Secure Multiparty Computation with Sublinear Preprocessing

Elette Boyle[1], Niv Gilboa[2], Yuval Ishai[3], and Ariel Nof[3]

[1] Reichman University (IDC Herzliya), and NTT Research, Israel.
`eboyle@alum.mit.edu`
[2] Ben-Gurion University, Israel.
`gilboan@bgu.ac.il`
[3] Technion, Israel.
{`yuvali`,`ariel.nof`}`@cs.technion.ac.il`

Abstract. A common technique for enhancing the efficiency of secure multiparty computation (MPC) with dishonest majority is via *preprocessing*: In an offline phase, parties engage in an input-independent protocol to securely generate correlated randomness. Once inputs are known, the correlated randomness is consumed by a "non-cryptographic" and highly efficient online protocol.
The correlated randomness in such protocols traditionally comes in two flavors: multiplication triples (Beaver, Crypto '91), which suffice for security against semi-honest parties, and *authenticated* multiplication triples (Bendlin et al., Eurocrypt '11, Damgård et al., Crypto '12) that yield efficient protocols against malicious parties.
Recent constructions of pseudorandom correlation generators (Boyle et al., Crypto '19, '20) enable concretely efficient secure generation of multiplication triples with *sublinear communication complexity*. However, these techniques do not efficiently apply to authenticated triples, except in the case of secure two-party computation of arithmetic circuits over large fields.
In this work, we propose the first *concretely efficient* approach for (malicious) MPC with preprocessing in which the offline communication is *sublinear* in the circuit size. More specifically, the offline communication scales with the *square root* of the circuit size.
From a feasibility point of view, our protocols can make use of any secure protocol for generating (unauthenticated) multiplication triples together with any *additive* homomorphic encryption. We propose concretely efficient instantiations (based on strong but plausible "linear-only" assumptions) from existing homomorphic encryption schemes and pseudorandom correlation generators.
Our technique is based on a variant of a recent protocol of Boyle et al. (Crypto '21) for MPC with preprocessing. As a result, our protocols inherit the succinct correlated randomness feature of the latter protocol.

1 Introduction

Protocols for secure multiparty computation (MPC) [44, 28, 2, 18] enable a set of parties with private inputs to compute a joint function of their inputs while revealing nothing but the output. Optimizing the asymptotic and concrete efficiency of MPC protocols has been the topic of a large body of work. The question is particularly challenging when considering security against *malicious* adversaries, who can actively corrupt parties.

A successful approach for the design of such protocols is to employ *preprocessing*. Before the inputs are known, the parties run an offline protocol to generate correlated secret randomness, which is consumed by a lightweight and typically "non-cryptographic" (or "information-theoretic") online protocol.[4] This model, known also as the offline/online model, is in particular appealing when no honest majority can be guaranteed, since it allows to push the heavy "cryptographic" part of the protocol to the offline phase, minimizing the cost of the online protocol. Originating from the work of Beaver [1], who showed how to use "multiplication triples" for secure arithmetic computation with dishonest majority, many protocols for secure computation make extensive use of correlated randomness [3, 23, 31, 39, 21, 24, 22, 20, 11].

Most of the above works design protocols in the preprocessing model with security against *malicious* adversaries. A powerful recurring technique uses homomorphic MACs to authenticate the values produced by the online protocol [3, 23]; the resulting correlation is a form of "authenticated" multiplication triples. Indeed, the so-called "SPDZ" line of work serves as a leading approach in this area, spawning a range of optimizations, implementations, and improvements, e.g. [35, 36, 20, 5, 17]. Another recent approach includes compilers based on sublinear distributed zero-knowledge [6, 14]. As per design, in all these protocols the bulk of the work lies in the preprocessing phase. In particular, the typical communication complexity of this phase in existing protocols is by orders of magnitude higher than the size of the circuit being evaluated.

Recent constructions of *pseudorandom correlation generators* (PCGs) [9, 8, 10] demonstrate promising potential for improving the communication demands of certain preprocessing procedures. PCGs provide a means for parties to locally expand short correlated seeds into long pseudorandom instances of certain correlations, without communication. Indeed, recent PCG constructions based on Learning Parity with Noise (LPN) or its Ring-LPN variant enable *concretely efficient* secure generation of many multiplication triples, with *sublinear communication* and good concrete efficiency, including the secure generation of the seeds [9, 10]. This directly yields a practical, sublinear-communication preprocessing for MPC with semi-honest security.

However, these techniques do not generally apply to the more complex correlation of *authenticated* multiplication triples, necessary for extending this approach to protocols with malicious security. Concretely efficient PCGs for authenticated triples exist only in the limited setting of 2-party correlations over

[4] This can be formalized by requiring the existence of alternative correlated randomness, which is computationally indistinguishable from the one generated by the offline protocol, and given which the entire protocol is information-theoretically secure.

large fields [10].[5] Using PCG-generated pairwise authenticated triples in the style of "BDOZ" [3] may be feasible for arithmetic circuits and a small number of parties, but result in online communication that scales quadratically in the number of parties. In the case of 2-party evaluation of Boolean circuits, PCGs for OT can be used to generate multiplication triples over $\mathbb{F}_2$, enabling semi-honest online protocols with 2 bits per party per gate, whereas for malicious security one needs to communicate 2 elements of a big finite field per gate.

To conclude, current PCG machinery cannot be efficiently used to generate *authenticated* triples that support an online protocol that scales linearly with the number of parties, or alternatively even 2-party protocols for Boolean circuits. Consequently, all practical protocols for MPC with preprocessing in these settings require the communication complexity of preprocessing to be much bigger than the circuit size.

1.1 Our Results

We provide new feasibility and concrete efficiency results for secure multiparty computation (MPC) in the dishonest majority setting. Our general approach can be instantiated to give the first practical *sublinear-communication* methods for generating "SPDZ-style" correlations, in the sense of achieving malicious security with an online phase that is both non-cryptographic and has linear communication in both the circuit size and the number of parties.

Our approach does not require authenticated multiplication triples as in SPDZ [23], but rather makes use of only semi-honest (non-authenticated) triples together with additional preprocessing material that builds on a variant of a recent protocol of Boyle et al. [14]. As a consequence, our protocols inherit the succinct correlated randomness feature of the latter protocol—that is, the additional preprocessing material (beyond multiplication triples) is only sublinear in the circuit size. From a concrete efficiency point of view, our approach is also attractive for Boolean circuits in the 2-party case, as there are no concretely efficient PCG for generating binary authenticated triples, whereas PCG for OT can generate binary non-authenticated triples.

More concretely, our protocols support secure computation in the preprocessing model of arithmetic circuits over any finite field or ring $\mathbb{Z}_{2^k}$. We say that the online phase is "information theoretic" (or "non-cryptographic") in the sense that the correlated randomness distribution D is computationally indistinguishable from some distribution D' for which executing the online phase with D' induces true information theoretic security. The offline preprocessing phase has communication that scales with the *square root* of the circuit size. Our protocol can make use of any secure protocol for generating (unauthenticated) multiplication triples, together with any additively homomorphic encryption.[6]

This constitutes a new feasibility result.

[5] While these limitations can in some cases be circumvented [9, 40, 25], this comes at a big additional cost.

[6] Implied, e.g., by any of the Quadratic Residuosity, Learning with Errors, or Decisional Composite Residuosity assumptions.

Theorem 1 (Sublinear-setup MPC with preprocessing, informal). *Let C be an arithmetic circuit of size $|C|$ (counting multiplication gates, inputs and outputs) over a ring R, where R is either a finite field $\mathbb{F}$ or the ring $\mathbb{Z}_{2^k}$. Then, there exists a secure n-party MPC protocol in the preprocessing model that computes C with security against up to $n-1$ malicious parties and the following features:*

- Sublinear offline communication*, consisting of a number of R-elements that scales as $O(\sqrt{|C|})$.*
- Information-theoretic online phase
- Per-party online communication *that does not grow with n: namely, $O(|C|)$ R-elements.*

Security is based on the assumptions underlying two cryptographic building blocks:

- *additively homomorphic encryption;*
- *sublinear-communication protocol for semi-honest generation of pseudorandom (un-authenticated) multiplication triples over R: e.g., implied by LPN.*

Note that the information-theoretic nature of the online phase makes the task of achieving sublinear offline communication highly nontrivial, as opposed to trivial solution approaches which simply ignore the offline phase (zero offline communication) and perform a complete secure computation in the online phase.

We additionally propose concretely efficient instantiations of our main theorem (based on strong but plausible "linear-only" assumptions) from existing homomorphic encryption schemes and existing pseudorandom correlation generators.

The overhead of our online phase is quite modest and in many settings is dominated by the communication of the semi-honest baseline protocol, even over a fast 1 Gbps network, e.g. for an arithmetic circuit of 2^{20} gates over a prime field of 60 bits (and soundness error 2^{-50}).

1.2 Technical Overview

Starting point: Distributed zero knowledge and BGIN'21. We follow in line with a collection of recent prior works using sublinear distributed zero knowledge machinery toward low-communication solutions for compiling semi-honest to malicious security [6, 12–14].

The high-level structure of these protocols is as follows. Within the protocol, the parties begin by running the underlying semi-honest secure protocol aside from the final step. Then, before exchanging the final messages and revealing outputs, the parties first jointly execute a verification phase in which correctness of the first phase is asserted. This is done by generating and verifying *zero knowledge proofs on distributed data* (hereafter "Distributed zero knowledge"), which can be done with sublinear proof size for simple languages [6]. Distributed zero knowledge (DZK) proofs consider a setting with a single prover and multiple verifiers, who each hold pieces of the statement; the prover sends a share of the proof to each verifier, and the verifiers interact amongst themselves in order to verify the proof.

The specific structure and use of the DZK proof system within the MPC protocol application varies across works: either each party acts separately as prover to assert his own proper behavior [6, 12, 14], or the parties jointly emulate the prover on the collective set of generated values, which no single party knows in full [6, 14].

A complication arises in any multi-party setting with more than a single corrupted party, as collusion may take place between a proving and a verifying entity. In order to provide soundness, the statement being proved must be somehow robustly held across parties, such that a corrupt verifier cannot modify his piece of the statement to enable the proof to improperly be accepted. Such robustness is natural in the case of MPC with an honest majority, where the honest parties themselves holds sufficient information to determine the (secret) statement. For the case of dishonest majority, ensuring robustness is less clear. A main idea of [14] (hereafter referred to as "BGIN'21") is that correlated randomness generated during preprocessing can be designed so as to function as an additional "dealer" party whose actions must be determined independent of the inputs, but whose behavior is *guaranteed to be honest.*

Of course, for this approach to work, it is crucial that the correlated randomness is indeed generated honestly. In the idealized preprocessing model, as considered in [14], this comes for free: the parties are assumed to be given honestly generated samples from the correlated randomness from an ideal honest dealer. Our challenge is to remove this assumption, and to generate these values as part of the protocol in a manner that still suffices for overall security, in *sublinear communication.* Moreover, we would like to do so without resorting to expensive general-purpose tools such as fully homomorphic encryption. Instead, we will rely on any *additively homomorphic* encryption. Doing so will require us to modify the BGIN'21 correlation and protocol in the process.

The BGIN'21 dealer (slightly modified). Consider the DZK joint prover emulation approach. After the semi-honest execution, the parties wish to jointly prove that for every multiplication gate, the shares they hold of the output wire correspond correctly to the shares they hold of the two input wires. To compress this into a single verification instead of $|C|$, the parties sample random coefficients α^g for each multiplication gate g, and instead verify that this random linear combination of all gate-checks indeed verifies. This reduces the challenge to proving that single degree-2 multivariate polynomial evaluates to 0 on $O(|C|)$ secret-shared inputs.

