CycloSPDZ: Efficient MPC over 2-Power Cyclotomic Rings mod 2^k for Dishonest Majority

Jianguo Xie[1 [0000−0001−7113−8064]], Yutao Sun[2 [0000−0002−0064−9411]], Jingqiang Lin[1 [0000−0003−2639−3722]], Nenghai Yu[1 [0000−0003−4417−9316]], and Guozhen Shi[2 [0009−0000−6155−8161]]

[1] University of Science and Technology of China, Hefei 230027, China
[2] Beijing Electronic Science and Technology Institute, Beijing 100070, China
jgxie@mail.ustc.edu.cn, 20242010@mail.besti.edu.cn, linjq@ustc.edu.cn, ynh@ustc.edu.cn, sgz@besti.edu.cn

Abstract. Most multi-party computation (MPC) protocols support arithmetic over finite fields or integer rings. Polynomial arithmetic in applications such as convolution evaluation and distributed key generation motivates extending these protocols to the 2-power cyclotomic rings $R_k = \mathbb{Z}_{2^k}[X]/(X^N + 1)$, where N is a power of two. However, no native protocol is known for actively secure computation over these rings with a dishonest majority.

In this work, we present an information-theoretic MAC that is homomorphic over R_k and requires only a small increase in coefficient bit length. We apply this to construct CycloSPDZ, an actively secure MPC protocol for a dishonest majority in the preprocessing model that has efficiency comparable to the state-of-the-art field-based convolution protocols in Convolutions in Overdrive (PoPETs 2023). We also construct an efficient preprocessing protocol from authenticated integer-ring triples by adapting Toom–4 multiplication to power-of-two moduli via scaled interpolation and batched exact division, and combining it with Karatsuba multiplication in a hybrid scheme. As an application, we evaluate convolution workloads using polynomial encoding and blocking. We implement and benchmark the online phase of CycloSPDZ against our hybrid variant of SPD$\mathbb{Z}_{2^k}$ (CRYPTO 2018) and the function-dependent variant of the Matrix Rings (ASIACRYPT 2024), achieving speedups of up to 5× and 48×, respectively.

Keywords: Secure multiparty computation · Dishonest majority · 2-power cyclotomic rings.

1 Introduction

Secure multi-party computation (MPC) enables a set of p mutually distrusting parties $P_1, \ldots, P_p$, each holding a private input x_i, to jointly compute a function $f(x_1, \ldots, x_p)$ while guaranteeing input privacy and output correctness. Even when an active adversary controls a subset of parties and causes them to deviate arbitrarily from the protocol, the adversary can neither learn anything beyond the output nor compromise the correctness of the computation. This requirement is formalized via the real–ideal paradigm, which demands that a real protocol execution be indistinguishable from an ideal execution in which a trusted third party collects $(x_1, \ldots, x_p)$ and returns $f(x_1, \ldots, x_p)$ to the parties.

The universally composable (UC) framework [1] provides a unified foundation for such simulation-based definitions and guarantees security under arbitrary protocol composition.

Many concretely efficient arithmetic MPC protocols with active security, such as SPDZ, MASCOT, and Overdrive [2,3,4], are designed over finite fields. These protocols follow the offline–online paradigm, decoupling input-independent correlated randomness generation from input-dependent online evaluation, and rely on authenticated secret sharing to detect adversarial tampering [5,6]. Specifically, the parties hold additive shares of each secret value together with an information-theoretic message authentication code (MAC), where a secret-shared global key enforces a linear authentication relation between them. Preprocessing generates multiplication triples and other correlated randomness, allowing the parties to perform secret multiplications online by opening masked differences [7]. This modular design builds the online phase on a unified preprocessing interface, which can be instantiated using oblivious transfer (OT), homomorphic encryption (HE), or pseudorandom correlation generators (PCG) [3,4,8].

Subsequent work in the SPDZ paradigm has broadened authenticated arithmetic along two dimensions, the underlying algebraic domain and the granularity of computation. Regarding the algebraic domain, SPD$\mathbb{Z}_{2^k}$ generalizes actively secure dishonest-majority computation from finite fields to the integer ring $\mathbb{Z}_{2^k}$, aligning cryptographic arithmetic with native machine-word operations [9,10]. Damgård et al. subsequently implement this framework and construct dedicated ring-based primitives for equality testing, comparison, and truncation [11]. Regarding computational granularity, Chen et al. and MatriGear design specialized preprocessing protocols to generate authenticated matrix multiplication triples [12,13], whereas Matrix Rings introduces vector MACs to directly authenticate matrix-valued secrets [14]. Collectively, these advances extend preprocessing and authentication from individual scalars to higher-dimensional algebraic objects, enabling protocols to exploit the intrinsic algebraic structure of composite operations.

Like matrix operations, polynomial multiplication exhibits algebraic structure that can be exploited as a whole. When a polynomial product is unrolled into a scalar arithmetic, however, each constituent secret scalar multiplication consumes an independent preprocessing triple and produces masked values that must be opened and authenticated, incurring substantial communication and verification overhead. This motivates us to construct native MACs and multiplication triples directly for polynomial elements, enabling a single ring multiplication to process an entire vector of coefficients at once. Specifically, we study 2-power cyclotomic rings $R_k = \mathbb{Z}_{2^k}[X]/(X^N + 1)$ modulo 2^k, where the ring dimension N is a power of two. Each element of R_k is represented by N coefficients modulo 2^k, and ring multiplication corresponds to polynomial multiplication followed by reduction under the negacyclic relation $X^N = -1$. This algebraic structure preserves machine-word arithmetic modulo 2^k on the coefficients while treating fixed-dimensional polynomials as the atomic units of online computation and authentication.

Achieving this goal first requires an authentication mechanism tailored to the ring. Over a finite field, every nonzero error is invertible, so a MAC forgery equation uniquely determines the secret key. In R_k, a nonzero error may be a zero divisor, allowing a single

forgery equation to hold for many possible keys. For example, with a uniformly random key $\alpha \in R_k$, the nonzero error $\delta = 2^{k-1}(1 + X + \cdots + X^{N-1})$ satisfies $\alpha\delta = 0$ with probability $1/2$. Moreover, during batch MAC checks, linear combinations with random challenge polynomials may shift the aggregated error into a deeper level of the ideal chain. We also require an efficient algorithm to generate the authenticated multiplication triples over R_k needed by the online protocol.

1.1 Our Contributions

MAC and Actively Secure Computation for R_k. We construct an information-theoretic MAC over the extended cyclotomic ring $R = \mathbb{Z}_{2^{k+s}}[X]/(X^N + 1)$, where the global key is sampled uniformly from R and batch verification employs short polynomial challenges with s-bit coefficients. For forgeries that modify a payload value, we prove a tight success-probability upper bound of $(Ns + 4)2^{-(Ns+2)}$. With fresh zero-sum masks in each check and a finite batch limit fixed before the session begins, the same bound holds under key reuse when the session aborts at the first failed or incomplete check. The resulting SPDZ-style online protocol achieves statistical UC security with abort against a static malicious adversary corrupting up to $p - 1$ of the p parties.

Preprocessing from Authenticated Integer-Ring Triples. We design a protocol that UC-realizes the preprocessing functionality for synthesizing authenticated cyclotomic-ring triples and masks from authenticated triples over integer rings modulo powers of two. Since each cyclotomic-ring triple is generated via secret polynomial products, where every coefficient multiplication uses an integer-ring triple, we apply a hybrid recursion of ℓ-level Toom–4 and Karatsuba to reduce the number of such multiplications to $7^\ell 3^{n-2\ell}$ for $N = 2^n$. To overcome the non-invertibility of even denominators in Toom–4 interpolation, we introduce scaled interpolation, which multiplies the inverse interpolation matrix by eight to leave only invertible odd denominators. When $\ell > 0$, one batched exact division removes the accumulated scaling factor after negacyclic reduction, accounting for carries and preserving the MAC relations. Under the Multipars backend cost model [15], Karatsuba requires 1.6–3.5 times as much estimated amortized communication per authenticated ring triple as our hybrid strategy across the parameters in Table 2.

Implementing and Benchmarking. We apply our cyclotomic-ring arithmetic to 1D long-kernel and 2D depthwise convolutions by encoding input blocks and kernels as ring elements and extracting output coefficients unaffected by negacyclic wraparound. We implement our protocol using the communication components of MP-SPDZ [16], and benchmark under emulated LAN and WAN settings. In the WAN setting, CycloSPDZ achieves online speedups of up to 5.5× over a SPD$\mathbb{Z}_{2^k}$ Toom–4/Karatsuba hybrid baseline and 48.7× over the function-dependent variant of Matrix Rings [14]. Moreover, across 2D depthwise convolution layers, it reduces online communication by 13.68%–20.83% compared to finite-field implementation of Convolutions in Overdrive [17].

1.2 Overview of Our Techniques

For authentication, we extend the coefficient modulus as in SPD$\mathbb{Z}_{2^k}$ by setting $q = k + s$ and working over $R = \mathbb{Z}_{2^q}[X]/(X^N + 1)$. We authenticate $z \in R$ via $\gamma = \alpha z$ with α

sampled uniformly from R, where only the low k bits of each coefficient represent the payload. Thus, a value error δ and a tag error Δ constitute a successful forgery only if $\delta \notin 2^k R, \alpha\delta = \Delta$. The ideal-chain structure of R explains how this modulus extension bounds the forgery probability. Writing $\mathfrak{m} = (X + 1)$, we have $2^k R = \mathfrak{m}^{Nk}$. A payload-changing error δ therefore has $\mathfrak{m}$-adic valuation $v < Nk$, and $\alpha\delta$ is uniform over $\delta R = \mathfrak{m}^v$. Consequently,

$$\Pr_{\alpha}[\alpha\delta = \Delta] \leq \frac{1}{|\mathfrak{m}^v|} = 2^{-(Nq-v)} \leq 2^{-(Ns+1)},$$

showing that the s additional bits across all N coefficients jointly provide Ns bits of authentication redundancy.

To check a batch of T opened values, the parties sample every coefficient of the challenge polynomials $\rho^{(j)}$ independently and uniformly from $\{0, \ldots, 2^s - 1\}$. The aggregated value and tag deviations are $\mathcal{D} = \sum_{j=1}^{T} \rho^{(j)}\delta^{(j)}$ and $E = \sum_{j=1}^{T} \rho^{(j)}\Delta^{(j)}$, respectively. If e is the aggregate deviation in the submitted check shares, a batch with valid openings passes exactly when $\alpha\mathcal{D} - E + e = 0$. Fix a payload-changing batch before the challenges are sampled, and let $v = \min_{1 \leq j \leq T} v(\delta^{(j)})$. The challenges are uniform modulo $2^s R$, so $\mathcal{D}$ has valuation $v + \ell$ with probability $2^{-(\ell+1)}$ for $0 \leq \ell < Ns$. The remaining probability 2^{-Ns} lies in the ideal $\mathfrak{m}^{v+Ns}$. Combining these probabilities with the per-layer key-matching bounds yields the tight batch forgery bound

$$\Pr[\text{Forge}] \leq (Ns + 4)2^{-(Ns+2)},$$

which is independent of the batch size and requires no modulus slack beyond 2^q. To allow key reuse across batches, the parties mask their check shares with fresh zero-sum masks before commitment and opening. Since the protocol aborts upon the first failed or incomplete check, an accepting-transcript analysis gives the same forgery bound for any finite batch limit fixed before the session begins.

Building on this MAC scheme, we construct an SPDZ-style online protocol over R_k, whose UC security in the hybrid model of Section 4 reduces directly to the session forgery bound $(Ns + 4)2^{-(Ns+2)}$. To generate the required preprocessing material from authenticated arithmetic over integer rings modulo powers of two, we employ a unified polynomial multiplication procedure that computes both the triple product $c = ab$ and the ring MACs $\alpha a, \alpha b, \alpha c$.

To reduce integer-ring triple consumption, we employ a hybrid Toom–4/Karatsuba recursion, where each Toom–4 level requires seven subproducts instead of the nine needed by two levels of Karatsuba. Standard Toom–4 interpolation involves even denominators. For our evaluation matrix M, every entry of $8M^{-1}$ has an odd denominator. These denominators are invertible modulo any power of two, so the scaled interpolation map is well defined. Hence, given the evaluation products g for the target product coefficients c, the relation $g = Mc$ yields $(8M^{-1})g = 8c$. This scaled interpolation reduces to a public linear transformation that each party applies locally to its value and underlying MAC shares.

We accumulate the scaling factors across all ℓ Toom–4 levels into a single factor of 2^D with $D = 3\ell$. After folding the polynomial modulo $X^N + 1$, the parties return the folded shares directly if $D = 0$. If $D > 0$, they perform one batched exact division by 2^D

on the final N coefficients. Although each coefficient is divisible by 2^D, the remainders of its individual shares can introduce a carry. With $p = 2$, locally rounding one share up and the other down eliminates this carry. With $p > 2$, the parties recover the exact quotient through an authenticated masked opening. Both mechanisms preserve the underlying integer-ring MAC relation.

Because each exact division by 2^D consumes $D = 3\ell$ bits of modulus precision and the product $c = ab$ must subsequently be multiplied by α, we adopt the two-stage precision schedule

$$q + 2D \xrightarrow{ab} q + D \xrightarrow{\alpha c} q.$$

Selecting the Toom–4 depth $0 \le \ell \le \lfloor n/2 \rfloor$ therefore balances the $7^\ell 3^{n-2\ell}$ scalar multiplications per polynomial product against $2D = 6\ell$ extra bits of initial precision and the associated exact-division overhead. Finally, for each secret ring element z, the output value shares of the polynomial product αz at q-bit precision directly form its MAC shares in R, while the underlying integer-ring MACs safeguard the preprocessing computation itself.

1.3 Related Work

Authenticated secret sharing underlies BDOZ, SPDZ, MASCOT, and Overdrive, which combine linear MACs with offline generation of correlated randomness over finite fields [6,5,3,4]. Expanding the algebraic domain, SPD$\mathbb{Z}_{2^k}$ adapts this paradigm to integer rings modulo 2^k, where Overdrive2k and Multipars develop specialized HE-based preprocessing [9,10,15]. To further compress preprocessing communication, pseudorandom correlation generators (PCGs) expand short seeds locally into correlated randomness [8,18]; along this line, Abram and Scholl apply Ring-LPN to multiparty SPDZ triple generation [19], and Preprocessing for Life combines correlation generators with distributed proofs to reduce storage and dealer-to-party communication in dealer-assisted MPC [20]. On the verification side, Rushing at SPDZ identifies MAC-key leakage in concurrent implementations, demonstrating that check scheduling and shared authentication state across executions directly affect both integrity and privacy [21]. Authenticated sharing also enables specialized function evaluation. SHARK combines it with interactive function secret sharing for two-party computation, while CAVERN combines it with verifiable incremental distributed point functions for spline evaluation in an honest-majority $(2 + 1)$-party setting [22,23].

Beyond elementwise scalar protocols, ring extensions and secret packing provide a general route to amortizing authentication and computation. Escudero et al. combine Galois rings with reverse multiplication-friendly embeddings (RMFEs) for dishonest-majority computation over $\mathbb{Z}_{2^k}$ [24], and degree-D embeddings extend this approach to higher-degree products and structured preprocessing correlations [25]; because extension-ring arithmetic operates on larger ring elements, its amortized efficiency hinges on the embedding rate. Alternatively, SuperPack achieves constant online communication through packed secret sharing when a fixed constant fraction of parties remains honest [26].

Rather than packing generic scalars, structured linear-algebra and tensor protocols lift preprocessing and authentication directly to matrix and convolution operations. Chen

et al. construct authenticated matrix multiplication triples using homomorphic encryption, and MatriGear accelerates their generation through optimized HE packing over prime fields [12,13]. To further reduce the preprocessing and storage overhead of matrix triples, Liu et al. introduce vector MACs for matrix-valued secrets in place of entrywise authentication [14]. Applying matrix-oriented protocols to convolutions, however, requires reshaping inputs into matrices; in the construction of Chen et al., this introduces replication overhead that grows with the filter size. Convolutions in Overdrive avoids this expansion through dedicated convolution triples and tensor-level protocols [17]. Notably, these matrix and convolution constructions operate over finite fields. Over $\mathbb{Z}_{2^k}$, MD-ML and MD-SONIC reduce the online cost of dot products and matrix operations through circuit-dependent preprocessing while retaining scalar MACs [27,28]. However, requiring the circuit to be fixed in advance limits flexibility when the computation structure is determined online, while generating circuit-specific masks and correlations adds to the preprocessing workload.

1.4 Organization

In Section 2 we introduce preliminaries, including notations, security model, and algebraic background. In Section 3, we present the information-theoretic MAC scheme and its security analysis. In Section 4, we show how to use our scheme to realise the functionality $\mathcal{F}_{\text{Online}}^{R_k}$. In Section 5, we present an efficient preprocessing protocol that constructs authenticated cyclotomic-ring triples from authenticated integer-ring triples using our hybrid Toom–4/Karatsuba multiplication scheme. In Section 6, we present our implementation of CycloSPDZ and benchmark it on convolution workloads. In Section 7, we conclude our work. In Appendix A, we prove Proposition 1, which characterizes the chain-ring structure of R. In Appendix B, we present the ideal functionalities and UC security proofs for our protocol.

To help the reader, Fig. 1 illustrates the different dependencies among our functionalities, including the section where the protocol is described.

2 Preliminaries

2.1 Notation and Algebraic Setting

In this paper, all rings we use are commutative with identity. For an arbitrary ring B, an integer $t \geq 1$, and elements $a^{(1)}, \ldots, a^{(t)} \in B$, we write $(a^{(1)}, \ldots, a^{(t)})_B := \sum_{i=1}^{t} a^{(i)} B$ for the ideal they generate, and we write $(a)_B := aB$ for the principal ideal. For brevity, ring subscripts are omitted when their meaning is unambiguous. Moreover, we write $B^\times$ for the unit group and $\ker(\varphi)$ for the kernel of a homomorphism φ.

We fix $k, s, n \in \mathbb{Z}_{\geq 1}$, and set $q := k + s$, $N = 2^n \geq 2$, $L := Nq$. Then the working ring and the payload ring are $R := \mathbb{Z}_{2^q}[X]/(X^N + 1)$ and $R_k := \mathbb{Z}_{2^k}[X]/(X^N + 1)$, respectively. Here k, q, s denote the payload bit length, the working bit length, and the redundancy bit length, respectively. For any $f \in \mathbb{Z}_{2^q}[X]$, denote its residue class in R by $[f] := f + (X^N + 1)_{\mathbb{Z}_{2^q}[X]} \in R$. And when there is no risk of misunderstanding, we just write f in place of $[f]$.

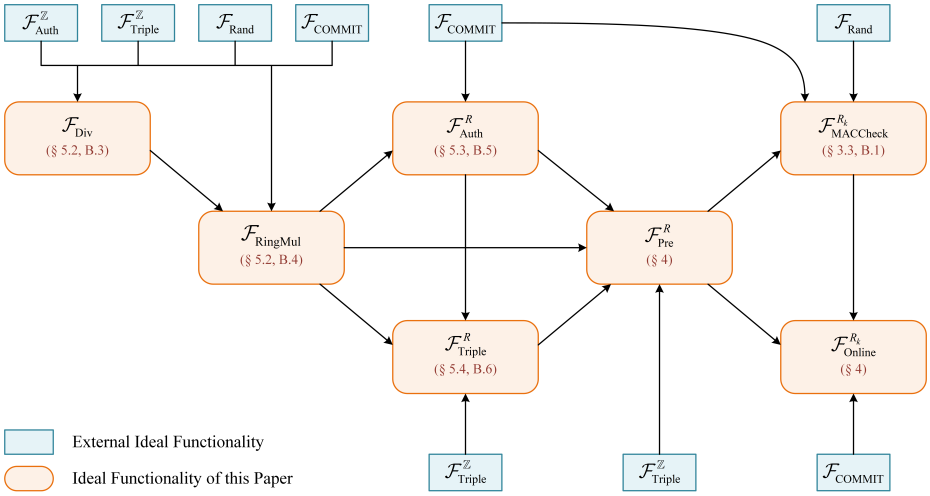

Fig. 1. Functionalities and their dependencies

Let $\pi_k \colon R \longrightarrow R_k$ denote the natural surjective ring homomorphism induced by coefficientwise reduction modulo 2^k. Then we set

$$I := \ker(\pi_k) = 2^k R, \qquad w := [X+1] \in R, \qquad \mathfrak{m} := (w)_R = wR.$$

With the convention $\mathfrak{m}^0 := R$, for any $z \in R \setminus \{0\}$, we define $v(z) := \max\{t \in \{0,\dots,L\} : z \in \mathfrak{m}^t\}$, and we stipulate that $v(0) := \infty$. We call $v(z)$ the $\mathfrak{m}$-adic order of z. And for any $a \in R$, we write its annihilator as $\mathrm{Ann}_R(a) := \{r \in R : ra = 0_R\}$.

For a polynomial z, we write $z[t]$ for its X^t-coefficient; meanwhile, for a vector z, $z[t]$ denotes its t-th coordinate. Moreover, for a finite nonempty set S, the notation $a \xleftarrow{\$} S$ denotes uniform sampling from S. Moreover, for any condition E, we write $\mathbf{1}_E$ for its indicator, which equals 1 if E holds and 0 otherwise.

For $z \in R$, an authenticated sharing $[\![z]\!]_\alpha$ under the global MAC key α consists of local pairs $(z_i, \gamma_{z,i})$. Let α_i denote the key share held by party P_i. Then it follows that

$$\alpha = \sum\nolimits_i \alpha_i, \qquad z = \sum\nolimits_i z_i, \qquad \gamma_z = \sum\nolimits_i \gamma_{z,i} = \alpha z \qquad \text{in } R.$$

In the sharing above, the payload is $\pi_k(z)$, whereas its MAC authenticates the full-ring value z.

A batch has size T, written $T^{(b)}$ for batch b. Party P_i's MAC share for item j is $\gamma_i^{(j)}$, or $\gamma_i^{(b,j)}$ in batch b. The check over R weights each item's equation by a challenge $\rho^{(j)}$ sampled from

$$\mathcal{T}_s := \left\{ \sum_{t=0}^{N-1} r^{(t)} X^t \;\middle|\; r^{(t)} \in \{0,\dots,2^s-1\} \right\} \subseteq R.$$

Using these challenges, each party P_i computes a check share, denoted by σ_i. We write σ_i^* and $\hat{\sigma}_i$ for its prescribed and actually submitted values, respectively. When the check

shares are randomized, $\xi_i \in R$ denotes the zero-sum masks added to the share of P_i such that $\sum_i \xi_i = 0$.

In addition to authenticated sharing over the polynomial ring R, we use authenticated sharing over integer rings. Here μ denotes the arithmetic precision in bits, and τ the additional authentication bits. The sharing $[\![x]\!]_{\beta,\mu}$ consists of local pairs $(x_i, \eta_{x,i})$ over $\mathbb{Z}_{2^{\mu+\tau}}$, satisfying

$$\beta = \sum\nolimits_i \beta_i, \quad x = \sum\nolimits_i x_i, \quad \eta_x = \sum\nolimits_i \eta_{x,i} = \beta x \pmod{2^{\mu+\tau}}.$$

Here each $\beta_i \in \mathbb{Z}_{2^\tau}$ is interpreted in $\mathbb{Z}_{2^{\mu+\tau}}$ using its least nonnegative representative. For polynomial arguments, $[\![\cdot]\!]_{\beta,\mu}$ applies coefficientwise.

2.2 Security Model

Adversarial Model. In this paper, we consider $p \geq 2$ parties $P_1, \ldots, P_p$ and a static malicious adversary $\mathcal{A}$ that can corrupt at most $p-1$ parties before the protocol begins. Let $C \subseteq \{1, \ldots, p\}$ denote the set of indices of corrupted party and $\mathcal{H} := \{1, \ldots, p\} \setminus C$ the honest ones. Unless otherwise specified, summations over party indices range from 1 to p. The adversary can obtain all the local states of the corrupted parties and control their behaviors. Therefore, the corrupted parties may deviate arbitrarily from the protocol, including modifying their shares, sending arbitrary messages, aborting the execution and so on. When analyzing statistical security in the ideal models considered below, we allow computationally unbounded adversaries.

