Scalable Multiparty Computation from Non-linear Secret Sharing

Sanjam Garg* Abhishek Jain† Pratyay Mukherjee‡ Mingyuan Wang§

Abstract

A long line of work has investigated the design of scalable secure multiparty computation (MPC) protocols with computational and communication complexity independent of the number of parties (beyond any dependence on the circuit size). We present the first unconditionally-secure MPC protocols for arithmetic circuits over *large fields* with total computation $\mathcal{O}(|C| \log |F|)$, where $|C|$ and $|F|$ denote the circuit and field size, respectively.

Prior work could either achieve similar complexity only in *communication*, or required highly structured circuits, or expensive circuit transformations. To obtain our results, we depart from the prior approach of share packing in linear secret-sharing schemes; instead, we use an "unpacking" approach via *non-linear* secret sharing.

*UC Berkeley sanjamg@berkeley.edu

†NTT Research and Johns Hopkins University abhishek@cs.jhu.edu

‡Supra Research pratyay85@gmail.com

§UC Berkeley mingyuan@berkeley.edu

[1]The first and fourth authors were supported in part by DARPA under Agreement No. HR00112020026, AFOSR Award FA9550-19-1-0200, NSF CNS Award 1936826, and research grants by the Sloan Foundation, Visa Inc, the BAIR Commons Meta Fund, and the Stellar Development Foundation. The second author was supported in part by NSF CNS-1814919, NSF CAREER 1942789, Johns Hopkins University Catalyst award, AFOSR Award FA9550-19-1-0200, JP Morgan Faculty Award, and research gifts from Ethereum, Stellar and Cisco. Any opinions, findings and conclusions, or recommendations in this material are those of the authors and do not necessarily reflect the views of the United States Government or DARPA.

Contents

1 Introduction

Secure multiparty computation (MPC) [Yao86, GMW87, BGW88, CCD88] enables a group of mutually distrusting parties to jointly compute an arbitrary function on their private inputs in a manner such that the participants learn nothing beyond the output of the function. Over the years, MPC has emerged as a general-purpose cryptographic tool for processing sensitive distributed data without introducing a single point of failure.

Large-Scale MPC. Many emerging applications of MPC naturally involve a large number of parties. For instance, in privacy-preserving machine learning, one can derive more robust conclusions when a large number of participants contribute their inputs. High participation is also desirable for better security assurance; indeed, in practice, the privacy threshold of an MPC protocol is less likely to be breached in a large-scale setting. It is therefore desirable to build MPC protocols where the communication and computational complexity has *minimal* (and ideally, no) dependence on the number of parties beyond any dependence on the circuit size.

The work of [DN07] constructed unconditionally-secure MPC for general arithmetic circuits where the per-party communication and computation is independent of the number of parties. Subsequently, a long line of works has attempted to achieve *total* (as opposed to per-party) complexity independent of the number of parties for circuits of sufficiently large width and size. For instance, [GPS21, GPS22] recently constructed MPC protocols with total *communication* $\mathcal{O}(|C| \log |F|)$, where $|C|$ denotes the number of gates in the circuit and $|F|$ denotes the field size. MPC protocols with similar *computational* complexity are only known for structured circuits (such as SIMD) [FY92, BGJK21]. For general circuits, one can use circuit transformations [DIK10] to achieve the desired structured form; however, this incurs an overhead of $\mathcal{O}(\log |C|)$ in circuit size and depth, which is eventually reflected in the communication, computation, and the round complexity.

Our Work. In this work, we study scalable MPC for general (unstructured) arithmetic circuits. In particular, we give a scalable MPC protocol (in both *computation* and communication) for any arithmetic circuit over a *large field*. Technically, we employ a non-linear secret sharing recently introduced to the MPC literature [GJM+23], which is a significant departure from all prior scalable MPC protocols based on linear secret sharing schemes.

1.1 Our Results

Main Result. We construct a scalable MPC protocol for arithmetic circuits over a prime field F where $\log |F|$ grows quadratically in the number of parties. In particular, the overall communication and computation of our protocol, measured at bit-level,[2] is $\mathcal{O}(|C| \cdot \log |F|)$.[3] Our protocol achieves unconditional security in the honest-majority setting against any $t < \frac{1}{2} - \delta$ fraction of semi-honest corruptions for any constant $\delta > 0$. The sub-optimal corruption threshold in our result is necessary and in keeping in known negative results for strict honest majority [DLN19]. The construction is presented in detail in Section 4. Here we provide the informal theorem.

Theorem 1 (Informal). *For any constant $\delta \in (0, 1/2)$, any prime field F_p such that $\log p = \Omega(n^2 \log n)$, and any arithmetic circuit C over F, there is an unconditional secure MPC protocol realizing C among n*

[2] Throughout this work, we measure communication and computation complexity at a bit level. Our protocol employs a non-linear secret sharing scheme where each party performs computation over a different field. This is in contrast to linear secret sharing where all parties work over the same field. Therefore, we inevitably have to measure complexities at the bit level.

[3] Throughout this work, when we state computational complexities, we ignore some $\mathsf{loglog}|F|$ terms. Note that, even an insecure evaluation of the circuit takes more than $\mathcal{O}(|C| \cdot \log |F|)$ at a bit level. This is because multiplying two $\log |F|$-bit number takes quasilinear time. We refer the readers to item 1 in Section 3.3 for a detailed discussion on this.

parties against $t = (1/2 - \delta) \cdot n$ semi-honest corruptions. This protocol is perfectly correct and $\mathsf{poly}(|C|/|F|)$*-statistically secure. Moreover, the total communication and computation (measured at bit-level) of the protocol is* $\mathcal{O}(|C| \cdot \log |F|)$.[4]

Dishonest-Majority MPC. Next, we extend our protocol to the *dishonest majority* setting in the correlated randomness model. Our correlated randomness setup includes secret shares of Beaver triples [Bea92] computed using the secret-sharing scheme based on the Chinese Remainder Theorem (CRT) [Mig83, AB83, GRS99]. Our protocol achieves unconditional security against any $t < 1 - \delta$ fraction of semi-honest corruptions for any constant $\delta > 0$. The construction is presented in Section 5. Here we provide the informal theorem.

Theorem 2 (Informal)**.** *For any constant $\delta \in (0, 1/2)$, any prime field F_p such that $\log p = \Omega(n^2 \log n)$, and any arithmetic circuit C over F, in the correlated randomness model, there is an unconditional secure MPC protocol realizing C among n parties against $t = (1 - \delta) \cdot n$ semi-honest corruptions. This protocol is perfectly correct and* $\mathsf{poly}(|C|/|F|)$*-statistically secure. Moreover, the total communication and computation (measured at bit-level) of the protocol is $\mathcal{O}(|C| \cdot \log |F|)$.*

MPC over Rings. Finally, we highlight that our protocols can be readily extended to work over rings. In particular, as long as p is coprime with $2, 3, \ldots, n-1$, all of our results directly extend to $\mathbb{Z}_p$ without any change. The problem of MPC over rings has been studied in many works (e.g., [CFIK03, ACD+19, ES21, EXY22, DEN22]) recently. Our results provide an alternative approach to this problem.

1.2 Discussion: MPC over Large Fields

Applications. Typically, unconditionally-secure MPC protocols require field size $\geqslant n$. Our results require significantly larger fields. We believe that this requirement can be met in emerging applications of MPC where MPC is used for privacy-preserving delegation of computation of resource-intensive cryptographic algorithms. One such example is zero-knowledge succinct non-interactive arguments (zkSNARGs) where the underlying field is naturally large — typically, $\log F$ is proportional to the security parameter. Recent works [OB22, GGJ+23, CLMZ23] demonstrated that zkSNARG proof generation, a resource-expensive process, can be performed using scalable MPC to enable fast proof generation in a privacy-preserving manner.

Our protocol is well-suited to such applications. In particular, similar to prior work [OB22, GGJ+23, CLMZ23], it can also support black-box use of cryptography. As an example, the distributed proof generation process in these works (in the context of group-based zkSNARGs) transforms a sharing $[\![x]\!]$ of x into g^x, where g is the generator of a cryptographic group $\mathbb{G}$. This can be done by exploiting linearity (e.g., Lagrange interpolation in the exponent). Our protocol can also support such transformations: given sharing $[\![r]\!]$ of a random mask r and g^r in the clear, one can reconstruct $y = x + r$ and compute $g^x = g^y / g^r$. More crucially, these randomness terms can be generated in a batch manner using the randomness extraction techniques we develop in this work.

Our protocol also achieves some key *advantages* over prior work. In order to achieve fast proof generation using scalable MPC, prior works inevitably rely on the highly repetitive structure of the proof generation circuit. Although some parts of the proof generation enjoy such a structure, there are parts that do not. This results in undesirable outcomes: for instance, the clients outsourcing the computation

[4] This informal theorem statement ignores $\mathsf{polylog}\, n$ terms on the computational complexity. We refer the readers to Theorem 3 for a formal statement.

need to compute the extended witness[5] themselves, which already involves $\mathcal{O}(|C|)$ computation. Furthermore, some works [GGJ+23] also require a king server with large memory and computational resources to handle non-repetitive computation. Our protocol eliminates such drawbacks.

Finally, we remark that, while our current result requires $\log p = \Omega\big(n^2 \cdot \log n\big)$ for technical reasons, it is conceivable that this can be reduced to $\log p = \Omega(n \cdot \log n)$ (we discuss this point at length in Section 2). Thus, our approach could eventually lead to a scalable MPC protocol where the total work could be divided among $\approx \lambda$ number of parties for such applications.

An Alternative Approach. We discuss an alternative approach to scalable MPC for large fields.[6] One can compile an arithmetic circuit C over a field F into a boolean circuit C' of size $\approx |C| \cdot \log F$ [vzGS91]. When the field is sufficiently large, e.g., $\log F \geqslant n$, the resulting boolean circuit will have a highly repetitive form and one can employ existing works [BGJK21] to obtain a scalable MPC protocol.

Compared to Theorem 1, this approach is less preferable. Asymptotically, it introduces a large $\operatorname{polylog} n$ overhead which affects the communication, computation and round complexity complexity of the final protocol. This overhead is introduced in the compilation process and then one must use a sufficiently large extension field to employ Shamir secret sharing-based MPC. This approach is also undesirable from a concrete efficiency perspective due to the hidden constant and lower-order terms in the asymptotic expression. Indeed, to avoid specifically such overheads, prior work has, for example, studied methods for direct garbling of arithmetic circuits [AIK11, BLLL23] (without first translating them to boolean circuits and using Yao's garbling [Yao86]). See further discussion on the overhead of this transformation in [AIK11, BLLL23].

For these and other reasons, this approach is also not well-suited for the application to outsourcing cryptography discussed above.

1.3 Related Work

Classical honest-majority MPC protocols [BGW88, CCD88] incur per-gate communication and computation that grows at least quadratically in the number of parties. There are two primary techniques for reducing this overhead: share packing [FY92], and king-based computation with batched randomness generation [DN07]. Either of these techniques can be used individually to reduce the per-gate overhead to $\mathcal{O}(n)$. However, the former technique is only directly applicable to SIMD computations, while the latter technique works for general circuits (of size greater than the number of parties).

Starting from [DIK+08, DIK10], a sequence of works has explored the "marriage" of both of these techniques to obtain more efficient protocols for circuits of sufficiently large width (in particular, larger than the number of parties). Specifically, [DIK10] presented an unconditionally secure MPC protocol with a total computation of $\mathcal{O}\big(|C| \cdot \log |C| + d^2 \cdot n\big)$ field operations, where d denotes the circuit depth. For circuits with width much larger than the depth, the effective overhead is $\mathcal{O}(\log |C|)$.[7] The work of [GIP15] obtained constant-factor improvements in the setting of malicious corruptions. Subsequently, [BGJK21] constructed MPC with a total computation of $\mathcal{O}(|C|)$ field operations for highly repetitive circuits. Recently, [GPS21, GPS22] constructed the first MPC protocols that incur only constant communication per-gate for general circuits. The computational complexity of their protocols, however, still remains $n \cdot |C|$ field operations.

All of the above works that achieve per-gate communication sublinear in n rely on sub-optimal corruption thresholds. The work of [DLN19] proved the necessity of a small gap δ from a strict honest majority

[5] Let $\mathcal{R}$ be an NP relation with statement x and witness w. Suppose this relation is described by an arithmetic circuit C, i.e., $(x, w) \in \mathcal{R}$ if and only if $C(x, w) = 1$. The extended witness stands for all the intermediate values when one evaluates the arithmetic circuit $C(x, w)$.

[6] We thank an anonymous reviewer for pointing this out.

[7] For circuits with large depth d, [DIK10] propose an alternative solution with overhead $\mathcal{O}(\log |C| \log(d))$.

for MPC protocols with constant communication per-gate.

Our core techniques rely on non-linear secret sharing built using Chinese Remainder Theorems (CRT). The use of CRT-based secret sharing to the realm of secret-sharing based MPC was recently introduced in [GJM+23]. Their focus is to study efficient weighted MPC, namely, the trust assumption is skewed by weights as in a majority of the weights are honest. They show that CRT-based secret sharing could be more efficient than linear secret sharing schemes for realizing weighted MPC. In comparison, our focus in this work is to study the scalability of MPC in the standard unweighted setting. We extend their core techniques by bringing the batched randomness generation techniques [DN07] to the realm of CRT secret sharing-based MPC.

There are other areas of secure computations, such as distributed RSA key generation [ICG08, HTX20] arithmetic garbled circuits [AIK11, Hea24], and doubly efficient PIR [LMW22] where CRT-based techniques were used. However, they did not consider CRT-based (non-linear) secret sharing.

Using computational assumptions such as fully homomorphic encryption [Gen09], one may get MPC protocols [MW16, AJL+12] that achieve optimal communication and round complexity. In this paper we focus on unconditional security.

2 Technical Overview

We now present an overview of the key techniques underlying our results. Let C be an arithmetic circuit over some prime field F_{p_0}. Let n be the number of parties and t be the number of (semi-honest) corruptions. We start by describing our ideas for the honest majority setting and then present some extensions.

The Necessity of Download Rate $\mathcal{O}(1)$. All existing information-theoretic MPC protocols employ secret sharing schemes to emulate the circuit evaluation in a gate-by-gate manner. That is, parties start by secret sharing their inputs. Afterward, for every gate such that the input wires are already secret shared, parties will collectively derive appropriate secret sharing of the output wire. Evidently, for every gate being computed, the overall computation done by all parties will inevitably grow with the total share size of the secret sharing scheme. Therefore, if one hopes to achieve $|C| \log p_0$ computational complexity, it is necessary that the ratio between the secret size and the total share size, namely, the *download rate*, is $\mathcal{O}(1)$.