To achieve this, each party takes part in 3 phases: (1) "Joint-Prover," computing its contribution of the jointly generated proof of correctness, (2) "Verifier query," jointly generating a verifier query challenge (a form of coin toss protocol), and (3) "Verifier answer," performing its role as one of the multiple verifiers.

The dealer (in addition to generating the semi-honest correlation material) must effectively perform each of these actions acting as an additional honest party, and commit to its answers to be revealed in the online phase. Concretely, for arithmetic circuit C over ring R, the dealer must perform the following generation tasks:

1. **Semi-honest multiplication triples** (for semi-honest computation): For each gate g, additive shares of random $r_1^g, r_2^g \in R$ and $r_1^g \cdot r_2^g \in R$.
2. **Random compressing coefficients**: Selecting random (or sufficiently high-entropy) linear coefficients α^g for each gate g, so that correctness of all multiplication gates can be checked in a compressed manner via a random linear α^g-combination of the individual verification polynomials. A concise description of all α^g is shared across the parties to be revealed and reconstructed in the online phase during verification. In [14], each α^g is taken to be the gth power of single random value $\alpha \leftarrow R$.
3. **Prover contribution**: The "dealer party's share" of the jointly computed DZK proof. This consists of simply proof-size many random share values $s_i \in R$, secret shared to the parties in such a way that the value of each s_i cannot be changed by malicious parties in the online phase.
4. **Verifier query generation**: Random polynomial evaluation point $\tau \in R$.
5. **Verifier answer contribution** Computation of verifier query answer as a function of the dealer's share of the statement (a function of items 1 and 2), its shares of the proof (item 3), and the verifier query (item 4).

When using a D-ZK proof with multiple rounds of interaction (as in BGIN'21), Step 3 above is replaced by a sequence of: (a) generating random Verifier challenges, and (b) computing the Prover contribution to this round of interaction.

Some parts of items 1-5 above are not a problem toward our goal. As discussed, recent constructions of PCGs enable concretely efficient secure generation of many semi-honest multiplication triples with sublinear communication [9, 10], taking care of item 1. In addition, we can easily support sampling and committing to random shares and of a random point as in items 3 & 4.

The problems are items 2 and 5. In the current state as in BGIN'21, the computation of item 5 requires performing a secure computation that is both linear in the circuit size, *and* high degree. The degree comes from multiple places: from the high powers of α^g, from the recursive DZK proof structure (typically executed interactively), and the DZK verifier answer procedure which is computed as a function on top of these (already high-degree) expressions.

Expressing dealer with bilinear structure. We thus devise a new approach. Our idea is to modify the BGIN'21 protocol, so as to make the corresponding items 2-5 of dealer computation expressible as a single *bilinear* pairing computation between two types of values: (1) ones that are (already) held additively secret shared across the parties, e.g. from the semi-honest multiplication triples, and (2) a small, sublinear-size set of other secret values.

Given this structure, we can securely emulate the dealer with low communication using additively homomorphic encryption. Namely, the parties will generate *encryptions* of the small set of secret values (requiring sublinear communication, comparable to the small number of values). Then they homomorphically compute encrypted shares of the desired paired output using the public ciphertexts and their additive shares, *without* communication. Since the size of the dealer's output is short—equivalently, the bilinear maps are highly compressing—the parties can then execute a standard protocol for jointly decrypting the resulting ciphertexts, again with sublinear communication.

In turn, it hence remains to achieve the above structural goal for the dealer's computation.

First, we observe that replacing the recursive multi-round DZK with a simpler non-recursive DZK construction of [6] (albeit with $\sqrt{|C|}$ instead of $\log |C|$ communication cost) immediately alleviates one source of cost. The new prover and verifier procedures become simply degree 2. In particular, in this construction, the computation of a verifier's answer corresponds to interpreting $\sqrt{|C|}$-size collections of symbols from the proof and statement as coefficients of a polynomial, and evaluating each at the point τ chosen as the verifier query (in dealer Step 4). Said in different words, each polynomial evaluation is an inner product between the corresponding $\sqrt{|C|}$-length coefficient vector with the vector formed by the corresponding $\sqrt{|C|}$ powers $\tau^0, \tau^1, \ldots, \tau^{\sqrt{|C|}}$ of τ. Note that while there are several different blocks of symbols, they are each inner-producted with the *same* powers-of-τ vector.

If we could reach a state where the symbols of the proof and statement were held as additive secret shares, then we would in fact have reached our goal, where the powers-of-τ form the sublinear-size set of other values (to be encrypted). However, this is not yet the case. This is because the "statement" that must be proved is formed by the *α^g-linear-compressed combination* of the multiplication triple values, as opposed to the values themselves.

It is worth emphasizing an important difference between our setting and typical usage of the α^g-linear-compression technique, which is employed broadly in protocol design. Typically, coefficients α^g are chosen *after* the parties are already committed to their the semi-honest protocol execution, in which case there is no need for secrecy, and they are simply public values. In contrast, in this setting (as in BGIN'21), for the dealer to function as an honest party, the selection of these coefficients must be made already in the *preprocessing* phase, but kept secret through the semi-honest protocol execution. In turn, the traditionally simple linear compression here constitutes a nontrivial secure computation.

In fact, because of this, there is a problem even if the parties *already* somehow held secret shares of random values α^g. This is because multiplication by a coefficient α^g still amounts to an additional degree of secure multiplication. Since computation of the dealer's contribution to the verification polynomial already requires degree 2 terms, this additional multiplication pushes out of hope for turning the computation into a bilinear operation.

Instead, we identify a new method for computing the same overall verification polynomial evaluation expression between the "dealer" party and online parties, which enables pushing more work to the online parties, but which means that the linear combination coefficients do not need to be used by the dealer at all in his calculations. (In some sense, this comes from an alternative perspective and goal than from BGIN'21, which focused primarily on optimizing the online portion of the protocol while assuming an honest dealer.) This step uses the fact that the online parties hold secret shares not only of the masks r_1, r_2 of the input wires of any multiplication gate, but also shares of their product, r_1r_2, leveraging the specific structure of semi-honest multiplication triples. In particular, terms in the verification polynomial of the form $\alpha^g r_1 r_2$, which were computed by the dealer in BGIN'21, can be jointly computed by the online parties if the α^g is made public, using the (r_1r_2) additive shares.

In doing so, we not only remove the extra secure multiplication of scaling by the α^g, but also the remaining question of how these coefficients α^g should be generated. What results is a successful expression of the dealer in the desired bilinear form, between additively secret shared values and powers of τ, which we will denote by π_{BL}.

Putting the pieces together. Obtaining the feasibility result follows in a few steps.

We prove that the above-described modifications to the dealer's ideal functionality still suffice, with corresponding adjustments in the online portion, to yield security in the overall protocol.

Combining the above ideas, we then obtain a *semi-honest* secure protocol that securely evaluates the (new) dealer functionality with sublinear communication. Namely, the parties: (1) run the sublinear-communication protocol for semi-honest generation of pseudorandom (un-authenticated) multiplication triples; (2) run a secure protocol for randomly sampling τ (as dictated by the [6] DZK; either from the ring R, or an extension ring if R is small) and generating AHE encryptions c_i of the powers $\tau^0, \tau^1, \ldots, \tau^{\sqrt{|C|}}$, (3) locally evaluate encrypted shares of the bilinear form π_{BL} by computing the compressing linear combination of the AHE ciphertexts c_i with the corresponding secret shares of the relevant values; (4) exchanging the resulting output-share ciphertexts and additively combining across parties, resulting in encryptions of the outputs of π_{BL}; and (5) executing a secure protocol for jointly decrypting the resulting ciphertexts.

In order to achieve the same dealer emulation with *malicious* security, we leverage a generic communication-preserving compiler of Naor and Nissim [38] (building on [28, 37]). Using this compiler a semi-honest secure protocol can be compiled into a maliciously secured protocol for the same functionality with sublinear additive communication cost. Since the compiler only requires collision-resistant hash functions, which are implied by additively homomorphic encryption [32], this does not require introducing new assumptions. This implies the final result. In the full version of the paper, we propose a maliciously secure protocol with concrete efficiency that is based on making two stronger, but quite plausible, assumptions on the AHE: being "linear-only" and having a threshold encryption variant. See discussion at the end of Section 6.

2 Preliminaries

Notation. Let $P_1, \ldots, P_n$ be the parties participating in the protocol. We use $[n]$ to denote the set $\{1, \ldots, n\}$. Let R be a ring which is either a finite field $\mathbb{F}$

or the ring $\mathbb{Z}_{2^k}$ and let $|R|$ be its size. Finally, let κ be the security parameter. We use bold letters to represent vectors and $\boldsymbol{v}[j]$ to denote the jth entry of the vector $\boldsymbol{v}$. When we write $\boldsymbol{u} \cdot \boldsymbol{v}$ we refer to the *inner product* between the two vectors. We use $[\![x]\!]$ to denote an additive sharing of x. When we write $[\![\boldsymbol{x}]\!]$, we mean that *each* entry in $\boldsymbol{x}$ is additively shared across the parties.

2.1 Security Definitions

In our setting, there is a set of n parties who wish to jointly run some computation. We assume that all parties are connected via point-to-point secure channels, which enable them to send messages to each other. In this work, we will typically consider secure computation of arithmetic circuits (with addition and multiplication gates) over a finite ring R, where R can either be fixed or be given as part of the circuit description. In particular, the case of Boolean circuits is captured by $R = \mathbb{F}_2$. For security, we use the standard ideal/real world definition from [27, 16].

MPC with Preprocessing Our main result refers to an MPC protocol that employs a "cryptographic" input-independent offline protocol, followed by a "non-cryptographic" input-dependent online protocol. As a building block, we will rely on MPC protocols in a hybrid model in which the offline protocol is replaced by an ideal source of correlated randomness $\mathcal{D}$ that is generated by a trusted dealer.

In this correlated randomness model, we consider protocols for arithmetic circuits that offer security up to an "additive" attack on intermediate wires in the circuits. Most information-theoretic protocols that offer the weaker form of *semi-honest* security also satisfy this notion of security with additive attacks.

We formalize the notion of "security-up-to-additive-attacks" [26], in the setting of MPC with dishonest majority in the correlated randomness model. This security model applies to the class $\mathcal{F}$ of arithmetic circuits over a ring R, and allows the ideal-world adversary $\mathcal{S}$ to (blindly) pick a tampering function that adds a chosen value from R for each wire of the circuit. Concretely, we allow additive attacks on *input wires to multiplication gates and on the circuit's output wires*. The trusted party in the ideal world then determines the output of the honest parties by computing the circuit over the parties' inputs, applying the chosen additive error to each wire.

MPC with non-cryptographic online phase. An MPC protocol in the preprocessing model is similar to the above model of MPC with correlated randomness, except that the correlated randomness $\mathcal{D}$ is securely generated by an *offline protocol*, instead of being distributed by a trusted dealer. As before, the correlated randomness $\mathcal{D}$ can then be consumed by an online protocol. The main advantage of this offline-online paradigm is that it allows the online protocol to be "non-cryptographic" (or information-theoretic), which typically translates to good concrete efficiency. We formalize this notion by requiring that the correlated randomness produced by an honest execution of the offline protocol can be replaced by a *computationally indistinguishable* distribution $\mathcal{D}$, such that given $\mathcal{D}$, the online protocol is *information-theoretically* secure.