Communication and Ideal Functionalities. In this paper, we assume that parties can access consistent broadcast and ideal functionalities for public randomness or commitments. Once the states to be checked are fixed, the public randomness functionality samples independent challenges uniformly at random from the domain specified by the protocol, using fresh randomness independent of the preceding state. The commitment functionality $\mathcal{F}_{\text{COMMIT}}$ records the submitted ring element according to the batch identifier and the sender identifier. It reveals only the commitment before opening and releases the committed value upon opening. In the protocol, honest parties open commitments sequentially once they receive all the commitments for a batch. Missing commitments or invalid openings cause an abort.

Nontrivial Forgery and Authentication Security. Let $T \geq 1$, and let $y^{(1)}, \ldots, y^{(T)} \in R$ be the values represented by the initial valid authenticated sharings. For each $j \in \{1, \ldots, T\}$, let $y^{(j)\prime} \in R$ denote the corresponding opened value, and define its deviation as $\delta^{(j)} := y^{(j)\prime} - y^{(j)}$. We call a forgery a nontrivial forgery if it changes the payload of at least one opened value, that is $\exists j \in \{1, \ldots, T\}$ such that $\pi_k(\delta^{(j)}) \neq 0$. Throughout this paper, when analyzing the security bound, we consider only nontrivial forgeries as described above.

Here we consider a single invocation of the batched MAC-checking protocol. Let V denote the view of the adversary $\mathcal{A}$ immediately before challenge generation, which includes the public transcript, the corrupted parties' local states, and the adversary's

randomness used so far. For this single-batch analysis, the global MAC key α remains uniformly distributed over R and is independent of V, whereas the value and tag deviations are fully determined by V.

Let Forge denote the event that a nontrivial forgery occurs and the current batch is nevertheless accepted, i.e.,

$$\text{Forge} := \{\text{accepted}\} \cap \left\{\exists j \in \{1, \ldots, T\} : \pi_k(\delta^{(j)}) \neq 0_{R_k}\right\}.$$

For every adversary we consider here, if $\Pr[\text{Forge}] \leq \varepsilon$, then the protocol has statistical soundness error at most ε. For a target statistical security level λ, we require $\varepsilon \leq 2^{-\lambda}$.

3 Information-Theoretic MAC Scheme

In this section, we construct a linear MAC over R that supports public linear operations on authenticated sharings and can verify a batch of openings through a single ring equation obtained via a random linear combination. We analyze the security of this batched verification using the ideal chain of R to characterize the aggregate deviation and its product with the MAC key. Using s-bit challenge coefficients, the protocol requires Ns random bits for each challenge and achieves the tight single-batch forgery bound $(Ns + 4)2^{-(Ns+2)}$. Moreover, by adding fresh zero-sum masks to check shares, we extend this single-batch bound to the entire session under the same MAC key for any finite batch limit fixed in advance.

3.1 Chain-Ring Structure and Product Distributions

The soundness analysis requires counting the solutions to multiplication equations over the ring R, and the counts depend on the size of principal ideals and their annihilators. We relate these counts to the possible orders of corresponding deviations to obtain a forgery bound covering all nontrivial deviations which actually change the payload. Since a deviation changes the payload exactly if and only if it lies outside the kernel of the projection, that is $\ker(\pi_k)$, the position of this kernel in the ideal chain determines the range of orders the bound must cover. The chain structure of R characterizes both this range and the cardinalities needed for the counting argument.

Proposition 1 (Repeated-Root Chain-Ring Structure). *Let $k, s \geq 1$ and $N = 2^n \geq 2$. Then $R = \mathbb{Z}_{2^q}[X]/(X^N + 1)$ is a finite local chain ring with unique maximal ideal $\mathfrak{m} = (X + 1)$, and $R/\mathfrak{m} \cong \mathbb{F}_2$. There exists a unit $g \in R^\times$ such that $w^N = 2g$. Moreover, all of the ideals of R are precisely $R = \mathfrak{m}^0 \supsetneq \mathfrak{m} \supsetneq \mathfrak{m}^2 \supsetneq \ldots \supsetneq \mathfrak{m}^L = \{0\}$, and for every $0 \leq t \leq L$, one has $|\mathfrak{m}^t| = 2^{L-t}, \mathrm{Ann}_R(w^t) = \mathfrak{m}^{L-t}$. In particular, for $0 \leq j \leq q$, one has $2^j R = \mathfrak{m}^{Nj}$, and $\ker(\pi_k) = \mathfrak{m}^{Nk}, R^\times = 1 + \mathfrak{m}$.*

The proof of proposition above is given in Appendix A.

From Proposition 1, we derive the following product law for a fixed deviation and the uniform global MAC key $\alpha \in R$. The public challenges are sampled independently from the short-coefficient set $\mathcal{T}_s$. Their residues are uniform in $R/2^s R$, a property we use below to determine the distribution of the aggregate value deviation across the relevant ideal layers.

Proposition 2 (Distribution of a Product with a Uniform Random Factor). *Let $\delta \in R \setminus \{0\}$, write $d = v(\delta)$, and let a be sampled uniformly from R. Then one has $\delta R = \mathfrak{m}^d$, and δa is uniformly distributed over $\mathfrak{m}^d$. To be more specific, for every $c \in R$, it holds*

$$\Pr[\delta a = c] = \begin{cases} 2^{-(L-d)}, & c \in \mathfrak{m}^d, \\ 0, & c \notin \mathfrak{m}^d. \end{cases}$$

Hence, for every fixed $\Delta \in R$, the random value $\delta a - \Delta$ is uniformly distributed over the coset $-\Delta + \mathfrak{m}^d$.

Proof. Since $d = v(\delta)$, which means $\delta \in \mathfrak{m}^d = (w^d)$, there must be some $u \in R$ such that $\delta = w^d u$. If $u \in \mathfrak{m}$, then $\delta \in \mathfrak{m}^{d+1}$, which conflicts with the fact that $d = v(\delta)$. Therefore, one must have $u \notin \mathfrak{m}$, and then, according to Proposition 1, u must be a unit. Consequently, one has $\delta R = w^d u R = w^d R = \mathfrak{m}^d$. Then the image of the additive group homomorphism $\mu_\delta : R \to R, a \mapsto \delta a$ is $\mathfrak{m}^d$. Since u is a unit, Proposition 1 further gives $\ker(\mu_\delta) = \mathrm{Ann}_R(\delta) = \mathrm{Ann}_R(w^d) = \mathfrak{m}^{L-d}$.

For any $c \in \mathfrak{m}^d$, fix a preimage $a^{(0)} \in R$ such that $\delta a^{(0)} = c$. Then $\delta a = c$ holds if and only if $\delta(a - a^{(0)}) = 0$, which also means $a - a^{(0)} \in \ker(\mu_\delta)$. Hence, all preimages of c exactly make up the coset $a^{(0)} + \ker(\mu_\delta)$, and its cardinality is $|\ker(\mu_\delta)| = |\mathfrak{m}^{L-d}| = 2^d$. Since a is uniform over R and $|R| = 2^L$, it follows that

$$\Pr[\delta a = c] = \frac{2^d}{2^L} = 2^{-(L-d)}.$$

For $c \notin \mathfrak{m}^d$, the equation $\delta a = c$ has no solution, and consequently the probability that $\delta a = c$ is equal to zero. Finally, since the mapping $z \mapsto z - \Delta$ is a bijection from $\mathfrak{m}^d$ to $-\Delta + \mathfrak{m}^d$, $\delta a - \Delta$ is uniform over $-\Delta + \mathfrak{m}^d$. □

3.2 Single-Batch MAC Checking

The verification of opened values authenticated under the same global MAC key can be delayed until a batch has accumulated, provided that all intervening computation and communication preserve privacy even if those opened values are maliciously altered. And to make the verification of batched values more efficient, we combine their authentication equations using independently and uniformly sampled polynomials with short coefficients as weights. It can be proven that the probability of successful forgery against this verification has a tight bound that is independent of the batch size.

Assume that the batch contains $T \geq 1$ original values $y^{(j)}$, all authenticated under a global MAC key uniformly sampled from R. During protocol execution, let $y^{(j)\prime}$ denote the value actually opened by parties, and let $\gamma_i^{(j)\prime}$ denote the corresponding tag share of P_i. The honest parties keep their original shares of the tags and the key without changing throughout the protocol execution.

To be more specific, once the opened values and the corresponding tags are fixed, all parties sample $\rho^{(1)}, \ldots, \rho^{(T)}$ independently and uniformly from $\mathcal{T}_s$ via the public randomness interface. The adversary cannot control the sampling of the challenge, nor can it reselect the opened values in the current batch or the original tag deviations after

observing the challenges. For each party P_i, define the share determined by its fixed local state as

$$\sigma_i^* = \sum_{j=1}^{T} \rho^{(j)} \left(y^{(j)\prime} \alpha_i - \gamma_i^{(j)\prime} \right).$$

Each honest party submits its actual share $\hat{\sigma}_i = \sigma_i^*$ through the commitment interface $\mathcal{F}_{\text{COMMIT}}$ specified in Section 2.2, while corrupted parties may submit arbitrary values of $\hat{\sigma}_i$ and pretend that these values are the actual shares σ_i^* they hold. Then, all parties sequentially open their commitments for the current batch. Corrupted parties may open their commitments after all honest parties or simply refuse to open their commitments. The protocol aborts if any opening is missing or invalid. Once all openings have been validated, each party calculates $\hat{\sigma} = \sum_i \hat{\sigma}_i$. The protocol accepts if and only if $\hat{\sigma} = 0$ holds in the full ring R; otherwise, it aborts.

The sum of the shares to be verified depends on the deviation of the opened values, tags, and final check shares. We define

$$\delta^{(j)} = y^{(j)\prime} - y^{(j)}, \quad \Delta^{(j)} = \sum_i \left(\gamma_i^{(j)\prime} - \gamma_i^{(j)} \right),$$

$$\mathcal{D} = \sum_{j=1}^{T} \rho^{(j)} \delta^{(j)}, \quad E = \sum_{j=1}^{T} \rho^{(j)} \Delta^{(j)}, \quad e = \sum_{i \in C} \left(\hat{\sigma}_i - \sigma_i^* \right).$$

In particular, if the protocol aborts before all the submissions are complete, we set $e = 0$. And when all the commitments are opened correctly, the original authentication relation yields

$$\begin{aligned} \hat{\sigma} &= \sum_i \hat{\sigma}_i = \sum_i \sigma_i^* + e \\ &= \sum_{j=1}^{T} \rho^{(j)} \left(\alpha y^{(j)\prime} - \sum_i \gamma_i^{(j)\prime} \right) + e \\ &= \alpha \mathcal{D} - E + e. \end{aligned}$$

For valid openings, the check accepts precisely when $\alpha \mathcal{D} - E + e = 0$. In an honest execution, all value, tag, and check-share deviations vanish. For a fixed nonzero $\mathcal{D}$ and fixed target $E - e$, Proposition 2 bounds the probability of acceptance under a uniform key in terms of $v(\mathcal{D})$. Combining this bound with the distribution of $\mathcal{D}$ under short-coefficient challenges gives the following result.

Theorem 1 (Tight Forgery Bound for Short-Coefficient Challenges). *Let k, s, n be positive integers, with $N = 2^n$, and $q = k + s$. In single-batch verification with challenges sampled independently and uniformly from $\mathcal{T}_s$, the probability of nontrivial forgery satisfies*

$$\Pr[\text{Forge}] \leq (Ns + 4) 2^{-(Ns+2)}$$

for every batch size $T \geq 1$ and every adversary $\mathcal{A}$. In particular, for each $T \geq 1$, there must exist an adversary that corrupts only one party and attains equality, which implies that the bound given above is tight.

Proof. A nontrivial forgery requires at least one payload to be changed, which means there must exist $\pi_k(\delta^{(j)}) \neq 0$ for some j. Fix a view V before challenge sampling in which a payload changes. This fixes the value and tag deviations. Then, let $v = \min_{1\le j\le T} v(\delta^{(j)})$ denote the minimum order of the value deviations and set $M = L - v$. Since $\ker(\pi_k) = \mathfrak{m}^{Nk}$ given by Proposition 1, it follows that $v \le Nk - 1, M = L - v \ge Ns + 1$. Suppose that t is an index satisfying $v(\delta^{(t)}) = v$, and fix all challenges except the t-th one. Then, it follows that $\mathcal{D} = \rho^{(t)}\delta^{(t)} + c$, where $c = \sum_{j\neq t} \rho^{(j)}\delta^{(j)} \in \mathfrak{m}^v$ is fixed. According to Proposition 1, we have $\mathfrak{m}^{Ns} = 2^s R$. Therefore, reduction modulo this ideal is equal to the reduction of each coefficient modulo 2^s. Hence, $\mathcal{T}_s$ contains exactly one representative of each coset in $R/\mathfrak{m}^{Ns}$. Then, since $\rho^{(t)}$ is sampled uniformly from $\mathcal{T}_s$, each coset is selected with probability 2^{-Ns}, and $\rho^{(t)} + \mathfrak{m}^{Ns}$ is therefore uniformly distributed over $R/\mathfrak{m}^{Ns}$.

According to Proposition 2, one has $\delta^{(t)}R = \mathfrak{m}^v$, so it follows that $\delta^{(t)}\mathfrak{m}^{Ns} = \mathfrak{m}^{v+Ns}$. Hence, multiplication by $\delta^{(t)}$ maps $R/\mathfrak{m}^{Ns}$ to $\mathfrak{m}^v/\mathfrak{m}^{Ns+v}$, and the cardinality formula given in Proposition 1 yields that both quotient groups have 2^{Ns} elements. Since the map is surjective and the two finite groups have the same number of elements, this map must be bijective. Together with the fact that, when $\rho^{(j)}, j \neq t$ has been fixed, $\rho^{(t)} + \mathfrak{m}^{Ns}$ is uniform over $R/\mathfrak{m}^{Ns}$, the bijection above yields that $\rho^{(t)}\delta^{(t)} + \mathfrak{m}^{v+Ns}$ is uniform over $\mathfrak{m}^v/\mathfrak{m}^{Ns+v}$. Adding a fixed term $c \in \mathfrak{m}^v$ preserves this distribution, regardless of the choice of c. Hence, $\mathcal{D} + \mathfrak{m}^{v+Ns}$ is uniform over $\mathfrak{m}^v/\mathfrak{m}^{Ns+v}$ conditioned on the view V.

It follows from the uniformity over the quotient group established above and cardinality formula given in Proposition 1 that, for $0 \le \ell \le Ns$, one has

$$\Pr[\mathcal{D} \in \mathfrak{m}^{v+\ell} \mid V] = \frac{|\mathfrak{m}^{v+\ell}|}{|\mathfrak{m}^v|} = \frac{2^{L-v-\ell}}{2^{L-v}} = 2^{-\ell}.$$

The layer of exact order $v, \ldots, v + Ns - 1$, together with the remaining ideal $\mathfrak{m}^{v+Ns}$, which contains all elements of order at least $v + Ns$, partitions all possible values of $\mathcal{D}$. Consequently, the probability that $\mathcal{D}$ lies in each possible layer or the remaining ideal is given by

$$\begin{aligned}
\Pr[v(\mathcal{D}) = v + \ell \mid V] &= \Pr[\mathcal{D} \in \mathfrak{m}^{v+\ell} \mid V] - \Pr[\mathcal{D} \in \mathfrak{m}^{v+\ell+1} \mid V] \\
&= 2^{-\ell} - 2^{-(\ell+1)} \\
&= 2^{-(\ell+1)}, \qquad 0 \le \ell < Ns, \\
\Pr[\mathcal{D} \in \mathfrak{m}^{v+\ell} \mid V] &= 2^{-Ns}, \qquad \ell = Ns.
\end{aligned}$$

The contribution of each layer to the forgery probability is given by the product of the probability that $\mathcal{D}$ belongs to it and its conditional acceptance probability. Moreover, when all challenges ρ and the adversary's independent private randomness $r_{\mathcal{A}}$ used after challenge sampling are fixed, $\mathcal{D}$ and E are therefore fixed. And since the commitment phase only reveals submission status and fixes the check shares belonging to the corrupted parties, e is also fixed by V, ρ and $r_{\mathcal{A}}$. Hence, in the acceptance equation $\alpha\mathcal{D} = E - e$, both $\mathcal{D}$ and the target $E - e$ are fixed, while the global key α remains uniform over R. Therefore, for $\mathcal{D} \neq 0$, Proposition 2 yields

$$\Pr[\text{Forge} \mid V, \rho, r_{\mathcal{A}}] \le \Pr[\alpha\mathcal{D} = E - e \mid V, \rho, r_{\mathcal{A}}] \le 2^{-(L-v(\mathcal{D}))}.$$

When $\mathcal{D}$ lies in the layer of exact order $v + \ell$, where $0 \le \ell < Ns$, we can substitute $L - v(\mathcal{D}) = L - (v + \ell) = M - \ell$ into the bound above and obtain the probability of a conditional forgery of at most $2^{-(M-\ell)}$. And as for the remaining condition $\mathcal{D} \in \mathfrak{m}^{v+Ns}$, the conditional forgery probability is at most 1. Every view with a payload change follows that $M \ge Ns + 1$, while for all other views the probability of a nontrivial forge is zero. Hence, applying the law of total probability to these conditions and averaging over V yields

$$\begin{aligned}\Pr[\text{Forge}] &\le \sum_{\ell=0}^{Ns-1} 2^{-(\ell+1)} \cdot 2^{-(Ns+1-\ell)} + 2^{-Ns} \cdot 1 \\ &= Ns\, 2^{-(Ns+2)} + 2^{-Ns} \\ &= (Ns+4)2^{-(Ns+2)}.\end{aligned}$$

To prove that the bound is tight, we can assume that a single corrupted party adds w^{Nk-1} to the share of the first opened value and leaves all other values and tags unchanged. The corrupted party also commits to and correctly opens its prescribed check shares. This strategy yields $\delta^{(1)} = w^{Nk-1}, \delta^{(j)} = 0$ for $j > 1, E = e = 0$. Since $\delta^{(1)} \notin \mathfrak{m}^{Nk} = \ker(\pi_k)$, the payload of the first value changes, with $v = Nk - 1$ and $M = Ns + 1$. In this condition, one has $\mathcal{D} = \rho^{(1)}\delta^{(1)}$ and all commitments are opened correctly. Then, a nontrivial forgery succeeds if and only if $\alpha\mathcal{D} = 0$.

For the remaining ideal $\mathfrak{m}^{v+Ns}$ in the preceding partition, the quotient group isomorphism established above and the fact that $\mathcal{T}_s \cap \mathfrak{m}^{Ns} = \{0\}$ yields $\mathcal{D} \in \mathfrak{m}^{v+Ns} \iff \rho^{(1)} \in \mathfrak{m}^{Ns} \iff \rho^{(1)} = 0$.

Thus, the remaining condition $\mathcal{D} \in \mathfrak{m}^{v+Ns}$ occurs with probability 2^{-Ns} and forces $\mathcal{D} = 0$, which makes the conditional forgery probability equal to 1. As for each of the preceding layers, $v(\mathcal{D}) = v + \ell$, where $0 \le \ell < Ns$. Since the target 0 lies in $\mathcal{D}R$, according to Proposition 2, the probability of a successful nontrivial forgery on each layer is given by $2^{-(Ns+1-\ell)}$. These probabilities are independent of the view V, and the strategy only changes the first opened value. Hence, for every batch size $T \ge 1$, the security bound of such a strategy is given by the law of total probability as

$$\begin{aligned}\Pr[\text{Forge}] &= \sum_{\ell=0}^{Ns-1} 2^{-(\ell+1)} \cdot 2^{-(Ns+1-\ell)} + 2^{-Ns} \cdot 1 \\ &= (Ns+4)2^{-(Ns+2)}.\end{aligned}$$

Therefore, for every admissible parameter choice and every batch size $T \ge 1$, there exists an adversary that attains the security bound, which implies that the bound given above is tight. □

3.3 Multi-Batch MAC Checking

The single-batch forgery bound given in Section 3.2 relies on a prerequisite that the MAC key is uniform over the full ring R, conditioned on the adversary's view before challenge sampling. However, in real-world use cases where multiple batches of checks are performed on openings authenticated using the same key, this prerequisite does not

always hold. When the same key is used for subsequent checks, the preceding public transcript and the outputs of previous verifications may change the possible distribution of the key. Therefore, to ensure the security of multi-batch authentication, we add fresh zero-sum masks using independent randomness across batches to the check shares in each batch.

For fixed authentication state, challenges, and corrupted masks, the honest check vector is uniform subject to its sum. The masks thus hide information carried by individual honest shares beyond their aggregate, while their zero sum preserves the verification equation. Hence, with this randomization, the probability of nontrivial forgery over the entire real-world session satisfies the tight single-batch bound given in Section 3.2, even when the same MAC key is used across all batches. We formalized the aforementioned process into Protocol Π_{MACCheck}, as illustrated in Fig. 2.

Protocol Π_{MACCheck}

Procedure for checking the public opened values $y^{(b,j)\prime}$, $j = 1, \ldots, T^{(b)}$, in batch b. Each party P_i holds its key share α_i and the corresponding current MAC shares $\gamma_i^{(b,j)\prime}$. All computations are over R. If $T^{(b)} = 0$, return accept.

Open phase:

1. Once the opened values and their current MAC shares are fixed, each party P_i obtains a fresh private mask $\xi_i^{(b)}$. The vector $(\xi_1^{(b)}, \ldots, \xi_p^{(b)}) \in R^p$ is uniform subject to $\sum_i \xi_i^{(b)} = 0$ and independent of the preceding state.
2. The parties obtain public challenges $\rho^{(b,1)}, \ldots, \rho^{(b,T^{(b)})}$ that are independently and uniformly distributed over $\mathcal{T}_s$ and independent of the preceding state and the masks.

MAC check phase:

1. Each party P_i computes the public aggregate $\bar{y}'$, its MAC aggregate G_i, and its prescribed check share $\sigma_i^{(b)*}$:
$$\bar{y}' = \sum_{j=1}^{T^{(b)}} \rho^{(b,j)} y^{(b,j)\prime}, \quad G_i = \sum_{j=1}^{T^{(b)}} \rho^{(b,j)} \gamma_i^{(b,j)\prime}, \quad \sigma_i^{(b)*} = \xi_i^{(b)} + \bar{y}'\alpha_i - G_i.$$
2. Each party P_i commits to $\sigma_i^{(b)*}$. After all commitments have been received, the parties open them sequentially. Let $\hat{\sigma}_i^{(b)}$ denote the value actually committed to and opened by P_i. Missing commitments or missing or invalid openings cause an abort.
3. Check whether $\sum_i \hat{\sigma}_i^{(b)} = 0$ holds in R. If it does, return accept; otherwise, abort.

Fig. 2. Batched MAC-check protocol Π_{MACCheck}

To analyze repeated invocations of this protocol and prove the security bound, we introduce the reference state and the accepting-transcript experiment.

Definition 1 (Reference State). *Start from the values, the tags, and the key shares each party holds at the very beginning, and fix the messages made public in the real execution up to the current point, including the deviations introduced by the adversary in those messages. Keeping these messages unchanged, update each party's shares according to the prescribed local steps. Each step uses the previously derived shares and the real*

opened messages for that step. Then, the resulting collection of value and tag shares for all current authenticated objects is called the reference state.

For each authenticated object z in the reference state, the prescribed public linear updates preserve the authentication relations, and therefore yield

$$z = \sum_i z_i, \qquad \gamma_z = \sum_i \gamma_{z,i} = \alpha z.$$

For each honest party, its actual shares equal to their derived shares. Value and tag deviations are determined by the differences between corrupted parties' actual shares and their derived shares.

Having specified the reference state, we next introduce an auxiliary experiment that supports comparison with accepting transcripts of the real execution.