Limitations of Linear Secret Sharing. Linear secret sharing, by definition, considers secret sharing schemes, where both the secret and each individual secret share come from the same underlying field F_{p_0}. Naturally, linear secret sharing schemes have a download rate $\leqslant 1/n$. Consequently, the total computation done by all parties, even for evaluating, say, an addition gate, is still n times the necessary computation required for evaluating addition insecurely (even though emulating addition gates does not require communication at all). Since most existing information-theoretic MPC protocols are based on linear secret sharing schemes, they are all affected by this bottleneck. In order to address this barrier, prior works employ a technique known as *share packing* [FY92]. That is, one uses a *single instance* of linear secret sharing to share *multiple secrets*. Through this packing technique, performing operations on secret shares allows parties to collectively evaluate multiple gates at the same time. Intuitively, this allows for an *amortized download rate* $\mathcal{O}(1)$.

However, this technique of utilizing amortization comes at a cost. Since one *must* evaluate multiple gates simultaneously, this technique is only compatible with circuits that come with specific repetitive patterns, which a general circuit might not conform to. Prior works tried to address this issue through different perspectives. For instance, [DIK10] considered compiling any circuit to a special form before applying this technique, [BGJK21] considered a larger (than SIMD) family of repetitive circuits where this technique is applicable, [GPS21, GPS22] used this technique to reduce the communication (but not the

computation) for general circuits. None of these works succeeds in constructing an MPC protocol for general circuits with $\mathcal{O}(|C| \cdot \log F)$ overall computation.

Our Key Insight. In this work, we completely depart from this amortization approach based on share packing — the only known technique for addressing the aforementioned barrier. Instead, we ask

Is there an MPC-friendly secret-sharing scheme with a non-amortized *download rate* $\mathcal{O}(1)$*?*

If the answer is positive, the hope is that one can directly use this secret sharing scheme to emulate circuit evaluation in a straightforward manner to achieve $\mathcal{O}(|C| \cdot \log F)$ complexity, which will allow us to completely bypass the issues that come with packing.

"Unpacked" Secret Sharing. We positively answer the above question through a non-linear secret-sharing scheme based on the Chinese remainder theorem [Mig83, AB83, GRS99]. While this type of secret sharing scheme has been known for a long time, it was only recently introduced to the MPC literature by [GJM+23].

Let us start by recalling the basics of CRT-based secret sharing. CRT-based secret sharing schemes are parametrized by moduli $p_1, \ldots, p_n$, which are required to be pairwise coprime (including p_0). To share a secret $x \in F_{p_0}$, one picks a random integer X such that $X = x \mod p_0$. The i-th secret share is defined as $x_i = X \mod p_i$. To reconstruct the secret from secret shares, one uses the Chinese remainder theorem to reconstruct the integer X first and then derive $x = X \mod p_0$. When X is drawn from suitable distributions (in particular, uniformly randomly from *consecutive* $x, x+p_0, x+2p_0, \ldots, x+L\cdot p_0$ integer representatives of x), one may prove the statistical security of CRT-based secret-sharing schemes (refer to Lemma 3). Later, we will simply use $[\![X]\!]$ to represent one instance of CRT sharing of the secret $X \mod p_0$.

Unlike linear secret sharing, one may observe that the CRT-based secret sharing scheme does not require each individual secret share to be as long as the secret. In fact, it is possible that the length of the secret is comparable to the *total length* of the secret shares, i.e.,

$$\log p_0 = \Theta(\log p_1 + \log p_2 + \cdots + \log p_n) = \Theta(\log P_{\mathsf{all}}),$$

where $P_{\mathsf{all}} = p_1 p_2 \cdots p_n$. Therefore, CRT-based secret sharing could have a non-amortized download rate $\mathcal{O}(1)$. Conceptually, one may think of our approach as an *unpacking* technique. Namely, instead of packing many small secrets into one sharing to achieve download rate $\mathcal{O}(1)$, our approach unpacks a single large secret into smaller shares to directly achieve $\mathcal{O}(1)$. Before we proceed to discuss CRT Secret Sharing-based MPC, we make two important observations.

1. **Necessity of Large Field.** Since $\log P_{\mathsf{all}}$ necessarily grows linearly with n, if one hopes to achieve download rate $\mathcal{O}(1)$, it is inevitable that the field size $\log p_0$ linearly depends on the number of parties. Ideally, the optimal result one could hope for is an MPC protocol with $\mathcal{O}(1)$ overhead for any field whose size is linear in n. For technical reasons, which we shall explain shortly, we achieve an $\mathcal{O}(1)$ overhead MPC protocol when $\log p_0$ grows quadratically in n.

2. **Necessity of Ramp.** Note that it is also inevitable that such a secret sharing scheme does not have a *sharp threshold* as in any t parties can not learn any information and any $t+1$ parties can recover the secret. The reason is the following: for any sharp threshold secret sharing scheme, irrespective of whether it is linear or non-linear, it must hold that each share size is as long as the secret size [CDSGV91]. Therefore, this means that the download rate of the secret sharing scheme must be $\leqslant 1/n$. Hence, for a download rate $\mathcal{O}(1)$ secret sharing, one must need significantly more than $t+1$ parties to reconstruct the secret. Looking ahead, this means the MPC protocol based on these schemes can not be strictly honestly majority $n = 2t+1$, but will be in honest super-majority $t < (1/2 - \delta) \cdot n$ regime for some constant

δ. In fact, this is not only necessary for us, but for *all* MPC protocols that achieve $\mathcal{O}(|C| \cdot \log p_0)$ total communication as shown by [DLN19]. Our observation here is aligned with this known lower bound.

MPC from CRT Secret Sharing. Although CRT secret sharing is *non-linear*, it bears a lot of resemblances, as observed by [GJM+23], to MPC-friendly linear secret sharing scheme, e.g., Shamir's secret sharing scheme. In particular, it also satisfies "local homomoprhism". Namely, under suitable constraints, when parties locally add/multiply their shares, it becomes a secret sharing of the sum/product of the underlying secrets. For Shamir's secret sharing, addition comes for free; but after each multiplication, the random polynomial encoding the secret goes from degree t to $2t$. Hence, one needs a "degree-reduction" protocol to restore the degree back to t. Similarly, for CRT secret sharing, one has to manage the size of the integer $X \in \mathbb{N}$ representing the secret $x \in F_{p_0}$. Otherwise, if X becomes too large (in particular, $X \geqslant P_{\mathsf{all}}$), the CRT invocation on the secret shares $\{X \mod p_i\}_{i\in[n]}$ will not give X but some $X' < P_{\mathsf{all}}$, resulting in a correctness issue when $X' \neq X \mod p_0$. Therefore, CRT secret sharing-based MPC protocols also need a similar *"integer-reduction" protocol* to restore the size of the secret integer. Just like degree-reduction, this can be done by employing two secret sharing $[\![R_t]\!]$ and $[\![R_{2t}]\!]$ of the same secret mask r. Parties reconstruct $S + R_{2t}$ in the clear and locally shift their share $[\![R_t]\!]$ by $(S + R_{2t}) \mod p_0$.

At this point, it is helpful to summarize the online phase of the CRT-based MPC protocol, assuming that parties are given access to sufficiently many masks $[\![R_t]\!]$ and $[\![R_{2t}]\!]$ and that all p_i have approximately the same length $\ell = \log p_i$. Parties start by sharing their input x as $[\![X]\!]$. These integers are of length approximately $t \cdot \ell$ to ensure that corrupted parties' secret shares are uniformly random.[8] For the addition gate, the integer grows slowly. In particular, for $|C|$ number of addition gates, the length of the integer will grow by an additive term of $\log |C|$. Therefore, no integer-reduction is needed for the addition gate. For the multiplication gate, however, the integer grows fast. In particular, two $(t \cdot \ell)$-bit integers X and Y will result in a $(2 \cdot t \cdot \ell)$-bit integer $X \cdot Y$.[9] Therefore, after each multiplication gate,[10] parties need to consume one pair of masks $[\![R_t]\!], [\![R_{2t}]\!]$ to restore the integer back to $(t \cdot \ell)$ bits. Therefore, with enough random masks, parties can collectively emulate the circuit evaluation gate by gate. Assuming that the CRT secret sharing has a download rate $\mathcal{O}(1)$, this already gives an MPC protocol with *online* communication and computation $\mathcal{O}(|C| \cdot \log p_0)$.[11]

Offline Batch Random Masks Generation. To enable the online phase, parties must collectively sample sufficiently many (corresponding to the number of multiplication gates) masks $[\![R_t]\!]$ and $[\![R_{2t}]\!]$ in the offline phase. Moreover, this has to be done efficiently, i.e., with $\mathcal{O}(|C| \cdot \log p_0)$ total complexity. In particular, every pair of masks $[\![R_t]\!]$ and $[\![R_{2t}]\!]$ is consumed for one gate, and therefore, the cost of generating them should be comparable to $\log p_0$. Note that, for security reasons, no random masks can be generated solely by a particular party since all security is lost if this particular party is corrupted. One may consider asking each party P_i to prepare a pair of $\left[\!\left[R_t^{(i)}\right]\!\right]$ and $\left[\!\left[R_{2t}^{(i)}\right]\!\right]$ and then use $[\![R_t]\!] = \sum_i \left[\!\left[R_t^{(i)}\right]\!\right]$ and $[\![R_{2t}]\!] = \sum_i \left[\!\left[R_{2t}^{(i)}\right]\!\right]$ as the pair of masks. While this solution is secure, it is unacceptably costly; the total communication and computation is comparable to $n \cdot \log p_0$, incurring an overhead of n.[12]

Earlier works also run into a similar issue of how to generate random masks efficiently in the offline phase. In particular, [DN07] introduced the celebrated Vandermonde randomness extraction technique

[8] This is not precise. They need to be exponentially larger than $t \cdot \ell + \log p_0$ to ensure exponentially small statistical security. But we ignore this for the ease of presentation.

[9] As a sanity check, $X \cdot Y$ needs to be smaller than P_{all} of length $n \cdot \ell$. Hence, it is required that $n > 2t$, similar to Shamir-based protocols.

[10] This is also needed for scalar multiplication gate.

[11] Although [GJM+23] focuses on the weighted setting, their techniques are sufficient to derive this. Our innovation, thus far, is the observation that the CRT techniques can be applied to the unweighted setting as well; and when the field is large, it will achieve download rate $\mathcal{O}(1)$.

[12] This is exactly why [GJM+23] incurs an overhead of n.

for *linear secret-sharing schemes*. This is done as follows. Every party P_i will prepare a pair of random masks $\left[\!\left[S_t^{(i)}\right]\!\right]$ and $\left[\!\left[S_{2t}^{(i)}\right]\!\right]$. Note that t of these pairs belong to the corrupted parties and are insecure to use. However, one may still extract $n-t$ secure pairs of masks by

$$\begin{pmatrix} \left[\!\left[R_t^{(1)}\right]\!\right] \\ \vdots \\ \left[\!\left[R_t^{(n-t)}\right]\!\right] \end{pmatrix} = V \cdot \begin{pmatrix} \left[\!\left[S_t^{(1)}\right]\!\right] \\ \left[\!\left[S_t^{(2)}\right]\!\right] \\ \vdots \\ \left[\!\left[S_t^{(n)}\right]\!\right] \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} \left[\!\left[R_{2t}^{(1)}\right]\!\right] \\ \vdots \\ \left[\!\left[R_{2t}^{(n-t)}\right]\!\right] \end{pmatrix} = V \cdot \begin{pmatrix} \left[\!\left[S_{2t}^{(1)}\right]\!\right] \\ \left[\!\left[S_{2t}^{(2)}\right]\!\right] \\ \vdots \\ \left[\!\left[S_{2t}^{(n)}\right]\!\right] \end{pmatrix}.$$

Here, $V \in F^{(n-t)\times n}$ is some fixed matrix. Note that, as long as $n - t = \Theta(n)$, the *amortized* cost of generating one pair of masks is indeed $\mathcal{O}(\log p_0)$. On the security side, the idea is that, for any set $I \subseteq [n]$ of corrupted parties, the resulting pairs $\left\{\left[\!\left[R_t^j\right]\!\right], \left[\!\left[R_{2t}^j\right]\!\right]\right\}_{j=1}^{n-t}$ are uniformly random to the adversary. [DN07] showed that, over any finite field, as long as V is *super-invertible*, this is indeed secure, where super-invertibility means that any $(n-t)\times(n-t)$ submatrix of V is full-rank. In particular, the Vandemonde matrix (Equation 1) is super-invertible.

Technical Challenge: Randomness Extraction over Integers. Can we use similar ideas for our purposes as well? Does the Vandermonde randomness extraction technique also work for CRT secret sharing? We note that, there is a crucial difference between CRT secret sharing and linear secret sharing. For CRT secret sharing, we have to work with *distributions over* $\mathbb{Z}$. In particular, suppose $Z_1, Z_2, \ldots, Z_{n-t}$ are the outputs of $n-t$ multiplication gates, which we aim to mask. We must be able to simulate

$$V \cdot \begin{pmatrix} \left[\!\left[S_{2t}^{(1)}\right]\!\right] \\ \left[\!\left[S_{2t}^{(2)}\right]\!\right] \\ \vdots \\ \left[\!\left[S_{2t}^{(n)}\right]\!\right] \end{pmatrix} + \begin{pmatrix} [\![Z_1]\!] \\ \vdots \\ [\![Z_{n-t}]\!] \end{pmatrix}$$

as a distribution over $\mathbb{Z}^{n-t}$ without the knowledge of the secret values $\{Z_i \mod p_0\}_i$. Essntially, we need to perform *randomness extraction over integers*.

Arguing statistical closeness between two distributions over integers is not common. To our best knowledge, all of these arguments (for instance, the argument used to prove the statistical security of CRT secret sharing) are variants of the so-called *smudging lemma* (refer to Lemma 2). It states that, if $\mathcal{U}$ is uniformly distributed over $\{1, 2, \ldots, M\}$, the $m + \mathcal{U}$ and $\mathcal{U}$ are m/M statistically close. Here, we are facing a drastically different question. The random masks that we generate, namely, each row of $V \cdot \vec{S}$, are *not uniformly distributed over consecutive integers*. In fact, they might be very structured. For instance, if the first row of V consists of only even numbers, then the first mask we generate, i.e., the first coordinate of $V \cdot \vec{S}$ is only supported on even integers. This raises the following key technical question.

What matrix V over $\mathbb{Z}$ suffices for randomness extraction over integers?

Let us begin with some helpful observations.