Definition 1 (PMPC with non-cryptographic online phase). *Let $\mathcal{F}$ be the class of n-party functionalities represented by arithmetic circuits C. An MPC protocol for $\mathcal{F}$ with preprocessing and non-cryptographic online phase (or PMPC protocol for short) is defined by a pair of protocols $\Pi = (\Pi_{\mathsf{offline}}, \Pi_{\mathsf{online}})$ such that:*

- *Π_{offline} is invoked with public inputs $(1^\kappa, 1^n, 1^{|C|}, R)$, where $|C|$ is the size of an arithmetic circuit $C \in \mathcal{F}$ over $\mathcal{R}$. It terminates with each party P_i outputting a local random output $\boldsymbol{Z}_i$.*
- *Π_{online} is invoked on public inputs $(1^\kappa,\ 1^n,\ C)$ and local inputs $(\boldsymbol{Z}_i,\ x_i)$ held by each P_i and ends with P_i outputting y_i.*

We make the following security requirements:

- *The protocol obtained by first running Π_{offline} and then Π_{online} securely realizes $\mathcal{F}$.*
- *There exists an ideal correlation generator $\mathcal{D}(1^\kappa, 1^n, 1^{|C|}, \mathcal{R})$, outputting $(\boldsymbol{Z}'_1, ..., \boldsymbol{Z}'_n)$, such that:*
  1. *The output of $\mathcal{D}$ is computationally indistinguishable from the output of an honest execution of Π_{offline}.*
  2. *If we feed Π_{online} with the output of $\mathcal{D}$, the resulting protocol realizes $\mathcal{F}$ with* statistical *(information-theoretic) security.*

In fact, to capture a minimal notion of MPC with non-cryptographic online phase, it suffices to relax the latter requirement (on Π_{online} with $\mathcal{D}$) to information-theoretic *semi-honest* security. However, the protocols we consider here satisfy the stronger property.

2.2 Fully Linear Proof Systems

A main technical building block in [14] is a *fully linear* proof system [6], enabling information-theoretic sublinear-communication zero-knowledge proofs on secret-shared or distributed input statements. In a nutshell, *zero-knowledge fully linear interactive oracle proof* (zk-FLIOP) is an information-theoretic proof system in which a prover P wishes to prove that some statement about an input x to a verifier V. In each round of the protocol, P produces a proof which, together with x, can be queried by V using *linear queries* only. Then, a public random challenge is generated and the parties proceed to the next round. At the end, the verifier V accepts or rejects based on the answers it received to its queries.

Definition 2 (Public-coin zk-FLIOP [6]). *A public-coin fully linear interactive proof system over R with ρ-round and ℓ-query and message length $(u_1, \ldots, u_\rho) \in \mathbb{N}^t$, consists of a randomized prover algorithm P and a deterministic verifier algorithm V. Let the input to P be $x \in R^m$ and let $r_0 = \perp$. In each round $i \in [\rho]$:*

1. *P outputs a proof $\pi_i \in R^{u_1}$, computed as a function of x, $r_1, \dots, r_{i-1}$ and $\pi_1, \dots, \pi_{i-1}$.*
2. *A random public challenge r_i is chosen uniformly from a finite set S_i.*
3. *ℓ linear oracle queries $q_1^i, \dots, q_\ell^i \in R^{m+u_i}$ are determined based on $r_1, \dots, r_i$. Then, V receives ℓ answers $(\langle q_1^i, x||\pi_i\rangle, \dots, \langle q_\ell^i, x||\pi_i\rangle)$.*

At the end of round ρ, V outputs accept *or* reject *based on the random challenges and all the answers to the queries.*

Let $\mathcal{L} \subseteq R^m$ be an efficiently recognizable language. We say that ρ-round ℓ-query interactive fully linear protocol $(\mathsf{P_{FLIOP}}, \mathsf{V_{FLIOP}})$ *over R is* zero-knowledge fully linear interactive oracle proof *system for $\mathcal{L}$ with soundness error ϵ if it satisfies the following properties:*

– Completeness: *If $x \in \mathcal{L}$, then $\mathsf{V_{FLIOP}}$ always outputs* accept*.*
– Soundness: *If $x \notin \mathcal{L}$, then for all P^*, the probability that $\mathsf{V_{FLIOP}}$ outputs* accept *is at most $2^{-\epsilon}$.*
– Zero knowledge: *There exists a simulator $\mathcal{S}_{\mathsf{FLIOP}}$ such that for all $x \in \mathcal{L}$ it holds that $\mathcal{S}_{\mathsf{FLIOP}} \equiv \mathsf{view}_{[\mathsf{P_{FLIOP}}(x), \mathsf{V_{FLIOP}}]}(\mathsf{V_{FLIOP}})$ (where the verifier's view $\mathsf{view}_{[\mathsf{P_{FLIOP}}(x), \mathsf{V_{FLIOP}}]}(\mathsf{V_{FLIOP}})$ consists of $\{r_i\}_{i\in[\rho]}$, $\{(q_1^i, \dots, q_\ell^i)\}_{i\in[\rho]}$ and $\big(\langle q_1^i, x||\pi_i\rangle, \dots, \langle q_\ell^i, x||\pi_i\rangle\big)_{i\in[\rho]}$.*

In this paper, we will use this tool for degree-d languages. That is, languages for which membership can be checked using a degree-d polynomial. The following theorem, which will be used by us, states that for degree-d languages, there are zk-FLIOP protocols with sublinear communication and rounds in the size of the input and number of monomials.

Theorem 2 ([6]). *Let $q : R^m \to R$ be a polynomial of degree-d with M monomials, and let $\mathcal{L}_q = \{x \in R^m \mid q(x) = 0\}$. Let ϵ be the required soudness error. Then, there is a zk-FLIOP for $\mathcal{L}_q$ with the following properties:*

– Constant rounds, $d = 2$: *It has 1 round, proof length $O(\eta\sqrt{m})$, challenge length $O(\eta)$ and the number of queries is $O(\sqrt{m})$, where $\eta = \log_{|R|}\left(\frac{\sqrt{m}}{\epsilon}\right)$ when R is a finite field, and $\eta = \log_2\left(\frac{\sqrt{m}}{\epsilon}\right)$ when $R = \mathbb{Z}_{2^k}$. The computational complexity is $\tilde{O}(M)$ and the proof generation is a degree-2 function of the input x and the prover's secret randomness, determined by the circuit C and the public randomness.*
– Logarithmic rounds, $d \geq 2$: *It has $O(\log M)$ rounds, proof length $O(d\eta \log M)$, challenge length $O(\eta \log M)$and the number of queries is $O(d + \log M)$, where $\eta = \log_{|R|}\left(\frac{d \log m}{\epsilon}\right)$ when R is a finite field, and $\eta = \log_2\left(\frac{d \log m}{\epsilon}\right)$ when $R = \mathbb{Z}_{2^k}$. The computational complexity is $O(dM)$.*

2.3 Additively-Homomorphic Encryption (AHE)

Our main protocol can be based on any *additively homomorphic encryption* (AHE) scheme. An AHE scheme consists of algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Add})$ such that $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ satisfy the usual correctness and semantic security requirements of a public-key encryption scheme, and Add enables to add (more generally, linearly combine) a vector of encrypted messages.

More concretely, Gen is a key-generation algorithm which takes as input the security parameter 1^κ and outputs a pair of secret and public keys (sk, pk), where pk includes a description of a finite plaintext ring R, Enc is an encryption algorithm which takes the public key pk and a message $m \in R$ as inputs and outputs a ciphertext c, Dec is a deterministic decryption algorithm which takes the secret key sk and ciphertext c as inputs and outputs a message m (or a symbol $\perp$ in case of failure), and the randomized algorithm Add takes as input ciphertexts $\{c_j \in \mathsf{Enc}_{pk}(m_j)\}_{j\in[M]}$ and ring elements $a_0, \ldots, a_M \in R$ and outputs a *fresh* ciphertext $c \in_R \mathsf{Enc}_{pk}(a_0 + \sum_{j=1}^{M} a_j \cdot m_j)$. We use the notation $\mathsf{Add}\left((a_k)_{k=0}^{M}, (c_k)_{k=1}^{M}\right)$ for this operation. When we simply want to add M ciphertexts (i.e., $a_0 = 0$ and $a_k = 1$ for each $k \geq 1$), we simply write $\mathsf{Add}(c_1, \ldots, c_M)$.

The above definition captures a simple version of AHE in which one can combine an unbounded number of encrypted messages and the resulting ciphertext is distributed identically to a fresh encryption of the correct value. This notion can be satisfied by standard number-theoretic encryption schemes such as the Goldwasser-Micali scheme [29] or its generalized version due to Benaloh [19]. To accommodate other instantiations, such lattice-based schemes [42, 33], or using Pailler's encryption scheme [41] over a chosen plaintext modulus, one needs to slightly relax the definition to allow a statistical error in the output of Add that depends on the number of ciphertexts that are combined. While we chose the strict definition for simplicity, our results can be extended to use the relaxed variants of AHE that support such alternative instantiations.

2.4 Ideal Functionalities and Basic Building Blocks

$\mathcal{F}_{\text{coin}}$ - *coin tossing.* This ideal functionality who gives the parties fresh random coins. It can be implemented using any secure coin tossing protocol. In the context of our protocol, we can minimize the number of calls to $\mathcal{F}_{\text{coin}}$ by having the parties call $\mathcal{F}_{\text{coin}}$ once to obtain a seed from which they can locally derive a long vector of ring elements.

$\mathcal{F}_{\mathsf{bc}}$ - *Broadcast with selective abort.* This ideal functionality allows the parties to deliver a message msg to all the other parties, while giving the adversary the ability to cause any party to abort. Therefore, whenever we say throughout the paper that P_i broadcasts a message, we mean that it sends a message to all parties via $\mathcal{F}_{\mathsf{bc}}$. This functionality can be easily implemented by having P_i sends its message to all the parties, and then running an additional round where all parties compare the message they received. To amortize away the second-round comparison, a standard optimization technique is to batch the check for many messages by taking a random linear combination of all these message before the end of the protocol.

Authenticated secret sharing $\langle\cdot\rangle$. In some cases, we will need the parties to hold a secret sharing of x in an authenticated way, i.e., that allows the parties to securely reveal (with abort) the secret. We denote by $\langle x\rangle$ such a secret sharing and denote by open the opening procedure which receives $\langle x\rangle$ and guarantee that at the end either the parties will obtain x or abort the protocol (up to a negligible failure probability). To implement this over a field, one can use SPDZ-style information-theoretic MACs [3, 23]. We stress that this tool is used a *sublinear* number of times in our protocol, and so its cost is amortized away.

3 The BGIN Compiler [14]

In this section, we review the verification procedure of Boyle et al.(BGIN) [14], that enables compiling a semi-honest protocol in the pre-processing model into a maliciously secured protocol, with sublinear additional amount of correlated randomness. There are two conditions that the semi-honest protocol should satisfy for the compiler to work:

1. Additive security: The adversary is restricted to only adding errors to a set of wires W in the circuit (see Definition **??**).
2. "Star-sharing" Compliance: For each circuit's wire $w \in W$, the parties hold a masked value $\hat{x}_w = x_w - r_w$ and additive shares of the mask r_w. The dealer knows r_w and its shares.

For formal definition, we refer the reader to [14]. The above requirements are satisfied by the semi-honest protocol based on Beaver triples [1]. Here the set W consists of all input wires for multiplication gates and output wires of the circuit. To maintain the star-sharing invariant in the circuit-independent version of the protocol, the parties first locally convert their star shares on each input wire to a multiplication gate, into *additive* shares of the output. Then, they can carry-out linear operations over the additive shares. When they arrive to the next input wire to a multiplication gate, they interact to reveal the masked input and so on.

To achieve malicious security, SPDZ-style [23] protocols use authenticated Beaver triples. This means that for each multiplication gate g, the parties receive from the dealer shares of $(r_1, r_2, r_1 \cdot r_2)$ and $(r_1 \cdot \theta, r_2 \cdot \theta, r_1 \cdot r_2 \cdot \theta)$, where θ is a global secret key. To achieve statistical security of κ bits, the MACed triple should be generated over a large ring (e.g., if the computation is carried-out over $\mathbb{F}_2$, then the MACed triple should be over $\mathbb{F}_{2^\kappa}$). This implies a correlated randomness overhead of $O(|C| \cdot \kappa)$ ring elements for small fields.