Definition 2 (Accepting-Transcript Experiment). *Fix a static adversary $\mathcal{A}$. Use the initial inputs of the real-world session, sample the shares of the initial values, tags, and MAC key from the joint distribution specified by the protocol, and send each party the shares they should hold. The adversary receives all corrupted parties' initial information and determines their local operations and the messages they send based on the initial information, the randomness they hold, and the messages they have received during the execution. During the commitment phase of each batch b, the commitment interface only records the honest parties' status, which implies that the honest parties have submitted their check shares, but leaves the actual values of those check shares unbound. Meanwhile, the commitment interface still binds the check shares actually submitted by corrupted parties and verifies their openings following the same rules as in the real-world execution. Once all corrupted parties' check shares $\hat{\sigma}_i^{(b)}$ have been bound, use fresh independent randomness to sample the honest opening vector $(t_i^{(b)})_{i\in\mathcal{H}}$ uniformly at random from the set*

$$\left\{ (u_i)_{i\in\mathcal{H}} \in R^{|\mathcal{H}|} \;\middle|\; \sum_{i\in\mathcal{H}} u_i = -\sum_{i\in C} \hat{\sigma}_i^{(b)} \right\}.$$

At the opening stage, the interface uses each $t_i^{(b)}$ as the valid opening of the check commitment for honest party P_i. All other protocol steps and abort rules remain unchanged. Then, the resulting experiment is called the accepting-transcript experiment, abbreviated as AT.

In AT, the bound corrupted shares determine the distribution of the honest opening vector. So, in AT, once all commitments have been opened, the check sum must be zero, and the check equation must hold. Meanwhile, acceptance in the real-world execution depends on the deviations of values, tags, and check shares for this batch.

In both AT and real-world execution, we calculate the deviations of the opened values and corresponding tags relative to their respective reference state. For the j-th item in batch b, let $y^{(b,j)}$ denote the derived value and $\gamma_i^{(b,j)}$ denote the derived tag share of

party P_i. Meanwhile, let $y^{(b,j)\prime}$ denote the consistently opened value and $\gamma_i^{(b,j)\prime}$ denote the current tag shares of party P_i. Then, the deviations and their aggregates follow

$$\delta^{(b,j)} = y^{(b,j)\prime} - y^{(b,j)}, \quad \Delta^{(b,j)} = \sum_{i\in C}\big(\gamma_i^{(b,j)\prime} - \gamma_i^{(b,j)}\big),$$

$$\mathcal{D}^{(b)} = \sum_{j=1}^{T^{(b)}} \rho^{(b,j)}\delta^{(b,j)}, \quad E^{(b)} = \sum_{j=1}^{T^{(b)}} \rho^{(b,j)}\Delta^{(b,j)}, \quad e^{(b)} = \sum_{i\in C}\big(\hat{\sigma}_i^{(b)} - \sigma_i^{(b)*}\big).$$

Here, $\sigma_i^{(b)*}$ denotes the prescribed check share, which includes the zero-sum mask, while $e^{(b)}$ denotes the aggregate deviation of the check shares actually committed to by the corrupted parties from their prescribed shares.

Also, in both AT and real-world execution, we define the event $A^{(b)} = \{\alpha\mathcal{D}^{(b)} - E^{(b)} + e^{(b)} = 0\}$. In the real-world execution, once all commitments have been opened correctly, the zero-sum masks cancel out exactly upon aggregation. Then, the authentication relations preserved by the linear updates yields $\hat{\sigma}^{(b)} = \sum_i \hat{\sigma}_i^{(b)} = \alpha\mathcal{D}^{(b)} - E^{(b)} + e^{(b)}$. Thus, a real-world execution passes the MAC check if and only if $A^{(b)}$ occurs.

On the other hand, in the real-world execution, condition on the MAC key, authentication state, challenges, and the corrupted parties' zero-sum masks and committed check shares for the current batch, the uniformity of the fresh zero-sum masks implies that the honest parties' check vector is uniformly distributed over the set of vectors satisfying

$$\sum_{i\in\mathcal{H}} \sigma_i^{(b)*} = \alpha\mathcal{D}^{(b)} - E^{(b)} + e^{(b)} - \sum_{i\in C}\hat{\sigma}_i^{(b)}.$$

When $A^{(b)}$ occurs, $\alpha\mathcal{D}^{(b)} - E^{(b)} + e^{(b)} = 0$ holds, and therefore one has $\sum_{i\in\mathcal{H}} \sigma_i^{(b)*} = -\sum_{i\in C}\hat{\sigma}_i^{(b)}$. Hence, under the fixed conditions above, whenever $A^{(b)}$ holds, the honest check share vector in the real-world execution has exactly the same distribution as the honest opening vector sampled in the accepting-transcript experiment.

Using the distribution relation established above, we compare the accepting transcripts from the real-world execution with those generated by AT on a batch-by-batch basis to analyze the forgery probability over the entire session. Assume that key shares are independently and uniformly distributed over R. Value material is generated with randomness independent of key initialization, without using key or MAC shares of honest parties. For each fresh authenticated value, the adversary may choose its value and MAC shares from its current view. Meanwhile, the honest shares are independently and uniformly completed using fresh randomness subject to the prescribed totals. Apart from check messages, honest parties' public messages depend only on their initial value state (excluding MAC key and MAC shares), the public transcript, and independent randomness. Under these conditions, the key α remains uniformly distributed over R even when conditioned on the adversary's view in AT. Consequently, the security bound established in the single-batch analysis can be applied to the auxiliary experiment. Let $\mathrm{Forge}_{\mathrm{sess}}$ denote the event that the real session accepts at least one batch containing an item with $\pi_k(\delta^{(b,j)}) \neq 0$. The following theorem establishes the same tight bound as in the single-batch case, independently of any finite batch limit fixed before the session begins.

Theorem 2 (Tight Forgery Bound for Multiple Batches under a Single MAC Key). *Let k, s, n be positive integers, with $N = 2^n$ and $q = k + s$. Under the initialization and state-update conditions above, consider using the same MAC key shares to verify multi-batch openings. Each batch invokes Π_{MACCheck} in Fig. 2, and the session terminates at the first check that rejects or fails to complete. Then, the probability of nontrivial session forgery satisfies*

$$\Pr_{\text{real}}[\mathrm{Forge}_{\mathrm{sess}}] \le (Ns + 4)2^{-(Ns+2)}$$

for every finite batch limit $B_{\max} \ge 1$ fixed before the session begins, every sequence of at most $B_{\max}$ nonempty batches, and every adversary $\mathcal{A}$. In particular, for each $B_{\max} \ge 1$ and first-batch size $T^{(1)} \ge 1$, there must exist an adversary that corrupts only one party and attains equality, which implies that the bound given above is tight.

Proof. In both AT and real-world execution, let b_{chg} denote the index of the first batch containing an item j such that $\pi_k(\delta^{(b,j)}) \neq 0$. If no such batch exists, then set $b_{\mathrm{chg}} = \infty$. Since the session terminates when a check rejects or fails to complete, a nontrivial forgery occurs if and only if $b_{\mathrm{chg}} < \infty$ and that batch completes normally and accepts. Thus, by the law of total probability, one has

$$\begin{aligned}\Pr_{\text{real}}[\mathrm{Forge}_{\mathrm{sess}}] &= \sum_{b=1}^{B_{\max}} \Pr_{\text{real}}[b_{\mathrm{chg}} = b] \cdot \Pr_{\text{real}}[\mathrm{Forge}_{\mathrm{sess}} \mid b_{\mathrm{chg}} = b] \\ &= \sum_{b=1}^{B_{\max}} \Pr_{\text{real}}[b_{\mathrm{chg}} = b \wedge \mathrm{Forge}_{\mathrm{sess}}].\end{aligned}$$

To bound each batch's contribution to the session forgery probability, we compare the probabilities of the corresponding transcripts in the real-world execution and AT. Conditioned on the same key, initial material, the randomness held by the adversary, and the zero-sum masks of the corrupted parties, the preceding conditional distribution identity applies at each batch. Hence, for any transcript in which the first b batches complete normally, its probability of occurring with all batches accepted in the real-world execution equals its probability in AT multiplied by $\prod_{\ell=1}^{b} \mathbf{1}_{A^{(\ell)}}$. Conditioned on $b_{\mathrm{chg}} = b$ and $A^{(b)}$, the probability that the b batches complete normally and all preceding acceptance conditions $A^{(1)}, \ldots, A^{(b-1)}$ hold is at most 1. Therefore, applying this bound to the preceding transcript probability relation and averaging over the common variables yields

$$\Pr_{\text{real}}[b_{\mathrm{chg}} = b \wedge \mathrm{Forge}_{\mathrm{sess}}] \le \Pr_{\mathrm{AT}}[b_{\mathrm{chg}} = b \wedge A^{(b)}].$$

To bound the probability in AT that the first payload change occurs in batch b and the acceptance $A^{(b)}$ holds, let $V^{(b)}$ consist of the adversary's view before challenge sampling in batch b, all value data and non-authentication auxiliary randomness generated before these challenges, and the adversary's entire private random tape. For $i_* \in \mathcal{H}$ and $t \in R$, since honest check opening in AT depends only on bound corrupted submissions and fresh randomness, the shifts $\alpha_{i_*} \mapsto \alpha_{i_*} + t$ and $\gamma_{z,i_*} \mapsto \gamma_{z,i_*} + tz$ for each fresh authenticated value z preserve the sampling distribution and $V^{(b)}$. Hence, given $V^{(b)}$, α remains uniformly distributed. Moreover, all $\delta^{(b,j)}, \Delta^{(b,j)}$ and the event $\{b_{\mathrm{chg}} = b\}$ are

determined by $V^{(b)}$. After further conditioning on the challenges, thereby fixing $\mathcal{D}^{(b)}$, $E^{(b)}$ and $e^{(b)}$, the key remains uniformly distributed over R.

On $\{b_{\text{chg}} = b\}$, at least one value deviation changes the payload. Let $v^{(b)}$ denote the minimum valuation of the value deviations in batch b. By Proposition 1, $\ker(\pi_k) = \mathfrak{m}^{Nk}$, so $v^{(b)} \le Nk - 1$ and $L - v^{(b)} \ge Ns + 1$. The layer calculation in the proof of Theorem 1 gives probability $2^{-(\ell+1)}$ that $\mathcal{D}^{(b)}$ has valuation $v^{(b)} + \ell$ for $0 \le \ell < Ns$. The probability that $\mathcal{D}^{(b)}$ lies in the tail ideal $\mathfrak{m}^{v^{(b)}+Ns}$ is 2^{-Ns}. For fixed challenges, Proposition 2 bounds the conditional acceptance probability on layer ℓ by $2^{-(L-v^{(b)}-\ell)} \le 2^{-(Ns+1-\ell)}$. Using a conditional bound of 1 on the tail ideal and summing over these cases gives

$$\begin{aligned}\Pr_{\text{AT}}[A^{(b)} \mid V^{(b)}] &\le \sum_{\ell=0}^{Ns-1} 2^{-(\ell+1)} 2^{-(Ns+1-\ell)} + 2^{-Ns} \\ &= (Ns+4)2^{-(Ns+2)}.\end{aligned}$$

Averaging over the $V^{(b)}$ for which $b_{\text{chg}} = b$ bounds the conditional acceptance probability $\Pr_{\text{AT}}[A^{(b)} \mid b_{\text{chg}} = b]$ by $(Ns+4)2^{-(Ns+2)}$. Substituting this bound into the factorization of each batch's joint probability and using the pairwise disjointness of the events $\{b_{\text{chg}} = b\}$, it follows that

$$\begin{aligned}\Pr_{\text{real}}[\text{Forge}_{\text{sess}}] &\le \sum_{b=1}^{B_{\max}} \Pr_{\text{AT}}[b_{\text{chg}} = b \wedge A^{(b)}] \\ &= \sum_{b=1}^{B_{\max}} \Pr_{\text{AT}}[b_{\text{chg}} = b] \cdot \Pr_{\text{AT}}[A^{(b)} \mid b_{\text{chg}} = b] \\ &\le (Ns+4)2^{-(Ns+2)} \sum_{b=1}^{B_{\max}} \Pr_{\text{AT}}[b_{\text{chg}} = b] \\ &\le (Ns+4)2^{-(Ns+2)}.\end{aligned}$$

To prove tightness, let a single corrupted party add w^{Nk-1} to the share of first opened value in the first batch, leave all other value shares and tags unchanged, and commit to and validly open its prescribed check share including the zero-sum mask. Then, one has $E^{(1)} = e^{(1)} = 0$. The deviation changes the payload, and the first batch accepts if and only if $\alpha\rho^{(1,1)}w^{Nk-1} = 0$. The tightness calculation in Theorem 1 shows that this probability is exactly $(Ns+4)2^{-(Ns+2)}$. Since acceptance of the first batch constitutes a session forgery, the bound is tight. □

Under the same conditions, we obtain the following UC security guarantee for the batched MAC-checking protocol.

Theorem 3. *The protocol* Π_{MACCheck} *implements* $\mathcal{F}_{\text{MACCheck}}^{R_k}$ *in the* $\mathcal{F}_{\text{Pre}}^{R}$*-hybrid model, with statistical distinguishing advantage at most* $(Ns+4)2^{-(Ns+2)}$.

The proof of Theorem 3 is given in Appendix B.1.

4 Online Phase

Our protocol is divided in two phases, a preprocessing phase and an online phase. The preprocessing, which is independent of each party's input, implements a functionality $\mathcal{F}_{\text{Pre}}^{R}$ that generates the necessary shared, authenticated values needed to compute the given function securely. This functionality is specified in Fig. 3. Its construction is presented in Section 5.

Functionality $\mathcal{F}_{\text{Pre}}^{R}$

Initialization. On input (Init) from all parties, sample $\alpha_1, \ldots, \alpha_p \in R$ independently and uniformly, privately deliver α_i to P_i, $\alpha = \sum_{i=1}^{p} \alpha_i$.
Authenticated sharing. To distribute $[\![z]\!]_\alpha$, receive the corrupted parties' value and MAC shares from the adversary. Sample the remaining shares uniformly subject to $\sum_i z_i = z$ and $\sum_i \gamma_{z,i} = \alpha z$, and privately send $(z_i, \gamma_{z,i})$ to each P_i.
Random value. On input (Random) from all parties, sample $r \in R$ uniformly and distribute $[\![r]\!]_\alpha$.
Input mask. On input (Input, P_ℓ) from all parties, sample $r \in R$ uniformly and distribute $[\![r]\!]_\alpha$. Privately deliver the full value r only to P_ℓ.
Multiplication triple. On input (Triple) from all parties, sample $a, b \in R$ independently and uniformly, compute $c = ab \in R$, and distribute $([\![a]\!]_\alpha, [\![b]\!]_\alpha, [\![c]\!]_\alpha)$.
Output mask. On input (OutputMask) from all parties, sample $r \in I$ uniformly and distribute the authenticated sharing $[\![r]\!]_\alpha$ to randomize the high bits of every coefficient in a whole-ring output.
Zero-sum masks. On input (Zero) from all parties, sample $\xi_1, \ldots, \xi_{p-1} \in R$ independently and uniformly, set $\xi_p = -\sum_{i=1}^{p-1} \xi_i$, and privately deliver ξ_i to P_i.
Abort. On input (Abort) from the adversary, or upon receiving inconsistent or invalid requests, deliver $\perp$ to all parties and abort.

Fig. 3. Ideal functionality for the preprocessing phase

Fig. 4 specifies the ideal online functionality $\mathcal{F}_{\text{Online}}^{R_k}$ for evaluating the agreed arithmetic circuit on private inputs in R_k. The functionality stores inputs and intermediate values under vid, reveals results upon output requests, and outputs $\perp$ on abort, where vid is a fresh, valid identifier.

We now present the online protocol Π_{Online}^{R} for realizing $\mathcal{F}_{\text{Online}}^{R_k}$. Using authenticated sharings over R and preprocessing material from $\mathcal{F}_{\text{Pre}}^{R}$, the protocol returns the designated public outputs or $\perp$. The protocol is specified in Fig. 5.

The protocol follows the communication and commitment conventions in Section 2.2. For vector commitments, each coordinate uses a distinct commitment identifier bound to the session, batch, and sender.

Each party stores its value and MAC shares by wire identifier and retains the MAC shares associated with opened values for verification by Π_{MACCheck} in Fig. 2. For each batch b, the parties invoke $\mathcal{F}_{\text{Pre}}^{R}$ with (Zero) to obtain fresh masks $\xi_i^{(b)}$, and then invoke $\mathcal{F}_{\text{Rand}}(\mathcal{T}_s^{T^{(b)}})$ to obtain public challenges, following the order specified in that protocol. After a successful check, the pending verification records for that batch are cleared. The

Functionality $\mathcal{F}_{\text{Online}}^{R_k}$

Initialization. On input (Init) from all parties, fix the arithmetic circuit and its execution order, and the owner of each input.

Input. On input $(\mathsf{Input}, P_\ell, \text{vid}, x)$ from P_ℓ and $(\mathsf{Input}, P_\ell, \text{vid})$ from the other parties, where $x \in R_k$, store (vid, x).

Linear operation. On input $(\mathsf{Linear}, (r_j, \text{vid}_j)_{j=1}^t, c, \text{vid}_u)$ from all parties, where $r_1, \ldots, r_t, c \in R$ are public coefficients, retrieve the value $x^{(j)}$ stored under vid_j for each $j = 1, \ldots, t$, compute $u = \pi_k(c) + \sum_{j=1}^t \pi_k(r_j) x^{(j)}$, and store (vid_u, u).

Multiplication. On input $(\mathsf{Multiply}, \text{vid}_1, \text{vid}_2, \text{vid}_3)$ from all parties, retrieve $(\text{vid}_1, x^{(1)})$, $(\text{vid}_2, x^{(2)})$, and store $(\text{vid}_3, x^{(1)} x^{(2)})$.

Output. On input $(\mathsf{Output}, \text{vid})$ from all honest parties, look for (vid, y) in memory and if present, deliver y to the adversary. If the adversary replies with $(\mathsf{Proceed})$, deliver y to all honest parties. If it replies with (Abort), deliver $\perp$ to all honest parties and abort.

Abort. At any stage, abort on (Abort) from the adversary or a request inconsistent with the agreed circuit or its execution order. Previously delivered outputs are retained.

Fig. 4. Ideal functionality for the online phase

session aborts on any missing, invalid, or inconsistent opening, or if a MAC check rejects or fails to complete.

Protocol Π^R_{Online}

Initialization. The parties invoke $\mathcal{F}^R_{\text{Pre}}$ with (Init) to obtain their MAC key shares α_i. They invoke $\mathcal{F}^R_{\text{Pre}}$ with (Triple) to get enough triples $(\llbracket a \rrbracket_\alpha, \llbracket b \rrbracket_\alpha, \llbracket c \rrbracket_\alpha)$. They invoke $\mathcal{F}^R_{\text{Pre}}$ with (Input, P_ℓ) to obtain random sharings $\llbracket r \rrbracket_\alpha$ where P_ℓ learns r.

Input. To share an input $x \in R_k$ held by P_ℓ:

1. P_ℓ lifts x coefficientwise to R by taking the unique representative of each coefficient modulo 2^k in $\{0, \ldots, 2^k - 1\}$ and interpreting it modulo 2^q. This yields $z \in R$ with $\pi_k(z) = x$. Party P_ℓ broadcasts $\epsilon = z - r$.
2. The parties locally compute its authenticated input shares as $z_i = r_i + \mathbf{1}_{i=1}\epsilon, \gamma_{z,i} = \gamma_{r,i} + \epsilon\alpha_i$.

Linear operation. To compute $\llbracket u \rrbracket_\alpha = c + \sum_{j=1}^t r_j \llbracket z^{(j)} \rrbracket_\alpha$, where $r_j, c \in R$ are public:

1. P_i locally computes
$$u_i = \sum_{j=1}^t r_j z_i^{(j)} + 1_{i=1} c, \quad \gamma_{u,i} = \sum_{j=1}^t r_j \gamma_{z^{(j)},i} + c\alpha_i.$$
2. P_i adopts $(u_i, \gamma_{u,i})$ as its shares of $\llbracket u \rrbracket_\alpha$. Here, $\mathbf{1}_{i=1}$ ensures that c is incorporated into one value share, while $c\alpha_i$ preserves MAC consistency. When no additive constant is involved, c is set to zero. This procedure naturally supports all linear operations locally.

Secret multiplication. To multiply $\llbracket z \rrbracket_\alpha$ and $\llbracket u \rrbracket_\alpha$:

1. P_i locally computes
$$d_i = z_i - a_i, \quad f_i = u_i - b_i, \quad \gamma_{d,i} = \gamma_{z,i} - \gamma_{a,i}, \quad \gamma_{f,i} = \gamma_{u,i} - \gamma_{b,i}.$$
2. The parties commit to and open (d_i, f_i) to reconstruct $d' = \sum_i d_i$ and $f' = \sum_i f_i$, retaining the opened values and MAC shares for subsequent verification via Π_{MACCheck}.
3. P_i locally computes $y_i = c_i + d' b_i + f' z_i, \gamma_{y,i} = \gamma_{c,i} + d'\gamma_{b,i} + f'\gamma_{z,i}$, thereby obtaining an authenticated sharing $\llbracket y \rrbracket_\alpha$ of the product.

Output. To output a value $\llbracket z \rrbracket_\alpha$:

1. Invoke Π_{MACCheck} on all pending values during multiplications.
2. P_i invoke $\mathcal{F}^R_{\text{Pre}}$ to obtain a fresh output mask $\llbracket r \rrbracket_\alpha$, where $r \in I$. They locally compute $\llbracket v \rrbracket_\alpha = \llbracket z \rrbracket_\alpha + \llbracket r \rrbracket_\alpha$. Commit to and open the value shares to reconstruct v'.
3. Invoke Π_{MACCheck} on v'. If the check accepts, output $\pi_k(v')$.

Abort. Explicit rejection, invalid or inconsistent openings, reuse of preprocessing material, specification violations, or a failed MAC check terminate the session.

Fig. 5. Protocol for reactive secure multi-party computation over R_k

Correctness. For an honest input, $r + \epsilon = z$. Linear operations apply the same public coefficients to the value shares and MAC shares, with the MAC on an additive constant distributed according to the key shares. In a multiplication, $d' = z - a$ and $f' = u - b$, so

$$y = c + (z - a)b + (u - b)z = zu, \quad \pi_k(y) = \pi_k(z)\pi_k(u).$$

The output mask leaves $\pi_k(z)$ unchanged, and zero-sum randomization preserves the sum of the check shares. Thus, an honest execution returns the designated circuit outputs.

Theorem 4. *The protocol Π^R_{Online} implements $\mathcal{F}^{R_k}_{\text{Online}}$ in the $\mathcal{F}^R_{\text{Pre}}$-hybrid model, with statistical distinguishing advantage at most $(Ns + 4)2^{-(Ns+2)}$.*

The proof of Theorem 4 is given in Appendix B.2.

5 Preprocessing phase

We derive cyclotomic-ring triples, authenticated random values, and input/output masks from authenticated multiplication triples over power-of-two integer rings. Secret-shared polynomial multiplication uses these triples for coefficient products and local linear operations for their combinations, enabling the computation of ring-triple products ab and MACs αz. To reduce secret multiplications, we combine Toom–4 and Karatsuba, using extra intermediate precision and one batched exact division whenever the recursion includes a Toom–4 level.

5.1 Authenticated Integer-Ring Triples and Secret Multiplication

We use authenticated integer-ring triples supplied by SPD$\mathbb{Z}_{2^k}$ preprocessing, which relies on oblivious transfer [9]. Alternative constructions, such as Overdrive2k [10] and Multipars [15], employ HE for triple generation. We denote the initialization and triple generation by $\mathcal{F}^{\mathbb{Z}}_{\text{Triple}}$, and the integer-ring authentication interface by $\mathcal{F}^{\mathbb{Z}}_{\text{Auth}}$.

Let $\mu_{\max}$ denote the maximum bit length for which the triples guarantee correct multiplication, with each invocation using $\mu \le \mu_{\max}$. During initialization, each party P_i obtains a key share $\beta_i \in \mathbb{Z}_{2^\tau}$, sampled independently and uniformly for honest parties. The global key β remains fixed throughout the session. Multiplication is performed modulo 2^μ, while authentication is defined modulo $2^{\mu+\tau}$.