- **Super-invertibility over $\mathbb{Z}_{p_0}$ is necessary.** If one can simulate the masked integers over $\mathbb{Z}$, one must also be able to simulate the masked integers modulo p_0, i.e., over $\mathbb{Z}_{p_0}$. Therefore, the masked integers modulo p_0 must be uniformly random. Otherwise, it is hopeless to simulate them without the knowledge of the underlying secrets. Based on this, one can arrive at the conclusion that it is necessary that V modulo p_0 is super-invertible.

- **Is super-invertibility sufficient?** Now that we know super-invertibility is necessary, a natural question is whether it also sufficient? Upon closer inspection, one can conclude that super-invertibility over $\mathbb{Z}_{p_0}$ alone is *not* a sufficient condition. Given any super-invertible V, we may multiply the first row of V by, say, 2 (over integers). The resulting matrix is still super-invertible over $\mathbb{Z}_{p_0}$, but now has an even first row. Then, the random mask being generated cannot be used to mask any distributions that are sensitive to the parity distinguisher, i.e., the distinguisher that outputs the parity of the integer. While the distributions that we aim to mask are not necessarily sensitive to parity distinguisher, this serves as an example to show that no matter what argument we use, it must be strong enough to eliminate *all* such attacks.
- **Natural attempt at strengthening super-invertibility.** In the previous example, one may wonder if the example of the entire first row being even is just a pathological case. A natural attempt to avoid such pathological examples is to enforce that each row contains a "1" at a distinct entry. Then, for each mask we generated $\sum_{j\in[n]} v_{i,j} \cdot S^j_{2t}$, one of the summand $v_{i,j} \cdot S^j_{2t}$ satisfies $v_{i,j} = 1$ and now we have the distribution S^j_{2t}, which is uniformly random over consecutive integers. Is this sufficient to mask the i-th row? If one looks at the *marginal distribution* of the i-th mask, the answer is yes; it is simulatable due to the smudging lemma. However, it is unclear if the *joint distribution* is simulatable. This is because the term S^j_{2t} does not solely contribute to the i-th mask; different multiples of S^j_{2t} contribute to different rows as well. Therefore, it remains elusive if this correlation leaks anything.

Key Observation: High Dimensional Smudging Lemma. We now present our solution for this problem. Without loss of generality, let us assume that the first $n - t$ parties are honest and remove the contribution of the corrupted parties from the masks. This simplifies our question and gives us a square matrix $V \in \mathbb{Z}^{(n-t)\times(n-t)}$. To address the integer randomness extraction problem, we derive the following techniques for it. In particular, we note that, while a matrix V over $\mathbb{Z}$ is not necessarily invertible, it must have an *adjugate matrix* $\mathsf{adj}(V)$ such that

$$V \cdot \mathsf{adj}(V) = \mathsf{adj}(V) \cdot V = \det(V) \cdot I,$$

where I is the identity matrix. We refer the reader to Section 3.1 for recalling the basic facts on the adjugate matrix. By relying on that we derive the following key technical lemma, which we call *high-dimensional smudging lemma.*

Lemma 1 (High-dimensional Smudging Lemma)**.** *Let $V \in \mathbb{Z}^{(n-t)\times(n-t)}$. Suppose $S_1, \ldots, S_{n-t}$ are independent distribution uniformly over consecutive integers $\{1, 2, \ldots, L\}$. Let $\Delta_1, \ldots, \Delta_{n-t}$ be fixed shifts that are* divisible *by $\det(V)$. We have*

$$V \cdot \begin{pmatrix} S_1 \\ S_2 \\ \vdots \\ S_{n-t} \end{pmatrix} \approx V \cdot \begin{pmatrix} S_1 \\ S_2 \\ \vdots \\ S_{n-t} \end{pmatrix} + \begin{pmatrix} \Delta_1 \\ \Delta_2 \\ \vdots \\ \Delta_{n-t} \end{pmatrix}.$$

The closeness is upper-bounded by $(n-t)\cdot\frac{\mathsf{max}_{\mathsf{adj}}\cdot\mathsf{max}_{\Delta}}{L}$, where $\mathsf{max}_{\mathsf{adj}}$ and max_{Δ} denotes the maximum absolute values in the adjugate of V and the Δ-vector.

Proof. Consider the following derivation:

$$V \cdot \begin{pmatrix} S_1 \\ S_2 \\ \vdots \\ S_{n-t} \end{pmatrix} = V \cdot \begin{pmatrix} S_1 \\ S_2 \\ \vdots \\ S_{n-t} \end{pmatrix} - \begin{pmatrix} \Delta_1 \\ \Delta_2 \\ \vdots \\ \Delta_{n-t} \end{pmatrix} + \begin{pmatrix} \Delta_1 \\ \Delta_2 \\ \vdots \\ \Delta_{n-t} \end{pmatrix}$$

$$= V \cdot \left(\begin{pmatrix} S_1 \\ S_2 \\ \vdots \\ S_{n-t} \end{pmatrix} - \mathsf{adj}(V) \cdot \begin{pmatrix} \Delta_1/\det(V) \\ \Delta_2/\det(V) \\ \vdots \\ \Delta_{n-t}/\det(V) \end{pmatrix} \right) + \begin{pmatrix} \Delta_1 \\ \Delta_2 \\ \vdots \\ \Delta_{n-t} \end{pmatrix}.$$

Now, the closeness is reduced to the closeness between

$$\begin{pmatrix} S_1 \\ S_2 \\ \vdots \\ S_{n-t} \end{pmatrix} - \mathsf{adj}(V) \cdot \begin{pmatrix} \Delta_1/\det(V) \\ \Delta_2/\det(V) \\ \vdots \\ \Delta_{n-t}/\det(V) \end{pmatrix} \qquad \text{and} \qquad \begin{pmatrix} S_1 \\ S_2 \\ \vdots \\ S_{n-t} \end{pmatrix},$$

for which one can apply standard smudging lemma directly. This closeness is determined by the upper bounds on the shift and the size of the support of the distribution S_i (plus a union bound on the number of rows).

Note that this proof crucially relies on the fact that all shifts Δ_i's are divisible by $\det(V)$. Otherwise, S_1 and $1/2 + S_1$, for instance, will not be close. □

Remark 1. *As a sanity check for this lemma, suppose the first row of V consists solely of even numbers, its determinant will be an even number. This lemma states that the extracted masks can only mask even shifts, which is aligned with our previous discussions.*

This useful tool allows us to simulate the masked integers and argue their closeness. In particular, it implies that we can simulate the view of the protocol as long as the distribution of the secret integer in the simulated view and the real view are shifted by multiple of $\det(V)$. Therefore, as a *security invariant*, we shall consistently enforce that, for any CRT sharing of a wire, the distributions $\mathcal{S}_1$ and $\mathcal{S}_2$ of the secret integer corresponding to different secrets s_1 and s_2 always satisfy $\mathcal{S}_1 = \mathcal{S}_2 + \Delta$ for some Δ that is divisible by $\det(V)$.

To enforce this invariant, for any secret $s \in \mathbb{Z}_{p_0}$, we will always reformat it as an integer S such that $S = s \mod p_0$ and $S = 0 \mod \det(V)$. (Note that this is only possible if p_0 and $\det(V)$ are coprime, which is aligned with our earlier discussion that V must be super-invertible over F_{p_0}.) Consequently, the different integer representatives of different secrets are always separated by a multiple of $\det(V)$. At this point, the reader might wonder the following: the determinant $\det(V)$ should depend on which subset of parties are corrupted; how can the protocol specification depend on which subset of parties are corrupted? A simple fix for this is to pick the least common multiple of $\det(V)$ for all possible V's. For example, for a Vandermonde matrix such as Equation 1, this can simply be $\prod_{1 \leqslant i < j \leqslant n}(j - i)$.

Dependence between $\log p_0$ and n. We are now ready to explain the quadratic dependence between $\log p_0$ and n. Note that, for correctness, the random masks $S^{(i)}$ generated need to be smaller than $p_1 p_2 \cdots p_n$. On the other hand, our high dimensional smudging lemma require them to be random over a support significantly larger than $\mathsf{max}_{\mathsf{adj}}$. Similarly, our reformat of the secret also increases the secret size from p_0 to $p_0 \cdot \det(V)$. For a Vandermonde super-invertible matrix, both $\det(V)$ and $\mathsf{max}_{\mathsf{adj}}$ are of the order n^{n^2} (refer to Section 3.1). Consequently, it implies that we must have $n^{n^2} \leqslant p_1 p_2 \cdots p_n$.

In other words, the cumulative length of the secret shares must be $\Omega\big(n^2 \log n\big)$ in order to utilize our high dimensional smudging lemma. We require a CRT sharing with download rate $\mathcal{O}(1)$; this implies $\log p_0$ must be $\Omega\big(n^2 \log n\big)$ too.

It is worth noting that this is a consequence of our proof technique. We are not aware of any explicit

attacks. To illustrate this point, consider the question of determining the closeness between

$$V \cdot \begin{pmatrix} S_1 \\ S_2 \\ \vdots \\ S_{n-t} \end{pmatrix} \qquad \text{and} \qquad V \cdot \begin{pmatrix} S_1 \\ S_2 \\ \vdots \\ S_{n-t} \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix},$$

where V is any Vandermonde matrix. While our techniques cannot prove that they are close, it remains unclear if they are close or far apart.

Our proof techniques also motivate studying the following research question. Does there exist super-invertible matrix V such that $\det(V)$ and $\mathsf{max}_{\mathsf{adj}}$ are small? In particular, $\exp(\mathcal{O}(n))$ small? If one constructs such a matrix, then it can be modularly plugged into our protocol to obtain MPC protocols for any field F such that $\log |F|$ is linear in n.

Extensions. Our protocol can be readily extended to achieve several useful results of independent interest. We discuss them next.

1. **Ring Computation.** So far, we focused on the setting where the arithmetics is over a prime field $\mathbb{Z}_{p_0}$. However, our protocol is (almost) oblivious to whether p_0 is a prime number or not. The only case where it matters is the super-invertibility of V over $\mathbb{Z}_{p_0}$. For our choice of the Vandermonde matrix (Equation 1), this property holds as long as p_0 is coprime with $\{2, 3, \ldots, n-1\}$. Therefore, our protocol readily extends to all p_0 that are coprime with $\{2, 3, \ldots, n-1\}$. In fact, our main technical sections only assume that $\mathbb{Z}_{p_0}$ is a ring.

2. **Dishonest Majority.** A recent work [GPS22] shows that the packed secret-sharing techniques can be extended to the dishonest majority setting as well. We observe that similar ideas can also be applied to our techniques. In particular, given suitable correlated setups,[13] namely, Beaver triples [Bea92], one may instantiate the standard framework of secret sharing-based dishonest majority protocol with our CRT secret sharing scheme to obtain a dishonest majority MPC protocol with $\mathcal{O}(|C| \cdot \log p_0)$ total communication and computation.

   Interestingly, we still need an integer-reduction step after each multiplication gate, even given Beaver triples. This is because scalar multiplication is not free for CRT secret sharing-based MPC protocols.

3 Preliminaries

All logarithms in this work are of base 2. Throughout the paper, the statistical security parameter is denoted by κ. For any $n \in \mathbb{N}$, we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. For any set S, $s \leftarrow S$ means that s is drawn uniformly randomly from the set S. For two discrete distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ over the same support $\mathcal{U}$, their *statistical distance* is defined as $\mathsf{SD}(\mathcal{D}_1, \mathcal{D}_2) = \frac{1}{2} \cdot \sum_{u \in \mathcal{U}} |\Pr[\mathcal{D}_1 = u] - \Pr[\mathcal{D}_2 = u]|$. We also use $\mathcal{D}_1 \approx_\varepsilon \mathcal{D}_2$ to denote that $\mathsf{SD}(\mathcal{D}_1, \mathcal{D}_2) \leqslant \varepsilon$. For any distribution $\mathcal{D}$ over a ring $\mathbb{R}$ and an element $\alpha \in \mathbb{R}$, we use $\alpha + \mathcal{D}$ to denote the distribution of $\alpha + d$, where d is drawn according to $\mathcal{D}$.

This work considers the integer ring $\mathbb{Z}$ and the ring $\mathbb{Z}_p$. Looking ahead, for our MPC protocol among n parties, every party P_i works in a *different ring* $\mathbb{Z}_{p_i}$ for all $i \in [n]$. To avoid ambiguity, we adopt the following notations. The standard $+$, $-$, and $\cdot$ is reserved for arithmetic operations over the integer ring $\mathbb{Z}$. The arithmetic operations over the ring $\mathbb{Z}_{p_i}$ is denoted by $\oplus_i$, $\ominus_i$, and $\odot_i$ instead.

Often, we need to switch between ring element $\alpha \in \mathbb{Z}_p$ and integers. When we treat a ring element α as an integer, we use the *canonical representation* of α as the representative integer. That is, we enforce $\alpha \in \{0, 1, \ldots, p-1\}$.

[13] This is necessary for information-theoretic dishonest majority protocols.

Finally, the following smudging lemma shall be useful. This lemma is usually stated for the special case $\Delta = 1$. Here, we state a slightly generalized version.

Lemma 2 (Smudging Lemma [AJL+12]). *Let M be a positive integer. Let $\mathcal{D}$ denote a distribution over $\mathbb{Z}$, which is uniform over $\{a, a+\Delta, a+2\cdot\Delta, \ldots, a+M\cdot\Delta\}$ for some $a, \Delta \in \mathbb{Z}$. For any $m \in \mathbb{Z}$, it holds that*

$$\mathsf{SD}\left(\mathcal{D}, \mathcal{D} + m\cdot\Delta\right) \leqslant \frac{|m|}{M},$$

where $|m|$ denotes the absolute value of m.

3.1 Linear Algebra Basics and Some Upper Bounds on the Determinant

For any square matrix M over a commutative ring $\mathbb{R}$, we use $\det(M)$ to denote its determinant. In this work, we will only be concerned with the integer ring $\mathbb{Z}$.

Let I denote the identity matrix. While a matrix M over a ring may not necessarily be invertible, it always has an *adjugate* matrix, denoted by $\mathsf{adj}(M)$, which satisfies

$$\mathsf{adj}(M)\cdot M = M\cdot\mathsf{adj}(M) = \det(M)\cdot I.$$

In particular, the (i,j)-th entry of the adjugate matrix $\mathsf{adj}(M)$ is defined as $(-1)^{(i+j)}\cdot\det\left(M_{-j,-i}\right)$, where $M_{-j,-i}$ stands for the matrix M without the j-th row and i-th column.