The novel verification protocol of BGIN [14] avoids this and requires the dealer to provide only sublinear amount of correlated randomness beyond the semi-honest protocol. The idea works as follows. For each wire $w \in W$, the parties need to verify the consistency of the values shared on this wire, with the values shared on the wires that precede it. In other words, the parties verify that they hold a sharing of the correct value on w, given the sharings they hold on wires that feed w. Specifically, let G_w be the set of multiplication gates that feed w (i.e., that between their output wire and w there are no other multiplication gates). For each $g \in G_w$, let x_1^g, x_2^g be the two input wires to g. The parties wish to verify that $x_w - \sum_{g \in G_w} x_1^g \cdot x_2^g = 0$. Instead, it suffices for the parties to verify that

$$\begin{aligned} p &= \sum_{w \in W} \alpha_w \cdot (x_w - \sum_{g \in G_w} x_1^g \cdot x_2^g) \\ &= \sum_{w \in W} \alpha_w \cdot \left(\hat{x}_w + r_w - \sum_{g \in G_w} (\hat{x}_1^g + r_1^g) \cdot (\hat{x}_2^g + r_2^g) \right) = 0 \end{aligned} \tag{1}$$

where the α_ws are random elements given to the parties by the dealer (it suffices for the dealer to give a seed α from which all randomness is derived).

Next, note that each gate g_ℓ can feed several wires. For each multiplication gate g_ℓ, let W^{g_ℓ} be the set of wires w for which $g_\ell \in G_w$ (i.e., that g_ℓ's output feed these wires). Then, let $\gamma_\ell = \sum\limits_{w \in W^{g_\ell}} \alpha_w$. Thus, Eq. (1) can be written as

$$\begin{aligned} p &= \sum_{w \in W} \alpha_w \cdot (\hat{x}_w + r_w) - \sum_{g_\ell \in \mathsf{mult}} \gamma_\ell \cdot ((\hat{x}_1^{g_\ell} + r_1^{g_\ell}) \cdot (\hat{x}_2^{g_\ell} + r_2^{g_\ell})) \\ &= \sum_{w \in W} \alpha_w \cdot \hat{x}_w - \sum_{g_\ell \in \mathsf{mult}} \gamma_\ell \cdot (\hat{x}_1^{g_\ell} \cdot \hat{x}_2^{g_\ell}) + \sum_{w \in W} \alpha_w \cdot r_w \\ &\quad - \sum_{g_\ell \in \mathsf{mult}} \gamma_\ell \cdot (\hat{x}_1^{g_\ell} \cdot r_2^{g_\ell} + \hat{x}_2^{g_\ell} \cdot r_1^{g_\ell}) + \sum_{g_\ell \in \mathsf{mult}} \gamma_\ell \cdot (r_1^{g_\ell} \cdot r_2^{g_\ell}) \end{aligned}$$

Now, setting

$$\Lambda = \sum_{w \in W} \alpha_w \cdot \hat{x}_w - \sum_{g_\ell \in \mathsf{mult}} \gamma_\ell \cdot (\hat{x}_1^{g_\ell} \cdot \hat{x}_2^{g_\ell}) \tag{2}$$

$$\Gamma_i = \sum_{w \in W} \alpha_w \cdot r_{w,i} - \sum_{g_\ell \in \mathsf{mult}} \gamma_\ell \cdot (\hat{x}_1^{g_\ell} \cdot r_{2,i}^{g_\ell} + \hat{x}_2^{g_\ell} \cdot r_{1,i}^{g_\ell}) \tag{3}$$

and $\Omega = \sum\limits_{g_\ell \in \mathsf{mult}} \gamma_\ell \cdot (r_1^{g_\ell} \cdot r_2^{g_\ell})$, it follows that checking that $p = 0$ is equivalent to checking that $\Lambda + \sum_{i=1}^n \Gamma_i + \Omega = 0$.

Observe that the parties can locally compute Λ, each party P_i can locally compute Γ_i and the dealer can locally compute Ω. Leveraging this, the verification protocol in [14] works thus as follows:

1. Each party P_i computes Λ and Γ_i, while the dealer computes Ω.
2. The dealer star shares Ω to the parties by sending $\hat{\Omega} = \Omega - \omega$ and also hands a mask s_i to each party P_i.
3. Each party P_i star shares Γ_i to the parties by broadcasting $\hat{\Gamma}_i = \Gamma_i - s_i$.

4. Each party P_i proves that it shared the correct Γ_i using a zk-FLIOP proof system (see explanation below).
5. If all proofs terminated successfully, then the dealer sends $\sum_{i=1}^{n} s_i + \omega$ to all parties.
6. The parties locally compute p and check equality to 0. If it holds, the parties output accept. Otherwise, they output reject.

The two main observations here are: (i) Γ_i is computed via a 2-degree polynomial. Thus, by Theorem 2, there exists a zk-FLIOP to prove that Eq. (3) holds. (ii) All inputs to the zk-FLIOP are either known to *all* parties or known to the dealer. Thus, we can run the zk-FLIOP by letting P_i emulate the prover's role, and letting all the other parties *together with the dealer* emulate the verifier's role. Specifically, the prover star-shares the proof across the parties and the dealer, and then each verifier makes the linear queries over its shares of the proof and the inputs. The fact that each piece of information is known to an honest participant (i.e., an honest party or the dealer) is what guarantee soundness. Leveraging the fact that from Theorem 2 the amount of communication in the proof can be made sublinear, we have that the amount of data sent by the verifying dealer is also sublinear in the size of the statement which, in our case, is similar to the size of the computed circuit. Finally, since all the computations made by the dealer in this protocol are over random data, it follows that the dealer can preprocess its messages and secret share it to the parties in an authenticated way, before the online computation begins. Overall, the amount of correlated randomness given to the parties for this protocol is sublinear.

Distributing the dealer. The authors in [14] did not provide a distributed protocol to compute the dealer. Instead, they viewed the dealer's role as a circuit to be computed, and proposed to compute it using any general-purposed MPC. Then, they showed that the number of multiplication gates in the dealer's circuit is approximately $4|C| + n \cdot 2|C|$ for n parties. That is, the size of the dealer's circuit grows linearly with circuit C and the number of parties n.

4 A New Simplified Verification Protocol

In this section, we present a modified verification protocol that will allow us eventually to distribute the dealer with sublinear communication. Our primary aim is to make the computation performed by the dealer be low depth (i.e., it is possible to represent the dealer using a low-depth circuit). A secondary goal is to make the size of the dealer's circuit be independent of the number of parties.

To this end, we first observe that the parties can locally compute an additive sharing of Ω as well. This holds since γ_ℓ is public and they hold shares of $r_1^{g_\ell} \cdot r_2^{g_\ell}$ (these are part of the semi-honest correlated randomness). It follows that now

$$\Gamma_i = \sum_{w \in W} \alpha_w \cdot r_{w,i} - \sum_{g_\ell \in \mathsf{mult}} \gamma_\ell \cdot \left(\hat{x}_1^{g_\ell} \cdot r_{2,i}^{g_\ell} + \hat{x}_2^{g_\ell} \cdot r_{1,i}^{g_\ell} + (r_1 \cdot r_2)_i^{g_\ell} \right)$$

Next, instead of letting each party prove that it star-shared the correct Γ_i, we will run a single proof where the parties emulate together the role of the zk-FLIOP prover. The dealer, however, still serves as a verifier, but now only *once* instead of n times. The goal now will be to prove that $\Gamma = \sum_{i=1}^{n} \Gamma_i$ is correct. That is, that

$$\Gamma = \sum_{w \in W} \alpha_w \cdot r_w - \sum_{g_\ell \in \mathsf{mult}} \gamma_\ell \cdot (\hat{x}_1^{g_\ell} \cdot r_2^{g_\ell} + \hat{x}_2^{g_\ell} \cdot r_1^{g_\ell} + (r_1 \cdot r_2)^{g_\ell}) \tag{4}$$

is correct given the star shares of Γ and the inputs. Our protocol to prove this is based on the constant round zk-FLIOP construction from [6] (the first item in Theorem 2) and requires $O(\sqrt{|C|})$ communication (and so $O(\sqrt{|C|})$ correlated randomness from the dealer). The idea is as follows. First, observe that letting

$$\boldsymbol{a} = ((\alpha_w)_{w \in W}, (-\gamma_\ell \cdot \hat{x}_1^{g_\ell}, -\gamma_\ell \cdot \hat{x}_2^{g_\ell}, -\gamma_\ell)_{g_\ell \in \mathsf{mult}}) \tag{5}$$

and

$$\boldsymbol{b} = ((r_w)_{w \in W}, (r_2^{g_\ell}, r_1^{g_\ell}, (r_1 \cdot r_2)^{g_\ell})_{g_\ell \in \mathsf{mult}}) \tag{6}$$

we have that $\Gamma = \boldsymbol{a} \cdot \boldsymbol{b}$ *where* $\boldsymbol{a}$ *is known to the parties and* $\boldsymbol{b}$ *is known to the dealer*. Note that both vectors are of size $|W| + 3|\mathsf{mult}| = 5|\mathsf{mult}| = 5|C|$. In our verification protocol, each party P_i first computes Γ_i and then star-shares it to the other parties, by broadcasting $\hat{\Gamma}_i = \Gamma_i - t_i$ to the other parties. Then, the parties locally compute $\hat{\Gamma} = \sum_{i=1}^{n} \hat{\Gamma}_i$. The parties then wish to verify that

$$\hat{\Gamma} + t - \boldsymbol{a} \cdot \boldsymbol{b} = 0 \tag{7}$$

where $t = \sum_{i=1}^{n} t_i$. The main observation here is that Eq. (7) represents a 2-degree polynomial over the inputs $\boldsymbol{a}, \boldsymbol{b}, \hat{\Gamma}, s$. Thus, by Theorem 2, there exists a zk-FLIOP to prove that Eq. (7) holds, where the proof's size is sublinear in the size of the input. However, unlike the BGIN compiler [14], here there does not exist a single prover who knows the entire input. We thus need to let the parties *emulate* jointly the role of the prover. In particular, the parties can locally compute additive shares of the proof and then star-share them to the other parties (exactly as with Γ_i). To enable this, we use the constant round zk-FLIOP from Theorem 2, where the proof generation itself is a degree-2 function of the input and prover's randomness. Since in Eq. (7) each input is either known to all parties, or, known to the dealer and *additivley* shared across the parties, it follows that each party can locally compute an additive share of the proof. As for the verifiers, the zk-FLIOP queries can be made over their shares, since the queries are linear. As in the BGIN compiler, we crucially rely on the fact that in our protocol, each piece of information (i.e., the inputs and the proof) is known either by each party or by the dealer, and so we can use the dealer as a verifier as well. By the linearity of the queries, it thus follow that the queries' answers can be reconstructed by each party and the dealer alone, thereby guaranteeing that an honest party will receive the correct answers. This means that if cheating took place and the statement is incorrect, then by the soundness property of the zk-FLIOP, it will be detected by any honest verifier (except for a small probability). As for privacy, the zero-knowledge property together with the fact that each party sees only masked values, guarantee that no private information is leaked during the execution.

$\underline{\Pi_{\mathsf{vrfy}}}$: Let $(\mathsf{P}_{\mathsf{FLIOP}}, \mathsf{V}_{\mathsf{FLIOP}})$ be a zk-FLIOP protocol with 1 round, ℓ queries, and message length $u \in \mathbb{N}$ for the polynomial in Eq. (7).