For a polynomial $\tilde{z}$ with coefficients in $\mathbb{Z}_{2^{\mu+\tau}}$, we denote its coefficientwise authenticated sharing by $[\![\tilde{z}]\!]_{\beta,\mu}$. The corresponding polynomial $z \in R$ is obtained by coefficientwise reduction modulo 2^q, with polynomial products reduced modulo $X^N + 1$.

Each invocation of $\mathcal{F}^{\mathbb{Z}}_{\text{Triple}}$ provides an authenticated triple

$$\big([\![A]\!]_{\beta,\mu_{\max}}, [\![B]\!]_{\beta,\mu_{\max}}, [\![C]\!]_{\beta,\mu_{\max}}\big).$$

Each party P_i receives $(A_i, B_i, C_i, \eta_{A,i}, \eta_{B,i}, \eta_{C,i})$, satisfying

$$C \equiv AB \pmod{2^{\mu_{\max}}}, \qquad \sum_i \eta_{U,i} \equiv \beta U \pmod{2^{\mu_{\max}+\tau}}, \quad U \in \{A, B, C\}.$$

Here, A and B are sampled independently and uniformly from $\mathbb{Z}_{2^{\mu_{\max}+\tau}}$. The lower $\mu_{\max}$ bits of C match those of AB, while its upper τ bits are independently uniform. The global key β authenticates the full values, and fresh randomness is used for each invocation. For any $\mu \le \mu_{\max}$, reducing the value and MAC shares modulo $2^{\mu+\tau}$ preserves both multiplication and authentication relations. Authenticated random coefficients and masks are obtained from the A or B components of fresh triples.

Public linear operations are performed locally. For secret coefficient multiplication, we use the SPD$\mathbb{Z}_{2^k}$ authenticated Beaver multiplication protocol $\Pi^{\mathbb{Z}}_{\text{Mult}}$. Given authenticated sharings $\left(\llbracket u^{(j)} \rrbracket_{\beta,\mu}, \llbracket v^{(j)} \rrbracket_{\beta,\mu}\right)_{j=1}^{T}$ at working precision μ, it outputs $\llbracket y^{(j)} \rrbracket_{\beta,\mu}$ satisfying $y^{(j)} \equiv u^{(j)} v^{(j)} \pmod{2^{\mu}}$. Each multiplication consumes a fresh integer-ring triple, and values opened in the same batch are jointly verified by Open.

The procedure Open opens and checks authenticated integer-ring values at precision μ. Before opening each value, it adds an authenticated high-bit mask $2^{\mu} r$, using independent fresh masks obtained from $\mathcal{F}^{\mathbb{Z}}_{\text{Auth}}$. It then performs the batched MAC check modulo $2^{\mu+\tau}$ and, on acceptance, returns the low μ bits of each opened value.

We use $\mathsf{Zero}(b, m)$ to generate zero-sum shares over $(\mathbb{Z}_{2^b})^m$. For each pair $i < j$, parties P_i and P_j independently sample uniform vectors, commit to them, and open them over their private channel. Let t_{ij} denote the sum of these vectors. Each party P_i computes

$$\xi_i = \sum_{j>i} t_{ij} - \sum_{j<i} t_{ji}, \qquad \sum_i \xi_i = 0.$$

The variant of the opening procedure with zero-sum masks is denoted by Open_{ξ}. Each party P_i adds a fresh zero-sum mask $\xi_i \in \mathbb{Z}_{2^{\mu+\tau}}$ to its batched MAC-check share σ_i, obtaining $\sigma_i^* = \sigma_i + \xi_i$. The parties commit to and open these masked shares. Writing $\hat{\sigma}_i$ for the actually opened share, the check accepts if all openings are valid and $\sum_i \hat{\sigma}_i = 0 \pmod{2^{\mu+\tau}}$.

5.2 Authenticated Polynomial Multiplication

We now construct an authenticated polynomial multiplication protocol. Given $\llbracket u \rrbracket_{\beta,\mu}$ and $\llbracket v \rrbracket_{\beta,\mu}$, each with $N = 2^n$ coefficients, the goal is to compute $uv \bmod (X^N + 1)$ with fewer secret multiplications while preserving coefficient authentication.

The direct construction computes the ordinary polynomial product $c = uv = \sum_{t=0}^{2N-2} c[t] X^t$ and applies negacyclic reduction using $X^N = -1$,

$$c[t] = \sum_{j+l=t} u[j] v[l], \qquad (uv \bmod (X^N + 1))[t] = c[t] - c[t+N], \quad 0 \le t < N,$$

where out-of-range coefficients are defined as zero. This requires N^2 secret coefficient multiplications, with all remaining operations performed locally. Karatsuba reduces four subproducts to three by splitting each input into two blocks, while Toom–4 uses seven evaluation products for four blocks, compared with nine under two levels of Karatsuba. Since the powers of two in Toom–4 interpolation are not invertible modulo 2^k, we employ scaled interpolation, reconstructing eight times the product coefficients at each Toom–4 node. With ℓ Toom–4 levels along each recursive path, the accumulated

scaling factor is 2^D, where $D = 3\ell$. After negacyclic reduction, the parties return the folded shares directly if $D = 0$. If $D > 0$, one batched exact division recovers the value and MAC shares of the final N coefficients. We use D additional bits of intermediate precision to preserve the required output precision.

The recursion uses ℓ Toom–4 levels followed by $n - 2\ell$ Karatsuba levels, where $0 \le \ell \le \lfloor n/2 \rfloor$. The parties locally evaluate the input polynomials, batch the $7^\ell 3^{n-2\ell}$ secret coefficient multiplications at the leaves using $\Pi^{\mathbb{Z}}_{\text{Mult}}$, and reconstruct the product bottom-up. For example, with $N = 32$ and $\ell = 2$, the construction requires $7^2 \cdot 3 = 147$ secret multiplications and introduces a scaling factor of $2^6 = 64$. After negacyclic reduction, a single batched exact division recovers the lower $\mu - 6$ bits of the product.

We describe the reconstruction at each type of node. For Karatsuba, each input is split into two blocks of m coefficients. Let $Y = X^m$, $u = u_0 + Yu_1$, and $v = v_0 + Yv_1$. The three recursive products are

$$g_0 = u_0 v_0, \qquad g_1 = (u_0 + u_1)(v_0 + v_1), \qquad g_2 = u_1 v_1,$$

giving $uv = g_0 + Y(g_1 - g_0 - g_2) + Y^2 g_2$.

For Toom–4, each input is split into four blocks, with $U(Y) = \sum_{j=0}^{3} u_j Y^j$ and $V(Y) = \sum_{j=0}^{3} v_j Y^j$, where $Y = X^m$. The parties locally evaluate both polynomials at $S = \{0, 1, -1, 2, -2, 3\}$ and infinity:

$$U(t) = \sum_{j=0}^{3} u_j t^j \quad (t \in S), \qquad U(\infty) = u_3,$$

and similarly for V. Let $g_t = U(t)V(t)$, $g_\infty = u_3 v_3$, and $C(Y) = U(Y)V(Y) = \sum_{j=0}^{6} c_j Y^j$. Writing

$$g = (g_0, g_1, g_{-1}, g_2, g_{-2}, g_3, g_\infty)^{\mathsf{T}}, \quad c = (c_0, \ldots, c_6)^{\mathsf{T}},$$

we obtain $g = Mc$, where each finite-point row of M is $(1, t, \ldots, t^6)$ and the infinity row is $(0, \ldots, 0, 1)$. Since $\det(M)$ is even, M is not invertible over power-of-two rings. The following proposition provides a locally computable scaled interpolation.

Proposition 3 (Scaled Toom–4 interpolation). *For the evaluation matrix M defined above, $8M^{-1}$ has entries with odd denominators. Thus, it defines a public linear transform over $\mathbb{Z}_{2^{\mu+\tau}}$ satisfying*

$$8M^{-1}g = 8c \pmod{2^{\mu+\tau}}.$$

Proof. Since g_∞ is the leading coefficient, Lagrange interpolation gives

$$C(Y) = g_\infty Y^6 + \sum_{t \in S} (g_t - g_\infty t^6) \frac{\prod_{r \in S \setminus \{t\}} (Y - r)}{\prod_{r \in S \setminus \{t\}} (t - r)}.$$

For $t = 0, 1, -1, 2, -2, 3$, the denominators are $-12, 12, 24, -24, -120, 120$, respectively. The largest power of two dividing these denominators is 8. Thus, multiplying by 8 leaves only odd denominators, which are invertible modulo $2^{\mu+\tau}$. The resulting transform therefore yields $8c$.

The parties apply the scaled interpolation to both value and MAC shares, then recombine the coefficient blocks using $Y = X^m$, summing overlapping coefficients while retaining the scaling factor. Since each leaf product is correct modulo 2^μ, its error is a multiple of 2^μ, which is preserved under linear reconstruction and negacyclic reduction. Thus, the resulting authenticated polynomial W satisfies

$$W \equiv 2^D uv \pmod{2^\mu,\ X^N + 1}, \qquad D = 3\ell.$$

For $\mu > D > 0$, each coefficient of W is divisible by 2^D, and exact division recovers the product modulo $2^{\mu-D}$. When $D = 0$, W already represents the product modulo 2^μ. In both cases, setting $\mu = b + D$ gives b correct output bits.

We next construct the authenticated exact-division protocol. Given an authenticated value $x \in 2^a\mathbb{Z}_{2^{\mu+\tau}}$, the goal is to obtain an authenticated quotient with arithmetic precision $\mu - a$, whose value and MAC shares lie in $\mathbb{Z}_{2^{\mu-a+\tau}}$. Since individual shares may not be divisible by 2^a, division must account for carries among their remainders. In the two-party setting, this is achieved by rounding one share up and the other down. For more than two parties, the parties first refresh the value shares and mask x using a uniformly random authenticated value r. They then invoke Open_ξ to obtain $d = (x - 2^a r) \bmod 2^\mu$ and reconstruct the authenticated quotient as $r + d/2^a$. The protocol is given in Fig. 6.

Correctness. In an honest execution, let $y^{(j)} = \sum_i y_i^{(j)} \bmod 2^{\mu-a+\tau}$. We show that the protocol recovers the correct quotient in its lower $\mu - a$ bits while preserving authentication,

$$y^{(j)} \equiv \frac{x^{(j)} \bmod 2^\mu}{2^a} \pmod{2^{\mu-a}}, \qquad \sum_i \eta_{y^{(j)},i} \equiv \beta y^{(j)} \pmod{2^{\mu-a+\tau}}.$$

For $p = 2$, we omit the batch index and write $x_i = 2^a q_i + r_i$, where $0 \le r_i < 2^a$. Since the shared value is divisible by 2^a, we have $r_1 + r_2 \in \{0, 2^a\}$. If $r_1 + r_2 = 0$, both shares divide exactly; otherwise, $r_1 > 0$ and rounding up the first share accounts for the carry. Thus,

$$\left\lceil \frac{x_1}{2^a} \right\rceil + \left\lfloor \frac{x_2}{2^a} \right\rfloor = \frac{x_1 + x_2}{2^a},$$

yielding the correct quotient modulo $2^{\mu-a+\tau}$. Similarly, the MAC shares satisfy $\sum_i \eta_{x,i} \equiv \beta x \pmod{2^{\mu+\tau}}$. Since βx is divisible by 2^a, the same rounding argument gives $\sum_i \eta_{y,i} \equiv \beta y \pmod{2^{\mu-a+\tau}}$, preserving authentication.

For $p > 2$, zero-sum refreshing preserves both the shared value and its MAC relation. The masked opening returns $d = (x - 2^a r) \bmod 2^\mu$. Since x is divisible by 2^a, so is d. The high-bit mask used by Open_ξ does not affect the lower μ bits. Consequently, the reconstructed quotient satisfies

$$y = r + \frac{d}{2^a} \equiv \frac{x \bmod 2^\mu}{2^a} \pmod{2^{\mu-a}}.$$

By linearity of authentication, the output MAC shares satisfy

$$\sum_i \eta_{y,i} \equiv \beta r + \beta \frac{d}{2^a} = \beta y \pmod{2^{\mu-a+\tau}}.$$

Protocol Π_{Div}

Input. $(\mu, a, (\llbracket x^{(j)} \rrbracket_{\beta,\mu})_{j=1}^{T})$, where $0 < a < \mu$, and $x^{(j)}$ belongs to $2^a \mathbb{Z}_{2^{\mu+\tau}}$.

Output. Authenticated sharings $(\llbracket y^{(j)} \rrbracket_{\beta,\mu-a})_{j=1}^{T}$ whose low $\mu - a$ bits satisfy

$$y^{(j)} \equiv \frac{x^{(j)} \bmod 2^{\mu}}{2^a} \pmod{2^{\mu-a}}.$$

1. If $p = 2$, the parties take the nonnegative representatives of their value shares and MAC shares modulo $2^{\mu+\tau}$. For each j, compute
$$y_1^{(j)} = \left\lceil x_1^{(j)} / 2^a \right\rceil, \quad \eta_{y^{(j)},1} = \left\lceil \eta_{x^{(j)},1} / 2^a \right\rceil,$$
$$y_2^{(j)} = \left\lfloor x_2^{(j)} / 2^a \right\rfloor, \quad \eta_{y^{(j)},2} = \left\lfloor \eta_{x^{(j)},2} / 2^a \right\rfloor.$$
Reduce the resulting shares modulo $2^{\mu-a+\tau}$ and proceed to Step 3.
2. If $p > 2$, perform masked opening and reconstruction:
   (a) Obtain independent uniform authenticated masks $(\llbracket r^{(j)} \rrbracket_{\beta,\mu})_{j=1}^{T}$. Invoke $\mathsf{Zero}(\mu + \tau, T + 1)$ to generate zero-sum masks $(\xi_i^x[1], \ldots, \xi_i^x[T], \xi_i)$. Each party P_i refreshes its value shares as
   $$x_i^{(j)} \leftarrow x_i^{(j)} + \xi_i^x[j] \pmod{2^{\mu+\tau}},$$
   leaving the MAC shares unchanged.
   (b) Invoke Open_{ξ} using the masks ξ_i to open $(\llbracket x^{(j)} - 2^a r^{(j)} \rrbracket_{\beta,\mu})_{j=1}^{T}$, obtaining $d^{(j)} \in \mathbb{Z}_{2^\mu}$. Abort if any $d^{(j)} \not\equiv 0 \pmod{2^a}$.
   (c) Compute $t^{(j)} = d^{(j)} / 2^a$. Reduce the value and MAC shares of $r^{(j)}$ modulo $2^{\mu-a+\tau}$. Each party P_i computes
   $$y_i^{(j)} = r_i^{(j)} + \mathbf{1}_{i=1} t^{(j)}, \qquad \eta_{y^{(j)},i} = \eta_{r^{(j)},i} + t^{(j)} \beta_i.$$
3. Each P_i returns $(y_i^{(j)}, \eta_{y^{(j)},i})_{j=1}^{T}$ as its shares of the authenticated quotients.

Fig. 6. Exact division of authenticated values

Finally, we consider the output distribution. For fixed x, the uniform mask r makes $d = (x - 2^a r) \bmod 2^\mu$ uniform over $2^a\mathbb{Z}_{2^\mu}$. Conditioned on d, the lower $\mu - a$ bits of r are fixed, while its remaining τ bits modulo $2^{\mu-a+\tau}$ remain uniform. Therefore, $y = r + d/2^a$ is uniform over the 2^τ representations whose lower $\mu - a$ bits equal the correct quotient.

Theorem 5. *The protocol Π_{Div} implements $\mathcal{F}_{\mathrm{Div}}$ in the $\big(\mathcal{F}^{\mathbb{Z}}_{\mathrm{Auth}}, \mathcal{F}^{\mathbb{Z}}_{\mathrm{Triple}}, \mathcal{F}_{\mathrm{Rand}}, \mathcal{F}_{\mathrm{COMMIT}}\big)$-hybrid model, with statistical distinguishing advantage at most $(\tau + 1)2^{-\tau}$.*

The proof of Theorem 5 is given in Appendix B.3.

For a working precision $D < \mu \le \mu_{\max}$, we combine the linear reconstruction with exact division to obtain the authenticated ring multiplication protocol in Fig. 7. All leaf multiplications can be batched in one call. When $D > 0$, the parties apply exact division to the final N coefficients after negacyclic reduction at the root.

Protocol Π_{RingMul}

Input. $\mu, [\![u]\!]_{\beta,\mu}, [\![v]\!]_{\beta,\mu}$.
Output. $[\![y]\!]_{\beta,\mu-D}$, whose low $\mu - D$ bits equal $uv \bmod (X^N + 1)$.

1. Each party P_i constructs a recursion tree with ℓ Toom–4 levels followed by $n - 2\ell$ Karatsuba levels. At each Karatsuba node, the input blocks u_0, u_1 and v_0, v_1 form three pairs (u_0, v_0), (u_0+u_1, v_0+v_1), (u_1, v_1). At each Toom–4 node, four input blocks form seven pairs $(U(t), V(t))$ for $t \in \{0, 1, -1, 2, -2, 3, \infty\}$, where $U(t) = \sum_{j=0}^{3} t^j u_j$, $U(\infty) = u_3$, and similarly for V. All leaf coefficient products are computed in a single batched invocation of $\Pi^{\mathbb{Z}}_{\mathrm{Mult}}$ at precision μ.
2. The parties reconstruct the product bottom-up. At each Karatsuba node, the three subproducts g_0, g_1, g_2 are combined as $g_0 + X^m(g_1 - g_0 - g_2) + X^{2m} g_2$. At each Toom–4 node, the seven subproducts form g in the order $(0, 1, -1, 2, -2, 3, \infty)$, and the scaled reconstruction is
$$\sum_{j=0}^{6} X^{jm} (8M^{-1} g)_j,$$
as defined in Proposition 3. Here, m denotes the block length, and scaling factors are preserved throughout the recursion.
3. The parties apply negacyclic reduction modulo $X^N + 1$ to obtain $[\![W]\!]_{\beta,\mu}$. If $\ell = 0$, they return this sharing. Otherwise, they invoke
$$\Pi_{\mathrm{Div}}\big(\mu, D, ([\![W[t]]\!]_{\beta,\mu})_{t=0}^{N-1}\big), \quad D = 3\ell,$$
and assemble the resulting coefficients into $[\![y]\!]_{\beta,\mu-D}$.

Fig. 7. Authenticated polynomial multiplication

Correctness. For inputs satisfying the preconditions, an honest execution of the protocol in Fig. 7 returns $[\![y]\!]_{\beta,\mu-D}$ satisfying

$$y \equiv uv \pmod{2^{\mu-D},\, X^N + 1}, \qquad \sum_i \eta_{y[t],i} \equiv \beta y[t] \pmod{2^{\mu-D+\tau}}$$

for every $0 \le t < N$.

Each leaf result agrees with its coefficient product in the low μ bits. At each internal node, Karatsuba reconstruction preserves the scaling factor, while Toom–4 introduces an additional factor of eight by Proposition 3. By induction, a subtree with h Toom–4 levels computes 8^h times its polynomial product modulo 2^μ. Since all reconstruction operations are linear, they preserve both the error divisibility by 2^μ and the MAC authentication relation.

At the root, $h = \ell$ and $D = 3\ell$. After negacyclic reduction, each coefficient satisfies

$$W[t] \equiv 2^D (uv \bmod (X^N + 1))[t] + 2^\mu E[t] \pmod{2^{\mu+\tau}}$$

for some error polynomial E. Since $\mu > D$, every coefficient of W is divisible by 2^D. For $D > 0$, the exact-division protocol in Fig. 6 recovers an authenticated quotient with $\mu - D$ correct bits; otherwise, the parties set $y = W$. Thus,

$$y[t] \equiv (uv \bmod (X^N + 1))[t] \pmod{2^{\mu-D}}.$$

The authentication relation follows from the linearity of reconstruction and the correctness of authenticated exact division.

Theorem 6. *The protocol* Π_{RingMul} *implements* $\mathcal{F}_{\text{RingMul}}$ *in the* $\left(\mathcal{F}^{\mathbb{Z}}_{\text{Auth}}, \mathcal{F}^{\mathbb{Z}}_{\text{Triple}}, \mathcal{F}_{\text{Rand}}, \mathcal{F}_{\text{COMMIT}}\right)$*-hybrid model, with statistical distinguishing advantage at most* $(\tau + 1)2^{-\tau}$.

The proof of Theorem 6 is given in Appendix B.4.

5.3 Creating the MACs for R_k

To create a MAC for R_k, the parties compute shares of $\gamma_z = \alpha z$. Let $\alpha = \sum_{j=0}^{N-1} \alpha[j] X^j$ and $z = \sum_{j=0}^{N-1} z[j] X^j$. By negacyclic reduction, the t-th coefficient of the ring MAC is

$$\gamma_z[t] = \sum_{j+h=t} \alpha[j] z[h] - \sum_{j+h=t+N} \alpha[j] z[h] \pmod{2^q}, \qquad 0 \le t < N.$$

Thus, the MAC is the negacyclic product of α and z and can be generated by invoking Π_{RingMul}.

During session initialization, the parties invoke the Init command of the integer-ring triple functionality to obtain their shares of β. They then use N independent random components from fresh integer-ring triples to construct $[\![\tilde{\alpha}]\!]_{\beta,\mu_{\max}}$. The parties set $\alpha = \tilde{\alpha} \bmod 2^q$, refresh the value shares of its lower q bits, and provide the resulting shares α_i to the online phase. The coefficientwise authenticated sharing of $\tilde{\alpha}$ is retained throughout the session for subsequent MAC generation.

To compute a MAC modulo 2^q, Π_{Auth} invokes Π_{RingMul} at working precision $\mu = q + D$. The lower q bits of the resulting product yield $\gamma_z = \alpha z$ in R. Since ring-triple generation requires an initial polynomial multiplication at precision $q + 2D$, the underlying integer-ring triples must satisfy $\mu_{\max} \ge \max\{\tau, q + 2D\}$.

Theorem 7. *The protocol* Π_{Auth} *perfectly implements* $\mathcal{F}^R_{\text{Auth}}$ *in the* $\left(\mathcal{F}_{\text{RingMul}}, \mathcal{F}_{\text{COMMIT}}\right)$*-hybrid model. Instantiating* $\mathcal{F}_{\text{RingMul}}$ *with* Π_{RingMul} *yields an implementation in the* $\left(\mathcal{F}^{\mathbb{Z}}_{\text{Auth}}, \mathcal{F}^{\mathbb{Z}}_{\text{Triple}}, \mathcal{F}_{\text{Rand}}, \mathcal{F}_{\text{COMMIT}}\right)$*-hybrid model with statistical distinguishing advantage at most* $(\tau + 1)2^{-\tau}$.

The proof of Theorem 7 is given in Appendix B.5.

Protocol Π_{Auth}

Input. $[\![\tilde{z}]\!]_{\beta,q+D}$.
Output. $[\![z]\!]_\alpha$, where $z = \sum_{t=0}^{N-1} (\tilde{z}[t] \bmod 2^q) X^t \in R$.

1. Reduce the sharing of $\tilde{\alpha}$ modulo $2^{q+D+\tau}$. Invoke Π_{RingMul} on
$$\big(q + D, [\![\tilde{\alpha}]\!]_{\beta,q+D}, [\![\tilde{z}]\!]_{\beta,q+D}\big)$$
to obtain $[\![\tilde{\gamma}]\!]_{\beta,q}$.
2. Invoke $\mathsf{Zero}(q, 2N)$ to obtain independent zero-sum masks $(\xi_i^z)_{i=1}^p$ and $(\xi_i^\gamma)_{i=1}^p$ for refreshing the value and MAC shares, respectively.
3. Each P_i computes
$$z_i = \sum_{t=0}^{N-1} \big((\tilde{z}_i[t] + \xi_i^z[t]) \bmod 2^q\big) X^t, \gamma_{z,i} = \sum_{t=0}^{N-1} \big((\tilde{\gamma}_i[t] + \xi_i^\gamma[t]) \bmod 2^q\big) X^t,$$
and returns these two elements of R as its shares.