We would use the following simple upper bounds for the determinant and the entries of the adjugate matrix of an $m\times m$ Vandermonde matrix with $1\leqslant x_1 < x_2 < \cdots < x_m \leqslant n$:

$$V = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_m \\ \vdots & \vdots & \ddots & \vdots \\ (x_1)^{m-1} & (x_2)^{m-1} & \cdots & (x_m)^{m-1} \end{pmatrix}, \tag{1}$$

It has a determinant $\prod_{1\leqslant i<j\leqslant m}(x_j - x_i)$, which can be upper-bounded by n^{m^2}. Each entry in its adjugate $\mathsf{adj}(V)$ is $(-1)^{(i+j)}\cdot\det\left(V_{-j,-i}\right)$, which can be upper-bounded by

$$(m-1)!\cdot\left(n^{m-1}\right)^{m-1} \leqslant n^{m^2}.$$

This is derived by writing down the determinant by definition and noticing that there are $(m-1)!$ terms, each of which is a product of $m-1$ entries upper-bounded by n^{m-1}.[14]

Looking ahead, these upper bounds will be useful for our security proof (e.g., Lemma 1).

3.2 Chinese Remainder Theorem-based Secret Sharing

We start with formally defining a ramp threshold secret-sharing scheme which shares secrets from a ring $\mathbb{R}$ to a set of sharings in possibly different rings $\mathbb{R}_1, \mathbb{R}_2, \ldots$.

Definition 1 ((n,T,t)-Threshold Secret-sharing Scheme). *Let $t, T \in \mathbb{N}$ denote the privacy threshold and reconstruction threshold, respectively, such that $T \geqslant t$. S secret-sharing scheme among n ($\geqslant T$) parties consists of two algorithms* $(\mathsf{Share}, \mathsf{Reconst})$ *with the following syntax:*

[14] The determinant of the minors of the Vandermonde matrix (i.e., $V_{-j,-i}$) is well-understood. There are explicit formulas for it and, thus, it is possible to derive tighter upper bounds. However, a coarse upper bound, such as the one stated above, suffices for our purpose.

- $\mathsf{Share}_{n,T,t}(s)$ *is a randomized algorithm that takes a ring element* $s \in \mathbb{R}$ *as input and outputs* n *shares* $(s_1, \ldots, s_n) \in \mathbb{R}_1 \times \cdots \times \mathbb{R}_n$.
- $\mathsf{Reconst}_{n,T,t}(\{s_i\}_{i\in I})$ *is a deterministic algorithm that takes a set of shares* $\{s_i\}_{i\in I}$ *such that* $|I| \geqslant T$ *and reconstruct the secret* s.

It should satisfy the perfect correctness and statistical security as follows.

- ***(Perfect) Correctness.*** *For any secret* $s \in \mathbb{R}$ *and any set* $I \subseteq [n]$ *of size* $|I| \geqslant T$ *we have that:*

$$\Pr\left[s' = s \ : \ \begin{matrix}(s_1, s_2, \ldots, s_n) \leftarrow \mathsf{Share}_{n,T,t}(s) \\ s' = \mathsf{Reconst}_{n,T,t}\left(\{s_i\}_{i\in I}\right)\end{matrix}\right] = 1.$$

- ε***-(Statistical) Security.*** *For any set* $I' \subset [n]$ *such that* $I' < t$ *and any two secrets* $s, s' \in \mathbb{F}$, *it holds that the following two distributions are* ε*-statistically close.*

$$\left\{\begin{matrix}(s_1, s_2, \ldots, s_n) \leftarrow \mathsf{Share}_{n,T,t}(s) \\ \textit{Output } \{s_i\}_{i\in I'}\end{matrix}\right\} \approx \left\{\begin{matrix}(s'_1, s'_2, \ldots, s'_n) \leftarrow \mathsf{Share}_{n,T,t}(s') \\ \textit{Output } \{s'_i\}_{i\in I'}\end{matrix}\right\}.$$

In this work, we will use the Chinese Remainder Theorem (CRT) based (non-linear) ramp threshold secret sharing scheme [Mig83, AB83, GRS99, GJM+23]. We briefly describe the scheme below for parameters (n, t, T) closely following [GJM+23].

Let $p_0, p_1, \ldots, p_n$ be pairwise coprime. CRT-based secret sharing scheme takes secrets from the ring $\mathbb{Z}_{p_0}$ and produces sharings from $\mathbb{Z}_1 \times \cdots \times \mathbb{Z}_n$. Define $P_{\mathsf{min}} := \min_{|I|\geqslant T}\left(\prod_{i\in I} p_i\right)$ and $P_{\mathsf{max}} := \max_{|I'|<t}\left(\prod_{i\in I'} p_i\right)$ (Note that $P_{\mathsf{max}} < P_{\mathsf{min}}$). Choose another parameter L, which will depend on n, t, T. Then the procedures $\mathsf{Share}, \mathsf{Reconst}$ work as follows:

- $\mathsf{Share}_{n,T,t}(s)$:
  - Choose uniform random $u \leftarrow \{0, \ldots, L-1\}$
  - Define $S := s + u \cdot p_0$.
  - Output $(s_1, \ldots, s_n)$ where $s_i = S \bmod p_i$
- $\mathsf{Reconst}_{n,T,t}(\{s_i\}_{i\in I})$:
  - Use CRT reconstruction to compute the unique integer $S \bmod P_I$ where $P_I = \prod_{i\in I} p_i$.
  - Output $s := S \bmod p_0$

The following was proven in [GRS99, GJM+23]. We refer the readers to [GJM+23] for formal proofs. It is also proved in an inline fashion within our security proofs (e.g., Claim 1) for our special case when $n = T$.

Lemma 3. *The above* (n, T, t)*-secret sharing scheme is correct as long as* $(L+1) \cdot p_0 < P_{\mathsf{min}}$ *and* ε*-secure for* $\varepsilon \leqslant P_{\mathsf{max}}/L$.

Notation. In our MPC protocol, we use the CRT-based secret sharing scheme with $n = T$; henceforth, $P_{\mathsf{min}} = \prod_{i\in[n]} p_i$. We use notations $\mathsf{Share}_{n,t}$ and $\mathsf{Reconst}_{n,t}$ and call it a (t, n)-sharing. We will use two different L parameters, namely L_t and L_{2t}, in our protocol. For better intuition, we write $[\![X_t]\!] \leftarrow \mathsf{Share}_{n,t}(x)$ to denote the secret sharing of x (as opposed to $[\![x_t]\!]$). This choice makes explicit the integer re-randomization X_t of the secret x. As usual, we write $[\![X_t]\!]_i$ for the i-th share of a (t, n)-sharing; consequently, $[\![X_t]\!]_i = X_t \mod p_i$.

3.3 Computational Complexity of Certain Computations

When considering the computational complexity of our MPC protocol, it is helpful to discuss the complexity of performing several operations, which are commonplace in our protocol. We emphasize that these complexities are measured *at bit level* in order to compare operations over different rings fairly.

1. **Ring Multiplication.** As mentioned, in our protocol, each party performs ring arithmetics over $\mathbb{Z}_p$ for some p. Elements in $\mathbb{Z}_p$ are of bit-length $\log p$. Therefore, it is clear that addition/subtraction can be performed in $\mathcal{O}(\log p)$ time. Moreover, we note here that multiplication can also be performed in time quasilinear in the bit length [SS71, Für07, HVDHL16, HVDH21], e.g., $\mathcal{O}(\log p \cdot \mathsf{loglog}\, p \cdot \mathsf{logloglog}\, p)$.[15] We refer the readers to [HVDH21] for more discussions on the relevant works. Throughout the paper, when we state the computational complexities, we ignore the lower-order $\mathsf{loglog}\, p$ terms.

2. **CRT Sharing/Reconstruction.** Throughout the protocol, parties need to do CRT share and reconstruct repetitively. Sharing requires parties to sample a random integer X and compute $[\![X]\!]_i = X \mod p_i$ for every i. Reconstruction requires parties to take as input $[\![X]\!]_1, \ldots, [\![X]\!]_n$ and reconstruct X. Since X is of bit-length roughly $\log p_1 + \log p_2 + \cdots + \log p_n$, computing $X \mod p_i$ for all $i \in [n]$ seems to require $(\log p_1 + \log p_2 + \cdots + \log p_n) \cdot n$ bit computation. Here, we note that it is well-known that this can be done in $(\log p_1 + \log p_2 + \cdots + \log p_n) \cdot \log n$ time through divide-and-conquer techniques (a.k.a., remainder tree techniques). Similar ideas apply to the reconstruction as well.[16] We refer the readers to, for instance, the note [Ber04] for relevant literature and discussion.

3. **Vandermonde Matrix Multiplication.** Finally, in our protocol, parties need to compute a special type of matrix multiplication $\vec{r} = V \cdot \vec{s}$, where $\vec{r}$ and $\vec{s}$ are vectors and V is a Vandermonde matrix (See Equation 1). Here, one may apply the standard FFT techniques to compute this matrix multiplication over any ring $\mathbb{Z}_p$ in $\mathcal{O}(n \log^2 n)$ ring arithmetics.[17] In particular, we emphasize that, since different parties are performing this matrix multiplication over different rings, the i-th party takes $\mathcal{O}(n \log n) \cdot \log p_i$ bit-operations to perform this matrix multiplication.

In the rest of the paper, we ignore the $\mathsf{loglog}\, p$ factor in ring operations. We sometimes use the $\widetilde{\mathcal{O}}()$ notation, which hides the $\log^2 n$ factor for CRT operations and Vandermonde matrix multiplications.

Remark 2 (Comparison with prior works)**.** *We emphasize that the lower-order terms* $\log \log\, p_0$ *and* $\log^2 n$ *for Ring arithmetics and Vandermonde matrix multiplication also exist for all prior works. Moreover, for prior works, generating Shamir's secret sharing also incurs a* $\log n$ *factor for the use of FFT similar to our CRT sharing. Our work focuses on the computational complexities of our MPC protocol and, hence, we explicitly argue these complexities.*

3.4 Secure Multiparty Computation

This section includes the standard definition of secure multiparty computation in the semi-honest and statistical security setting (see, e.g., [AL17]).

[15] The original FFT-based techniques from [SS71] already gives a practically efficient algorithm with asymptotic complexity $\mathcal{O}(\log p \cdot \mathsf{loglog}\, p \cdot \mathsf{logloglog}\, p)$. Asymptotically, the state-of-the-art is given by the recent work [HVDH21] with complexity $\mathcal{O}(\log p \cdot \mathsf{loglog}\, p)$. This is *conjectured* to be optimal.

[16] For instance, to compute $X \mod p_i$ for $i = 1, 2, 3, 4$. One starts by computing $X \mod p_1 p_2 p_3 p_4$. Utilizing this, one computes $X \mod p_1 p_2$ and $X \mod p_3 p_4$. Utilizing $X \mod p_1 p_2$, one may further compute $X \mod p_1$ and $X \mod p_2$, and so on. Clearly, the reverse procedure also gives a quasilinear time for reconstruction.

[17] We note that, given any input $\vec{s}$, computing $V^\intercal \cdot \vec{s}$ corresponds to batch polynomial evaluations, which is well-known that it can be computed over any ring $\mathbb{Z}_p$ in $\mathcal{O}(n \log^2 n)$ ring arithmetics [BM74]. The task that we need to perform is slightly different, i.e., $V \cdot \vec{s}$ instead of $V^\intercal \cdot \vec{s}$. For this, we note that there is a beautiful theorem, which states: for any matrix A, the time complexity of performing $A \cdot \vec{x}$ with input $\vec{x}$ is near-identical (except for an additive difference linear in n) to the time complexity of performing $A^\intercal \cdot \vec{x}$. We refer the readers to [Rud23, Theorem 4.3.1] for a statement and proof.

Definition 2. *Let $C : \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_n \to \mathcal{Y}$ be some function, which is modeled as an arithmetic circuit. We say that a protocol π ε-securely realized C with corruption threshold t if the following holds. For any input $\vec{x}$ and any subset $I \subset [n]$ such that $|I| \leqslant t$, there exists an efficient simulator $\mathcal{S}$ such that*

$$\mathsf{SD}\left(\left(\mathcal{S}(I, \vec{x}_I, C(\vec{x})), C(\vec{x})\right), \left(\mathsf{view}_I^{\pi}(\vec{x}), \mathsf{Output}^{\pi}(\vec{x})\right)\right) \leqslant \varepsilon.$$

In particular, when the marginal distributions $C(\vec{x})$ and $\mathsf{Output}^{\pi}(\vec{x})$ are identical, this protocol is perfectly correct.

4 Scalable Honest Majority MPC

This section presents our honest majority scalable MPC protocol. We consider the semi-honest and information-theoretic security setting. First, we provide the parameters used in the protocol:

Parameters. This protocol is parameterized by (1) the moduli $p_1, p_2, \ldots, p_n$ of each party and (2) two secret sharing parameters L_t and L_{2t} to be used for the secret-sharing schemes $(\mathsf{Share}_{n,t}, \mathsf{Reconst}_{n,t})$ and $(\mathsf{Share}_{n,2t}, \mathsf{Reconst}_{n,2t})$ respectively. Looking ahead, the correctness and security, as well as the communication and computation complexity, will depend on these parameters. Now, we present a small gadget used in the main protocol. This small gadget ensures that the different *representative integers* $\bar{x}$ we select for different secret elements x are always distanced by a multiple of $\prod_{1 \leqslant i < j \leqslant n}(j - i)$, which is crucial for our security proof.

$$\mathsf{ReFormat}(x) \to \bar{x}.$$

This (deterministic) algorithm takes an integer $x \in \mathbb{Z}_{p_0}$ as input and uses CRT to output the unique integer $\bar{x}$ satisfying all of the below:

- $0 \leqslant \bar{x} < p_0 \cdot \prod_{1 \leqslant i < j \leqslant n}(j - i)$.
- $\bar{x} = x \bmod p_0$
- $\bar{x} = 0 \bmod \prod_{1 \leqslant i < j \leqslant n}(j - i)$

4.1 Protocol Description and the Main Theorem

The full description of the protocol is presented in Figure 1 and Figure 2. In particular, Figure 1 presents the offline phase of the protocol, where parties generate a batch of random masks that will be used in the online phase. This part of the protocol only depends on the size of the circuit; in particular, they are independent of the input of each party. Figure 2 presents the online phase of the protocol, where parties emulate the circuit evaluation gate by gate as similarly done in all prior works. In particular, after each multiplication gate, parties will utilize the random masks generated in the offline phase to perform an "integer reduction" on the random integer involved in the CRT secret sharing.[18]

Next, we state the main theorem for our protocol. Section 4.2 interprets our main theorem through different settings. The proof of the main theorem is presented in Section 4.3.