1. The trusted dealer $\mathcal{D}$:
   (a) Chooses a random mask $t_i \in \mathbb{F}$ for each $i \in [n]$ and hands it to P_i
   (b) Chooses a random mask $s_i \in \mathbb{F}^u$ for each $i \in [n]$ and hands it to P_i.
2. The parties call $\mathcal{F}_{\text{coin}}$ to receive α_w for each $w \in W$ (recall that W is the set of the circuit's output wire and multiplication gates' output wires). Alternatively, the parties call $\mathcal{F}_{\text{coin}}$ to receive α and expand it to α_w by setting $\alpha_w = \alpha^w$ or via a PRG.
3. The parties locally compute Λ (see Eq.(2)).
4. Each party P_i computes Γ_i and star-shares it to the parties by broadcasting $\hat{\Gamma}_i = \Gamma_i - t_i$.
5. The parties locally compute $\hat{\Gamma} = \sum_{i=1}^{n} \hat{\Gamma}_i$ (note that $\hat{\Gamma} = \Gamma - t$ where $t = \sum_{i=1}^{n} t_i$).
6. The parties jointly prove that Γ is correct:
   Let $I = (\hat{\Gamma}, t, \boldsymbol{a}, \boldsymbol{b})$ be the vector of inputs to the zk-FLIOP protocol. Let $I^P = (\hat{\Gamma}, 0, \boldsymbol{a}, 0)$ (i.e., a vector obtained by replacing all inputs not known to all parties in I with 0) and $I^D = (0, t, 0, \boldsymbol{b})$ (i.e., a vector obtained by replacing all inputs not known to the dealer in I with 0).
   Observe that: $I = I^P + I^D$.
   (a) Let $\pi = \mathsf{P}_{\mathsf{FLIOP}}(I)$ be the proof generated by the prover in the zk-FLIOP protocol. Then, each party P_i locally computes its share of the proof π^i and broadcasts $\hat{\pi}^i = \pi^i - s_i$ to all the other parties.
   (b) The dealer $\mathcal{D}$ choose a random challenge τ and hands to the parties.
   (c) Let $\hat{\pi} = \sum_{i=1}^{n} \hat{\pi}^i$ and $s = \sum_{i=1}^{n} s_i$. Let $q_1, \ldots, q_\ell$ be the query vector determined by $\mathsf{V}_{\mathsf{FLIOP}}$ based on τ. Then, the parties locally compute
   $$\hat{a}_1, \ldots, \hat{a}_\ell \leftarrow \langle q_1, \boldsymbol{I}^P || \hat{\pi} \rangle, \ldots, \langle q_\ell, \boldsymbol{I}^P || \hat{\pi} \rangle$$
   whereas the dealer computes
   $$\tilde{a}_1, \ldots, \tilde{a}_\ell \leftarrow \langle q_1, \boldsymbol{I}^D || s \rangle, \ldots, \langle q_\ell, \boldsymbol{I}^D || s \rangle.$$
   (d) The dealer $\mathcal{D}$ hands $\tilde{a}_1, \ldots, \tilde{a}_\ell$ to the parties, who locally compute
   $$a_1 = \hat{a}_1 + \tilde{a}_1, \ldots, \ a_\ell = \hat{a}_\ell + \tilde{a}_\ell$$
   (e) The parties run the decision predicate of $\mathsf{V}_{\mathsf{FLIOP}}$ on $a_1, \ldots, a_\ell$. If any party received reject, then it output reject. Otherwise, the parties proceed to the next step.

7. The parties locally compute $\hat{p} = \Lambda + \hat{\Gamma}$.
8. The dealer hands the parties t.
9. The parties locally compute that $p = \hat{p} + t$. If $p = 0$, they output accept. Otherwise, they output abort.

We begin by proving correctness, soundness and privacy for Π_{vrfy}, when Π_{vrfy} is ran over a finite field $\mathbb{F}$. The proof appears in the full version.

Proposition 1. *Assume that Π_{vrfy} is executed over a finite field $\mathbb{F}$, let W be the set of all output wires and input wires to multiplication gates and let Δ_w be the additive error on wire $w \in W$. Then, Π_{vrfy} satisfies the following properties:*

1. Correctness: *If $\forall w \in W : \Delta_w = 0$ and all parties follow the protocol's instructions, then the honest parties always output* accept.
2. Soundness: *If $\exists w \in W : \Delta_w \neq 0$, then the honest parties output* accept *with probability at most $\frac{|W|}{|\mathbb{F}|} + \epsilon$, where ϵ is the soundness error of the zk-FLIOP protocol.*
3. Privacy: *For any adversary $\mathcal{A}$ controlling a subset of parties T of size$\leq n-1$, there exists a simulator $\mathcal{S}$ who receives $(\hat{x}_w, \Delta_w, r_{w,i})_{w \in W}$ and $\{(r_1 \cdot r_2)_i^{g_\ell}\}_{g_\ell \in \mathsf{mult}}$ for all $i \in T$ as input, and outputs a transcript $\mathsf{view}_{\mathcal{S}}$ such that $\mathsf{view}_{\mathcal{S}} \equiv \mathsf{view}_{A}^{\Pi_{\mathsf{vrfy}}}$.*

Working over small fields or the ring $\mathbb{Z}_{2^k}$. The soundness error of our protocol depends on the size of the field $\mathbb{F}$. When we compute the circuit over small fields, it is possible to run Π_{vrfy} over an extension field to reduce the error. This is carried-out by lifting each input to the verification protocol into the extension field. Suppose that we want the error to be $2^{-\sigma}$. Then, one can choose an extension field $\tilde{\mathbb{F}}$ such that $\frac{|W|}{|\tilde{\mathbb{F}}|} + \epsilon \leq 2^{-\sigma}$.

Similarly, when the circuit is computed over the ring $\mathbb{Z}_{2^k}$, we will run Π_{vrfy} over the extension ring $\mathbb{Z}_{2^k}[x]/f(x)$, i.e., the ring of polynomials with coefficients from $\mathbb{Z}_{2^k}$ modulo a polynomial $f(x)$ which is of the right degree and is irreducible over $\mathbb{Z}_2$. As shown in [6, 12], taking f of degree d, the number of roots of a polynomial of degree δ over $\mathbb{Z}_{2^k}[x]/f(x)$ is at most $2^{(k-1)d}\delta + 1$. Thus, the probability that $p = 0$ is at most $\frac{2^{(k-1)d} \cdot (|W|)}{2^{kd}} \approx \frac{|W|}{2^d}$. Hence, by choosing d appropriately, we can achieve a desired soundness error.

From an online to an offline dealer. In Π_{vrfy} the dealer only sends messages that depend on random data. Therefore, it can preprocess all its messages and secret share it to the parties in an authenticated way. This includes the masks t_i, the masks s_i and its share of the queries' answers.

Complexity. The amount of correlated randomness that we obtain depends on the size of the proof u and the size of the queries' answers ℓ in the zk-FLIOP emulation. By Theorem 2 both ℓ and u is square-root of the input's size to the relation given in Eq. (7). Thus, the amount of correlated randomness is $O(\sqrt{|W|})$.

The communication cost of the protocol per party includes star sharing Γ_i, sending the share of the proof π^i and opening the correlated randomness which is shared in an authenticated way. Thus, it depends on the size of the proof and size of the correlated randomness. Hence, as the correlated randomness, it is of size $O(\sqrt{|W|})$.

Finally, the computational work includes computing the random coefficients α_w, computing Λ, and the work in the zk-FLIOP protocol. The number of arithmetic operations for the first two computations is linear in the size of $|W|$, whereas by Theorem 2, the number of operations for the latter is $\tilde{O}(M)$, where M is the number of monomials in Eq. (7). As the number of monomials is $O(|W|)$, we obtain that the computaional work is $\tilde{O}(|W|)$.

Summing the above and using Theorem 2 we obtain:

Proposition 2. *Let ε be a statistical error bound. Then, Protocol Π_{vrfy} has communication cost of $O(\sqrt{|W|}\cdot\kappa)$ per party, computational work $\tilde{O}(|W|)$ and the amount of correlated randomness provided by the dealer is $O(\sqrt{|W|}\cdot\kappa)$, where $\kappa = \log_{|\mathbb{F}|}\left(\frac{\sqrt{|W|}}{\varepsilon}\right)$ when the input is defined over a finite field $\mathbb{F}$, $\kappa = \log_2\left(\frac{\sqrt{|W|}}{\varepsilon}\right)$ when the input is defined over the ring $\mathbb{Z}_{2^k}$ (where W is the set of the ciruit's output wires and input wires to multiplication gates).*

4.1 A Concrete Instantiation for the zk-FLIOP Protocol

In this section, we present a concrete instantiation for the constant round zk-FLIOP protocol used in Π_{vrfy} based on the fully linear PCP construction given in [6]. Consider a prover $\mathcal{P}$ who wants to prove that $c - \boldsymbol{a}\cdot\boldsymbol{b} = 0$ to a verifier V, where $\boldsymbol{a}$ and $\boldsymbol{b}$ are vector of elements of size m.

To prove the correctness of the statement, the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ are divided into M vectors, i.e., $\boldsymbol{a} = \boldsymbol{a}_1||\cdots||\boldsymbol{a}_M$ and $\boldsymbol{b} = \boldsymbol{b}_1||\cdots||\boldsymbol{b}_M$. This means that the statement to be proven can be written as $c - \sum_{k=1}^{M}\boldsymbol{a}_k\cdot\boldsymbol{b}_k$. Denote the number of elements in each vector by L, and so $L\cdot M = m$.

The first step of $\mathcal{P}$ is to choose random vectors $\boldsymbol{a}_0$ and $\boldsymbol{b}_0$. The, define L polynomials of degree-M such that:

$$\forall e \in [L] : f_e(0) = \boldsymbol{a}_0[e], \ldots, f_e(M) = \boldsymbol{a}_M[e].$$

That is, the evaluation of f_e on the point k is the eth entry of the vector $\boldsymbol{a}_k$. Similarly, define additional L polynomials of degree-M by setting:

$$\forall e \in [L] : g_e(0) = \boldsymbol{b}_0[e], \ldots, g_e(M) = \boldsymbol{b}_M[e].$$

Next, define an additional polynomial q by letting $q(x) = \sum_{e=1}^{L} f_e(x)\cdot g_e(x)$. By the way q is defined it follows that:

- For each $k \in \{0,\ldots,M\}$ it holds that $q(k) = \boldsymbol{a}_k\cdot\boldsymbol{b}_k$
- The degree of q is $2M$.

To fully define q the prover $\mathcal{P}$ thus computes M more additional points on q. This can be done by first interpolating the f_e and g_e polynomials to compute these polynomials on the points $M+1,\dots,2M$ and then computing $q(x)$ for each $x \in \{M+1,\dots,2M\}$.

The protocol then proceeds as follows:

1. **Proof**: The prover $\mathcal{P}$ defines the proof by setting
$$\pi = \left(\{f_e(0)\}_{e=1}^{L}, \{g_e(0)\}_{e=1}^{L}, q(0), \dots, q(2M)\right)$$
2. **Challenge**: A random point τ is chosen (such that $\tau \notin \{0,\dots,M\}$).
3. **Query**: The linear queries over the proof and inputs are defined such that the verifier V is given:
   - For each $e \in [L]$: $f_e(\tau)$ and $g_e(\tau)$.
   - $q(\tau)$
   - $c - \sum_{k=1}^{M} q(k)$
4. **Decision**: The verifier V checks that
   (a) $q(\tau) = \sum_{e=1}^{L} f_e(\tau) \cdot g_e(\tau)$
   (b) $c - \sum_{k=1}^{M} q(k) = 0$
   If both equations hold, then V outputs accept. Otherwise, it outputs reject.

Observe that in check 4a, the verifier ensures that P defined the proof correctly, i.e., computed the polynomial q honestly. Then, it can verify that the statement holds via check 4b. Note also that privacy (zero-knowledge) is maintained in this protocol by the additional random point defined for each polynomial (i.e., $f_e(0)$ and $g_e(0)$). These random points make the evaluation of each polynomial on the point τ look completely random.