Fig. 8. Generating authenticated sharings for R_k

5.4 Creating Multiplication Triples for R_k

An authenticated multiplication triple consists of $[\![a]\!]_\alpha$, $[\![b]\!]_\alpha$, and $[\![c]\!]_\alpha$, with $c = ab$ in R. We obtain independent random coefficients from integer-ring triples to form $\tilde{a}$ and $\tilde{b}$, compute $\tilde{c}$ using Π_{RingMul}, and authenticate all three polynomials using Π_{Auth}.

Computing $\tilde{c}$ and then its MAC requires two successive multiplications. Since each invocation of Π_{RingMul} reduces the arithmetic precision by D bits, we use the following input and output precisions:

$$\tilde{a}\tilde{b} : q + 2D \longrightarrow q + D. \qquad \tilde{\alpha}\tilde{a},\ \tilde{\alpha}\tilde{b},\ \tilde{\alpha}\tilde{c} : q + D \longrightarrow q.$$

The first multiplication produces $[\![\tilde{c}]\!]_{\beta,q+D}$. Reducing the retained sharings of $\tilde{a}$ and $\tilde{b}$ modulo $2^{q+D+\tau}$ puts all three inputs at the same precision, allowing their MACs to be generated in one batch.

Protocol Π_{Triple}

Output. $([\![a]\!]_\alpha, [\![b]\!]_\alpha, [\![c]\!]_\alpha)$, where a, b are independent and uniform in R, and $c = ab$ in R. Upon receiving a (Triple) request from each party, execute the following steps:

1. Obtain $2N$ independent authenticated random coefficients from fresh integer-ring triples to construct $[\![\tilde{a}]\!]_{\beta,\mu_{\max}}$ and $[\![\tilde{b}]\!]_{\beta,\mu_{\max}}$.
2. Reduce copies of both sharings modulo $2^{q+2D+\tau}$ and invoke Π_{RingMul} with
$$\big(q + 2D, [\![\tilde{a}]\!]_{\beta,q+2D}, [\![\tilde{b}]\!]_{\beta,q+2D}\big)$$
to obtain $[\![\tilde{c}]\!]_{\beta,q+D}$.
3. Reduce copies of the sharings of $\tilde{a}$ and $\tilde{b}$ modulo $2^{q+D+\tau}$. Invoke Π_{Auth} in parallel on $[\![\tilde{a}]\!]_{\beta,q+D}$, $[\![\tilde{b}]\!]_{\beta,q+D}$, and $[\![\tilde{c}]\!]_{\beta,q+D}$ to obtain their authenticated shares. Fix the order of batches $1, \ldots, j$ and the opened values in each before sampling fresh random challenges for batch j. Reveal the check shares for batch j only after batches $1, \ldots, j-1$ have all passed their checks.
4. Upon successful completion, P_i outputs
$$\big((a_i, \gamma_{a,i}), (b_i, \gamma_{b,i}), (c_i, \gamma_{c,i})\big)$$
as its local shares of the ring triple $([\![a]\!]_\alpha, [\![b]\!]_\alpha, [\![c]\!]_\alpha)$.

Fig. 9. Triple generation protocol

In an honest execution, the correctness of Π_{RingMul} ensures that reduction modulo 2^q yields $c = ab$ in R, while Π_{Auth} provides the corresponding MAC shares. The independent uniform sampling of the input coefficients, followed by reduction modulo 2^q, ensures that a and b are independent and uniform in R.

Theorem 8. *The protocol Π_{Triple} perfectly implements $\mathcal{F}^R_{\text{Triple}}$ in the $\left(\mathcal{F}^{\mathbb{Z}}_{\text{Triple}}, \mathcal{F}_{\text{RingMul}}, \mathcal{F}^R_{\text{Auth}}\right)$-hybrid model. Instantiating $\mathcal{F}_{\text{RingMul}}$ and $\mathcal{F}^R_{\text{Auth}}$ with Π_{RingMul} and Π_{Auth}, respectively, yields an implementation in the $\left(\mathcal{F}^{\mathbb{Z}}_{\text{Auth}}, \mathcal{F}^{\mathbb{Z}}_{\text{Triple}}, \mathcal{F}_{\text{Rand}}, \mathcal{F}_{\text{COMMIT}}\right)$-hybrid model with statistical distinguishing advantage at most $(\tau + 1)2^{-\tau}$.*

The proof of Theorem 8 is given in Appendix B.6.

5.5 Input and output masks

Input masks. To answer a Random request from Section 4, obtain N independent authenticated random coefficients from integer-ring triples to form $[\![\tilde{r}]\!]_{\beta,\mu_{\max}}$. Reduce a copy to precision $q + D$ and invoke Π_{Auth} to obtain $[\![r]\!]_\alpha$. For an input mask, Π_{Private} uses the retained sharing of $\tilde{r}$ to deliver r privately to P_ℓ and verify consistency. Choose $q \le \mu \le \mu_{\max}$, $\nu \ge 1$, and $\nu(\mu - q + 1) \ge \lambda$, where $2^{-\lambda}$ is the target error bound for an honest recipient's consistency check.

Protocol Π_{Private}

Input. Recipient P_ℓ, public parameters μ, ν, and the sharings $[\![\tilde{r}]\!]_{\beta,\mu_{\max}}$ and $[\![r]\!]_\alpha$, where $r = \tilde{r} \bmod 2^q$.

Output. Deliver r privately to P_ℓ and release $[\![r]\!]_\alpha$ for use, or abort.

1. Reduce a copy of the sharing of $\tilde{r}$ modulo $2^{\mu+\tau}$ and set $[\![x]\!]_{\beta,\mu} = [\![\tilde{r}]\!]_{\beta,\mu}$. If $\mu > q$, randomize the high bits by adding $2^q[\![u]\!]_{\beta,\mu}$, where u is an independent random polynomial with authenticated coefficients.
2. Obtain ν independent authenticated random challenges $[\![b^{(j)}]\!]_{\beta,\mu}$ from fresh integer-ring triples. Use $\Pi^{\mathbb{Z}}_{\text{Mult}}$ to compute $[\![c^{(j)}[t]]\!]_{\beta,\mu}$ satisfying $c^{(j)}[t] \equiv b^{(j)}x[t] \pmod{2^\mu}$, for $1 \le j \le \nu$ and $0 \le t < N$.
3. Use $\mathsf{Zero}(\mu, (\nu+1)N)$ to independently refresh the value shares of $x, c^{(1)}, \ldots, c^{(\nu)}$ modulo 2^μ. Each $P_i, i \ne \ell$, sends its refreshed shares to P_ℓ. After receiving all shares, P_ℓ broadcasts an acknowledgment. Only then do the parties open $([\![b^{(j)}]\!]_{\beta,\mu})_{j=1}^{\nu}$ using the authenticated Open procedure.
4. From the received shares and its own, P_ℓ reconstructs x' and $c^{(j)\prime}$, and checks
$$c^{(j)\prime}[t] \equiv b^{(j)}x'[t] \pmod{2^\mu} \qquad \text{for all } j, t.$$
On success, it returns $x' \bmod 2^q$ and broadcasts completion, releasing $[\![r]\!]_\alpha$ for use; otherwise, abort.

Fig. 10. Private Opening of Input Masks

Whole-ring output masks. An output mask is uniform in $I = \ker(\pi_k) = 2^k R$, randomizing the high s bits without changing the low k output bits. Sample $[\![\tilde{u}]\!]_{\beta,\mu_{\max}}$ with independent uniform coefficients from integer-ring triples. Reduce this sharing and

a copy of the sharing of $\tilde{\alpha}$ modulo $2^{s+D+\tau}$, and invoke Π_{RingMul} at precision $s + D$. Let u and v be the coefficientwise reductions of $\tilde{u}$ and the returned product modulo 2^s. Reducing their value shares modulo 2^s and multiplying by 2^k yields shares of

$$r = 2^k u, \qquad \gamma_r = 2^k v = \alpha r \quad \text{in } R.$$

Independently refresh the two sets of shares using $\mathsf{Zero}(q, 2N)$ to obtain $[\![r]\!]_\alpha$. Uniformity of the coefficients of u modulo 2^s ensures that r is uniform in I.

Selective output masks. For a convolution block, let $J \subseteq \{0, \ldots, N-1\}$ be the public set of output positions. Obtain N independent authenticated random coefficients from fresh integer-ring triples and reduce their sharings to precision $q + D$, as for an input mask. At each $t \in J$, multiply the coefficient's value shares and integer-ring MAC shares by 2^k. Use the sampled shares directly at every $t \notin J$. Invoke Π_{Auth} on the resulting coefficientwise authenticated polynomial to obtain $[\![r_J]\!]_\alpha$ with ring MAC αr_J. The coefficients $r_J[t]$ are independent and uniform in $2^k\mathbb{Z}_{2^q}$ for $t \in J$ and in $\mathbb{Z}_{2^q}$ for $t \notin J$.

6 Efficiency Analysis and Evaluation

We evaluate the offline communication and online performance of CycloSPDZ, using one-dimensional long-kernel and two-dimensional depthwise convolutions as applications.

6.1 Evaluation Setup

Implementation and evaluation setup. We implement the online protocol and our Toom–4/Karatsuba hybrid scheme of CycloSPDZ[3] and all baselines in a common C++ framework using the networking and serialization components of MP-SPDZ [16]. We evaluate all protocols with two parties ($p = 2$) on an Intel Core Ultra X7 358H running Ubuntu 24.04.4, with code compiled using g++ at -O3 optimization. The two parties run as separate single-threaded processes. We use netem to emulate LAN (1 ms RTT, 1 Gbit/s) and WAN (100 ms RTT, 100 Mbit/s) conditions with shared bidirectional bandwidth in an isolated Linux network namespace. Each benchmark includes one warm-up run and three measured runs, reporting the median of the slower party's execution time. Offline communication is evaluated analytically and amortized over large batches. Online timing starts with authenticated inputs and preprocessing material already available from a trusted dealer, and ends after output authentication and reconstruction. Communication is summed over both parties.

Workloads. All workloads use inputs and weights in $[0, 255]$, batch size and stride 1, and exact 32-bit accumulation ($k = 32$). We evaluate causal one-dimensional depthwise convolutions with $W = 8192$ and $F \in \{256, 512, 1024, 2048, 4096, 8192\}$, based on the Hyena-s-8K operator in FlashFFTConv [29]. We also evaluate seven two-dimensional depthwise layers from MobileNetV1 (M1), MobileNetV2 (M2), and ConvNeXt V2 (CN2) [30,31,32], using same zero padding and 3×3 or 7×7 kernels.

[3] `https://anonymous.4open.science/r/cycloSPDZ-45C0/`

We compare CycloSPDZ with Convolutions in Overdrive (CoO) [17], SPD$\mathbb{Z}_{2^k}$ [9], MatriGear (MG) [13], and Matrix Rings (MR) [14]. CoO evaluates convolutions using convolution triples. For SPD$\mathbb{Z}_{2^k}$, we implement convolution using either Karatsuba or our hybrid Toom–4/Karatsuba recursion. For MG and MR, we use CoO's image-to-column (im2col) transformation and square-matrix blocking, omitting products that are publicly known to be zero. We evaluate both the function-independent (FI) and function-dependent (FD) preprocessing interfaces of MR. CycloSPDZ computes each convolution output block using one multiplication of the encoded input and kernel in R_k, with authentication over R. For the block's public set J of output positions, the parties add a fresh authenticated selective mask r_J from Section 5.5 to the product. They open the entire masked ring element and verify it with a ring MAC check. Upon acceptance they return only the coefficients in J modulo 2^k. Each coefficient outside J is uniformly masked over $\mathbb{Z}_{2^q}$.

We select the parameters to satisfy a common conditional authentication-error bound of $\varepsilon_{\text{auth}} < 2^{-57}$ per convolution, assuming correct preprocessing, ideal commitments, and public randomness. Table 1 summarizes the message spaces and corresponding authentication-error bounds.

Table 1: Evaluation parameters

Scheme	**Message space**	**Authentication-error bound**
CoO 64-bit [17]	$\mathbb{F}_{p_{64}}$	$4/p_{64} < 2^{-57}$
CoO 128-bit [17]	$\mathbb{F}_{p_{128}}$	$4/p_{128} < 2^{-57}$
MG [13]	$\mathbb{F}_{p_{64}}$	$4/p_{64} < 2^{-57}$
MR FI [14]	$\mathrm{Mat}_h(\mathbb{F}_{p_{64}})$	$(o+2)/p_{64} < 2^{-57}$
MR FD [14]	$\mathrm{Mat}_h(\mathbb{F}_{p_{64}})$	$4/p_{64} < 2^{-57}$
SPD$\mathbb{Z}_{2^k}$ [9]	$\mathbb{Z}_{2^k}$	$132 \cdot 2^{-65} < 2^{-57}$
Ours	R_k	$(N+4)2^{-(N+2)} < 2^{-57}$

$k = 32$, $p_{64} = 2^{64} - 59$, and p_{128} is a 128-bit prime. Here h is the matrix dimension, and $o \le 52$ is the number of MR FI output matrices in the evaluated configurations.

The adapted field baselines use $p_{64} = 2^{64}-59$. Two batched checks yield an authentication-error bound of $4/p_{64}$ for CoO 64-bit, MG, and MR FD, whereas MR FI uses one batched check and o individual output checks, yielding $(o+2)/p_{64}$. MG and MR FD use matrix dimensions $h = 128$ and $h = 64$, respectively. For comparison, we also include CoO's published 128-bit-field configuration, which uses a 40-bit statistical security parameter. SPD$\mathbb{Z}_{2^k}$ uses $s_Z = 65$, with three additional working bits for each of the ℓ Toom–4 levels along a multiplication path. CycloSPDZ uses $k = 32$, $s_R = 1$, and $N \ge 64$, giving $q = k + s_R = 33$ bits per coefficient. By Theorem 2, its session-wide forgery bound is $(N+4)2^{-(N+2)}$ for any finite batch limit fixed before the session begins.

6.2 Preprocessing Evaluation

Communication cost of triple generation. We estimate the amortized communication cost of generating one authenticated triple over R_k. The backend model is based on the native Multipars configuration [15], with 128-bit data, 64-bit authentication redundancy, and a 57-bit statistical security parameter. We adapt this configuration to supply scalar material of width $\mu \in \{64, 128\}$. Each native backend instance costs 18,900 bits of communication, including modulus switching.

Let $q = k + 1$. Recursive multiplication with ℓ Toom–4 levels and $\log_2 N - 2\ell$ Karatsuba levels requires $n_{\text{mul}} = 7^{\ell} 3^{\log_2 N - 2\ell}$ scalar multiplications. The backend source width must satisfy $\mu \geq \max\{64, q + 6\ell\}$ to accommodate products and authentication tags. The required number of native backend instances is $a_T = 6n_{\text{mul}} + N$ for $\mu = 64$ and $a_T = 8n_{\text{mul}} + 2N$ for $\mu = 128$. The total amortized communication, in bits, is $C_T = 18{,}900a_T + 4n_{\text{mul}}(4q + 15\ell + 2560) + 12Nq$.

Table 2 compares Schoolbook ($n_{\text{mul}} = N^2, \ell = 0$), Karatsuba ($\ell = 0$), maximum-depth Toom–4 ($\ell = \lfloor(\log_2 N)/2\rfloor$), and our Hybrid scheme, which minimizes C_T over feasible choices of ℓ and μ.

Table 2: Communication of triple generation (MB)

(k, N)	$(48, 64)$	$(40, 256)$	$(32, 1024)$	$(32, 4096)$	$(32, 8192)$
Schoolbook	63.861	1,018.853	16,277.418	260,409.049	1,041,616.437
Karatsuba	11.494	102.559	918.969	8,258.375	24,765.246
Maximum-depth Toom–4	7.270	49.946	263.962	2,406.772	7,200.760
Ours (Hybrid)	**7.021**	**48.653**	**263.962**	**2,363.305**	**7,080.035**

Preprocessing communication for convolutions. We evaluate single-channel 1D convolution with $W = 8192$, $C = 1$, and $k = 32$. For ring dimension N and kernel length F, the number of output blocks is $T = \lceil W/(N - F + 1)\rceil$. Each block requires one authenticated ring triple and one selective output mask. For a full block, $J = \{F - 1, \ldots, N - 1\}$. If the last block is partial, J contains only the positions in this range corresponding to the remaining outputs. In increasing order of F in Table 3, the selected (N, T) pairs are $(1024, 11)$, $(1024, 16)$, $(2048, 8)$, $(4096, 4)$, $(8192, 2)$, and $(16384, 1)$.

The total amortized communication is

$$C_{\text{Pre}} = 18{,}900a + T(D_T + D_J) + 2(T + 1)(Nq + 512),$$

where a is the amortized number of native backend instances. The costs D_T and D_J count the bits used to construct one authenticated ring triple and one selective mask r_J (Section 5.5), respectively, excluding backend material generation. The final term accounts for pregenerated verification payloads. Table 3 reports the protocol overhead $C_{\text{Pre}} - 18{,}900a$ and total communication C_{Pre}. Comparisons with the baselines use total communication.

Table 3: Preprocessing communication for Conv1D (GB)

F	**256**	**512**	**1024**	**2048**	**4096**	**8192**
MG [13]	3.454	6.800	13.165	24.589	42.214	56.576
MR FI [14]	3.840	6.336	11.824	22.170	38.428	53.208
CoO 128-bit [17]	0.02334	0.02806	0.02815	0.02831	0.02851	0.03735
Ours (Hybrid only)	**0.320**	**0.466**	**0.698**	**1.001**	**1.501**	**1.439**
Ours (total)	**3.636**	**5.289**	**7.903**	**11.303**	**16.925**	**21.011**

6.3 Online Evaluation

Online communication. Each coefficient is serialized in $\lceil q/8 \rceil = 5$ bytes, so a ring element occupies $A = N\lceil q/8 \rceil = 5N$ bytes. Let $c = 64$ bytes be the overhead per commit-and-open operation per party. A MAC check with a challenges has communication cost

$$Q_R(a) = 2\left(\left\lceil \frac{aN}{8} \right\rceil + A + 2c\right).$$

For G multiplications, the parties check all $2G$ opened masked differences in one batch. They then run one separate check for each of the M output blocks. The total communication in bytes is

$$C_R = 4G(A + c) + Q_R(2G) + 2M(A + c) + MQ_R(1).$$

For one-dimensional convolution, $G = M = CT$. For two-dimensional convolution, the number of output blocks per channel is 15 for M1-112 and 1 for the remaining layers.

Table 4 presents the one-dimensional communication. CycloSPDZ reduces communication by 36.37%–55.28% relative to CoO 128-bit. Avoiding scalar decomposition and window expansion reduces communication by more than one order of magnitude relative to SPD$\mathbb{Z}_{2^k}$ Hybrid and more than two relative to MG and MR FI.

Table 4: Online communication for Conv1D (MB)

F	**256**	**512**	**1024**	**2048**	**4096**	**8192**
MG [13]	133.432	262.407	507.773	948.175	1,627.653	2,181.301
MR FI [14]	134.359	268.578	537.020	939.674	1,744.981	3,355.595
MR FD [14]	17.179	17.441	17.966	19.015	21.112	25.308
CoO 128-bit [17]	1.066	1.082	1.115	1.180	1.311	1.573
CoO 64-bit [17]	0.533	0.541	0.558	0.590	0.656	0.787
SPD$\mathbb{Z}_{2^k}$ Hybrid	7.163	10.563	12.262	18.212	22.145	33.027
Ours	**0.477**	**0.688**	**0.694**	**0.711**	**0.751**	**0.832**

In column order, MR FI uses $h = 256, 512, 1024, 1024, 1024, 1024$.

Table 5: Online communication for Conv2D (MB)

layer shape	**M1-112** $112^2\times32$	**M1-56** $56^2\times128$	**M1-28** $28^2\times256$	**M1-14** $14^2\times512$	**M2-56** $56^2\times144$	**M2-28** $28^2\times192$	**CN2-56** $56^2\times96$
MG [13]	321.127	248.775	138.544	78.709	302.810	107.578	979.567
MR FI [14]	359.878	183.568	95.504	58.734	229.500	71.628	734.104
MR FD [14]	83.845	71.848	39.662	27.305	85.696	29.747	253.788
CoO 128-bit [17]	51.399	51.455	25.838	13.141	57.886	19.379	38.837
CoO 64-bit [17]	25.700	25.728	12.919	6.571	28.943	9.690	19.419
SPD$\mathbb{Z}_{2^k}$ Hybrid	93.278	93.278	46.639	29.161	104.937	34.980	171.715
Ours	**20.347**	**21.488**	**10.857**	**5.672**	**24.169**	**8.145**	**16.126**

In column order, MR FI uses $h = 256, 128, 64, 256, 128, 64, 128$, and CycloSPDZ uses $N = 1024, 4096, 1024, 256, 4096, 1024, 4096$.

Table 5 presents two-dimensional communication. CycloSPDZ has the lowest communication cost among the evaluated schemes on all seven layers, with reductions of 13.68%–20.83% relative to CoO 64-bit and 56.84%–60.41% relative to CoO 128-bit. The reductions relative to CoO are partly due to CycloSPDZ's smaller coefficient encodings and binary MAC challenges.

Online runtime. As shown in Table 6, for one-dimensional long-kernel convolution, CycloSPDZ achieves a 3.94–13.23× speedup over CoO 64-bit in the LAN setting. Polynomial multiplication reduces local computation relative to direct convolution, which requires $WF - F(F-1)/2$ multiply-accumulate operations. In the WAN setting, CycloSPDZ achieves speedups of 1.37× and 1.94× for $F = 2048, 4096$, respectively. Compared with SPD$\mathbb{Z}_{2^k}$ Hybrid scheme, CycloSPDZ achieves 5.21–6.30× and 1.93–3.31× speedups in the LAN and WAN settings, respectively. Relative to MR FD, CycloSPDZ achieves 7.53–48.72× speedups in the WAN settings.

As shown in Table 7, for two-dimensional depthwise convolution, CoO 64-bit has lower LAN runtime on most layers, where direct evaluation has low arithmetic cost for 3×3 kernels. Compared with SPD$\mathbb{Z}_{2^k}$ Hybrid scheme, CycloSPDZ achieves 1.16–3.47× and 2.33–5.51× speedups in the LAN and WAN settings, respectively. Relative to MR FD, CycloSPDZ achieves 3.40–15.84× speedups in the WAN settings.

Table 6: Online runtime for Conv1D (s)

	Scheme	$F = 256$	$F = 512$	$F = 1024$	$F = 2048$	$F = 4096$
LAN	CoO 64-bit [17]	0.096	0.160	0.295	0.630	0.989
	MG [13]	11.139	21.496	41.651	77.977	132.150
	MR FI [14]	15.073	57.546	231.022	462.575	921.386
	MR FD [14]	3.774	7.153	13.528	24.982	43.239
	SPD$\mathbb{Z}_{2^k}$ Karatsuba	0.327	0.469	0.703	1.080	1.868
	SPD$\mathbb{Z}_{2^k}$ Hybrid	0.139	0.191	0.228	0.321	0.390
	Ours	**0.024**	**0.033**	**0.040**	**0.051**	**0.075**
WAN	CoO 64-bit [17]	0.746	0.807	0.927	1.267	1.799
	MG [13]	21.498	41.255	79.619	146.616	252.327
	MR FI [14]	25.691	77.642	268.759	527.893	1048.554
	MR FD [14]	5.956	9.341	15.699	27.339	45.263
	SPD$\mathbb{Z}_{2^k}$ Karatsuba	2.598	3.476	4.612	6.592	9.366
	SPD$\mathbb{Z}_{2^k}$ Hybrid	1.528	1.862	2.051	2.698	3.072
	Ours	**0.791**	**0.904**	**0.908**	**0.922**	**0.929**

SPD$\mathbb{Z}_{2^k}$ Hybrid uses Toom–4/Karatsuba depths $(\ell, r) = (4,0), (4,1), (5,0), (5,1), (6,0)$, in column order.