Theorem 3 (Main Theorem). *For any $n, t \in \mathbb{N}$ such that $t < n/2$ consider parameters $p_0, p_1, \ldots, p_n$, $L_t, L_{2t} \in \mathbb{N}$ for which $2, 3, \ldots, n, p_0, p_1, \ldots, p_n$ are pairwise coprime.[19] Let $P_{\mathsf{all}} = \prod_{i \in [n]} p_i$ and $P_{\mathsf{max}} = \mathsf{max}_{I \subset [n], |I| \leqslant t} \left(\prod_{i \in I} p_i\right)$. Then, for any arithmetic circuit C over $\mathbb{Z}_{p_0}$, the protocol in Figure 1 and 2 realizes C among n parties allowing at most t passive corruptions with the following guarantees:*

[18] This is similar to the "degree reduction" on the random polynomials involved in Shamir's secret sharing.

[19] For example, $p_0, p_1, \ldots, p_n$ can be distinct prime numbers $> n$. In this particular case, $\mathbb{Z}_{p_0}$ is a finite field.

- ***Correctness.*** *It is perfectly correct as long as*

$$\log P_{\mathsf{all}} - 3 > \max \left\{ \begin{array}{c} 2\log|C| + 2n\log n + 2\log L_t + 2\log p_0, \\ 2\log|C| + 2n^2\log n + 2\log p_0, \\ n\log n + \log p_0 + \log L_{2t} \end{array} \right\}.$$

- ***Security.*** *It is ε-statistically secure where*

$$\varepsilon = \mathcal{O}\left(\frac{|C|^3 \cdot p_0^2 \cdot n^{3n^2} \cdot P_{\mathsf{max}} \cdot L_t}{L_{2t}} + \frac{|C| \cdot p_0 \cdot n^{2n^2} \cdot P_{\mathsf{max}}}{L_t} \right).$$

- ***Efficiency.*** *For a non-king party P_i (with associated modulo p_i), the communication and computation complexity are*

$$\mathcal{O}\left(\log P_{\mathsf{all}} \cdot \frac{|C|}{n-t} + |C| \cdot \log p_i \right) \quad \textit{and} \quad \mathcal{O}\left(\left(\log P_{\mathsf{all}} \cdot \frac{|C|}{n-t} + |C| \cdot \log p_i \right) \cdot \log^2 n \right).$$

For a king party, there is an additional communication and computation complexity of

$$\mathcal{O}(\log P_{\mathsf{all}} \cdot |C|) \quad \textit{and} \quad \mathcal{O}(\log P_{\mathsf{all}} \cdot |C| \cdot \log n).^{20}$$

The overall communication and computation complexity for all parties are

$$\mathcal{O}(\log P_{\mathsf{all}} \cdot |C|) \quad \textit{and} \quad \mathcal{O}\big(\log P_{\mathsf{all}} \cdot |C| \cdot \log^2 n\big).$$

Additionally, similar to prior works (e.g., [BGW88, DN07]), the round complexity of this protocol is linear in the depth of the circuit.

4.2 Implication of the Main Theorem

Theorem 3 is a general theorem for all possible choices of parameters. Next, we interpret this theorem through different instantiations.

Large Field Regime. First, when the field size is sufficiently large, in particular $\log p_0 = \Omega\big(n^2 \log n\big)$, our protocol achieves total communication and computation $\mathcal{O}(|C| \cdot \log F)$ (except for a $\log^2 n$ factor on the computational complexity). To see this, let us define

$$\ell := c \cdot \left(\frac{\log p_0}{n - 2t} \right)$$

for some constant c. Set each p_i to be a distinct prime number of length ℓ. Furthermore, let

$$\begin{aligned} \log L_t &= t \cdot \ell + \log p_0 \\ \log L_{2t} &= 2 \cdot (t \cdot \ell) + 3 \cdot \log p_0 \end{aligned}$$

Intuitively, by setting the parameter in this way, we ensure that

$$\log P_{\mathsf{all}} = \Theta(\log p_0),$$

i.e., our secret sharing is "rate-1". By our main theorem, one may verify that as long as (1) one picks c appropriately and (2) the ring size p_0 is superpolynomial in the circuit size $|C|$, then we have the following corollary.

[20] If parties take turns to be the king party, then this complexity can be distributed among the n parties. In particular, it can be evenly distributed if parties act as the king party equally often.

Offline Phase. Each party P_i:

- Generates a pair of random masks by sampling $s^{(i)} \leftarrow \{0, 1, \ldots, p_0 - 1\}$.
- Compute sharing $\left[\!\left[S_t^{(i)}\right]\!\right] \leftarrow \mathsf{Share}_{n,t}(s^{(i)})$ and $\left[\!\left[S_{2t}^{(i)}\right]\!\right] \leftarrow \mathsf{Share}_{n,2t}(p_0 - s^{(i)})$.
- Send $\left[\!\left[S_t^{(i)}\right]\!\right]_j$ and $\left[\!\left[S_{2t}^{(i)}\right]\!\right]_j$ to P_j for every $j \in [n]$.
- After receiving the shares of masks from all parties, each party P_i locally extracts $(n-t)$ masks as:

$$\begin{pmatrix} \left[\!\left[R_t^{(1)}\right]\!\right]_i \\ \vdots \\ \left[\!\left[R_t^{(n-t)}\right]\!\right]_i \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 3 & \cdots & n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 2^{n-t-1} & 3^{n-t-1} & \cdots & n^{n-t-1} \end{pmatrix} \odot_i \begin{pmatrix} \left[\!\left[S_t^{(1)}\right]\!\right]_i \\ \left[\!\left[S_t^{(2)}\right]\!\right]_i \\ \vdots \\ \left[\!\left[S_t^{(n)}\right]\!\right]_i \end{pmatrix}$$

and

$$\begin{pmatrix} \left[\!\left[R_{2t}^{(1)}\right]\!\right]_i \\ \vdots \\ \left[\!\left[R_{2t}^{(n-t)}\right]\!\right]_i \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 3 & \cdots & n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 2^{n-t-1} & 3^{n-t-1} & \cdots & n^{n-t-1} \end{pmatrix} \odot_i \begin{pmatrix} \left[\!\left[S_{2t}^{(1)}\right]\!\right]_i \\ \left[\!\left[S_{2t}^{(2)}\right]\!\right]_i \\ \vdots \\ \left[\!\left[S_{2t}^{(n)}\right]\!\right]_i \end{pmatrix}.$$

Note that the above computation done by P_i is under modulo p_i. This step is repeated for $\lceil |C|/(n-t) \rceil$ times to generate (double-sharings) of enough masks for the online phase, namely, $\left\{\left[\!\left[R_t^{(1)}\right]\!\right], \left[\!\left[R_{2t}^{(1)}\right]\!\right]\right\}, \left\{\left[\!\left[R_t^{(2)}\right]\!\right], \left[\!\left[R_{2t}^{(2)}\right]\!\right]\right\}, \ldots, \left\{\left[\!\left[R_t^{(|C|)}\right]\!\right], \left[\!\left[R_{2t}^{(|C|)}\right]\!\right]\right\}$.

Figure 1: Offline phase of our protocol parameterized by $p_1, \ldots, p_n$, L_t, and L_{2t}.

Corollary 4 (Large Field Regime)**.** *For any number of parties n, corruption threshold $t = (1/2 - \delta) \cdot n$ for some constant δ and ring size p_0 with $\log p_0 = \Omega\big(n^2 \log n\big)$ that is coprime with $\{2, 3, \ldots, n\}$, our protocol realizes any arithmetic circuit C over $\mathbb{Z}_{p_0}$ with the following guarantees.*

- *The overall communication is $\mathcal{O}(|C| \cdot \log p_0)$ bits.*
- *The overall computation is $\mathcal{O}\big(|C| \cdot \log p_0 \cdot \log^2 n\big)$ bit operation.*
- *This protocol is perfectly correct.*
- *This protocol is $\mathsf{poly}(|C|/p_0)$-secure, which is exponentially small for exponentially large fields.*

Small Field Regime. When the field size is not large enough, in particular, $\log p_0 = \mathrm{o}\big(n^2 \log n\big)$, the n^{n^2} terms in the correctness and security terms become dominant and one no longer can pick $p_1, \ldots, p_n$ such that $\log P_{\mathsf{all}} = \Theta(\log p_0)$. Instead, one have to make sure that

$$\log P_{\mathsf{all}} = \Theta\big(n^2 \log n\big).$$

Therefore, we can define

$$\ell := c \cdot \left(\frac{n^2 \log n}{n - 2t}\right)$$

Online Phase. The online phase follows the general structure of the DN [DN07] protocol. That is, parties start by sharing their input. Next, they emulate the circuit gate by gate, while maintaining an invariant that parties collectively hold a (t, n)-sharing for all wires as the protocol proceeds. The addition gates do not require communication, while the multiplication gates do. In particular, parties consume one random mask $\left\{ \left[\!\left[R_t^{(i)} \right]\!\right], \left[\!\left[R_{2t}^{(i)} \right]\!\right] \right\}$ per multiplication gate to reduce the size of the integer being shared and maintain the invariant. Finally, once the computation reaches the final level, parties learn the output of the protocol by broadcasting (and reconstructing) the secret shares of the output wires.

- **Input Sharing.** Each party P_i, holding input $x^{(i)} \in \mathbb{Z}_{p_0}$, compute $\bar{x}^{(i)} \leftarrow \mathsf{ReFormat}(x^{(i)})$ and $\left[\!\left[\bar{X}_t^{(i)} \right]\!\right] \leftarrow \mathsf{Share}_{n,t}(\bar{x}^{(i)})$. It sends $\left[\!\left[\bar{X}_t^{(i)} \right]\!\right]_j$ to all party P_j.
- **Circuit Emulation.** Parties emulate the circuit evaluation gate by gate as follows.
  - *Addition* $x + y \in \mathbb{Z}_{p_0}$. Let $[\![X_t]\!]$ and $[\![Y_t]\!]$ denote the (t, n)-sharing of the input wires x and y. Each party P_i simply sets $[\![X_t]\!]_i \oplus_i [\![Y_t]\!]_i$ as its (t, n)-sharing of the output wire.
  - *Multiplication* $x \cdot y \in \mathbb{Z}_{p_0}$.[a] Let $[\![X]\!]$ and $[\![Y]\!]$ denote the secret sharing of the input wires x and y. Let $\{[\![R_t]\!], [\![R_{2t}]\!]\}$ be the sharing of the next pair of random masks that is not consumed yet. Then:
    - Each party P_i sends to the king party
    $$m_i := ([\![X_t]\!]_i \odot_i [\![Y_t]\!]_i) \oplus_i [\![R_{2t}]\!]_i .$$
    - After receiving $\{m_1, \ldots, m_n\}$, the king party reconstructs
    $$m \leftarrow \mathsf{Reconst}_{n,2t}(\{m_i\}_{i\in[n]})$$
    and then computes $\bar{m} \leftarrow \mathsf{ReFormat}(m)$. Then it sends $\bar{m}$ to all non-king parties (Note that $\bar{m}$ is of length $\geqslant \log p_0$, which is large. To avoid $n \cdot \log p_0$ communication, the king party can simply send $\bar{m} \mod p_i$ to the i-th party).
    - Each party (king and non-king) sets own (t, n)-sharing as $[\![R_t]\!] \oplus_i \bar{m}$.
- **Output Reconstruction.** Let $[\![W_t]\!]$ denote the secret sharing of the output wire w. Each party P_i broadcast its share $w_i := [\![W]\!]_i$. After receiving $\{w_1, \ldots, w_n\}$, parties reconstruct $w \leftarrow \mathsf{Reconst}_{n,t}(\{w_i\}_{i\in[n]})$.

[a] For multiplication with a (public) scalar, we also need to perform the "integer reduction" step as described next. This is different from prior works due to the non-linearity nature of the CRT secret sharing. Since this is a special case of multiplication of two privately held values, we skip the details of the same here.

Figure 2: Online phase of our protocol parameterized by $p_1, \ldots, p_n$, L_t, and L_{2t}.

for appropriate constant c. Furthermore, one sets $\log p_i = \ell$ and

$$\begin{aligned} L_t &= t \cdot \ell + \log p_0 \\ L_{2t} &= 2t \cdot \ell + 3 \log p_0 \end{aligned}$$

By our main theorem, we can similarly obtain the following corollary.

Corollary 5 (Small Field Regime)**.** *For any number of parties n, corruption threshold $t = (1/2 - \delta) \cdot n$ for some constant δ and ring size p_0 with $\log p_0 = o\big(n^2 \log n\big)$ that is coprime with $\{2, 3, \ldots, n\}$, our protocol realizes any arithmetic circuit C over $\mathbb{Z}_{p_0}$ with the following guarantees.*

- *The overall communication is $\mathcal{O}\big(|C| \cdot (n^2 \log n)\big)$ bits.*
- *The overall computation is $\mathcal{O}\big(|C| \cdot (n^2 \log n) \cdot \log^2 n\big)$ bit operation.*
- *This protocol is perfectly correct.*
- *This protocol is $\mathsf{poly}(|C|/n^{n^2})$-secure, which is exponentially small in n.*

Note that, when $\log p_0$ is close to $n^2 \log n$, the total communication and computation complexity is near-optimal.

4.3 Proof of the Main Theorem

This section proves Theorem 3. The efficiency and correctness are easier to see, so we prove them first. After that, we argue the statistical security of our protocol.

Efficiency. We measure the communication and computation at the bit level. For example, the communication of sending an element modulo p_i is $\log p_i$; the computation complexity of computing modulo p_i arithmetics is also $\widetilde{\mathcal{O}}\left(\log p_i\right)$. We refer readers to Section 3.3 for relevant discussion on the computation complexity of multiplication, CRT invocation, and Vandermonde matrix multiplication.

In the offline phase, P_i samples and shares $|C|/(n-t)$ pair of integers $\left[\!\left[S_t^{(i)}\right]\!\right]$ and $\left[\!\left[S_{2t}^{(i)}\right]\!\right]$, which are of length $\leqslant \log(L_t \cdot p_0)$ and $\leqslant \log(L_{2t} \cdot p_0)$. The computation complexity of sampling and computing the secret shares are $\widetilde{\mathcal{O}}\left(\log P_{\mathsf{all}}\right)$. The communication complexity of sending the secret shares is also $\mathcal{O}(\log P_{\mathsf{all}})$.

After receiving the shares, each party locally computes the Vandermonde matrix multiplication, which is of computational complexity $\mathcal{O}\big(n \cdot \log^2 n \cdot \log p_i\big)$ per batch of $n - t$ random masks generated. Overall, party P_i performs
$\mathcal{O}\big(|C| \cdot \log^2 n \cdot \log p_i\big)$ bit computation for all matrix multiplication during the offline phase.

In the online phase, the initial input sharing step and the final output reconstruction step incur a one-time cost of communication/computation complexity $\mathcal{O}(\log P_{\mathsf{all}})$ per party. During the circuit emulation step, for party P_i, each gate incurs a constant number of modulo-p_i arithmetics, and hence, the overall communication and computation complexity is $\mathcal{O}(|C| \cdot \log p_i)$.