Soundness. A malicious prover can succeed only if check 4a passes although $q(x)$ is not defined correctly. By the Schwartz-Zippel lemma, this event can happen when working over a finite field with probability bounded by $\frac{2M}{|\mathbb{F}|}$.

When the statement is defined over the ring $\mathbb{Z}_{2^k}$, then the verification protocol itself is executed over the extension ring $\mathbb{Z}_{2^k}[x]/f(x)$ (see above). In this case, if f is of degree d, then the cheating probability is bounded by $\frac{2^{(k-1)d}\cdot(2M)}{2^{kd}} \approx \frac{2M}{2^d}$.

Concrete Costs. Recall that L and M are parameter to choose under the constraint that $m = L \cdot M$. If we set $L = M = \sqrt{m}$, the cost becomes sublinear in m. Concretely, the size of the proof is $2L + 2M + 1 = 4\sqrt{m} + 1$. In addition, the size of the answers to the queries is $2L + 2 = 2\sqrt{m} + 2$.

Plugging in the above into our protocol, recall that $m = |W| + 3|\mathsf{mult}|$ (see Eq. (5) and (6)). Now, in the emulation of the zk-FLIOP in Π_{vrfy}, the parties communicate to star-share their additive shares of the proof π^i and communicate to reconstruct the queries' answers (here the parties open the shared masks of the answers which was given to them by the dealer). Overall, the communication cost is therefore roughly $6\sqrt{|W| + 3|\mathsf{mult}|} + 2$ elements sent by each party to the other parties.

The amount of correlated randomness handed by the dealer, however, is exactly $2\sqrt{|W| + 3|\mathsf{mult}|} + 1$ elements. This is due to the fact that the dealer acts only as a verifier and does not hold a share of the proof (more accurately, the dealer only provides masks for the shares of the proof held by the parties; however, this is private randomness which is not required to be shared across the parties).

4.2 The Dealer's Ideal functionality $\mathcal{F}_{\mathsf{Dealer}}$

Based on the concrete instantiation described in the previous section, we now define an ideal functionality for the trusted dealer. Later, when we show how to distribute the dealer's work, we will show how to securely compute this functionality.

Recall that the dealer's work includes the following: (i) choosing random masks for the circuit's wires as part of the semi-honest correlated randomness; (ii) choosing random private masks for the parties to mask Γ_i and to mask the additive shares of the proof π^i; (iii) choosing a random challenge τ and (iv) computing the zk-FLIOP queries' answers based on i,ii and iii.

Observe that (iv) involves computing a random point on many polynomials. Recall that in the zk-FLIOP protocol, the verifier should receive $f_e(\tau), g_e(\tau)$ for each $e \in [L]$ and $q(\tau)$. The first $2L$ polynomials are of degree M, whereas q is of degree $2M$. In our protocol, the f polynomials correspond to the vector $\boldsymbol{a}$ (see Eq. (5)), which is being determined in the online computation and is known to *all* parties. In contrast, the g polynomials correspond to the vector $\boldsymbol{b}$ (see Eq. (6)) which is known to the dealer. Finally, the polynomial q is star-shared across the parties, meaning that it is additively shared between each party and the dealer.

To define the ideal functionality in a simple way, we first define a procedure π_{BL} that performs the bilinear computation between the coefficients of a set of polynomials and the powers of a point, which is what required to evaluate these polynomials on the point.

$\pi_{\mathsf{BL}}(\boldsymbol{G}, \tau, d)$:

1. Parse $\boldsymbol{G}$ as vectors $\boldsymbol{G}_1||\boldsymbol{G}_2||\ldots$ of size $d+1$.
2. For each $\boldsymbol{G}_e$: Let g_e be a degree-d polynomial defined using the points $\boldsymbol{G}_e[0], \ldots, \boldsymbol{G}_e[d]$. Then, covert the points into polynomial coefficients $\beta_{e,0}, \ldots, \beta_{e,d}$
3. Compute $\tau, \ldots, \tau^d$.
4. For each polynomial g_e, return $y_e = \sum_{j=0}^{d} \beta_{e,j} \cdot \tau^d$

Fig. 1. The procedure π_{BL}

Next, observe that in our protocol, some values are given to the parties at the beginning of the protocol, whereas some values are revealed during the execution. In the definition of the functionality, we split the outputs into three types:

1. Values that are given to the parties in the clear before the beginning of the protocol.
2. Values that are additively shared across the parties at the beginning of the protocol. These are marked using our notation $[\![\cdot]\!]$.
3. Values that should be revealed during the execution to the parties. To realize this, the dealer can secret share these values in an authenticated way. These values are marked using the notation $\langle\cdot\rangle$. As explained in Section 2.4, one can realize it via SPDZ-style information-theoretic MACs [3, 23].

The dealer's ideal functionality to produce correlated randomness for both the semi-honest computation and the verification protocol is formally defined in Functionality 3.

FUNCTIONALITY 3 (The Ideal Functionality $\mathcal{F}_{\mathsf{Dealer}}$)
Let mult be the set of multiplication gates. Denote by W_I, W_{mult} and W_o the set of the circuit's input wires, the set of input wires to multiplication gates and the set of the circuit's output wires, respectively. Let M and L be parameters given to the functionality. Let $\mathcal{A}$ be the ideal-world adversary controlling a set of parties of size$\leq n-1$.
The functionality $\mathcal{F}_{\mathsf{Dealer}}$ works as follows:

– For each wire $w \in W_I \cup W_{\mathsf{mult}} \cup W_o$ sample a random $r_w \in R$.
  Sample random masks $s_0, \ldots, s_{2M} \in R$.
  Sample a random $\boldsymbol{b}_0 \in R^L$
  Sample a random point $\tau \in R$.
– Let $\boldsymbol{b} = ((r_w)_{w \in W_{\mathsf{mult}} \cup W_o}, (r_2^{g_\ell}, r_1^{g_\ell}, (r_1 \cdot r_2)^{g_\ell})_{g_\ell \in \mathsf{mult}})$.
  Then, split $\boldsymbol{b}$ into M vectors of size L, i.e., $\boldsymbol{b} = \boldsymbol{b}_1 || \cdots || \boldsymbol{b}_M$.
– For each $e \in [L]$: let $\boldsymbol{G}_e$ be a vector of size $M+1$ defined as $\boldsymbol{G}_e = \boldsymbol{b}_0[e] || \cdots || \boldsymbol{b}_M[e]$.
  Set $\boldsymbol{G} = \boldsymbol{G}_1 || \cdots || \boldsymbol{G}_L$. Then, compute: $y_1, \ldots, y_L \leftarrow \pi_{\mathsf{BL}}(\boldsymbol{G}, \tau, M)$ (see Fig. 1).
– Let $\boldsymbol{S} = (s_0, \ldots, s_{2M})$. Then, compute $z \leftarrow \pi_{\mathsf{BL}}(\boldsymbol{S}, \tau, 2M)$ (see Fig. 1) and $s = \sum_{k=1}^{M} s_k$.
– Give the parties

$$(\{\langle r_w \rangle\}_{w \in W_I \cup W_0}, [\![\boldsymbol{b}]\!], [\![\boldsymbol{b}_0]\!], [\![\boldsymbol{S}]\!], \langle y_1 \rangle, \ldots, \langle y_L \rangle, \langle \tau \rangle, \langle s \rangle, \langle z \rangle)$$

  while letting $\mathcal{A}$ choose the corrupted parties' shares.

Comparison with BGIN [14]. A drawback of our verification protocol compared to [14] is that we have $O(\sqrt{C} \cdot n)$ communication per party and they have $O(\log(|C|) \cdot n)$ communication. Nevertheless, in both protocols, communication is sublinear in the size of the circuit. Our main advantage is in the work of the dealer. First, the size of the dealer's circuit (i.e., number of multiplications operations) does not depend on the number of parties n, which gives rise to efficient implementation even for large number of parties. Much more important is the fact that our dealer is represented by computation of *low-depth*. We will explore this property in Section 6 and see how it is crucial for achieving the goal of distributing the dealer with sublinear communication. We stress that the dealer in [14] *cannot* be represented this way. To see this, it suffices to recall that in their work the dealer needs to compute $\sum_{g_\ell \in \mathsf{mult}} \gamma_\ell \cdot (r_1^{g_\ell} \cdot r_2^{g_\ell})$. This implies that it needs to compute the coefficients γ_ℓ and so the random coefficient α_w for each wire w. To avoid linear correlated randomness, the authors in [14] suggested to use a single α from which all α_w are derived by computing: $\alpha_w = \alpha^w$. This yields a computation with depth that is linear in the size of the circuit, which makes it impossible to use the techniques presented in the next sections. In our protocol, the dealer does not need to compute anything based on the random coefficients and so this is completely avoided.

5 Online Computation with a Trusted Dealer

We are now ready to present the main protocol to compute any arithmetic circuits with malicious security in the $\mathcal{F}_{\mathsf{Dealer}}$-hybrid model (i.e., in the presence of a trusted dealer that gives the parties the correlated randomness). Informally, our protocol takes a secure-up-to-additive attack and star-sharing compliant protocol, and compiles it into malicious security, by adding a verification step, where the parties run the protocol Π_{vrfy} from Section 4. The formal description appears in the full version of this paper.

Concrete costs. We estimate the concrete communication and computation overhead of our verification protocol compared to the base semi-honest protocol. The communication overhead is a small additive term that is completely dominated by the communication complexity of the semi-honest protocol. The concrete computational overhead of the verification protocol for arithmetic circuits is often dominated by the communication cost of the semi-honest protocol even when the traffic is exchanged over a fast 1 Gbps network. To make our following estimates somewhat easier we assume an imbalance between the degree of each polynomial M, which is set to $M = \sqrt{|C|}$, and the number of polynomials $L = 5\sqrt{|C|}$.

In the verification protocol, the parties communicate to star-share Γ_i and then emulate the zk-FLIOP. When using the instantiation presented in Section 4.1, each party sends roughly $4L = 20\sqrt{|C|}$ elements to star-share its share of the proof. Then, the parties need to open the correlated randomness, which yields $2M = 2\sqrt{|C|}$ additional elements sent per party (see the end of Section 4.1). Overall, each party sends approximately $22\sqrt{|C|}$ elements. This overhead is dominated by the cost of the baseline semi-honest protocol (which requires interaction for each gate) even for moderately large circuits.

The computational cost of a party is dominated by interpolation to compute M extra points on each of L polynomials of degree M. Polynomial interpolation can be efficiently performed by the Discrete Fourier Transform (DFT) and therefore the total computation is dominated by L DFT operations. To give one data point, consider an arithmetic circuit of size $|C| = 2^{20}$ over $\mathbb{F}_p$, such that p is a 60-bit prime and 2^{10} divides $p-1$. In this setting, the soundness error is $\approx \frac{M}{|C|} = 2^{-50}$. Based on Shoup [43], a single multiplication of polynomials of degree $M = \sqrt{|C|} = 2^{10}$ over $\mathbb{F}_p$ takes 56 microseconds on a single standard core. The multiplication requires three DFT operations, one on each input polynomial and then one inverse DFT (which is also a DFT) to return the output polynomial to coefficient representation. Extrapolating these numbers to our case of a single DFT, it is safe to estimate that the interpolation of a single polynomial takes at most 20 microseconds, and interpolating $L = 5 \cdot 2^{10}$ polynomials requires at most 100 milliseconds.

The communication cost of the semi-honest protocol is two field elements per multiplication gate for a total of $120 \cdot 2^{20}$ bits. Even if the underlying network has 1 Gbps bandwidth then the total time for communication is roughly 120 milliseconds which exceeds the time for computation.