Table 7: Online runtime for Conv2D (s)

	Scheme	M1-112	M1-56	M1-28	M1-14	M2-56	M2-28	CN2-56
LAN	CoO 64-bit [17]	0.564	0.557	0.287	0.148	0.627	0.224	1.030
	MG [13]	25.082	19.333	10.751	6.102	23.562	8.323	79.662
	MR FI [14]	46.065	14.225	4.231	8.277	17.435	3.177	58.269
	MR FD [14]	7.994	7.041	3.756	2.337	8.359	2.812	28.866
	SPD$\mathbb{Z}_{2^k}$ Karatsuba	2.925	3.247	1.349	0.775	3.425	1.006	6.093
	SPD$\mathbb{Z}_{2^k}$ Hybrid	1.519	1.516	0.771	0.503	1.698	0.593	3.068
	Ours	**0.710**	**1.312**	**0.390**	**0.145**	**1.450**	**0.293**	**1.040**
WAN	CoO 64-bit [17]	3.355	3.410	1.942	1.335	3.670	1.729	3.242
	MG [13]	49.392	38.245	21.671	12.736	46.634	17.129	150.572
	MR FI [14]	72.759	28.511	12.282	13.492	35.021	9.481	111.880
	MR FD [14]	15.078	13.458	7.748	5.373	15.686	6.028	47.968
	SPD$\mathbb{Z}_{2^k}$ Karatsuba	14.921	14.779	7.909	4.812	16.649	6.132	26.699
	SPD$\mathbb{Z}_{2^k}$ Hybrid	9.367	9.221	5.094	3.581	10.376	4.120	16.678
	Ours	**3.209**	**3.805**	**2.164**	**1.453**	**4.123**	**1.772**	**3.028**

7 Conclusion

We introduced CycloSPDZ, an actively secure dishonest-majority MPC protocol over power-of-two cyclotomic rings R_k. Our native information-theoretic homomorphic MAC over R_k admits a tight forgery bound for batched verification and allows us to achieve the target statistical security with only a small increase in coefficient bit length. We also developed an efficient preprocessing protocol from authenticated integer-ring triples. By combining scaled interpolation with batched exact division, our protocol supports hybrid Toom–4/Karatsuba multiplication over power-of-two moduli while preserving authentication. Furthermore, we proved that CycloSPDZ achieves universally composable (UC) security with abort against static malicious adversaries in the preprocessing model. Finally, we demonstrated the practical benefits of CycloSPDZ by implementing its online phase and evaluating convolution workloads, including long one-dimensional convolutions and two-dimensional depthwise convolutions. CycloSPDZ achieves speedups of up to 5× and 48× over our hybrid variant of SPD$\mathbb{Z}_{2^k}$ and the function-dependent Matrix Rings baseline, respectively. These findings underscore the advantage of native polynomial-ring authentication in MPC. Future work includes reducing the communication cost of preprocessing and applying CycloSPDZ to distributed key generation for homomorphic encryption schemes such as TFHE.

References

1. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. 42nd Annual IEEE Symposium on Foundations of Computer Science 2001, pp. 136–145. https://doi.org/10.1109/SFCS.2001.959888
2. Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P. Smart. Practical covertly secure MPC for dishonest majority – or: Breaking the SPDZ limits. In: Computer Security – ESORICS 2013, LNCS, vol. 8134, pp. 1–18. Springer, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40203-6_1
3. Marcel Keller, Emmanuela Orsini, and Peter Scholl. MASCOT: Faster malicious arithmetic secure computation with oblivious transfer. ACM CCS 2016, pp. 830–842. https://doi.org/10.1145/2976749.2978357
4. Marcel Keller, Valerio Pastro, and Dragos Rotaru. Overdrive: Making SPDZ great again. EUROCRYPT 2018, pp. 158–189. https://doi.org/10.1007/978-3-319-78372-7_6
5. Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. CRYPTO 2012, pp. 643–662. https://doi.org/10.1007/978-3-642-32009-5_38
6. Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. EUROCRYPT 2011, pp. 169–188. https://doi.org/10.1007/978-3-642-20465-4_11
7. Donald Beaver. Efficient multiparty protocols using circuit randomization. In: Advances in Cryptology – CRYPTO 1991, LNCS, vol. 576, pp. 420–432. Springer, Berlin, Heidelberg (1992). https://doi.org/10.1007/3-540-46766-1_34
8. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators from Ring-LPN. CRYPTO 2020, pp. 387–416. https://doi.org/10.1007/978-3-030-56880-1_14

9. Ronald Cramer, Ivan Damgård, Daniel Escudero, Peter Scholl, and Chaoping Xing. SPD$\mathbb{Z}_{2^k}$: Efficient MPC mod 2^k for dishonest majority. CRYPTO 2018, pp. 769–798. https://doi.org/10.1007/978-3-319-96881-0_26 Extended version revised March 31, 2022: https://eprint.iacr.org/2018/482
10. Emmanuela Orsini, Nigel P. Smart, and Frederik Vercauteren. Overdrive2k: Efficient secure MPC over $\mathbb{Z}_{2^k}$ from somewhat homomorphic encryption. In: Topics in Cryptology – CT-RSA 2020, LNCS, vol. 12006, pp. 254–283. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-40186-3_12
11. Ivan Damgård, Daniel Escudero, Tore Kasper Frederiksen, Marcel Keller, Peter Scholl, and Nikolaj Volgushev. New primitives for actively-secure MPC over rings with applications to private machine learning. IEEE Symposium on Security and Privacy 2019, pp. 1102–1120. https://doi.org/10.1109/SP.2019.00078
12. Hao Chen, Miran Kim, Ilya Razenshteyn, Dragos Rotaru, Yongsoo Song, and Sameer Wagh. Maliciously secure matrix multiplication with applications to private deep learning. Advances in Cryptology – ASIACRYPT 2020, pp. 31–59. https://doi.org/10.1007/978-3-030-64840-4_2
13. Hyunho Cha, Intak Hwang, Seonhong Min, Jinyeong Seo, and Yongsoo Song. MatriGear: Accelerating authenticated matrix triple generation with scalable prime fields via optimized HE packing. IEEE Symposium on Security and Privacy 2025, pp. 2453–2471. https://doi.org/10.1109/SP61157.2025.00174
14. Hongqing Liu, Chaoping Xing, Chen Yuan, and Taoxu Zou. Dishonest majority multi-party computation over matrix rings. In: Advances in Cryptology – ASIACRYPT 2024, LNCS, vol. 15489, pp. 299–327. Springer, Singapore (2025). https://doi.org/10.1007/978-981-96-0938-3_10
15. Sebastian Hasler, Pascal Reisert, Marc Rivinius, and Ralf Küsters. Multipars: Reduced-communication MPC over $\mathbb{Z}_{2^k}$. Proceedings on Privacy Enhancing Technologies 2024, no. 2, pp. 5–28. https://doi.org/10.56553/popets-2024-0038
16. Marcel Keller. MP-SPDZ: A versatile framework for multi-party computation. ACM CCS 2020, pp. 1575–1590. https://doi.org/10.1145/3372297.3417872
17. Marc Rivinius, Pascal Reisert, Sebastian Hasler, and Ralf Küsters. Convolutions in Overdrive: Maliciously secure convolutions for MPC. Proceedings on Privacy Enhancing Technologies 2023, no. 3, pp. 321–353. https://doi.org/10.56553/popets-2023-0084
18. Zhe Li, Chaoping Xing, Yizhou Yao, and Chen Yuan. Efficient pseudorandom correlation generators over $\mathbb{Z}/p^k\mathbb{Z}$. CRYPTO 2025, pp. 207–240. https://doi.org/10.1007/978-3-032-01884-7_7
19. Damiano Abram and Peter Scholl. Low-communication multiparty triple generation for SPDZ from Ring-LPN. PKC 2022, pp. 221–251. https://doi.org/10.1007/978-3-030-97121-2_9
20. Elette Boyle, Niv Gilboa, Matan Hamilis, Yuval Ishai, and Ariel Nof. Preprocessing for life: Dishonest-majority MPC with a trusted or untrusted dealer. In: 2025 IEEE Symposium on Security and Privacy (SP), pp. 2433–2452. IEEE (2025). https://doi.org/10.1109/SP61157.2025.00041
21. Alexander Kyster, Frederik Huss Nielsen, Sabine Oechsner, and Peter Scholl. Rushing at SPDZ: On the practical security of malicious MPC implementations. In: 2025 IEEE Symposium on Security and Privacy (SP), pp. 2491–2508. IEEE (2025). https://doi.org/10.1109/SP61157.2025.00176
22. Kanav Gupta, Nishanth Chandran, Divya Gupta, Jonathan Katz, and Rahul Sharma. SHARK: Actively secure inference using function secret sharing. IEEE Symposium on Security and Privacy 2025, pp. 2472–2490. https://doi.org/10.1109/SP61157.2025.00175

23. Yang Liu and Liang Feng Zhang. CAVERN: Efficient honest-majority maliciously secure (2+1)-PC for $\mathbb{Z}_{2^n}$ via DPF. IEEE Symposium on Security and Privacy 2026, pp. 1921–1936. https://doi.org/10.1109/SP63933.2026.00089
24. Daniel Escudero, Chaoping Xing, and Chen Yuan. More efficient dishonest majority secure computation over $\mathbb{Z}_{2^k}$ via Galois rings. CRYPTO 2022, pp. 383–412. https://doi.org/10.1007/978-3-031-15802-5_14
25. Daniel Escudero, Cheng Hong, Hongqing Liu, Chaoping Xing, and Chen Yuan. Degree-D reverse multiplication-friendly embeddings: Constructions and applications. In: Advances in Cryptology – ASIACRYPT 2023, LNCS, vol. 14438, pp. 106–138. Springer, Singapore (2023). https://doi.org/10.1007/978-981-99-8721-4_4
26. Daniel Escudero, Vipul Goyal, Antigoni Polychroniadou, Yifan Song, and Chenkai Weng. SuperPack: Dishonest majority MPC with constant online communication. EUROCRYPT 2023, pp. 220–250. https://doi.org/10.1007/978-3-031-30617-4_8
27. Boshi Yuan, Shixuan Yang, Yongxiang Zhang, Ning Ding, Dawu Gu, and Shi-Feng Sun. MD-ML: Super fast privacy-preserving machine learning for malicious security with a dishonest majority. In: 33rd USENIX Security Symposium, pp. 2227–2244. USENIX Association (2024). https://www.usenix.org/conference/usenixsecurity24/presentation/yuan
28. Yansong Zhang, Xiaojun Chen, Ye Dong, Qinghui Zhang, Rui Hou, Qiang Liu, and Xudong Chen. MD-SONIC: Maliciously-secure outsourcing neural network inference with reduced online communication. IEEE Transactions on Information Forensics and Security 2025, vol. 20, pp. 3534–3549. https://doi.org/10.1109/TIFS.2025.3550834
29. Daniel Y. Fu, Hermann Kumbong, Eric Nguyen, and Christopher Ré. FlashFFTConv: Efficient convolutions for long sequences with tensor cores. ICLR 2024, pp. 9455–9483.
30. Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861, 2017. https://doi.org/10.48550/arXiv.1704.04861
31. Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. CVPR 2018, pp. 4510–4520. https://doi.org/10.1109/CVPR.2018.00474
32. Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon, and Saining Xie. ConvNeXt V2: Co-designing and scaling ConvNets with masked autoencoders. CVPR 2023, pp. 16133–16142. https://doi.org/10.1109/CVPR52729.2023.01548

A Proof of Proposition 1

Throughout this appendix, we retain the notation of Section 2.1. Thus $k, s, n \in \mathbb{Z}_{\geq 1}$, $q = k + s$, $N = 2^n \geq 2$, $L = Nq$, and $R = \mathbb{Z}_{2^q}[X]/(X^N + 1)$, $w = [X + 1]$, $\mathfrak{m} = (w)$. Also, we set $R_k := \mathbb{Z}_{2^k}[X]/(X^N + 1)$, and let $\pi_k : R \longrightarrow R_k$ denote the natural surjective ring homomorphism induced by coefficientwise reduction modulo 2^k. As before, we write a polynomial in place of its residue class in R whenever no confusion can arise.

A.1 Locality and the Unit Group

We first prove that $\mathfrak{m} = (w) = (X + 1)$ is the unique maximal ideal of the quotient ring R.

Proof. In the coefficient ring $\mathbb{Z}_{2^q}$, it is evident that $2^q = 0$; that is, 2 is a nilpotent element of R. And it is for sure that every nilpotent element belongs to every maximal ideal. Thus, every maximal ideal of R contains (2). Reducing modulo 2, we can obtain

$$\begin{aligned} R/(2) &= \big(\mathbb{Z}_{2^q}[X]/(X^N+1)\big)/(2) \\ &\cong \mathbb{Z}_{2^q}[X]/(X^N+1,2) \\ &\cong \mathbb{F}_2[X]/(X^N+1). \end{aligned}$$

And since $N = 2^n$, repeated application of the Frobenius endomorphism in characteristic 2 gives $X^N + 1 = (X+1)^N$. Thus, we have $R/(2) \cong \mathbb{F}_2[X]/((X+1)^N)$. Writing $y = X + 1$, the ring on the right is isomorphic to $\mathbb{F}_2[y]/(y^N)$, whose unique maximal ideal is generated by the image of y. According to the correspondence theorem for ideals, R has the unique maximal ideal $(2, w)$.

Since $X^N + 1 = 0$ in the quotient ring R and $N = 2^n$ must be even, we have

$$X^N - 1 = (X+1)(X^{N-1} - X^{N-2} + \cdots + X - 1) = -2.$$

Equivalently,

$$(X+1)(-X^{N-1} + X^{N-2} - \cdots - X + 1) = 2.$$

Hence, $2 \in (w)$, and consequently $(2, w) = (w) = \mathfrak{m}$. Therefore, the quotient ring R is a local ring, and $\mathfrak{m}$ is its unique maximal ideal. □

The preceding identification of $\mathfrak{m} = (w)$ as the unique maximal ideal of R leads naturally to the following characterization of the units of R:

For every $a \in R$, one has

$$a \in R^\times \iff a \notin \mathfrak{m} \iff a + \mathfrak{m} = 1 + \mathfrak{m}.$$

Consequently, one also has

$$R^\times = R \setminus \mathfrak{m} = 1 + \mathfrak{m}.$$

Proof. Assume that $R^\times \cap \mathfrak{m} \neq \emptyset$. Then we can choose an element a such that $a \in R^\times \cap \mathfrak{m}$. Since a is a unit, its inverse a^{-1} exists and $a^{-1}a = 1$. Meanwhile, since $a \in \mathfrak{m}$ and $\mathfrak{m}$ is an ideal, one has $a \in \mathfrak{m} \Longrightarrow a^{-1}a \in \mathfrak{m} \Longrightarrow 1 \in \mathfrak{m}$. This contradicts the fact that $\mathfrak{m}$ is a maximal ideal (proper ideal) of R. Thus, the initial assumption is false, which means $R^\times \cap \mathfrak{m} = \emptyset$. Hence, one must have $R^\times \subseteq R \setminus \mathfrak{m}$.

Conversely, assume there exists $a \in R$ such that $a \notin \mathfrak{m}$ and $a \notin R^\times$. Then the principal ideal (a) is proper, and must be contained in some maximal ideal of R. Since the unique maximal ideal of R is $\mathfrak{m} = (X+1)$, one must have $a \in (a) \subseteq \mathfrak{m}$. This contradicts the assumption $a \notin \mathfrak{m}$. Therefore, no such element $a \in R$ exists, and $R \setminus \mathfrak{m} \subseteq R^\times$.

Combining the two inclusions above, one has $R^\times = R \setminus \mathfrak{m}$. Since $\mathfrak{m} = (X+1)$ and $N = 2^n$ is even, one has

$$\begin{aligned} R/\mathfrak{m} &\cong \mathbb{Z}_{2^q}[X]/(X^N+1, X+1) \\ &\cong \mathbb{Z}_{2^q}/\big((-1)^N+1\big) \\ &= \mathbb{Z}_{2^q}/(2) \\ &\cong \mathbb{F}_2. \end{aligned}$$

Therefore, the only nonzero residue class in $R/\mathfrak{m}$ is $1+\mathfrak{m}$. Thus, for every $a \in R$, the following equivalence holds:

$$a \notin \mathfrak{m} \Longleftrightarrow a + \mathfrak{m} = 1 + \mathfrak{m}.$$

Together with the fact that $R^\times = R \setminus \mathfrak{m}$, this yields $R^\times = 1 + \mathfrak{m}$. □

A.2 The Power Relation and the Nilpotency Index

A basic power relation in R is given by $w^N = 2g$ for some unit $g \in R^\times$.

Proof. In R, according to the binomial theorem, one has

$$w^N = (X+1)^N = X^N + 1 + \sum_{i=1}^{N-1} \binom{N}{i} X^i = \sum_{i=1}^{N-1} \binom{N}{i} X^i.$$

Since $N = 2^n$, for every $1 \le i \le N-1$, the binomial coefficient $\binom{N}{i}$ is even. Therefore, we can define $g := \sum_{i=1}^{N-1} \frac{\binom{N}{i}}{2} X^i \in R$, where the integer coefficients are reduced modulo 2^q. Then one has $w^N = 2g$.

Meanwhile, as described in Appendix A.1, one has $R/\mathfrak{m} \cong \mathbb{F}_2$. In this residue field, the image of X is 1. Evaluating the displayed integer polynomial at $X = 1$, with division by 2 performed in the integers before reduction, yields

$$\frac{1}{2} \sum_{i=1}^{N-1} \binom{N}{i} = 2^{N-1} - 1 \equiv 1 \pmod 2.$$

Therefore, one has $g + \mathfrak{m} = 1 + \mathfrak{m}$, and hence $g \in R^\times$ by Appendix A.1. That is, there must exist a unit g such that $w^N = 2g$ holds. □

The relation above also makes it possible to determine the nilpotency index of $\mathfrak{m} = (w)$. Since w^N differs from 2 only by a unit factor, powers of w can be compared directly with the corresponding powers of 2. Since $L = Nq$, one has $\mathfrak{m}^L = (0)$, $\mathfrak{m}^{L-1} \neq (0)$.

Proof. Since there exists a unit $g \in R^\times$ such that $w^N = 2g$ and $L = Nq$, one has $w^L = w^{Nq} = (2g)^q = 2^q \cdot g^q = 0$. Therefore, $\mathfrak{m}^L = (w^L) = (0)$ holds.

Meanwhile, since $L - 1 = N(q-1) + (N-1)$, the following identity holds:

$$\begin{aligned} w^{L-1} &= (w^N)^{q-1} \cdot w^{N-1} \\ &= (2g)^{q-1} \cdot w^{N-1} \\ &= 2^{q-1} \cdot g^{q-1} \cdot w^{N-1}. \end{aligned}$$

Since g^{q-1} is a unit, for $2^{q-1} \cdot g^{q-1} \cdot w^{N-1} = 0$ to hold, one must have $2^{q-1} \cdot w^{N-1} = 0$. And since $X^N + 1$ is monic, each element of R admits a unique representative of degree strictly less than N. Thus, the equality $2^{q-1} \cdot w^{N-1} = 0$ would force its unique representative of degree less than N, namely $2^{q-1} \cdot (X+1)^{N-1}$, to be the zero polynomial. However, since this representative has degree $N-1$, which is obviously less than N, and leading coefficient $2^{q-1} \neq 0$, one must have $2^{q-1} \cdot (X+1)^{N-1} \neq 0$. So, the assumption above is false, and $w^{L-1} = 2^{q-1} \cdot g^{q-1} \cdot w^{N-1} \neq 0$ must hold. Hence, one has $\mathfrak{m}^L = (0), \mathfrak{m}^{L-1} \neq (0)$. □

A.3 The Ideal Chain and Cardinalities of Each Layer

The nilpotency index obtained in Appendix A.2 provides the natural range for the powers of $\mathfrak{m}$. Their cardinalities are given explicitly as follows: for every integer $0 \leq t \leq L$, one has $|\mathfrak{m}^t| = 2^{L-t}$.

Proof. By Appendix A.2, when $t = L$, one has $\mathfrak{m}^L = (0), |\mathfrak{m}^L| = 1 = 2^{L-L}$, which satisfies the formula above. Moreover, since w has nilpotency index L, one has $w^j \neq 0$ for every $0 \leq j < L$. And since $\mathfrak{m}\mathfrak{m}^j = \mathfrak{m}^{j+1}$, the quotient $\mathfrak{m}^j/\mathfrak{m}^{j+1}$ is naturally a vector space over the residue field $R/\mathfrak{m} \cong \mathbb{F}_2$. Moreover, since $\mathfrak{m}^j = (w^j)$, this vector space is generated by $w^j + \mathfrak{m}^{j+1}$.

Suppose that $w^j \in \mathfrak{m}^{j+1}$, and then there must exist $r \in R$ such that $w^j = w^{j+1}r$. Hence, one has $w^j(1 - wr) = 0$. And since $-wr \in \mathfrak{m}$, the characterization of the unit group yields $1 - wr \in 1 + \mathfrak{m} = R^\times$. Therefore, for $w^j(1-wr) = 0$ to hold, it must be that $w^j = 0$, which contradicts $w^j \neq 0$. Thus $w^j \notin \mathfrak{m}^{j+1}$, and consequently $w^j + \mathfrak{m}^{j+1} \neq 0$ in $\mathfrak{m}^j/\mathfrak{m}^{j+1}$.

Therefore, $\mathfrak{m}^j/\mathfrak{m}^{j+1}$ is generated by a nonzero vector over $\mathbb{F}_2$, and consequently $|\mathfrak{m}^j/\mathfrak{m}^{j+1}| = 2$. Hence, for every $0 \leq t < L$, one has

$$|\mathfrak{m}^t| = |\mathfrak{m}^L| \cdot \prod_{j=t}^{L-1} |\mathfrak{m}^j/\mathfrak{m}^{j+1}| = \prod_{j=t}^{L-1} 2 = 2^{L-t}.$$

Together with the case $t = L$, for every integer $0 \leq t \leq L$, $|\mathfrak{m}^t| = 2^{L-t}$ always holds. □

The powers of $\mathfrak{m}$ considered above also yield a factorization of every nonzero element of R. For any $z \in R \setminus \{0\}$, one has $v(z) = \max\{a \in \{0, \ldots, L-1\} : z \in \mathfrak{m}^a\}$. Then there exists a unit $u \in R^\times$ such that $z = w^{v(z)}u$. Consequently, one has $zR = \mathfrak{m}^{v(z)}$.

Proof. By Appendix A.2, one has $\mathfrak{m}^L = (0)$. Since $z \neq 0$ and $z \in R = \mathfrak{m}^0$, the integer $v = v(z)$ is well defined, and the definition of v yields $z \in \mathfrak{m}^v, z \notin \mathfrak{m}^{v+1}$. Since $\mathfrak{m}^v = (w^v)$, there exists $u \in R$ such that $z = w^v u$. If $u \in \mathfrak{m}$, then it follows that $z = w^v u \in (w^v)\mathfrak{m} = \mathfrak{m}^{v+1}$. This contradicts the fact that $z \notin \mathfrak{m}^{v+1}$, and therefore implies that $u \notin \mathfrak{m}$. Moreover, by Appendix A.1, one has $u \in R^\times$, and therefore it follows that $zR = (w^v u)R = w^v R = \mathfrak{m}^v = \mathfrak{m}^{v(z)}$. □

The preceding cardinality computation naturally leads to the full description of the ideal structure of R. The ideals of R are precisely $\mathfrak{m}^t, 0 \leq t \leq L$, and they form the

strictly descending chain $R = \mathfrak{m}^0 \supsetneq \mathfrak{m} \supsetneq \mathfrak{m}^2 \supsetneq \ldots \supsetneq \mathfrak{m}^L = \{0\}$. Thus, R is a finite local chain ring.

Proof. By Appendix A.2, $\mathfrak{m}^L = (0) = \{0\}$. Let $J \neq (0)$ be an arbitrary ideal of R. Consider the set $S_J := \{t \in \{0, \ldots, L\} : J \subseteq \mathfrak{m}^t\}$. Since $J \subseteq R = \mathfrak{m}^0$, the set S_J is nonempty. Moreover, since $J \neq (0) = \mathfrak{m}^L$, the finite set S_J must have a largest element $d := \max(S_J)$, such that $0 \leq d < L$.