To sum up, the overall communication and computation complexity (ignoring $\log^2 n$ factor) for a non-king party P_i is

$$\mathcal{O}\left((\log p_1 + \log p_2 + \cdots + \log p_n) \cdot \frac{|C|}{n-t} + |C| \cdot \log p_i\right).$$

For the king party, additional communication and computation are required to reconstruct and broadcast the secrets after each multiplication gate. This complexity is

$$\mathcal{O}(|C| \cdot (\log p_1 + \log p_2 + \cdots + \log p_n)).$$

Since $t < n/2$, which means $n - t = \Omega(n)$, the overall communication and complexity for all parties, therefore, is

$$\mathcal{O}(|C| \cdot (\log p_1 + \log p_2 + \cdots + \log p_n)).$$

Correctness. Observe that this protocol is perfectly correct as long as the integer associated with each wire never exceeds $p_1 p_2 \cdots p_n$. This includes the intermediate integers computed during the multiplication step, i.e., the integer $X \cdot Y + R_{2t}^{(j)}$. Therefore, to ensure perfect correctness, we just need to upper bound the maximum possible integer being computed during the protocol.

Clearly, integers of the form $X \cdot Y + R_{2t}^{(j)}$ have the highest upper bound. Now, we upper bound each term separately. First, one can clearly upper-bound $R_{2t}^{(j)}$ as

$$n \cdot n^{n-t-1} \cdot (L_{2t} \cdot p_0) \leqslant n^n \cdot L_{2t} \cdot p_0.$$

For X and Y, they are integers representing some input wires. They will, at most, be the summation of $\leqslant |C|$ wires.[21] Each summand wire could be the output wire of a multiplication gate, which means the integer associated with it is of the form $R_t^{(j)} + m$. One can upper-bound it as $n \cdot n^{n-t-1} \cdot (L_t \cdot p_0) + p_0 \cdot \prod_{1 \leqslant i < j \leqslant n}(j - i)$. Therefore,

$$X \leqslant |C| \cdot \left(n^n \cdot L_t \cdot p_0 + n^{n^2} \cdot p_0\right)$$

and

$$X \cdot Y \leqslant |C|^2 \cdot \left(n^n \cdot L_t \cdot p_0 + n^{n^2} \cdot p_0\right)^2.$$

In conclusion, this protocol is perfectly correct as long as[22]

$$\log P_{\mathsf{all}} - 3 > \max \left\{ \begin{array}{c} 2\log|C| + 2n\log n + 2\log L_t + 2\log p_0; \\ 2\log|C| + 2n^2\log n + 2\log p_0; \\ n\log n + \log p_0 + \log L_{2t} \end{array} \right\}$$

Security. The description of the simulator is presented in Figure 3. We shall argue that the view simulated by Figure 3 is statistically close to the real view. This is broken into three parts. First, we argue that the distributions of the secret shares of the corrupted parties are statistically close. Second, we argue that the communication that happened during the circuit emulation step is statistically close. Finally, we argue that the communication that happened during the output reconstruction step is statistically close.

First, we argue that the secret shares of the input the corrupted parties see are indistinguishable between the real and simulated views.[23] Note that, for any distribution $\mathcal{D}$ over $\mathbb{Z}$, the distribution of $\{\mathcal{D} \mod p_i\}_{i \in I}$ is equivalent to the distribution of $\mathcal{D} \mod \prod_{i \in I} p_i$ by the CRT bijective mapping. Therefore, instead of arguing the distribution of the secret shares, we directly argue the distribution modulo $\prod_{i \in I} p_i$. This part of the proof is summarized as the following claim.

Claim 1. *Assume that $p_0, p_1, \ldots, p_n$ are coprime. For any integer x and L, the distribution of*

$$\left(x + p_0 \cdot \mathcal{U}_{\{0,1,\ldots,L-1\}}\right) \mod \prod_{i \in I} p_i$$

is $\frac{\prod_{i \in I} p_i}{L}$-close to uniform, where $\mathcal{U}_{\{0,1,\ldots,L-1\}}$ is the uniform distribution over $\{0, 1, \ldots, L-1\}$.

[21] Multiplication will not increase the size of the integer since the "integer-reduction" step after each multiplication gate brings a large integer back to be a smaller one.

[22] The "-3" ensures that each additive term in the upper bound is less than $p_1 p_2 \cdots p_n / 8$, which suffices.

[23] This is basically due to the statistical security of the CRT secret sharing. We include it below for completeness.

The simulator takes as input the set $I \subseteq [n]$ of corrupted parties, their input x_I, and the output of the protocol $C(\vec{x})$. It proceeds to simulate the view of the adversary as follows.

- **Offline Phase.** The simulator simulates the honest party as described by the protocol specification since this step does not require any input from the honest party.
- **Input Sharing.** The simulator simulates the honest party with their inputs set as 0.
- **Circuit Emulation.** The simulator simulates the honest party as described by the protocol specification. This is well-defined since the simulator holds honest parties' secret shares for all the input wires of the circuits (from the input sharing phase).
- **Output Reconstruction.** For the output wire w, the simulator first invokes the CRT on the honest parties' secret shares to find the unique $0 \leqslant W < \prod_{i\in[n]\setminus I} p_i$ such that $W \mod p_i$ equals to the honest parties' secret share for all $i \in [n] \setminus I$. This allows the simulator to recover the corrupted parties' secret shares as $W \mod p_i$ for all $i \in I$. Now, the simulator checks if $W \mod p_0$ is consistent with the output $C(\vec{x})$. If not, it performs the following additional steps.
  - It invokes CRT to find the integer $0 \leqslant \Delta < p_0 \cdot \left(\prod_{i\in I} p_i\right) \cdot \left(\prod_{i<j}(j-i)\right)$ such that

$$\Delta = W - C(\vec{x}) \mod p_0 \qquad \text{and} \qquad \Delta = 0 \mod \prod_{i\in I} p_i \cdot \prod_{i<j}(j-i).$$

  - It shifts every honest party's secret share of the output wire by $\Delta \mod p_i$ for all $i \in [n] \setminus I$.

  It proceeds to broadcast the honest parties' secret shares of the output wire.

Figure 3: The Simulator

Proof of Claim 1. Let $L' \leqslant L$ be the largest multiple of $\prod_{i\in I} p_i$. Observe that

$$\mathsf{SD}\left(\mathcal{U}_{\{0,1,\ldots,L-1\}}, \mathcal{U}_{\{0,1,\ldots,L'-1\}}\right) \leqslant \frac{L-L'}{L} \leqslant \frac{\prod_{i\in I} p_i}{L}.$$

On the other hand,

$$\left(x + p_0 \cdot \mathcal{U}_{\{0,1,\ldots,L'-1\}}\right) \mod \prod_{i\in I} p_i$$

is exactly uniformly random. This completes the proof. □

This shows that the adversary's secret shares of the input in the real and simulated views are statistically close, where there is a simulation error of $\mathcal{O}\left(n \cdot \frac{\prod_{i\in I} p_i}{L_t}\right)$. Here, the factor n is due to the union bound on the number of honest parties. This simulation error is dominated by the simulation error in the third part. Hence, the theorem statement of Theorem 3 does not include this term.

In the rest of the analysis, we fix the secret shares of the corrupted parties. This includes both the secret shares of the inputs and the preprocessed masks $S_t^{(i)}$ and $S_{2t}^{(i)}$.

Remark 3 (Conditional Distribution of CRT Sharing)**.** *Before we proceed, it is helpful to note that,* conditioned on the corrupted parties' secret shares*, CRT sharing distribution of the form* $x + p_0 \cdot \mathcal{U}_{\{0,1,\ldots,L-1\}}$ *would become*

$$x' + \left(p_0 \cdot \prod_{i\in I} p_i\right) \cdot \mathcal{U}_{\left\{0,1,\ldots,\left\lfloor \frac{L}{\prod_{i\in I} p_i} \right\rfloor - 1\right\}},$$

where $0 \leqslant x' < p_0 \cdot \prod_{i\in I} p_i$ *is obtained by invoking CRT on modulo* p_0 *and modulo* $\prod_{i\in I} p_i$*. That is,* $x' = x \mod p_0$ *and* $x' \mod \prod_{i\in I} p_i$ *is consistent with the corrupted parties' secret share. Intuitively, for the randomness* L *contained in a CRT secret sharing, its* lower-order randomness $L \mod \prod_{i\in I} p_i$ *is used to guarantee that the corrupted parties' secret shares are uniformly random. Its* higher-order randomness $\frac{L}{\prod_{i\in I} p_i}$ *is* independent of this conditioning *and will be used later for the indistinguishability for the rest of the simulation.*

Second, we proceed to prove the statistical closeness between the real and simulated views of the circuit emulation step. In this step, only the multiplication and scalar multiplication require communication. By our correctness guarantee, all the integer computation that happened during the protocol execution never exceeds $p_1 p_2 \cdots p_n$. Therefore, the view of the "integer-reduction" protocol is equivalent to the integer computed. For example, for a multiplication gate between a wire x and y, the view is equivalent to the integer $X \cdot Y + R_{2t}^{(j)}$.

Instead of proving the distribution of every single such integer is statistically close between the real view and simulated view, we shall prove $(n-t)$ such integers in a batch. That is, we shall prove

$$\begin{pmatrix} R_{2t}^{(1)} \\ R_{2t}^{(2)} \\ \vdots \\ R_{2t}^{(n-t)} \end{pmatrix} + \begin{pmatrix} A^{(1)} \\ A^{(2)} \\ \vdots \\ A^{(n-t)} \end{pmatrix} \approx \begin{pmatrix} R_{2t}^{(1)} \\ R_{2t}^{(2)} \\ \vdots \\ R_{2t}^{(n-t)} \end{pmatrix} + \begin{pmatrix} B^{(1)} \\ B^{(2)} \\ \vdots \\ B^{(n-t)} \end{pmatrix}. \tag{2}$$

Here, $A^{(1)}, \ldots, A^{(n-t)}$ denotes the distribution in the real view, while $B^{(1)}, \ldots, B^{(n-t)}$ denotes the distribution in the simulated view. They are either of the form $X \cdot Y$ or $\alpha \cdot X$ depending on whether the gate is multiplication or scalar multiplication. Since the simulator does not use the correct input for the honest parties, $A^{(i)}$ and $B^{(i)}$ may not be the same distribution. However, the masks $R_{2t}^{(i)}$ are honestly simulated, and therefore, they are identically distributed in both worlds. This gives the claim (Equation 2) that we need to prove. Openning up the definition of $R_{2t}^{(i)}$ gives

$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 3 & \cdots & n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 2^{n-t-1} & 3^{n-t-1} & \cdots & n^{n-t-1} \end{pmatrix} \cdot \begin{pmatrix} S_{2t}^{(1)} \\ S_{2t}^{(2)} \\ \vdots \\ S_{2t}^{(n)} \end{pmatrix} + \begin{pmatrix} A^{(1)} \\ A^{(2)} \\ \vdots \\ A^{(n-t)} \end{pmatrix}$$

$$\approx \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 3 & \cdots & n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 2^{n-t-1} & 3^{n-t-1} & \cdots & n^{n-t-1} \end{pmatrix} \cdot \begin{pmatrix} S_{2t}^{(1)} \\ S_{2t}^{(2)} \\ \vdots \\ S_{2t}^{(n)} \end{pmatrix} + \begin{pmatrix} B^{(1)} \\ B^{(2)} \\ \vdots \\ B^{(n-t)} \end{pmatrix}.$$

Since the adversary knows $S_{2t}^{(i)}$ entirely for all $i \in I$, it suffices to remove the contribution of these $S_{2t}^{(i)}$ from both sides of the equation. Let $[n] \setminus I = \{i_1, i_2, \ldots, i_{n-t}\}$. This gives

$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ i_1 & i_2 & i_3 & \cdots & i_{n-t} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (i_1)^{n-t-1} & (i_2)^{n-t-1} & (i_3)^{n-t-1} & \cdots & (i_{n-t})^{n-t-1} \end{pmatrix} \cdot \begin{pmatrix} S_{2t}^{(i_1)} \\ S_{2t}^{(i_2)} \\ \vdots \\ S_{2t}^{(i_{n-t})} \end{pmatrix} + \begin{pmatrix} A^{(1)} \\ A^{(2)} \\ \vdots \\ A^{(n-t)} \end{pmatrix}$$

$$\approx \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ i_1 & i_2 & i_3 & \cdots & i_{n-t} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (i_1)^{n-t-1} & (i_2)^{n-t-1} & (i_3)^{n-t-1} & \cdots & (i_{n-t})^{n-t-1} \end{pmatrix} \cdot \begin{pmatrix} S_{2t}^{(i_1)} \\ S_{2t}^{(i_2)} \\ \vdots \\ S_{2t}^{(i_{n-t})} \end{pmatrix} + \begin{pmatrix} B^{(1)} \\ B^{(2)} \\ \vdots \\ B^{(n-t)} \end{pmatrix}.$$

We shall write it succinctly as

$$V \cdot \begin{pmatrix} S_{2t}^{(i_1)} \\ S_{2t}^{(i_2)} \\ \vdots \\ S_{2t}^{(i_{n-t})} \end{pmatrix} + \begin{pmatrix} A^{(1)} \\ A^{(2)} \\ \vdots \\ A^{(n-t)} \end{pmatrix} \approx V \cdot \begin{pmatrix} S_{2t}^{(i_1)} \\ S_{2t}^{(i_2)} \\ \vdots \\ S_{2t}^{(i_{n-t})} \end{pmatrix} + \begin{pmatrix} B^{(1)} \\ B^{(2)} \\ \vdots \\ B^{(n-t)} \end{pmatrix}.$$

To prove this, we first show a useful claim, which essentially is a restatement of our high-dimensional smudging lemma (Lemma 1), albeit conditioned on the adversary's shares.

Claim 2. *Suppose* $(\Delta^{(1)}, \ldots, \Delta^{(n-t)})$ *is a fixed vector satisfying that every entry is divisible by* $\det(V) \cdot \prod_{i \in I} p_i$. *It holds that*

$$V \cdot \begin{pmatrix} S_{2t}^{(i_1)} \\ S_{2t}^{(i_2)} \\ \vdots \\ S_{2t}^{(i_{n-t})} \end{pmatrix} + \begin{pmatrix} \Delta^{(1)} \\ \Delta^{(2)} \\ \vdots \\ \Delta^{(n-t)} \end{pmatrix} \approx_\varepsilon V \cdot \begin{pmatrix} S_{2t}^{(i_1)} \\ S_{2t}^{(i_2)} \\ \vdots \\ S_{2t}^{(i_{n-t})} \end{pmatrix},$$

where $\varepsilon \leqslant \frac{\mathsf{max}_\Delta \cdot n^{2n}}{L_{2t}}$. *Here, we use* max_Δ *to denote the maximum* $\Delta^{(i)}$.

