls to $\mathcal{F}_{\text{MULT}}$ (where h denotes the number of multiplication gates in the circuit). To ensure correctness, it would suffice to ensure that at most e of the n-sets were faulty. To achieve this, we instead begin with n' sets of h outputs of $\mathcal{F}_{\text{MULT}}$, and perform a cut-and-choose test which checks the correctness of δ' of these sets, which are then discarded. Thus, we have $n' = n + \delta'$, and we need to verify that except with negligible probability, if the cut-and-choose test passes then at most e of the sets contain incorrect outputs of $\mathcal{F}_{\text{MULT}}$. Since

$$\Pr\left[\text{cut-and-choose passes} \wedge \text{more than } e \text{ copies contain errors}\right] \leq \frac{\binom{n'-e}{\delta'}}{\binom{n'}{\delta'}} \leq (1 - e/n')^{\delta'}$$

then to get $\mathsf{negl}(s)$ error, it suffices to set $\delta' = O(s)$. Table 3 shows that as $\mathcal{B}$ grows, $n'/\mathcal{B}$ approaches 2, so we obtain overhead 2 also when using passively-secure OLEs.

$\mathcal{B}$	e	δ	n	δ'	$n' = n + \delta'$	$n/\mathcal{B}$	$n'/\mathcal{B}$
1317	272	459	4640	514	5154	3.52	3.91
3065	362	669	8916	749	9665	2.91	3.15
6749	509	934	17402	1046	18448	2.58	2.73
14332	690	1362	34147	1525	35672	2.38	2.49
29864	987	1917	67493	2147	69640	2.26	2.33
61386	1369	2781	133769	3115	136884	2.18	2.23
125195	1964	3913	265987	4383	270370	2.12	2.16
253781	2778	5585	529690	6255	535945	2.09	2.11
512404	3951	7933	1056213	8885	1065098	2.06	2.08

Table 3: Concrete parameters of our protocol for 40-bit security. $\mathcal{B}$ is the block size, e is the number of tolerated errors, δ, δ' are the number of copies tested during the consistency and cut-and-choose tests (resp.), n' is the initial number of copies, where only n copies are used during the evaluation of the circuit. The two rightmost columns describe the overhead when using actively-secure and passively-secure OTs/OLEs (resp.).

[9] We note that the cut-and-choose analysis described in this paragraph holds over any field, but the concrete constants mentioned are obtained for large fields.

B Proof of Lemma 4.4 from [AHIV17]

In this section, we provide a direct proof of [AHIV17, Lemma 4.4] for the case when $e < (n-k+1)/3$, i.e., without relying on [AHIV17, Conjecture 4.1]. The proof was provided to us by the authors of [AHIV17] and is reproduced here verbatim with their permission. The proof of claim B.2 below is due to Ronny Roth and Gilles Zémor [RZ17].

Lemma B.1 (restatement of Lemma 4.4 from [AHIV17])**.** *Let e be a positive integer such that $e < (n - k + 1)/3$. Suppose $d(U, L^m) > e$. Then, for a random w^* in the row-span of U, we have*

$$\Pr[d(w^*, L) \leq e] \leq (n - k + 1)/|\mathbb{F}|.$$

Proof: Suppose that $d(U^*, L^m) > e$ and L^* is the span of the vectors in U^*. Assume towards a contradiction that $d(v^*, L) \leq e$ for all $v^* \in L^*$. Suppose $v_0^* \in L^*$ maximizes the distance from L. Since $d(U^*, L^m) > e$, there must be a row U_i^* such that $\Delta(U_i^*, L) \setminus \Delta(v_0^*, L) \neq \emptyset$. Let $v_0^* = u_0 + \chi_0$ and $U_i^* = u_i + \chi_i$ for $u_0, u_i \in L$ and χ_0, χ_i of weight $\leq e$. We argue that there exists $\alpha \in \mathbb{F}$ such that for $\hat{v} = v_0^* + \alpha U_i^*$ we have $d(\hat{v}, L) > d(v_0^*, L)$, contradicting the choice of v_0^*. This follows by a union bound, noting that for any $j \in \Delta(v_0^*, L) \cup \Delta(U_i^*, L)$ there is at most one choice of α such that $\hat{v}_j = 0$.

Now, it suffices to show that in any affine subspace of $\mathbb{F}^n$, either all points are e-close to L or almost all are not. This reduces to showing the following claim. We state an explicit version of the conjecture for the case of RS codes.

Claim B.2. *Let L be an arbitrary linear code over $\mathbb{F}$ of length n. Let e be a positive integer such that $e < (n - k + 1)/3$. Then for every $u, v \in \mathbb{F}^n$, defining an affine line $\ell_{u,v} = \{u + \alpha v : \alpha \in \mathbb{F}\}$, either (1) for every $x \in \ell_{u,v}$ we have $d(x, L) \leq e$, or (2) for at most $(n - k + 1)$ points $x \in \ell_{u,v}$ we have $d(x, L) \leq e$.*

We begin with the observation that for any two length n vectors u and v of weight at most e, $\ell_{u,v}$ contains N points at most distance e from L if and only if $\ell_{u,v+c}$ contains N points of distance at most e from L for any codeword $c \in L$. This means it suffices to prove the claim for vectors u and v of weight at most e.

We now prove the lemma in two cases

Case 1: $|Support(u) \cup Support(v)| \leq e$ This means that $\ell_{u,v}$ is entirely contained in the ball $B_e(\mathbf{0})$ where $\mathbf{0}$ is the all 0s vector which in turn means all the vectors in the line are at most t from L.

Case 2: $|Support(u) \cup Support(v)| \geq e + 1$ Since u and v each have weight at most e, the intersection of their supports can be of cardinality at most $e - 1$. For each of the coordinates in the intersection of the supports, there can be at most one vector in $\ell_{u,v}$ such that the entry in that coordinate is 0. Therefore, there are at most $e - 1$ vectors in $\ell_{u,v}$ that are contained in the ball $B_e(\mathbf{0})$ where $\mathbf{0}$ is the all 0s vector.

To conclude this case, we need to demonstrate that there exists no codeword $c \neq \mathbf{0}$ such that the line $\ell_{u,v}$ intersects with a vector inside the ball of radius e around c. Assume for contradiction there exists a codeword c and vector w of weight at most e such that $c + w \in \ell_{u,v}$. Then we have that

$$c + w = u + \alpha v$$

This means that c is equal to the sum of three vectors each of weight at most e. Now we arrive at a contradiction because the minimum distance of L is $(n - k + 1)$ and $e < (n - k + 1)/3$.

■