6 Distributing the Dealer with Sublinear Communication

We are now ready to show how to compute the dealer's functionality $\mathcal{F}_{\mathsf{Dealer}}$. The main observation behind our offline protocol is that, given a circuit C, the dealer's computation can be described using the next four steps:

1. Sample a vector $\boldsymbol{b}$ of semi-honest correlated randomness.
2. Sample a vector $\boldsymbol{v}$ of ring elements of size $O(\sqrt{|C|})$ using an arithmetic circuit of size $O(\sqrt{|C|})$.
3. Compute a bilinear function over $\boldsymbol{b}$ and $\boldsymbol{v}$ which outputs $O(\sqrt{|C|})$ ring elements. Denote the output vector by $\boldsymbol{y}$.
4. Give the parties $[\![\boldsymbol{b}]\!]$ and a subset of entries of $\boldsymbol{v}||\boldsymbol{y}$. Some of the entries may be secret shared (and possibly authenticated).

To see this, recall that the dealer needs to evaluate $O(\sqrt{|C|})$ polynomials on a random point τ. This can be done by computing a vector of the powers of τ, i.e., $(\tau, \tau^2, \ldots, \tau^{2M})$ and then multiplying the coefficients of each polynomial with this vector. Note that the task of computing $2M$ powers of τ is represented by a circuit of size $O(\sqrt{|C|})$ since $M = O(\sqrt{|C|})$. It should be noted that $\boldsymbol{b}$ consists of points on these polynomials and not coefficients, and so the dealer is required to do the conversion first.

We now proceed to describe a protocol which emulates the dealer. The idea behind the protocol is that we will generate encryptions of the powers of τ via an additively-homomorphic encryption scheme (AHE). Then, the parties can compute locally the bilinear operation, obtaining an encrypted version of the result, which can then be decrypted.

To this end, we define several ideal functionalities that will be used in the protocol.

The ideal functionality $\mathcal{F}_{\mathsf{triples}}$. This ideal functionality will be used to generate the semi-honest correlated randomness of the protocol, which consists of m multiplication triples. The $\mathcal{F}_{\mathsf{triples}}$ functionality can be realized with polylogarithmic communication complexity (in m) and good concrete efficiency via pseudorandom correlation generators (PCGs) based on Ring-LPN [10].

FUNCTIONALITY 4 (The Ideal Functionality $\mathcal{F}_{\mathsf{triples}}$)
Functionality $\mathcal{F}_{\mathsf{triples}}$ works with parties $P_1, \ldots, P_n$ and an ideal-world adversary $\mathcal{S}$ controlling a strict subset of the parties with indexes $I \subset [n]$ as follows: Upon receiving the command (Init, m) from all parties, it waits to receive $r^{\ell}_{1,i}, r^{\ell}_{2,i}, r^{\ell}_{3,i}$ for each $i \in I$ and $\ell \in [m]$ from $\mathcal{S}$. Then, it chooses for each $\ell \in [m]$ and $j \in [n] \setminus I$ random $r^{\ell}_{1,j}, r^{\ell}_{2,j}, r^{\ell}_{3,j}$ under the constraint that

$$\sum_{i=1}^{n} r^{\ell}_{3,i} = \sum_{i=1}^{n} r^{\ell}_{1,i} \cdot \sum_{i=1}^{n} r^{\ell}_{2,i}$$

Then, it hands $(r^{\ell}_{1,i}, r^{\ell}_{2,i}, r^{\ell}_{3,i})_{\ell \in [m]}$ to party P_i.

The ideal functionality $\mathcal{F}^{\mathsf{AHE}}_{\mathsf{EncPowers}}$. The next ideal functionality gives the parties encryptions of the powers of τ. It also gives an authenticated secret sharing of τ. In addition, it samples the masks that are used in the online verification protocol and secret shares the sum of them in an authenticated way. Recall that we can realize authenticated secret sharing with information-theoretic MACs, and so generating it can be represented by a small constant-size circuit.

FUNCTIONALITY 5 (The Ideal Functionality $\mathcal{F}^{\mathsf{AHE}}_{\mathsf{EncPowers}}$)
Functionality $\mathcal{F}^{\mathsf{AHE}}_{\mathsf{EncPowers}}$ works with parties $P_1, \ldots, P_n$ as follows:

– Upon receiving the command Init from all parties, the functionality runs $\mathsf{Gen}(1^{\kappa})$ to obtain (sk, pk), Then, it chooses shares sk_i for each $i \in [n]$ such that $sk = \sum_{i=1}^{n} sk_i$ and sends pk, sk_i to party P_i.
– Upon receiving the command $(\mathit{compute}, M)$ from all parties the functionality:
  1. Chooses a random τ, computes $\tau^2, \ldots, \tau^{2M}$ and $c_1, \ldots, c_{2M} = \mathsf{Enc}_{pk}(\tau), \ldots, \mathsf{Enc}_{pk}(\tau^{2M})$.
  2. Chooses random $s_0, \ldots, s_{2M}$ and computes $s = \sum_{k=1}^{M} s_k$.

  Then, it hands $\{[\![s_k]\!]\}_{k=0}^{2M}, c_1, \ldots, c_{2M}, \langle\tau\rangle, \langle s\rangle$ to the parties.

The parties compute the encryption of powers of τ using a simple, iterative protocol, beginning with a shared value $\mathsf{Enc}_{pk}(1)$, which is an encryption of $\tau^0 = 1$. Next, each party chooses its random share τ_i of τ. To compute $\mathsf{Enc}_{pk}(\tau^{j+1})$ given $\mathsf{Enc}_{pk}(\tau^j)$ and τ_i, each party homomorphically computes $\mathsf{Enc}_{pk}(\tau^j \cdot \tau_i)$ and sends the result to the other parties. Upon receiving all ciphertexts, each party homomorphically evaluates $\mathsf{Enc}_{pk}(\tau^j \cdot \sum_{i=1}^n \tau_i) = \mathsf{Enc}_{pk}(\tau^{j+1})$. Overall, this functionality can be computed with communication of $O(M \cdot \mathrm{poly}(\kappa, n)) = O(\sqrt{|C|} \cdot \mathrm{poly}(\kappa, n))$ ring elements. To achieve malicious security, we can use generically communication-preserving compilers; see below.

The ideal functionality $\mathcal{F}^{\mathsf{AHE}}_{\mathsf{AuthDec}}$. This functionality receives a ciphertext c from all parties and the secret-key share sk_i from each P_i and outputs $\langle u \rangle$ where $u = \mathsf{Dec}_{sk}(c)$ and $sk = \sum_{i=1}^n sk_i$ (i.e., the parties receive authenticated secret sharing of u). Note that it can be realized by a protocol with cost that is *independent* of the size of the computed circuit.

Realizing $\mathcal{F}_{\mathsf{Dealer}}$ with Semi-Honest Security Using the above functionalities it is easy to describe a semi-honest protocol to compute $\mathcal{F}_{\mathsf{Dealer}}$.

$\underline{\Pi^{\mathsf{SH}}_{\mathsf{dealer}}}$: Upon receiving a description of the circuit C as an input:

1. Sample semi-honest correlated randomness: the parties call $\mathcal{F}_{\mathsf{triples}}$ to receive $(r^{g_\ell}_{1,i}, r^{g_\ell}_{2,i}, r^{g_\ell}_{3,i})_{g_\ell \in \mathsf{mult}}$. For each input and output wire w of the circuit, each party P_i samples a random $r_{w,i}$ and then the parties generate $\langle r_w \rangle$ where $r_w = \sum_{i=1}^n r_{w,i}$.
2. The parties call $\mathcal{F}^{\mathsf{AHE}}_{\mathsf{EncPowers}}$ to receive back $sk_i, \{[\![s_k]\!]\}_{k=0}^{2M}, c_1, \ldots, c_{2M}, \langle \tau \rangle, \langle s \rangle$.
3. Each party P_i samples $\boldsymbol{b}_{0,i} \in R^L$ and then uses the shares $\boldsymbol{b}_{0,i}, \{r_{w,i}\}_{w \in W}, (r^{g_\ell}_{1,i}, r^{g_\ell}_{2,i}, r^{g_\ell}_{3,i})_{g_\ell \in \mathsf{mult}}$ to define the M-degree polynomials $g_{e,i}$ for each $e \in [L]$, and the shares $\{s_{k,i}\}_{k=0}^{2M}$ to define the $2M$-degree polynomial $\tilde{q}_i$. Then, each party locally converts its shares of the points on these polynomials to shares of the coefficients.
   Denote the coefficients of g_e (for each $e \in [L]$) by $g_{e,0}, \ldots, g_{e,M}$ and of $\tilde{q}$ by $\tilde{q}_0, \ldots, \tilde{q}_{2M}$.
4. For each $e \in [L]$, each party P_i locally computes $\mathsf{Enc}_{pk}(g_{e,i}(\tau)) \leftarrow \mathsf{Add}\left((g_{e,k,i})_{k \in \{0,\ldots,M\}}, (c_k)_{k \in [M]}\right)$.
   Similarly, each P_i computes $\mathsf{Enc}_{pk}(\tilde{q}_i(\tau)) \leftarrow \mathsf{Add}\left((\tilde{q}_{k,i})_{k \in \{0,\ldots,2M\}}, (c_k)_{k \in [2M]}\right)$.
5. Each P_i sends $\{\mathsf{Enc}_{pk}(g_{e,i}(\tau))\}_{e \in [L]}, \mathsf{Enc}_{pk}(\tilde{q}_i(\tau))$ to all the other parties.
6. The parties locally compute $\{\mathsf{Enc}_{pk}(g_e(\tau))\}_{e \in [L]}, \mathsf{Enc}_{pk}(\tilde{q}(\tau))$ by adding the received ciphertexts.
7. The parties obtain $\{\langle g_e(\tau) \rangle\}_{e \in [L]}$ and $\langle \tilde{q}(\tau) \rangle$ by calling $\mathcal{F}^{\mathsf{AHE}}_{\mathsf{AuthDec}}$. Denote $\langle \boldsymbol{y} \rangle = (\langle g_1(\tau) \rangle, \ldots, \langle g_L(\tau) \rangle)$ and $\langle z \rangle = \langle \tilde{q}(\tau) \rangle$

Output: Each party outputs his share of $\boldsymbol{b}$,$\boldsymbol{b}_0$ and $\{s_k\}_{k=0}^{2M}$, and his authenticated shares of τ, s, $\boldsymbol{y}$ and z.