Before we proceed to the proof, we remind the reader that these distributions are all conditioned on the corrupted parties' secret shares. Consequently, their distributions are in the form described by Remark 3.

Proof of Claim 2. Let $\mathsf{adj}(V)$ be the adjugate matrix of V. Consider the following derivation.

$$\begin{aligned} V \cdot \begin{pmatrix} S_{2t}^{(i_1)} \\ S_{2t}^{(i_2)} \\ \vdots \\ S_{2t}^{(i_{n-t})} \end{pmatrix} &= V \cdot \begin{pmatrix} S_{2t}^{(1)} \\ S_{2t}^{(2)} \\ \vdots \\ S_{2t}^{(n-t)} \end{pmatrix} - \begin{pmatrix} \Delta^{(1)} \\ \Delta^{(2)} \\ \vdots \\ \Delta^{(n-t)} \end{pmatrix} + \begin{pmatrix} \Delta^{(1)} \\ \Delta^{(2)} \\ \vdots \\ \Delta^{(n-t)} \end{pmatrix} \\ &= V \cdot \left(\begin{pmatrix} S_{2t}^{(1)} \\ S_{2t}^{(2)} \\ \vdots \\ S_{2t}^{(n-t)} \end{pmatrix} - \mathsf{adj}(V) \cdot \begin{pmatrix} \Delta^{(1)}/\det(V) \\ \Delta^{(2)}/\det(V) \\ \vdots \\ \Delta^{(n-t)}/\det(V) \end{pmatrix} \right) + \begin{pmatrix} \Delta^{(1)} \\ \Delta^{(2)} \\ \vdots \\ \Delta^{(n-t)} \end{pmatrix} \\ &\approx V \cdot \begin{pmatrix} S_{2t}^{(1)} \\ S_{2t}^{(2)} \\ \vdots \\ S_{2t}^{(n-t)} \end{pmatrix} + \begin{pmatrix} \Delta^{(1)} \\ \Delta^{(2)} \\ \vdots \\ \Delta^{(n-t)} \end{pmatrix} \end{aligned}$$

Evidently, the closeness we need to prove depends on the closeness of

$$\begin{pmatrix} S_{2t}^{(1)} \\ S_{2t}^{(2)} \\ \vdots \\ S_{2t}^{(n-t)} \end{pmatrix} \approx \begin{pmatrix} S_{2t}^{(1)} \\ S_{2t}^{(2)} \\ \vdots \\ S_{2t}^{(n-t)} \end{pmatrix} - \mathsf{adj}(V) \cdot \begin{pmatrix} \Delta^{(1)}/\det(V) \\ \Delta^{(2)}/\det(V) \\ \vdots \\ \Delta^{(n-t)}/\det(V) \end{pmatrix}.$$

First, note that we crucially require all $\Delta^{(i)}$ to be divisible by $\det(V)$. Otherwise, the RHS will only be supported on non-whole numbers, while the LHS is only supported on whole numbers; thus, these two distributions cannot be close. Next, $\Delta^{(i)}$ needs to be divisible by $\prod_{i\in I} p_i$ because, conditioned on the adversary's view, $S_{2t}^{(i)}$ are uniformly distributed over consecutive integers, all of which fall into the same class modulo $\prod_{i\in I} p_i$. That is,

$$s_{2t}^{(i)},\ s_{2t}^{(i)} + \prod_{i\in I} p_i,\ s_{2t}^{(i)} + 2\cdot\prod_{i\in I} p_i,\ \ldots,\ s_{2t}^{(i)} + \frac{L_{2t}}{\prod_{i\in I} p_i}\cdot\prod_{i\in I} p_i$$

for some $0 \leqslant s_{2t}^{(i)} < \prod_{i\in I} p_i$. Therefore, by Smudging Lemma (Lemma 2), the closeness of these two distributions is bounded by[24]

$$\left\| \mathsf{adj}(V) \cdot \begin{pmatrix} \Delta^{(1)}/\det(V) \\ \Delta^{(2)}/\det(V) \\ \vdots \\ \Delta^{(n-t)}/\det(V) \end{pmatrix} \right\|_\infty \cdot \frac{1}{L_{2t}} \cdot n \leqslant \frac{\mathsf{max}_\Delta \cdot n^{n^2}}{L_{2t}},$$

since every entry of $\mathsf{adj}(V)$ is upper bounded by $n^{(n-t)^2}$ (refer to Section 3.1). This completes the proof. □

Now, we remark that Claim 2 is stated for a deterministic shift vector $(\Delta^{(1)}, \ldots, \Delta^{(n-t)})$. Clearly, similar results hold when $(\Delta^{(1)}, \ldots, \Delta^{(n-t)})$ is a distribution over a universe of shift vectors, all satisfying the premise of Claim 2.

Now, in order to use Claim 2, we need to show that the difference between the real distribution $(A^{(1)}, A^{(2)}, \ldots, A^{(n-t)})$ and the simulated distribution $(B^{(1)}, B^{(2)}, \ldots, B^{(n-t)})$ satisfies the prerequisite of Claim 2. In particular, we also need to upper-bound how far apart they are, i.e., max_Δ. For this, we have the following claim.

Claim 3 (Security Invariant)**.** *For any wire w, the distribution of the CRT sharing integer W is the simulated view is identically distributed by shifting the distribution of the CRT sharing integer W in the real view by Δ, where Δ is divisible by*

$$\left(\prod_{1\leqslant i<j\leqslant n} (j-i)\right) \cdot \left(\prod_{i\in I} p_i\right)$$

and it holds that

$$\Delta \leqslant |C| \cdot p_0 \cdot \left(\prod_{1\leqslant i<j\leqslant n} (j-i)\right) \cdot \left(\prod_{i\in I} p_i\right).$$

[24] Here, $\|\vec{x}\|_\infty$ denotes the ℓ_∞ norm of a vector, which is the maximum entry. The factor n is due to the union bound on each row.

Proof of Claim 3. For both the input sharing and the "integer-reduction" protocol output, the distribution of the secret integer is always a fixed distribution (i.e., independent of the wire value) shifted by an integer m defined as (1) $m \mod p_0$ equals to the wire value and $m = 0 \mod \prod_{1 \leqslant i < j \leqslant n}(j-i)$. Therefore, the distribution of the real view and the simulated view will always be exactly shifted by some integer Δ, which satisfies (1) $\Delta \mod p_0$ is the difference of the wire values in the two views and (2) it is a multiple of $\prod_{1 \leqslant i < j \leqslant n}(j-i)$. If we further condition on the corrupted parties' shares, it gives that Δ is divisible by $\left(\prod_{1 \leqslant i < j \leqslant n}(j-i)\right) \cdot \left(\prod_{i \in I} p_i\right)$ and is upper-bounded by $p_0 \cdot \left(\prod_{1 \leqslant i < j \leqslant n}(j-i)\right) \cdot \left(\prod_{i \in I} p_i\right)$. Now, for any other wires in the circuit, which may be the summation of $|C|$ wires (either input wire or output wire of a multiplication gate), the upper bound would change to $|C| \cdot p_0 \cdot \left(\prod_{1 \leqslant i < j \leqslant n}(j-i)\right) \cdot \left(\prod_{i \in I} p_i\right)$. □

Going back to Equation 2, the shift between $A^{(i)}$ and $B^{(i)}$ will be, thus, upper bounded by[25]

$$\left(|C| \cdot p_0 \cdot \left(\prod_{1 \leqslant i < j \leqslant n}(j-i)\right) \cdot \left(\prod_{i \in I} p_i\right)\right) \cdot \left(|C| \cdot (n^n \cdot L_t \cdot p_0 + n^{n^2} \cdot p_0)\right),$$

which can be simplified as:

$$\mathfrak{C} = |C|^2 \cdot p_0^2 \cdot n^{2n^2} \cdot P_{\mathsf{max}} \cdot L_t.$$

Together with Claim 2, this imply that the closeness in Equation 2 is bounded by $n^{n^2} \cdot \mathfrak{C}$. Now, Equation 2 is just for one batch $(n-t)$ of gates. For the entire circuit, we would need to use a union bound of $|C|$.[26] Therefore, the simulation error in the circuit emulation part is overall upper-bounded by $n^{n^2} \cdot |C| \cdot \mathfrak{C}$., which is equal to:

$$\frac{|C|^3 \cdot p_0^2 \cdot n^{3n^2} \cdot P_{\mathsf{max}} \cdot L_t}{L_{2t}}.$$

Finally, we argue that the output reconstruction step is statistically close between the real and simulated view. We divide this into two cases.

- Case 1: If the output of the protocol only depends on the corrupted parties' inputs, then the sharing of the output wire is identical between the real and simulated view. In particular, it will be consistent with the circuit output $C(\vec{x})$. Hence, there is no simulation error in this case.
- Case 2: If the output of the protocol depends on the honest parties' inputs. Then, the CRT sharing integer of the output wire will depend on the randomness from the honest party. In particular, the distribution of this integer between the real and simulated view will be shifted by some Δ' (Claim 3). Now, once the simulator computes the shift Δ to ensure the correct output $\mod p_0$. The closeness between the real and simulated view will again follow from a variant of Claim 2, i.e., instead of considering $S_{2t}^{(i)}$, we are concerned with $S_t^{(i)}$ in this case. And, by a similar analysis, the closeness here would be bounded by

$$\frac{\mathsf{max}_\Delta \cdot n^{n^2}}{L_t} \leqslant \frac{|C| \cdot p_0 \cdot n^{2n^2} \cdot P_{\mathsf{max}}}{L_t}.$$

This completes the security proof and, hence, the proof of Theorem 3.

[25] The worst-case is when this is a multiplication gate $X \cdot Y$ and the shift for X and Y individually are amplified by the maximum possible value of X and Y, which is already derived in the correctness analysis.

[26] Note that the analysis of different batches of gates relies on the randomness coming from different batches of the random masks. Therefore, they are independent of each other.

5 Scalable Dishonest Majority MPC

In this section, we briefly discuss how to extend our protocol in Section 4 to dishonest majority setting. Similar to all prior works, a correlated setup is required to enable an information-theoretic online phase. We assume there are $\leqslant (1-\delta)\cdot n$ corruptions for some constant $\delta \in (0, 1/2)$.

An description of the protocol is included in Figure 4. We obtain the following theorem about our dishonest majority protocol.

Theorem 6 (Dishonest Majority Protocol)**.** *Consider any number of parties n, corruption threshold $t = (1-\delta)\cdot n$ for some constant δ and ring size p_0 with $\log p_0 = \Omega\big(n^2 \log n\big)$ that is coprime with $\{2, 3, \ldots, n\}$. Assuming the trusted correlated setup, the protocol in Figure 4 realizes any arithmetic circuit C over $\mathbb{Z}_{p_0}$ with the following guarantees.*

- *The overall communication is $\mathcal{O}(|C| \cdot \log p_0)$ bits.*
- *The overall computation is $\mathcal{O}\big(|C| \cdot \log p_0 \cdot \log^2 n\big)$ bit operation.*
- *This protocol is perfectly correct.*
- *This protocol is $\mathsf{poly}(|C|/p_0)$-secure, which is exponentially small for exponentially large fields.*

Overview of the Protocol and Proof Sketch. Similar to prior works, the dishonest majority protocol proceeds by first secret sharing the input and then emulating the circuit evaluation gate by gate. Unlike the honest majority setting, where parties can multiply two secret shares to obtain a secret sharing of the product of the secrets, this is infeasible in the dishonest majority setting. Instead, parties rely on the Beaver's triple to reduce the multiplication gate into a linear function. In particular, suppose parties need to compute $[\![X]\!] \cdot [\![Y]\!]$ and they hold sharings of the Beaver's triple $[\![A]\!], [\![B]\!], [\![C]\!]$, where $c = ab$. Parties first reconstruct $U = X + A$ and $V = Y + B$ in the clear. Next, one uses the standard technique to write

$$x \cdot y = -u \cdot v + u \cdot y + v \cdot x + c.$$

Therefore, parties may obtain the secret sharings of $x \cdot y$ by computing

$$(p_0 - 1) \cdot u \cdot v + u \cdot [\![Y]\!] + v \cdot [\![X]\!] + [\![C]\!].$$

Note that we replace (-1) with $p_0 - 1$ to ensure that the integer associated with the secret is always positive.

For prior works based on linear secret sharing scheme, this already gives a feasible solution. For CRT-based secret sharing scheme, however, scalar multiplication is not "free". It also makes the integer grow bigger. Therefore, we still need to perform an "integer-reduction" protocol to reduce the integer back to an acceptable size.

The integer growing for scalar multiplication is not as devastating as multiplying two secrets because of the following reason. For the dishonest majority, each sharing $[\![X]\!]$ of a wire x will roughly have length $\log X \approx \frac{t}{n} \cdot \log P_{\mathsf{all}} > \frac{1}{2} \cdot \log P_{\mathsf{all}}$. Therefore, if one multiplies $[\![X]\!]$ and $[\![Y]\!]$ directly, it will be $\log(XY) > \log P_{\mathsf{all}}$, and correctness will not hold. Instead, when one performs scalar multiplication, the integer will only grow by $\log p_0$ and becomes $\frac{t}{n} \cdot \log P_{\mathsf{all}} + \log p_0$. This is still lower than $\log P_{\mathsf{all}}$ as long as $\log p_0 < \frac{n-t}{n} \cdot \log P_{\mathsf{all}}$. Based on this intuition, we are calling the masks generated $\{[\![R_t]\!], [\![R_{t+\log p_0}]\!]\}$ instead of $\{[\![R_t]\!], [\![R_{2t}]\!]\}$.

Correlated Randomness Setup. We assume parties have holds $|C|$ many CRT sharing of "Beaver's Triple", i.e., $[\![A]\!]$, $[\![B]\!]$, $[\![C]\!]$ such that $a, b \leftarrow \mathbb{Z}_{p_0}$, and $c = a \cdot b$ over $\mathbb{Z}_{p_0}$.

Offline Phase. This is identical to the offline phase of the honest majority protocol (Figure 1). Unlike honest majority protocol, the pair of masks generated are $[\![R_t]\!]$ and $[\![R_{t+\log p_0}]\!]$ (instead of $[\![R_{2t}]\!]$). Note that, these masks are needed to mask sharing after a scalar multiplication. (In particular, they are not intended to mask the product of two sharing as in the honest majority protocol).

Online Phase. The online phase proceeds as follows.