By the maximality of d, one has $J \nsubseteq \mathfrak{m}^{d+1}$. Thus, we can choose $z \in J \setminus \mathfrak{m}^{d+1}$. Since $z \in \mathfrak{m}^d \setminus \mathfrak{m}^{d+1}$, one has $v(z) = d$. Then by the preceding result, it follows that $\mathfrak{m}^d = (z) \subseteq J \subseteq \mathfrak{m}^d$, and therefore, one must have $J = \mathfrak{m}^d$.

Hence, all nonzero ideals of R are exactly $\mathfrak{m}^t$ for integers t satisfying $0 \leq t < L$, while the zero ideal is $\mathfrak{m}^L$. Finally, for every integer t with $0 \leq t < L$ the cardinality formula yields $|\mathfrak{m}^{t+1}| = 2^{L-t-1} < 2^{L-t} = |\mathfrak{m}^t|$. So, all consecutive inclusions are strict. Thus, $R = \mathfrak{m}^0 \supsetneq \mathfrak{m} \supsetneq \mathfrak{m}^2 \supsetneq \ldots \supsetneq \mathfrak{m}^L = \{0\}$, and R is a finite local chain ring. □

A.4 Annihilators and Reduction Modulo Powers of Two

The description of the ideal structure in Appendix A.3 makes it possible to determine the annihilators of powers of w, to identify the ideals generated by powers of 2, and to describe the kernel of coefficientwise reduction modulo 2^k.

For every integer t with $0 \leq t \leq L$ and every unit $u \in R^\times$, it follows that

$$\mathrm{Ann}_R(w^t) = \mathrm{Ann}_R(w^t u) = \mathfrak{m}^{L-t}.$$

Moreover, for every integer j with $0 \leq j \leq q$, one has $2^j R = \mathfrak{m}^{Nj}$. In particular, $\ker(\pi_k) = 2^k R = \mathfrak{m}^{Nk}$ always holds.

Proof. Fix an integer t with $0 \leq t \leq L$, and set $J := \mathrm{Ann}_R(w^t) = \{a \in R : aw^t = 0\}$. As an annihilator, J is an ideal of R. By the description of the ideal structure obtained in Appendix A.3, there must exist a unique integer $j \in \{0, \ldots, L\}$ such that $J = \mathfrak{m}^j$.

Since $\mathfrak{m}^{L-t} = (w^{L-t})$, one has $\mathfrak{m}^{L-t} \cdot w^t = (w^{L-t}) \cdot w^t = (w^L) = \mathfrak{m}^L = (0)$. Thus, $\mathfrak{m}^{L-t} \subseteq J = \mathfrak{m}^j$. And since the powers of $\mathfrak{m}$ form a strictly descending chain, this inclusion implies that $j \leq L - t$.

Assume that $j < L - t$. Since $w^j \in \mathfrak{m}^j = J$, the definition of J gives $w^j \cdot w^t = w^{j+t} = 0$. However, $j < L - t$ implies $j + t < L$, whereas the nonvanishing result established in Appendix A.3 gives $w^i \neq 0$ for every $0 \leq i < L$. In particular, $w^{j+t} \neq 0$, which contradicts the assumption, and therefore, the assumption does not hold. Hence, one has $j \geq L - t$. Together with the fact that $j \leq L - t$, $j = L - t$ must hold. Thus, one has $\mathrm{Ann}_R(w^t) = J = \mathfrak{m}^{L-t}$.

Moreover, let $u \in R^\times$. For every $a \in R$, since multiplication by a unit is injective, one has $aw^t u = 0 \iff aw^t = 0$. Consequently, $\mathrm{Ann}_R(w^t u) = \mathrm{Ann}_R(w^t) = \mathfrak{m}^{L-t}$. By Appendix A.2, $w^N = 2g$ must hold for some unit $g \in R^\times$. Hence, for every integer j with $0 \leq j \leq q$, it follows that $\mathfrak{m}^{Nj} = w^{Nj} R = (w^N)^j R = (2g)^j R = 2^j R$, since g^j is a unit.

Finally, by the definition of π_k and the uniqueness of polynomial representatives of degree less than N, an element of R belongs to $\ker(\pi_k)$ if and only if each coefficient of the representative is divisible by 2^k. Hence, one has $\ker(\pi_k) = 2^k R = \mathfrak{m}^{Nk}$. □

B Ideal Functionalities and UC Security Proofs

B.1 Security of Π_{MACCheck}

To characterize the security goal of protocol Π_{MACCheck}, we define the ideal functionality $\mathcal{F}_{\text{MACCheck}}^{R_k}$ for the batched MAC check, which verifies a batch of opened values.

Functionality $\mathcal{F}_{\text{MACCheck}}^{R_k}$

This functionality reuses the reference state of the existing authenticated sharings. It could handle at most $B_{\max}$ nonempty batches under the same MAC key, where the batch limitation $B_{\max}$ is fixed before the session begins. Once it receives consistent requests from all parties for the batch size $T^{(b)}$ and the public opened values $y^{(b,j)\prime}$, it returns accept directly if $T^{(b)} = 0$; otherwise, it obtains the reference values $y^{(b,j)}$ from the reference state and fixes them together with the public opened values of this batch. Then, after receiving Proceed from the adversary, it returns accept to all parties and continues if $\pi_k\left(y^{(b,j)\prime} - y^{(b,j)}\right) = 0$ holds for every j; otherwise, it terminates the session. Moreover, whenever it receives Abort, invalid or inconsistent requests, or the protocol fails to complete, it also aborts.

Fig. 11. Ideal functionality for the batched MAC check

Proof (of Theorem 3). We prove the security of protocol Π_{MACCheck} via the UC framework. By leveraging the completeness of the dummy adversary, it is sufficient to construct a simulator S for the dummy adversary in the $\mathcal{F}_{\text{Pre}}^{R}$-hybrid model. S works as follows:

Overall simulation. S reuses the shared-state simulation, in which it knows the MAC key, the reference opened values, and the corrupted parties' reference state, and it keeps the message distribution outside the checks unchanged. For a nonempty batch b, once the public opened values and the current MAC shares are fixed, S delivers fresh zero-sum masks $\xi_i^{(b)}$ to the corrupted parties as the protocol specifies. It then samples the public challenges $\rho^{(b,j)}$ independently and uniformly from $\mathcal{T}_s$. Let the prescribed check shares of the corrupted parties be

$$b_i := \xi_i^{(b)} + \overline{y}'\alpha_i - \sum_{j=1}^{T^{(b)}} \rho^{(b,j)}\gamma_i^{(b,j)}, \qquad i \in C.$$

Where $\gamma_i^{(b,j)}$ are the MAC shares in the reference state. Then, S samples the check shares for the honest parties uniformly subject to the following sum constraint:

$$\sum_{i \notin C} \sigma_i^{(b)*} = \alpha \mathcal{D}^{(b)} - \sum_{i \in C} b_i.$$

After that, S submits the check shares of the honest parties above to the $\mathcal{F}_{\text{COMMIT}}$ which fixes these values. Once all commitments have been received, the parties open them

sequentially as the protocol specifies. The values that the corrupting party actually commits to and effectively opens are denoted by $\widehat{\sigma}_i^{(b)}$. If all commitments have been validly opened and the check equation $\sum_i \widehat{\sigma}_i^{(b)} = 0$ holds, S sends Proceed to the ideal functionality; otherwise, it sends Abort.

Then, we argue that the real-world execution and the simulated execution are statistically close. Given the key, the reference state, the challenges, and the corrupted parties' masks, the honest parties' current MAC shares equal their derived shares. And the fresh zero-sum masks make their check shares uniformly subject to the sum constraint above. Since the simulator S knows all items on the right side, its distribution of sampling matches that of the real-world execution. The honest check shares are fixed upon commitment and are opened sequentially as the protocol specifies. Hence, the adversary's choice to abort based on partial openings is simulated identically. When all openings are valid, one has $\sum_i \widehat{\sigma}_i^{(b)} = \alpha \mathcal{D}^{(b)} - E^{(b)} + e^{(b)}$.

Let Bad denote the event that some check accepts while the payload of at least one opened value has changed. Before the first Bad event occurs, the two executions agree on the message distribution and on their accept-or-abort behavior. If a check fails without any payload change, the simulator reproduces this by sending Abort. Each batch uses independent masks and challenges as specified by the protocol. The number of nonempty batches is bounded in advance, and the session terminates at the first batch that rejects or fails to complete. Hence, by Theorem 2, one has

$$\mathrm{Adv}_Z \le \Pr_{\text{real}}[\mathsf{Bad}] \le (Ns + 4)2^{-(Ns+2)}.$$

□

B.2 Security of Π_{Online}

Proof (of Theorem 4). We prove the security of protocol Π^R_{Online} via the UC framework. By leveraging the completeness of the dummy adversary, it is sufficient to construct a simulator S for the dummy adversary in the $\mathcal{F}^R_{\text{Pre}}$-hybrid model. S internally simulates the preprocessing functionality and the virtual honest parties. It maintains the existing keys, authenticated sharings, and the corrupted parties' reference state, and works as follows:

Simulation of the initialization and input phase. S internally generates the preprocessing material exactly as the ideal functionality $\mathcal{F}^R_{\text{Pre}}$ specifies. It uses 0 as the placeholder input of the honest parties, maintains the state of the virtual honest parties as the protocol specifies, and generates their messages. Hence, in such simulation, S has access to all key shares and all preprocessing randomness, and it can maintain the corrupted parties' reference state as in Definition 1. When a corrupted input party broadcasts ϵ, S uses the corresponding input mask r to extract its effective input as $x^* = \pi_k(r + \epsilon)$ and submits it to the ideal functionality $\mathcal{F}^{R_k}_{\text{Online}}$.

Simulation of linear operations. This part only involves local computation. S executes the prescribed operations on behalf of the virtual honest parties and updates the reference state of the corrupted parties accordingly.

Simulation of the multiplication. When the differences d, f are opened S uses a fresh triple, computes the virtual honest parties' shares of these differences as the

protocol specifies, and commits to and then opens them. Then, S executes the subsequent local computation based on the actually opened d', f'. It updates the reference state and stores the opened values pending verification together with their MAC shares.

Simulation of the output phase. In the output phase, S obtains the correct output value y from the ideal functionality $\mathcal{F}_{\text{Online}}^{R_k}$ and then invokes the MAC-check simulation given in Appendix B.1 for the values opened during multiplication that are pending verification. All verifications throughout the online phase reuse the same MAC key and share a single $\mathcal{F}_{\text{MACCheck}}^{R_k}$ session. In particular, empty batches are accepted directly. If any check fails, S sends Abort to the ideal functionality; otherwise, it internally provides shares of a fresh output mask and the corresponding MAC shares $\left(r_i, \gamma_{r,i}\right)$ to the corrupted parties, as the ideal functionality $\mathcal{F}_{\text{Pre}}^{R}$ specifies. Then, S sets $\overline{v}_i = \overline{z}_i + r_i$ and $\overline{\gamma}_{v,i} = \overline{\gamma}_{z,i} + \gamma_{r,i}$ for $i \in C$, where $\left(\overline{z}_i, \overline{\gamma}_{z,i}\right)$ are the reference shares of the current output before adding the mask, and $\left(\overline{v}_i, \overline{\gamma}_{v,i}\right)$ are the corresponding shares after masking. Then, S samples $\eta \xleftarrow{\$} I$, $v = \iota(y) + \eta$, where $\iota(y)$ is the fixed representative of y in R. S also samples the honest parties' corresponding value shares and MAC shares uniformly subject to

$$\sum_{i \notin C} v_i = v - \sum_{i \in C} \overline{v}_i, \qquad \sum_{i \notin C} \gamma_{v,i} = \alpha v - \sum_{i \in C} \overline{\gamma}_{v,i}.$$

After that, S commits and opens the output shares as the protocol specifies. For the actual reconstruction v', S invokes the above check simulation in the same session. If the check aborts, S sends Abort; otherwise, it sends Proceed.

Then, we argue that the real-world execution and the simulated execution are statistically close. In the initialization phase, S generates all materials as $\mathcal{F}_{\text{Pre}}^{R}$ specifies, and the distribution of these materials is therefore the same as in the real-world execution. As for linear operations, only local computation is involved, and there are no additional public messages. Hence, they would not introduce any new distinguishing advantage. Moreover, the public values in the input and multiplication phases are protected by fresh and uniform full-ring masks, so using placeholder inputs does not change the message distribution seen by the environment.

For the MAC checks in the output phase, the sampling rule of $\mathcal{F}_{\text{Pre}}^{R}$ and the public linear updates of the protocol satisfy the conditions of Theorem 2. Each output needs at most two nonempty checks, and the session terminates at the first check that rejects or fails to complete. Therefore, the simulation argument of Appendix B.1 applies.

Then, let Bad denote the event that some check accepts while the payload of at least one opened value has been changed. If the batched MAC check before the output passes and no Bad event occurs, then all deviations of the relevant values opened during multiplication lie in I. Hence, together with the input extraction and the protocol operations, the derived value z^{real} of this output in the real-world execution satisfies $\pi_k\left(z^{\text{real}}\right) = y$. The opened value $z^{\text{real}} + r$ is uniform over $z^{\text{real}} + I = \iota(y) + I$, which has the same distribution as v generated by the simulator. Given the reference opened value, key α, and the corrupted parties' reference shares, the honest parties' value shares and MAC shares are uniform subject to the sum constraints, just as in the simulation. And in the simulated execution, the deviations introduced by the corrupted parties remain in the candidate

output. So, if the final check accepts and no Bad event occurs, the corresponding deviation lies in the ideal I, so the honest parties' output payload is still y.

Therefore, the distinguishing advantage of the environment is at most the probability that a Bad event occurs. Hence, for a fixed circuit and a fixed finite number of checks, the session bound gives

$$\mathrm{Adv}_Z \le \Pr_{\text{real}}[\mathsf{Bad}] \le (Ns+4)2^{-(Ns+2)}.$$

□

B.3 Security of $\Pi_{\mathbf{Div}}$

To characterize the security goal of protocol Π_{Div}, we define the ideal functionality $\mathcal{F}_{\mathrm{Div}}$ for exact division, which converts authenticated input sharings that satisfy the divisibility condition into secret sharings of the quotient.

Functionality $\mathcal{F}_{\mathrm{Div}}$

This functionality reuses the existing keys and the state of the authenticated sharings. It accepts T division requests that satisfy the input conditions of Fig. 6. Its input shares belong to $\mathbb{Z}_{2^{\mu+\tau}}$, and its output shares belong to $\mathbb{Z}_{2^{\mu-a+\tau}}$.

Two-party case. It applies $\lceil \cdot/2^a \rceil$ and $\lfloor \cdot/2^a \rfloor$ to the input shares and MAC shares of the two parties, respectively, reduces the results to $\mathbb{Z}_{2^{\mu-a+\tau}}$, and privately delivers them to the corresponding parties.

Multi-party case. It receives the high-bits correction $e^{(j)} \in 2^{\mu-a}\mathbb{Z}_{2^{\mu-a+\tau}}$ and the corrupted parties' output shares from the adversary, and computes $Y^{(j)} = \frac{x^{(j)}}{2^a} + e^{(j)} \in \mathbb{Z}_{2^{\mu-a+\tau}}$. It keeps the corrupted parties' output shares and samples the honest parties' shares uniformly subject to all value shares summing to $Y^{(j)}$ and all MAC shares summing to $\beta Y^{(j)}$. And after receiving Proceed from the adversary, it privately delivers the output shares to the corresponding parties.

Abort. Upon receiving (Abort) from the adversary, or upon receiving inconsistent or invalid requests, deliver $\perp$ to all parties and halt.

Fig. 12. Ideal functionality for exact division

Proof (of Theorem 5). We prove the security of protocol Π_{Div} via the UC framework. By leveraging the completeness of the dummy adversary, it is sufficient to construct a simulator S for the dummy adversary in the $\left(\mathcal{F}^{\mathbb{Z}}_{\mathrm{Auth}}, \mathcal{F}^{\mathbb{Z}}_{\mathrm{Triple}}, \mathcal{F}_{\mathrm{Rand}}, \mathcal{F}_{\mathrm{COMMIT}}\right)$-hybrid model. S internally simulates the preprocessing functionality and the virtual honest parties. It maintains the existing keys, authenticated sharings, and the corrupted parties' reference state, and works as follows:

Simulation of the two-party case. This part only involves local computation. S updates the corrupted parties' reference state according to the rounding and reduction rules of the protocol and obtains their output shares and MAC shares.

Simulation of the multi-party case. S internally simulates the preprocessing functionality and provides a series of fresh random authenticated masks $\left(r_i^{(j)}, \eta_{r^{(j)},i}\right)$ to

the corrupted parties. Then, S simulates the visible interaction with the corrupted parties while obtaining $T+1$ groups of zero-sum masks, namely $\left(\xi_i^x[1],\ldots,\xi_i^x[T],\xi_i\right)$, and updates their reference shares. Let $\left(\widetilde{\overline{x}}_i^{(j)},\overline{\eta}_{x^{(j)},i}\right)$ denote the corrupted parties' value shares and MAC shares after the refresh. Then, S uses 0 as the placeholder value of $x^{(j)}$ and samples the virtual honest parties' refreshed value shares and placeholder MAC shares uniformly subject to

$$\sum_{i\notin C}\widetilde{x}_i^{(j)} = -\sum_{i\in C}\widetilde{\overline{x}}_i^{(j)}, \quad \sum_{i\notin C}\eta_{x^{(j)},i} = -\sum_{i\in C}\overline{\eta}_{x^{(j)},i} \pmod{2^{\mu+\tau}}.$$

The MAC shares stay unchanged before and after the refresh. Following the protocol, S computes the virtual honest parties' shares of the masked differences. It then simulates all interactions in Open_{ξ} and the integer-ring batched MAC check. This check uses the zero-sum mask ξ_i and random challenges. The parties commit to and then open their check shares. Let $d^{(j)\prime}\in\mathbb{Z}_{2^\mu}$ denote the return value of Open_{ξ}, and let $\delta^{(j)}$ denote the full value actually opened in the protocol minus its reference value. If any step aborts, or $\delta^{(j)}\not\equiv 0 \pmod{2^\mu}$ holds for some j, S sends Abort to the ideal functionality. Otherwise, S computes $t^{(j)} = d^{(j)\prime}/2^a$ and obtains the corrupted parties' output shares and MAC shares as the protocol specifies. Then, it submits these shares together with the high-bits correction $e^{(j)} = r^{(j)} + t^{(j)} \pmod{2^{\mu-a+\tau}}$ to the ideal functionality $\mathcal{F}_{\mathrm{Div}}$ and sends Proceed.

Then, we argue that the real-world execution and the simulated execution are statistically close. In the two-party case, the protocol only performs deterministic rounding and reduction locally on the existing sharings, and it therefore produces no new public transcript. And since S updates the reference state by the same rules and can keep the correspondence between inputs and outputs, it would introduce no new distinguishing advantage.

In the multi-part case, S generates the specified random authenticated masks and zero-sum masks, of which the distribution on the corrupted parties' view is the same as in the real-world execution. Given the corrupted parties' view before the opening, the real-world and simulated masked differences $x^{(j)} - 2^a r^{(j)}$ and $-2^a r^{(j)}$ are both uniform over $2^a\mathbb{Z}_{2^{\mu+\tau}}$, and their reference return values are both uniform over $2^a\mathbb{Z}_{2^\mu}$. Moreover, zero-sum masks could make the honest parties' shares uniform subject to the sum constraints. Hence, the opening messages have the same distribution in both executions.

For the MAC check, let $\delta^{(j)}$ denote the actual full opened value minus its reference value in the reference state, and set $D_{\mathrm{chk}} = \sum_j \chi^{(j)}\delta^{(j)}$, where each challenge $\chi^{(j)}\in\mathbb{Z}_{2^\tau}$ is sampled independently and uniformly. If b_i denotes the corrupted parties' prescribed check share, computed from the reference state using the actual opened values and ξ_i, the authentication relation gives

$$\sum_{i\notin C}\sigma_i^* = -\beta D_{\mathrm{chk}} - \sum_{i\in C} b_i \pmod{2^{\mu+\tau}}.$$

S knows all items in the right-hand side, while fresh independent zero-sum masks make the honest parties' check shares uniform subject to this sum. Hence, S can reproduce the distribution of check messages and the accept-or-abort behavior.

Let Bad denote the event that some check accepts while at least one opened value has its μ low-order bits changed. If the check passes and no Bad event occurs, the simulated execution satisfies $d^{(j)\prime} \equiv -2^a r^{(j)} \pmod{2^\mu}$, and therefore $e^{(j)} = r^{(j)} + \frac{d^{(j)\prime}}{2^a} \in 2^{\mu-a}\mathbb{Z}_{2^{\mu-a+\tau}}$ holds. Since shifts preserve uniformity, we can pair the real-world and simulated masks as $r^{R,(j)} = r^{(j)} + \frac{x^{(j)}}{2^a} \pmod{2^{\mu+\tau}}$, without changing either distribution. Under this correspondence, the masked differences and Open_{ξ} return values then agree in both executions. The full real-world output satisfies $Y^{(j)} = r^{R,(j)} + \frac{d^{(j)\prime}}{2^a} = \frac{x^{(j)}}{2^a} + e^{(j)} \pmod{2^{\mu-a+\tau}}$, which matches the full output of the ideal functionality. Therefore, the return values of Open_{ξ} and full the outputs have the same joint distribution in both executions. Moreover, the ideal functionality completes the honest parties' shares uniformly subject to the value and MAC sum constraints above. The fresh random authenticated masks and the independent zero-sum masks are uniform. Hence, given the public messages, the corrupted parties' reference state, and the sums above, the honest real-world output shares and MAC shares of honest parties have the same conditional distribution.

Together with the message simulation, this gives the same interaction distribution before the first Bad event. The simulator also aborts when a check fails. And if the checks pass and no Bad event occurs, the public messages and outputs have the same joint distribution. Thus, the distinguishing advantage of the environment is at most the probability that Bad event occurs. Since each batch uses fresh independent zero-sum masks and the session terminates at the first check that rejects or fails to complete, together with the authentication theorem of $\mathrm{SPD}\mathbb{Z}_{2^k}$ [9], we can obtain

$$\mathrm{Adv}_Z \le \Pr_{\text{real}}[\mathsf{Bad}] \le (\tau + 1)\, 2^{-\tau}.$$

□

B.4 Security of Π_{RingMul}

To characterize the security goal of protocol Π_{RingMul}, we define the ideal functionality $\mathcal{F}_{\text{RingMul}}$ for authenticated polynomial multiplication, which converts authenticated sharings of two input polynomials into an authenticated sharing of their product modulo $X^N + 1$.

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

This functionality reuses the existing keys and the state of the authenticated sharings. It receives multiplication requests as in Fig. 7, with secret-shared input polynomials u and v. Let ℓ denote the number of Toom-4 levels in the public multiplication tree, and set $D = 3\ell$.

Computation. It computes the leaf inputs $U^{(j)}$ and $V^{(j)}$ as specified in the local-evaluation step of Fig. 7. It receives the high-order deviations $\kappa_d^{(j)}, \kappa_e^{(j)}, \vartheta^{(j)} \in 2^{\mu}\mathbb{Z}_{2^{\mu+\tau}}$ from the adversary and sets $g^{(j)} = \left(U^{(j)} + \kappa_d^{(j)}\right)\left(V^{(j)} + \kappa_e^{(j)}\right) + \vartheta^{(j)} \pmod{2^{\mu+\tau}}$. Then, it applies reconstruction and negacyclic folding as specified in the reconstruction and negacyclic-folding steps of Fig. 7. For each $t \in \{0, \ldots, N-1\}$, if $D > 0$ in the multi-party case, it receives the high-order deviations $h^{(t)} \in 2^{\mu-D}\mathbb{Z}_{2^{\mu-D+\tau}}$; otherwise, it sets $h^{(t)} = 0$. Then the output coefficients are $Y[t] = \frac{W[t]}{2^D} + h^{(t)} \pmod{2^{\mu-D+\tau}}$.