Proposition 3. *Assuming the AHE scheme is semantically secure, protocol* $\Pi^{\mathsf{SH}}_{\mathsf{dealer}}$ *realizes* $\mathcal{F}_{\mathsf{Dealer}}$ *with* semi-honest security *in the* ($\mathcal{F}_{\mathsf{triples}}, \mathcal{F}^{\mathsf{AHE}}_{\mathsf{EncPowers}}, \mathcal{F}^{\mathsf{AHE}}_{\mathsf{AuthDec}}$)*-hybrid model.*

The proof can be found in the full version. As discussed above, $\mathcal{F}^{\mathsf{AHE}}_{\mathsf{EncPowers}}$ can be realized using a protocol with sublinear cost in the size of the computed circuit, while $\mathcal{F}^{\mathsf{AHE}}_{\mathsf{AuthDec}}$ can be realized with constant cost. Given $\mathcal{F}^{\mathsf{AHE}}_{\mathsf{AuthDec}}$ is called to decrypt sublinear amount of ciphertexts, the total cost of calling $\mathcal{F}_{\mathsf{dec}}$ is also sublinear. Thus, we get the following:

Corollary 1 (Distributing the dealer with semi-honest security). *Given (1) semi-honest protocol realizing $\mathcal{F}_{\mathsf{triples}}$ to compute m multiplication triples with communication $\alpha(\kappa, n, m, R)$, and (2) a semantically-secure AHE scheme, there exists a protocol that securely realizes $\mathcal{F}_{\mathsf{Dealer}}$ in the semi-honest model with communication per party of $O(\sqrt{|C|}) \cdot \mathrm{poly}(n, \kappa)$ ring elements and $\alpha(\kappa, n, |C|, R)$ additional bits.*

Achieving Malicious Security. In the context of feasibility, one can easily obtain a malicious security variant of Corollary 1 by applying the communication-efficient GMW-style compiler of Naor and Nissim [38] (building on [28, 37]). Using this compiler, a semi-honest secure protocol can be compiled into a maliciously secured protocol for the same functionality with sublinear additive communication cost of $\mathrm{poly}(\kappa, n, \log|C|, \log|R|)$ bits per party. Since the compiler only requires collision-resistant hash functions, which are implied by additively homomorphic encryption [32], this does not require introducing new assumptions. Note that the compiler also respects an augmented correlated randomness functionality that allows the adversary to pick its own output shares as we require in this work. We thus obtain the following:

Theorem 6 (Distributing the dealer with malicious security). *Given (1) semi-honest protocol realizing $\mathcal{F}_{\mathsf{triples}}$ to compute m multiplication triples with communication $\alpha(\kappa, m, R)$, and (2) a semantically-secure AHE scheme, there exists a protocol that securely realizes $\mathcal{F}_{\mathsf{Dealer}}$ in the malicious model with communication per party of $O(\sqrt{|C|}) \cdot \mathrm{poly}(n, \kappa)$ ring elements and $\alpha(\kappa, n, |C|, R) + \mathrm{poly}(\kappa, n, \log|C|, \log|R|)$ additional bits.*

Conclusion: MPC with sublinear preprocessing and non-cryptographic online phase. To obtain our main result, we combine the offline protocol of Theorem 6 with the online protocol described in Section 5, instantiated with Beaver's semi-honest MPC protocol [1] based on multiplication triples. This yields a protocol which satisfies the notion of Preprocessing MPC from Definition 1, where the offline communication complexity of Π_{offline} scales with $\sqrt{|C|}$ and the online communication and correlated randomness are the same as those of the baseline semi-honest protocol up to a sublinear additive term. The protocol relies on AHE, together with any low-communication protocol (e.g., one based on PCG) for generating random (unauthenticated) multiplication triples.

Theorem 7 (Sublinear preprocessing from AHE+triples). *Let f be an n-party functionality represented by an arithmetic circuit C over a ring R. Then, given (1) semi-honest protocol realizing m unauthenticated multiplication triples (see $\mathcal{F}_{\mathsf{triples}}$) with communication $\alpha(\kappa, n, m, R)$, and (2) an AHE scheme over R, there exists a PMPC protocol $(\Pi_{\mathsf{offline}}, \Pi_{\mathsf{online}})$ for f with non-cryptographic online phase (see Definition 1) with the following efficiency measures:*

– *The communication per party in Π_{offline} is $O(\sqrt{|C|}) \cdot \mathrm{poly}(n, \kappa)$ ring elements and $\alpha(\kappa, n, |C|, R) + \mathrm{poly}(\kappa, n, \log|C|, \log|R|)$ additional bits;*

– *The communication per party in Π_{online} is $O(|C|) + O(\sqrt{|C|}) \cdot \mathrm{poly}(n, \kappa)$ ring elements.*

Ingredient (1) in Theorem 7 can be instantiated with a PCG based on LPN or Ring-LPN [10] for better concrete efficiency, with $\alpha(\kappa, n, |C|, R) = n^2 \cdot \mathrm{poly}(\kappa) \cdot (\log |C| + \log |R|)$ bits of communication. Using this instantiation, the total communication cost of Π_{offline} is $O(\sqrt{|C|}) \cdot \mathrm{poly}(n, \kappa)$ ring elements.

Improving Concrete Efficiency. The protocol described previously for distributing the dealer uses generic tools to compile the semi-honest protocol to a malicious protocol. This approach is sufficient for good asymptotic efficiency, but not necessarily for good concrete efficiency. In the full version of the paper, we propose a maliciously secure protocol with improved performance that is based on making two stronger, but quite plausible, assumptions on the AHE: being "linear-only" and having a threshold encryption variant.

An AHE is "linear-only", or more precisely has Linear Targeted Malleability [4, 7], if linear functions, and only linear functions, can be computed homomorphically on the ciphertexts. It is widely assumed that popular AHE schemes such as Goldwasser-Micali [29], Paillier [41] and even certain parameter ranges for lattice based encryption systems [15] are all "linear-only". Encryption schemes with a threshold variant enable parties to share a secret key and distributively decrypt a ciphertext without interaction. Systems with such a threshold variant include GM [34], Paillier [30] and lattice based systems via noise flooding.

The proposed protocol to distribute the dealer assuming a circuit over a field $\mathbb{F}$ executes a generically malicious secure protocol for the generation of powers of τ and then the rest of the previous semi-honest protocol. The parties proceed by threshold decryption of the ciphertexts. The adversary can add errors to the decrypted values, which might even depend on the evaluation point τ. However, due to Linear Targeted Malleability the probability that any attack on the protocol does not cause an honest party to abort after verification in the online phase is $\Theta\left(\frac{M}{|\mathbb{F}|}\right)$. The complexity of the protocol is dominated by roughly $5|C|$ homomorphic operations and threshold decryption of $5\sqrt{|C|}$ ciphertexts.

Acknowledgments. We thank the Eurocrypt reviewers for helpful comments. E. Boyle supported by a Google Research Scholar Award, AFOSR Award FA9550-21-1-0046, ERC Project HSS (852952), and ERC Project NTSC (742754). N. Gilboa supported by ISF grant 2951/20, ERC grant 876110, and a grant by the BGU Cyber Center. Y. Ishai supported by ERC Project NTSC (742754), BSF grant 2018393, and ISF grant 2774/20. A. Nof supported by ERC Project NTSC (742754).

References

1. D. Beaver. Efficient multiparty protocols using circuit randomization. In *CRYPTO*, 1991.

2. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, 1988.
3. R. Bendlin, I. Damgård, C. Orlandi, and S. Zakarias. Semi-homomorphic encryption and multiparty computation. In *EUROCRYPT*, 2011.
4. N. Bitansky, A. Chiesa, Y. Ishai, O. Paneth, and R. Ostrovsky. Succinct non-interactive arguments via linear interactive proofs. In *Theory of Cryptography Conference*, pages 315–333, 2013.
5. A. R. Block, H. K. Maji, and H. H. Nguyen. Secure computation with constant communication overhead using multiplication embeddings. In D. Chakraborty and T. Iwata, editors, *Progress in Cryptology - INDOCRYPT*, volume 11356 of *Lecture Notes in Computer Science*, pages 375–398, 2018.
6. D. Boneh, E. Boyle, H. Corrigan-Gibbs, N. Gilboa, and Y. Ishai. Zero-knowledge proofs on secret-shared data via fully linear pcps. In *CRYPTO*, 2019.
7. D. Boneh, Y. Ishai, A. Sahai, and D. J. Wu. Lattice-based snargs and their application to more efficient obfuscation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 247–277, 2017.
8. E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, P. Rindal, and P. Scholl. Efficient two-round OT extension and silent non-interactive secure computation. In *ACM CCS*, 2019.
9. E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, and P. Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In *CRYPTO*, 2019.
10. E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, and P. Scholl. Efficient pseudorandom correlation generators from ring-lpn. In *CRYPTO*, 2020.
11. E. Boyle, N. Gilboa, and Y. Ishai. Secure computation with preprocessing via function secret sharing. In *TCC*, 2019.
12. E. Boyle, N. Gilboa, Y. Ishai, and A. Nof. Practical fully secure three-party computation via sublinear distributed zero-knowledge proofs. In *ACM CCS*, 2019.
13. E. Boyle, N. Gilboa, Y. Ishai, and A. Nof. Efficient fully secure computation via distributed zero-knowledge proofs. In *ASIACRYPT*, 2020.
14. E. Boyle, N. Gilboa, Y. Ishai, and A. Nof. Sublinear gmw-style compiler for MPC with preprocessing. In T. Malkin and C. Peikert, editors, *Advances in Cryptology - CRYPTO*, 2021.
15. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014.
16. R. Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptology*, 13(1):143–202, 2000.
17. D. Catalano, M. D. Raimondo, D. Fiore, and I. Giacomelli. Monz2ka: Fast maliciously secure two party computation on z2k. *IACR Cryptology ePrint Archive*, 2019.
18. D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols (extended abstract). In *STOC*, 1988.
19. J. D. Cohen and M. J. Fischer. A robust and verifiable cryptographically secure election scheme (extended abstract). In *FOCS 1985*, pages 372–382, 1985.
20. R. Cramer, I. Damgård, D. Escudero, P. Scholl, and C. Xing. Spdz2k: Efficient MPC mod 2k for dishonest majority. In *CRYPTO*, 2018.
21. I. Damgård, M. Keller, E. Larraia, V. Pastro, P. Scholl, and N. P. Smart. Practical covertly secure MPC for dishonest majority - or: Breaking the SPDZ limits. In *ESORICS*, 2013.

22. I. Damgård, J. B. Nielsen, M. Nielsen, and S. Ranellucci. The tinytable protocol for 2-party secure computation, or: Gate-scrambling revisited. In *CRYPTO*, 2017.
23. I. Damgård, V. Pastro, N. P. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. In *CRYPTO*, 2012.
24. I. Damgård and S. Zakarias. Constant-overhead secure computation of boolean circuits using preprocessing. In *TCC*, 2013.
25. P. S. Damiano Abram. Low-communication multiparty triple generation for spdz from ring-lpn. In *PKC 2022*, 2022.
26. D. Genkin, Y. Ishai, M. Prabhakaran, A. Sahai, and E. Tromer. Circuits resilient to additive attacks with applications to secure computation. In *STOC*, 2014.
27. O. Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.
28. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *STOC*, 1987.
29. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of computer and system sciences*, 28(2):270–299, 1984.
30. C. Hazay, G. L. Mikkelsen, T. Rabin, T. Toft, and A. A. Nicolosi. Efficient rsa key generation and threshold paillier in the two-party setting. *Journal of Cryptology*, 32(2):265–323, 2019.
31. Y. Ishai, E. Kushilevitz, S. Meldgaard, C. Orlandi, and A. Paskin-Cherniavsky. On the power of correlated randomness in secure computation. In *TCC*, 2013.
32. Y. Ishai, E. Kushilevitz, and R. Ostrovsky. Sufficient conditions for collision-resistant hashing. In *Theory of Cryptography, TCC*, pages 445–456, 2005.
33. C. Juvekar, V. Vaikuntanathan, and A. P. Chandrakasan. GAZELLE: A low latency framework for secure neural network inference. In *USENIX Security 2018*, pages 1651–1669, 2018.
34. J. Katz and M. Yung. Threshold cryptosystems based on factoring. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 192–205, 2002.
35. M. Keller, E. Orsini, and P. Scholl. MASCOT: faster malicious arithmetic secure computation with oblivious transfer. In *ACM CCS*, 2016.
36. M. Keller, V. Pastro, and D. Rotaru. Overdrive: Making SPDZ great again. In *EUROCRYPT*, 2018.
37. J. Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *ACM STOC*, pages 723–732, 1992.
38. M. Naor and K. Nissim. Communication preserving protocols for secure function evaluation. In *ACM STOC*, pages 590–599, 2001.
39. J. B. Nielsen, P. S. Nordholt, C. Orlandi, and S. S. Burra. A new approach to practical active-secure two-party computation. In *CRYPTO*, 2012.
40. C. Orlandi, P. Scholl, and S. Yakoubov. The rise of paillier: Homomorphic secret sharing and public-key silent OT. In *EUROCRYPT*, 2021.
41. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques*, pages 223–238, 1999.
42. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC 2005*, pages 84–93, 2005.
43. V. Shoup. Arithmetic software libraries. https://www.shoup.net/papers/akl-chapter.pdf.
44. A. C. Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, 1986.