- **Input Sharing.** Each party P_i, holding input $x^{(i)} \in \mathbb{Z}_{p_0}$, compute $\bar{x}^{(i)} \leftarrow \mathsf{ReFormat}(x^{(i)})$ and $\left[\!\left[\bar{X}_t^{(i)}\right]\!\right] \leftarrow \mathsf{Share}_{n,t}(\bar{x}^{(i)})$. It sends $\left[\!\left[\bar{X}_t^{(i)}\right]\!\right]_j$ to all party P_j.
- **Circuit Emulation.** Parties emulate the circuit evaluation gate by gate as follows.
  - *Addition* $x + y \in \mathbb{Z}_{p_0}$. Let $[\![X_t]\!]$ and $[\![Y_t]\!]$ denote the (t, n)-sharing of the input wires x and y. Each party P_i simply sets $[\![X_t]\!]_i \oplus_i [\![Y_t]\!]_i$ as its (t, n)-sharing of the output wire.
  - *Multiplication* $x \cdot y \in \mathbb{Z}_{p_0}$. Let $[\![X]\!]$ and $[\![Y]\!]$ denote the secret sharing of the input wires x and y. Let $[\![A]\!], [\![B]\!], [\![C]\!]$ be the next Beaver's triple that is not consumed yet. Let $\{[\![R_t]\!], [\![R_{t+\log p_0}]\!]\}$ be the next pair of random masks that are not consumed yet.
    - Each party sends to the king party
    $$u_i = [\![X]\!]_i \oplus_i [\![A]\!]_i \qquad \text{and} \qquad v_i = [\![Y]\!]_i \oplus_i [\![B]\!]_i .$$
    - After receiving $\{u_1, u_2, \ldots, u_n\}$ and $\{v_1, v_2, \ldots, v_n\}$, the king party reconstructs u and v by CRT. The king party sends u and v to all non-king parties.
    - Party locally computes the following modulo p_i.
    $$m = (p_0 - 1) \cdot u \cdot v + u \cdot [\![Y]\!] + v \cdot [\![X]\!] + [\![C]\!] + [\![R_{t+\log p_0}]\!] .$$
    They send their secret share of m to the king party, who will reconstruct m, reformat it $\bar{m} = \mathsf{ReFormat}(m)$, and send it back to the parties.
    - Parties set $\bar{m} \oplus_i [\![R_t]\!]_i$ as its sharing for the output wire.
- **Output Reconstruction.** Let $[\![W_t]\!]$ denote the secret sharing of the output wire w. Each party P_i broadcast its share $w_i := [\![W]\!]_i$. After receiving $\{w_1, \ldots, w_n\}$, parties reconstruct $w \leftarrow \mathsf{Reconst}_{n,t}(\{w_i\}_{i \in [n]})$.

Figure 4: Description of the dishonest majority protocol.

Therefore, in order to obtain the desired efficiency, we shall set

$$\log P_{\mathsf{all}} = \Omega\left(\frac{n}{n-t} \cdot \log p_0\right)$$
$$\log p_i = \Omega\left(\frac{\log p_0}{n-t}\right)$$

The download rate of the secret sharing scheme will be $\mathcal{O}(1)$ as long as $t < (1-\delta) \cdot n$ for a constant δ. All the communication and computation done in the protocol will be linear in $|C| \cdot \log P_{\mathsf{all}}$, which is $\mathcal{O}(|C| \cdot \log p_0)$ as desired.

The protocol will need to set the parameter L in the CRT secret sharings of $[\![A]\!], [\![B]\!], [\![C]\!]$ and the sharings of $\{[\![R_t]\!], [\![R_{t+\log p_0}]\!]\}$ appropriately. Since setting these parameters and the proof of the theorem is entirely analogous to the honest majority setting, we omit the details here.

References

[AB83] Charles Asmuth and John Bloom. A modular approach to key safeguarding. *IEEE Trans. Inf. Theory*, 29(2):208–210, 1983. 4, 7, 14

[ACD+19] Mark Abspoel, Ronald Cramer, Ivan Damgård, Daniel Escudero, and Chen Yuan. Efficient information-theoretic secure multiparty computation over $\mathbb{Z}/p^k\mathbb{Z}$ via galois rings. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019: 17th Theory of Cryptography Conference, Part I*, volume 11891 of *Lecture Notes in Computer Science*, pages 471–501, Nuremberg, Germany, December 1–5, 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-36030-619. 4

[AIK11] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. How to garble arithmetic circuits. In Rafail Ostrovsky, editor, *52nd Annual Symposium on Foundations of Computer Science*, pages 120–129, Palm Springs, CA, USA, October 22–25, 2011. IEEE Computer Society Press. doi:10.1109/FOCS.2011.40. 5, 6

[AJL+12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 483–501, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-29011-429. 6, 13

[AL17] Gilad Asharov and Yehuda Lindell. A full proof of the BGW protocol for perfectly secure multiparty computation. *Journal of Cryptology*, 30(1):58–151, January 2017. doi:10.1007/s00145-015-9214-4. 15

[Bea92] Donald Beaver. Efficient multiparty protocols using circuit randomization. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 420–432, Santa Barbara, CA, USA, August 11–15, 1992. Springer, Heidelberg, Germany. doi:10.1007/3-540-46766-134. 4, 12

[Ber04] Daniel J Bernstein. Scaled remainder trees. 2004. URL: https://cr.yp.to/arith/scaledmod-20040820.pdf. 15

[BGJK21] Gabrielle Beck, Aarushi Goel, Abhishek Jain, and Gabriel Kaptchuk. Order-C secure multiparty computation for highly repetitive circuits. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part II*, volume 12697 of *Lecture Notes in Computer Science*, pages 663–693, Zagreb, Croatia, October 17–21, 2021. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-77886-623. 3, 5, 6

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 1–10, Chicago, IL, USA, May 2–4, 1988. ACM Press. doi:10.1145/62212.62213. 3, 5, 17

[BLLL23] Marshall Ball, Hanjun Li, Huijia Lin, and Tianren Liu. New ways to garble arithmetic circuits. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part II*, volume 14005 of *Lecture Notes in Computer Science*, pages 3–34, Lyon, France, April 23–27, 2023. Springer, Heidelberg, Germany. doi:10.1007/978-3-031-30617-41. 5

[BM74] Allan Borodin and Robert Moenck. Fast modular transforms. *Journal of Computer and System Sciences*, 8(3):366–386, 1974. 15

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 11–19, Chicago, IL, USA, May 2–4, 1988. ACM Press. doi:10.1145/62212.62214. 3, 5

[CDSGV91] Renato M Capocelli, Alfredo De Santis, Luisa Gargano, and Ugo Vaccaro. On the size of shares for secret sharing schemes. In *Annual International Cryptology Conference*, pages 101–113. Springer, 1991. 7

[CFIK03] Ronald Cramer, Serge Fehr, Yuval Ishai, and Eyal Kushilevitz. Efficient multi-party computation over rings. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 596–613, Warsaw, Poland, May 4–8, 2003. Springer, Heidelberg, Germany. doi:10.1007/3-540-39200-937. 4

[CLMZ23] Alessandro Chiesa, Ryan Lehmkuhl, Pratyush Mishra, and Yinuo Zhang. Eos: Efficient private delegation of zkSNARK provers. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 6453–6469, Anaheim, CA, August 2023. USENIX Association. URL: https://www.usenix.org/conference/usenixsecurity23/presentation/chiesa. 4

[DEN22] Anders P. K. Dalskov, Daniel Escudero, and Ariel Nof. Fast fully secure multi-party computation over any ring with two-thirds honest majority. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022: 29th Conference on Computer and Communications Security*, pages 653–666, Los Angeles, CA, USA, November 7–11, 2022. ACM Press. doi:10.1145/3548606.3559389. 4

[DIK+08] Ivan Damgård, Yuval Ishai, Mikkel Krøigaard, Jesper Buus Nielsen, and Adam Smith. Scalable multiparty computation with nearly optimal work and resilience. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 241–261, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Heidelberg, Germany. doi:10.1007/978-3-540-85174-514. 5

[DIK10] Ivan Damgård, Yuval Ishai, and Mikkel Krøigaard. Perfectly secure multiparty computation and the computational overhead of cryptography. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 445–465, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-13190-523. 3, 5, 6

[DLN19] Ivan Damgård, Kasper Green Larsen, and Jesper Buus Nielsen. Communication lower bounds for statistically secure MPC, with or without preprocessing. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part II*, volume 11693 of *Lecture Notes in Computer Science*, pages 61–84, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-26951-73. 3, 5, 8

[DN07] Ivan Damgård and Jesper Buus Nielsen. Scalable and unconditionally secure multiparty computation. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 572–590, Santa Barbara, CA, USA, August 19–23, 2007. Springer, Heidelberg, Germany. doi:10.1007/978-3-540-74143-532. 3, 5, 6, 8, 9, 17, 19

[ES21] Daniel Escudero and Eduardo Soria-Vazquez. Efficient information-theoretic multi-party computation over non-commutative rings. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part II*, volume 12826 of *Lecture Notes in Computer Science*, pages 335–364, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-84245-112. 4

[EXY22] Daniel Escudero, Chaoping Xing, and Chen Yuan. More efficient dishonest majority secure computation over $\mathbb{Z}_{2^k}$ via galois rings. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part I*, volume 13507 of *Lecture Notes in Computer Science*, pages 383–412, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Heidelberg, Germany. doi:10.1007/978-3-031-15802-514. 4

[Für07] Martin Fürer. Faster integer multiplication. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 57–66, 2007. 15

[FY92] Matthew K. Franklin and Moti Yung. Communication complexity of secure computation (extended abstract). In *24th Annual ACM Symposium on Theory of Computing*, pages 699–710, Victoria, BC, Canada, May 4–6, 1992. ACM Press. doi:10.1145/129712.129780. 3, 5, 6

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 169–178, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press. doi:10.1145/1536414.1536440. 6

[GGJ+23] Sanjam Garg, Aarushi Goel, Abhishek Jain, Guru-Vamsi Policharla, and Sruthi Sekar. zkSaaS: Zero-Knowledge SNARKs as a service. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 4427–4444, Anaheim, CA, August 2023. USENIX Association. URL: https://www.usenix.org/conference/usenixsecurity23/presentation/garg. 4, 5

[GIP15] Daniel Genkin, Yuval Ishai, and Antigoni Polychroniadou. Efficient multi-party computation: From passive to active security via secure SIMD circuits. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 721–741, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-48000-735. 5

[GJM+23] Sanjam Garg, Abhishek Jain, Pratyay Mukherjee, Rohit Sinha, Mingyuan Wang, and Yinuo Zhang. Cryptography with weights: MPC, encryption and signatures. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part I*, volume 14081 of *Lecture Notes in Computer Science*, pages 295–327, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Heidelberg, Germany. doi:10.1007/978-3-031-38557-510. 3, 6, 7, 8, 14

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 218–229, New York City, NY, USA, May 25–27, 1987. ACM Press. doi:10.1145/28395.28420. 3

[GPS21] Vipul Goyal, Antigoni Polychroniadou, and Yifan Song. Unconditional communication-efficient MPC via hall's marriage theorem. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part II*, volume 12826 of *Lecture Notes in Computer Science*, pages 275–304, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-84245-110. 3, 5, 6

[GPS22] Vipul Goyal, Antigoni Polychroniadou, and Yifan Song. Sharing transformation and dishonest majority MPC with packed secret sharing. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part IV*, volume 13510 of *Lecture Notes in Computer Science*, pages 3–32, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Heidelberg, Germany. doi:10.1007/978-3-031-15985-51. 3, 5, 6, 12

[GRS99] Oded Goldreich, Dana Ron, and Madhu Sudan. Chinese remaindering with errors. In *31st Annual ACM Symposium on Theory of Computing*, pages 225–234, Atlanta, GA, USA, May 1–4, 1999. ACM Press. doi:10.1145/301250.301309. 4, 7, 14

[Hea24] David Heath. Efficient Arithmetic in Garbled Circuits. Cryptology ePrint Archive, Paper 2024/139, 2024. https://eprint.iacr.org/2024/139. URL: https://eprint.iacr.org/2024/139. 6

[HTX20] Mike Hamburg, Mike Tunstall, and Qinglai Xiao. Improvements to RSA key generation and CRT on embedded devices. Cryptology ePrint Archive, Report 2020/1507, 2020. https://eprint.iacr.org/2020/1507. 6

[HVDH21] David Harvey and Joris Van Der Hoeven. Integer multiplication in time o(nlog\,n). *Annals of Mathematics*, 193(2):563–617, 2021. 15

[HVDHL16] David Harvey, Joris Van Der Hoeven, and Grégoire Lecerf. Even faster integer multiplication. *Journal of Complexity*, 36:1–30, 2016. 15

[ICG08] Sorin Iftene, Stefan Ciobaca, and Manuela Grindei. Compartmented threshold RSA based on the Chinese remainder theorem. Cryptology ePrint Archive, Report 2008/370, 2008. https://eprint.iacr.org/2008/370. 6

[LMW22] Wei-Kai Lin, Ethan Mook, and Daniel Wichs. Doubly efficient private information retrieval and fully homomorphic RAM computation from ring LWE. Cryptology ePrint Archive, Report 2022/1703, 2022. https://eprint.iacr.org/2022/1703. 6

[Mig83] Maurice Mignotte. How to share a secret? In Thomas Beth, editor, *Advances in Cryptology – EUROCRYPT'82*, volume 149 of *Lecture Notes in Computer Science*, pages 371–375, Burg Feuerstein, Germany, March 29 – April 2, 1983. Springer, Heidelberg, Germany. doi:10.1007/3-540-39466-427. 4, 7, 14

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 735–763, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-49896-526. 6

[OB22] Alex Ozdemir and Dan Boneh. Experimenting with collaborative zk-SNARKs: Zero-knowledge proofs for distributed secrets. In Kevin R. B. Butler and Kurt Thomas, editors, *USENIX Security 2022: 31st USENIX Security Symposium*, pages 4291–4308, Boston, MA, USA, August 10–12, 2022. USENIX Association. 4

[Rud23] Atri Rudra. (dense structured) matrix vector multiplication, 2023. URL: https://cse.buffalo.edu/faculty/atri/courses/matrix/matrix-vect-notes.pdf. 15

[SS71] V Strassen and A Schönhage. Schnelle multiplikation großer zahlen. *Computing*, 7(3/4):281–292, 1971. 15

[vzGS91] Joachim von zur Gathen and Gadiel Seroussi. Boolean circuits versus arithmetic circuits. *Inf. Comput.*, 91(1):142–154, 1991. 5

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science*, pages 162–167, Toronto, Ontario, Canada, October 27–29, 1986. IEEE Computer Society Press. doi:10.1109/SFCS.1986.25. 3, 5