Output. It receives and keeps the corrupted parties' value shares and MAC shares submitted by the adversary. Then, it completes the honest parties' shares uniformly over $\mathbb{Z}_{2^{\mu-D+\tau}}$, subject to all value shares summing to $Y[t]$ and all MAC shares summing to $\beta Y[t]$, respectively. Once it receives Proceed from the adversary, it privately delivers the output shares to the corresponding parties.

Abort. Upon receiving (Abort) from the adversary, or upon receiving inconsistent or invalid requests, deliver $\perp$ to all parties and halt.

Fig. 13. Ideal functionality for authenticated polynomial multiplication

Proof (of Theorem 6). We prove the security of protocol Π_{RingMul} via the UC framework. By leveraging the completeness of the dummy adversary, it is sufficient to construct a simulator S for the dummy adversary in the $\left(\mathcal{F}_{\text{Auth}}^{\mathbb{Z}}, \mathcal{F}_{\text{Triple}}^{\mathbb{Z}}, \mathcal{F}_{\text{Rand}}, \mathcal{F}_{\text{COMMIT}}\right)$-hybrid model. S internally simulates the preprocessing functionality and the virtual honest parties. It maintains the existing keys, authenticated sharings, and the corrupted parties' reference state, and works as follows:

Simulation of the local evaluation. For corrupted parties, S updates the corrupted parties' reference state by local evaluation along the public recursion tree. For each $i \in C$, S splits the two input shares at the current node into coefficient blocks, denoted by $u_i^{(r)}$ and $v_i^{(r)}$. At a Karatsuba node, each input is split into 2 equal-length blocks; that is, the range of r is $\{0, 1\}$. Meanwhile, at a Toom-4 node, each input is split into 4 equal-length blocks; that is, the range of r is $\{0, 1, 2, 3\}$. Then, at a Karatsuba node, S forms the three input pairs $\left(u_i^{(0)}, v_i^{(0)}\right)$, $\left(u_i^{(0)} + u_i^{(1)}, v_i^{(0)} + v_i^{(1)}\right)$, and $\left(u_i^{(1)}, v_i^{(1)}\right)$. While at a Toom-4 node, for $t \in \{0, 1, -1, 2, -2, 3\}$, S generates $\left(\sum_{r=0}^{3} t^r u_i^{(r)}, \sum_{r=0}^{3} t^r v_i^{(r)}\right)$, and takes $\left(u_i^{(3)}, v_i^{(3)}\right)$ as the share pair for the point at infinity. S executes the same operation to the MAC shares. S repeats this at each child node until it obtains the corrupted parties' leaf input shares $U_i^{(j)}$ and $V_i^{(j)}$. For the virtual honest parties, S keeps the corrupted parties' shares and completes placeholder authenticated sharings of u and v with sum 0, as in the proof of Theorem 5. It then applies the same local evaluation.

Simulation of the leaf multiplications. S independently generates fresh scalar triples for each leaf multiplication, as specified by the preprocessing functionality. It records the allowed high-order differences $\vartheta^{(j)} = C^{(j)} - A^{(j)}B^{(j)}$. Using the current

leaf input shares and triples, S simulates Open for the differences and then performs the integer-ring batched MAC check. Let $\delta^{(j)}$ and $\varepsilon^{(j)}$ denote the deviations of the two full opened differences in Open. Failed checks and deviations of opened values are handled as in the proof of Theorem 5. If the check passes, let $d^{(j)}$ and $e^{(j)}$ denote the Open return values used in the leaf multiplication. Then, S computes the leaf outputs as specified and records the input deviations $\kappa_d^{(j)} = d^{(j)} + A^{(j)}$ and $\kappa_e^{(j)} = e^{(j)} + B^{(j)}$.

Simulation of the reconstruction and exact division. After simulating all leaf multiplications and opening checks, S reconstructs the result from the leaves to the root. It applies Karatsuba's linear reconstruction or the public Toom-4 transformation $8M^{-1}$ to the result shares of the child nodes at each level until it obtains the result at the root. Then, S folds the value shares and MAC shares negacyclically using $X^N = -1$ and let $\overline{W}$ denote the total value of the resulting simulated sharing. Let $\mathcal{L}$ denote the public linear transformation from the leaf products to the N folded coefficients. When $D > 0$, rounding, share refresh, and mask generation follow the proof of Theorem 5. And in the multi-party case, Open_{ξ} and the integer-ring batched MAC check use the current virtual sharings. Let $\overline{Y}$ be the total virtual output value and set $h^{(t)} = \overline{Y}[t] - \frac{\overline{W}[t]}{2^D} \pmod{2^{\mu-D+\tau}}$. In particular, when the two-party branch is taken, $h^{(t)} = 0$. Meanwhile, when $D = 0$, set $\overline{Y} = \overline{W}$ and $h^{(t)} = 0$. If all steps succeed, S submits $\kappa_d^{(j)}, \kappa_e^{(j)}, \vartheta^{(j)}, h^{(t)}$ together with the corrupted parties' reference outputs to $\mathcal{F}_{\text{RingMul}}$ and then sends Proceed. If any step aborts, S immediately sends Abort. It also sends Abort if any full opened value in Open or Open_{ξ} differs from its reference value modulo 2^{μ}.

Then, we argue that the real-world execution and the simulated execution are statistically close. The ideal functionality generates preprocessing material with the same distribution as in the real-world execution and linear operations produce no additional messages. Therefore, it remains to analyze the leaf openings and MAC checks, the batched exact division after folding at the root, and the final output shares.

For each leaf, given the corrupted parties' view before opening, $A^{(j)}$ and $B^{(j)}$ are independent and uniform over the full share ring. By shifting the honest shares of $A^{(j)}$ and $B^{(j)}$ according to the real and placeholder leaf inputs, respectively, the difference shares and MAC shares of both executions can be made equal. Since shifts preserve the distributions, Open and the integer-ring batched MAC check produce identically distributed messages in both executions.

The parties compute the leaf outputs $g_i^{(j)} = C_i^{(j)} + d^{(j)} B_i^{(j)} + e^{(j)} A_i^{(j)} + \mathbf{1}_{i=1} d^{(j)} e^{(j)}$ according to Beaver's rule. Summing the shares with $C^{(j)} = A^{(j)} B^{(j)} + \vartheta^{(j)}$ gives the total value $\left(A^{(j)} + d^{(j)}\right)\left(B^{(j)} + e^{(j)}\right) + \vartheta^{(j)}$. After the shifts mentioned above, the real-world total values of $A^{(j)}$ and $B^{(j)}$ equal the simulated total values plus $U^{(j)}$ and $V^{(j)}$, respectively. And since the simulated execution has $\kappa_d^{(j)} = A^{(j)} + d^{(j)}$ and $\kappa_e^{(j)} = B^{(j)} + e^{(j)}$, keeping $\vartheta^{(j)}$ unchanged yields

$$g^{(j)} = \left(U^{(j)} + \kappa_d^{(j)}\right)\left(V^{(j)} + \kappa_e^{(j)}\right) + \vartheta^{(j)}, \quad \overline{g}^{(j)} = \kappa_d^{(j)} \kappa_e^{(j)} + \vartheta^{(j)}.$$

For the output shares of the leaf multiplication, let Δ denote the difference between the real-world execution and the simulation. Then it yields

$$\Delta g_i^{(j)} = \Delta C_i^{(j)} + d^{(j)} \Delta B_i^{(j)} + e^{(j)} \Delta A_i^{(j)}.$$

The honest $C^{(j)}$ shares were not used in earlier openings or checks, so they remain uniform subject to the sum constraint. Thus, we can choose an honest party P_h and determine its share from the sum constraint. For each other honest party, we set $\Delta C_i^{(j)} = -d^{(j)}\Delta B_i^{(j)} - e^{(j)}\Delta A_i^{(j)}$, so that its leaf output shares agree in both executions. The shifts preserve uniformity, while the respective sums determine the remaining share of P_h. The same holds for the MAC shares. Therefore, among honest parties, only P_h may have different leaf output shares in the two executions. Its value and MAC differences are $g^{(j)} - \overline{g}^{(j)}$ and $\beta\left(g^{(j)} - \overline{g}^{(j)}\right)$, respectively.

Let Bad denote the event that some Open or Open_ξ check in a leaf multiplication or an exact division passes while at least one opened value differs from its reference value by a nonzero amount modulo 2^μ. Then, if the checks pass and no Bad event occurs, the correctness of the low-order bits returned by Open and Open_ξ, together with the triple property, guarantee that $\kappa_d^{(j)}, \kappa_e^{(j)}, \vartheta^{(j)}$ are multiples of 2^μ. Thus, $g^{(j)} \equiv U^{(j)}V^{(j)} \pmod{2^\mu}$ holds. The linear reconstruction preserves divisibility by 2^μ. Therefore, by Proposition 3 and the Karatsuba identity, with $E[t]$ denoting the high-order error of coefficient t, one has

$$W[t] = \mathcal{L}\big((g^{(j)})_j\big)[t] = 2^D\big(uv \bmod (X^N+1)\big)[t] + 2^\mu E[t] \pmod{2^{\mu+\tau}}.$$

Meanwhile, the simulated leaf outputs satisfy $\overline{g}^{(j)} = \kappa_d^{(j)}\kappa_e^{(j)} + \vartheta^{(j)} \equiv 0 \pmod{2^\mu}$. Together with the fact that reconstruction and folding are both linear, it yields $\overline{W}[t] \equiv 0 \pmod{2^\mu}$. Thus, since $D < \mu$, both $W[t]$ and $\overline{W}[t]$ are multiples of 2^D, satisfying the input condition for exact division.

After reconstruction, only P_h may still have different shares. Its value and MAC differences are $W - \overline{W}$ and $\beta\left(W - \overline{W}\right)$, respectively. In the two-party case, the rounding rule given in Fig. 6 divides this difference by 2^D. In the multi-party case, use the same zero-sum masks in both executions. Shift P_h's division mask and MAC share by $\left(W - \overline{W}\right)/2^D$ and $\beta\left(W - \overline{W}\right)/2^D$, respectively, to make the difference shares and MAC shares pending opening match. Hence, both executions produce the same division messages, with output deviation $\left(W - \overline{W}\right)/2^D$. Using the simulator's values $h^{(t)}$, it has

$$Y[t] = \overline{Y}[t] + \frac{W[t] - \overline{W}[t]}{2^D} = \frac{W[t]}{2^D} + h^{(t)} \pmod{2^{\mu-D+\tau}}.$$

The division's low-order correctness yields $h^{(t)} \in 2^{\mu-D}\mathbb{Z}_{2^{\mu-D+\tau}}$, and when no multiparty division is performed, one has $h^{(t)} = 0$. Therefore, the submitted parameters satisfy the functionality requirements, and the full real-world output matches the ideal output, with authentication relations preserved.

The distribution of the honest parties' final output shares needs further explanation. To prove their conditional uniformity, we use X^t and 1 as reference inputs and form the vector $v_t = (v_t^{(j)})_j$, where $v_t^{(j)}$ is the product at leaf j. Then, we use this vector to construct shifts affecting only the shares of coefficient t while preserving the distribution of the random material. By the reconstruction identity, one has $\mathcal{L}(v_t) =$

$8^{\ell}\left((X^t \cdot 1) \bmod (X^N + 1)\right) = 2^D X^t$. Choose any $z \in \mathbb{Z}_{2^{\mu-D+\tau}}$. Whenever this element is used in the larger share ring, use its least nonnegative representative. For each leaf j, if at least two parties are honest, increase one's $C^{(j)}$ value share by $zv_t^{(j)}$ and decrease another's by the same amount. Their reconstructed shares increase and decrease by $2^D z X^t$, respectively. If any multi-party division is executed, also increase and decrease the two parties' value shares of division mask t by z, respectively. The difference shares to be opened remain unchanged. After rounding and reconstruction, only the two parties' shares of coefficient t change, increasing and decreasing by z, respectively. These shifts preserve the messages, functionality parameters, and distribution of the random material, and can also be applied to the MAC shares as well as those of the corresponding division masks independently. Apply these shifts coefficientwise between the fixed honest party and each other honest party. This yields any output shares satisfying the sum constraints. Since these shifts preserve probabilities, every valid combination of output shares occurs with the same probability once the adversary's view and functionality parameters are fixed. Hence, the value shares and MAC shares are independently and uniformly distributed subject to their respective sum constraints, just as in the ideal functionality. In particular, when there is only one honest party, its shares are uniquely determined by the sums and the corrupted parties' shares.

Thus, when no Bad event occurs, both executions have the same joint distribution of messages and outputs, with identical abort behavior. There are at most two batches of checks. Bad occurs if the first batch with a low-order deviation is accepted. For an earlier accepted leaf check, fix the challenges for its m opened values as $\chi^{(1)}, \ldots, \chi^{(m)} \in \mathbb{Z}_{2^\tau}$, and set $c = \gcd\left(2^{\mu+\tau}, \chi^{(1)}, \ldots, \chi^{(m)}\right)$. The linear combination of free honest MAC shares of a fresh triple is uniform over $c\mathbb{Z}_{2^{\mu+\tau}}$. Subtracting key terms in this subgroup preserves the distribution. Once the checks pass, the last honest check share is determined by the other check shares. Thus, earlier accepted leaf checks constrain the key set only through their acceptance conditions. Accounting for all earlier acceptance probabilities and applying the key-query analysis in the proof of the authentication theorem of SPD$\mathbb{Z}_{2^k}$ [9], one has

$$\mathrm{Adv}_Z \le \Pr_{\text{real}}[\mathsf{Bad}] \le (\tau + 1)\, 2^{-\tau}.$$

□

B.5 Security of Π_{Auth}

To characterize the security goal of protocol Π_{Auth}, we define the ideal functionality $\mathcal{F}^R_{\text{Auth}}$ for polynomial-ring MAC generation, which converts coefficientwise authentication over the integer ring into authentication of the polynomial over R.

Functionality $\mathcal{F}^{R}_{\text{Auth}}$

This functionality reuses the existing keys and the state of the authenticated sharings. It accepts authentication requests from all parties for $[\![\widetilde{z}]\!]_{\beta,q+D}$. It sets $z = \widetilde{z} \bmod 2^q$ and $\alpha = \widetilde{\alpha} \bmod 2^q$, then computes $\gamma_z = \alpha z$ in R. After receiving the output shares $(z_i, \gamma_{z,i})_{i\in C}$ from the corrupted parties, it completes the honest parties' value shares and MAC shares independently and uniformly, subject to $\sum_{i=1}^{p} z_i = z$ and $\sum_{i=1}^{p} \gamma_{z,i} = \gamma_z$ in R. It privately delivers $(z_i, \gamma_{z,i}) \in R^2$ to party P_i, giving the authenticated sharing $[\![z]\!]_\alpha$. If it receives Abort from the adversary or an inconsistent or invalid request, it returns $\perp$ to all parties and aborts.

Fig. 14. Ideal functionality for authentication

Proof (of Theorem 7). We prove the security of protocol Π_{Auth} via the UC framework. By leveraging the completeness of the dummy adversary, it is sufficient to construct a simulator S for the dummy adversary in the $(\mathcal{F}_{\text{RingMul}}, \mathcal{F}_{\text{COMMIT}})$-hybrid model. S reuses the simulation of the existing keys and sharing state, and works as follows:

Overall simulation. S simulates the corrupted parties' interaction with $\mathcal{F}_{\text{RingMul}}$, handling the high-order deviations and output shares supplied by the adversary. It returns the adversary-specified shares $(\widetilde{\gamma}_i, \eta_{\widetilde{\gamma},i})_{i\in C}$ to the corrupted parties and records them as integer-ring authenticated shares of the intermediate product. Then, S simulates the corrupted parties' view of generating two independent groups of zero-sum masks. From their reference refresh amounts ξ_i^z and ξ_i^γ, it computes their output coefficientwise as $z_i = (\widetilde{z}_i + \xi_i^z) \bmod 2^q$ and $\gamma_{z,i} = (\widetilde{\gamma}_i + \xi_i^\gamma) \bmod 2^q$ for $i \in C$. If any step aborts, S sends Abort to $\mathcal{F}^{R}_{\text{Auth}}$. Otherwise, S submits these outputs to $\mathcal{F}^{R}_{\text{Auth}}$ and sends Proceed. The functionality then completes and privately delivers the honest parties' outputs.

Then, we argue that in this hybrid model, the two executions have the same distribution. The multiplication functionality returns an authenticated sharing of the product of the real inputs. Since the arithmetic precision is $q + D$, its high-order deviations disappear modulo 2^q. Meanwhile, zero-sum masks preserve the sums modulo 2^q, so the final output satisfies $\sum_i z_i = z, \sum_i \gamma_{z,i} = \widetilde{\gamma} \bmod 2^q = \alpha z$ in R. And this matches the ideal functionality $\mathcal{F}^{R}_{\text{Auth}}$.

The adversary specifies the corrupted shares returned by $\mathcal{F}_{\text{RingMul}}$, and the simulator handles these shares through the same interface. And the refresh messages depend only on fresh randomness, so the simulator generates them as specified. Thus, corrupted parties see identically distributed interactions in both executions. Given the corrupted parties' view, the two groups of honest zero-sum masks are generated with fresh randomness independently and uniformly subject to $\sum_{i\notin C} \xi_i^z = -\sum_{i\in C} \xi_i^z$ and $\sum_{i\notin C} \xi_i^\gamma = -\sum_{i\in C} \xi_i^\gamma$. Hence, the refreshed honest value shares and MAC shares are independently and uniformly subject to their respective sums, which matches the ideal functionality. In particular, when there is only one honest party, the sums and corrupted parties' shares uniquely determine its shares. Moreover, both executions also handle aborts identically. Thus, both executions have the same joint distribution of messages and outputs.

In real-world execution, we invoke Π_{RingMul} once. Therefore, by the UC composition theorem, replacing $\mathcal{F}_{\text{RingMul}}$ with Π_{RingMul} gives an overall distinguishing advantage of at most the bound from Theorem 6, namely

$$\mathrm{Adv}_Z \leq (\tau + 1)2^{-\tau}.$$

□

B.6 Security of Π_{Triple}

To characterize the security goal of protocol Π_{Triple}, we define the ideal functionality $\mathcal{F}^R_{\text{Triple}}$ for polynomial-ring triple generation, which generates authenticated multiplication triples under the same key.

Functionality $\mathcal{F}^R_{\text{Triple}}$

This functionality reuses the existing keys and the state of the authenticated sharings. Once it receives Triple requests from all parties, it samples a and b independently and uniformly from R, and then computes $c = ab$. After receiving the three groups of corrupted output shares and Proceed from the adversary, it keeps those shares. For $z \in \{a, b, c\}$, it completes the honest value shares and MAC shares independently and uniformly subject to $\sum_i z_i = z$ and $\sum_i \gamma_{z,i} = \alpha z$ in R. Then, it privately delivers $\left((a_i, \gamma_{a,i}), (b_i, \gamma_{b,i}), (c_i, \gamma_{c,i})\right)$ to party P_i. If it receives Abort from the adversary or an inconsistent or invalid request, it returns $\perp$ to all parties and aborts.

Fig. 15. Ideal functionality for triple generation

Proof (of Theorem 8). We prove the security of protocol Π_{Triple} via the UC framework. By leveraging the completeness of the dummy adversary, it is sufficient to construct a simulator S for the dummy adversary in the $\left(\mathcal{F}^{\mathbb{Z}}_{\text{Triple}}, \mathcal{F}_{\text{RingMul}}, \mathcal{F}^R_{\text{Auth}}\right)$-hybrid model. S reuses the simulation of the existing keys and sharing state, and works as follows:

Overall simulation. S simulates the corrupted parties' interaction with $\mathcal{F}^{\mathbb{Z}}_{\text{Triple}}$. It records the corrupted parties' integer-ring authenticated shares forming $\widetilde{a}$ and $\widetilde{b}$, reduces the shares before multiplication, and maintains their reference state. Then, S simulates the adversary's interaction with $\mathcal{F}_{\text{RingMul}}$ and submits the adversary's high-order deviations as well as the corrupted output shares. It records the returned $\left(\widetilde{c}_i, \eta_{\widetilde{c},i}\right)_{i\in C}$ as the corrupted parties' integer-ring authenticated shares of $\widetilde{c}$. After that, S reduces the shares before authentication, then invokes $\mathcal{F}^R_{\text{Auth}}$ three times in parallel. For each $z \in \{a, b, c\}$, it simulates the interaction using the corrupted shares specified by the adversary. And it records the returned shares $\left(z_i, \gamma_{z,i}\right)_{i\in C}$ as the corrupted parties' final triple shares. If any step aborts, S sends Abort to $\mathcal{F}^R_{\text{Triple}}$. Otherwise, S submits these outputs to $\mathcal{F}^R_{\text{Triple}}$ and sends Proceed. The functionality then completes and privately delivers the honest parties' outputs.

Then, we argue that in this hybrid model, the two executions have the same distribution. By the sampling rule of $\mathcal{F}^{\mathbb{Z}}_{\text{Triple}}$, the reduced a, b are independent and uniform over R. And since the first ring multiplication uses arithmetic precision $q + 2D$, the value $\widetilde{c}$ returned by $\mathcal{F}_{\text{RingMul}}$ agrees with $\widetilde{a}\widetilde{b}$ in the $q + D$ low-order bits, giving

$c = \widetilde{c} \bmod 2^q = ab$. The subsequent three invocations of $\mathcal{F}_{\text{Auth}}^R$ make the final triple shares satisfy $\sum_i z_i = z$, $\sum_i \gamma_{z,i} = \alpha z$, where $z \in \{a, b, c\}$, in R. This matches the result of the ideal functionality $\mathcal{F}_{\text{Triple}}^R$.

For visible interactions, the corrupted parties' shares of random coefficients are independent of the total values. The adversary can specify their output shares from ring multiplication and authentication. Therefore, such interactions do not leak a or b. Moreover, S uses the same functionality interfaces, so the view of the corrupted parties is distributed identically. For the final triples, according to the output rule of $\mathcal{F}_{\text{Auth}}^R$, when given the corrupted parties' view and the total values above, the three groups of honest value shares and MAC shares are independently and uniformly subject to their respective sum constraints. This also matches the ideal functionality. In particular, when there is only one honest party, the sums and corrupted parties' shares uniquely determine its shares. Moreover, both executions also handle aborts identically. Thus, both executions have the same joint distribution of messages and outputs.

In real-world execution, we invoke Π_{RingMul} once and then make three parallel invocations to Π_{Auth} for a, b, and c. Let Bad denote the event that any check under the common integer MAC key β accepts while the payload of at least one opened value changed. Appendices B.3–B.5 show that, when started from the same reference state, the real and ideal executions of each invocation agree on messages, outputs, and abort behavior until Bad occurs. The preceding paragraphs show that the ideal multiplication and authentication invocations yield exactly the view and output distribution of the ideal execution with $\mathcal{F}_{\text{Triple}}^R$. Before sampling fresh challenges for a batch, the parties fix the order and the opened values of all batches up to and including that batch. The three parallel invocations can be batch-checked separately, but the parties reveal the check shares of the current batch only after all preceding batches have passed their checks. Therefore, if Bad occurs, the first batch containing an incorrect opening must itself have passed, since no later batch can reveal its check shares until then.

For the probability calculation, generate the earlier check messages as if all earlier checks had passed. Appendices B.3 and B.4 show that, even if the reference values are fixed, these messages can be generated independently of any single honest party's share of β. Since the challenges are fresh and at least one key share is uniform, Lemma 1 of [9] applies to the first batch with an incorrect opening. Hence the MAC check equation holds with probability at most $(\tau + 1)2^{-\tau}$. For any finite sequence of batch MAC checks, we partition the auxiliary executions into mutually exclusive cases according to which batch first contains an incorrect opening, if any. By the law of total probability, the probability that any batch accepts an incorrect opening is therefore at most the single-batch bound, so the distinguishing advantage between the two executions satisfies

$$\mathrm{Adv}_Z \le \Pr_{\text{real}}[\mathsf{Bad}] \le (\tau + 1)2^{-\tau}.$$

□