Pseudorandom Correlation Functions from Ring-LWR

Sebastian Hasler, Pascal Reisert, and Ralf Küsters

University of Stuttgart
Stuttgart, Germany
{sebastian.hasler,pascal.reisert,ralf.kuesters}@sec.uni-stuttgart.de

Abstract. State-of-the-art actively secure multiparty computation protocols, like SPDZ (Damgård et al., CRYPTO 2012), use correlated randomness, like Beaver triples, to achieve a highly efficient online phase. For a long time, the generation of the correlated randomness in the offline phase relied on classical cryptographic primitives, like somewhat homomorphic encryption or oblivious transfer, that required significant communication.
More recently, Boyle et al. (FOCS 2020) defined a new primitive called pseudorandom correlation functions (PCFs) to generate correlated randomness non-interactively. PCFs set up keys for each party in an initial interactive phase, which can then be used by the parties to generate a large number of shares of the correlated randomness without further communication. In the random oracle model (ROM), two-party PCFs can be generically constructed based on evaluating a weak pseudorandom function (WPRF) using a powerful-enough homomorphic secret sharing scheme. However, the concrete efficiency of instantiations of this approach has not been analyzed so far. There are also some works that construct PCFs based on other approaches, but they cannot be used for correlations of degree ≥ 2 (e.g., Beaver triples) over large rings/fields (such as those used in SPDZ).
In this paper, we improve the complexity and concrete efficiency of PCFs over large rings/fields by presenting a new generic PCF based on the hardness of the ring-learning with rounding (Ring-LWR) problem and FHE. We only share BFV keys in the initial interactive phase. The two parties then use the random oracle to locally sample BFV (pseudo-)ciphertexts encrypting pseudorandom plaintexts. We use a new bootstrapping algorithm for these (pseudo-)ciphertexts that reduces initially saturated noise to a level where the parties can use the homomorphic properties of the BFV scheme to correlate the encrypted randomness locally. Both parties can then produce, without further interaction, shares of the correlated randomness with their secret key share. Our new PCF works with any form of correlated randomness that can be expressed as an arithmetic circuit over a base ring $\mathbb{Z}_t$ or field $\mathbb{F}_{p^d}$, e.g., Beaver or matrix triples.

Keywords: Pseudorandom Correlation Functions · Multiparty Computation · Ring-LWR · Threshold Fully Homomorphic Encryption

This report is the full version of our ASIACRYPT 2025 publication [HRK25].

1 Introduction

Multiparty computation (MPC) allows several parties to jointly compute a function on private input data without revealing information about the inputs apart from the result and what can be derived from it. Currently, the most efficient MPC protocols that protect against active corruption by even a majority of malicious parties are two-phase protocols, like SPDZ [DPSZ12] and its variants [KOS16, KPR18]. These protocols consist of an offline phase in which (structured) random data, classically in the form of Beaver triples [Bea92], is precomputed and an online phase in which the structured randomness is used to compute the desired output from the private inputs. The two-phase approach results in a fast online phase and an offline phase that is often magnitudes slower. However, the slower offline phase is usually considered acceptable since preprocessing may start long before the private inputs become available. Unfortunately, some of the fastest offline phases, like [KOS16, KPR18], do not only have a long runtime, but also require the parties to exchange large amounts of data. The resulting traffic further slows down the MPC computation in networks with low bandwidth. Additionally, it leads to high financial costs if the MPC protocol is deployed on commercial cloud servers.

A major breakthrough in reducing communication costs was suggested by Boyle et al. [BCGI18, BCG+19b, BCG+20c], who introduced a new primitive called pseudorandom correlation generator (PCG), which compresses a large target correlation, e.g., millions of Beaver triples, into small secret seeds. The size of the seeds is sublinear in the size of the target correlation, and the seeds can be generated with low communication costs in an initial interactive phase. Once each party has a seed, they can *expand* them into large batches of correlated randomness locally, i.e., without further communication. A positive side effect of this design principle is that the correlated randomness can be created locally shortly before the online phase starts and does not have to be stored as in previous approaches like [DPSZ12, KPR18]. The security of the PCGs of Boyle et al. are based on function secret sharing (FSS) [BGI15, BGI16b] and variants of the coding-theoretic learning parity with noise (LPN) assumption, e.g., Ring-LPN in the state-of-the-art PCG from [BCG+20c]. While the hardness of the LPN-variants has been conjectured and seems reasonable, a reduction from some standard lattice problem is, to the best of our knowledge, still missing.

One drawback of PCGs is that per secret seed, they can still only produce a certain (although comparably large) amount of correlated random numbers. A natural extension of PCGs, which solve that problem, are pseudorandom correlation functions (PCFs) [BCG+20a], where one seed can be expanded to a virtually unlimited amount of correlated randomness, i.e., PCF can be thought of as a PCG with constant seed size and communication cost. Boyle et al. [BCG+20a] suggest two PCF constructions: Firstly, a generic PCF construction based on any homomorphic secret sharing (HSS) [BGI16a] or FSS scheme that allows to evaluate a weak pseudorandom function (WPRF), and secondly, a construction that extends their PCG construction and relies on similar primitives. The latter is incompatible with the large plaintext moduli usually used in MPC protocols like SPDZ [CD23]. In this work, we therefore focus on the generic PCF approach, which is the only known way to obtain PCFs for correlations of degree ≥ 2 (such as Beaver triples) over large plaintext moduli. We present a new instantiation based on threshold FHE and a WPRF based on ring-learning with rounding (Ring-LWR) [BPR12] whose security follows from the hardness of standard Ring-LWE. Our resulting PCF allows the generation of arbitrary additively shared correlations that can be computed as an arithmetic circuit over a base field $\mathbb{F}_{p^k}$ or base ring $\mathbb{Z}_t$. We want to briefly outline the generic PCF construction and our optimizations.

Our PCF. First, recall that in an HSS scheme for a class of programs $\mathcal{Q}$ each party P_σ receives a share $\mathsf{share}_\sigma(k)$ such that $\mathsf{Eval}(\mathsf{share}_0(k), Q) + \mathsf{Eval}(\mathsf{share}_1(k), Q) = Q(k)$ holds (with high probability) for a public evaluation algorithm Eval, all $Q \in \mathcal{Q}$ and all inputs k to

Q (cf. [BGI16a] for details). Given an HSS scheme sufficiently powerful to evaluate a WPRF followed by a correlation, one can generically construct PCFs. Namely, let F_k be a WPRF with key k, let $\mathcal{C}$ be the circuit that computes the target correlation in plain (e.g., $\mathcal{C}(a,b) = (a,b,ab)$ in the case of Beaver triples), and let $\mathcal{Q} = \{Q_x \mid x \text{ input to } F_k\}$ with $Q_x(k) := \mathcal{C}(F_k(x))$ be the class of programs. Given shares $\mathsf{share}_\sigma(k)$, each party can compute $\mathsf{Eval}(\mathsf{share}_\sigma(k), Q_x)$, which is by definition (with high probability) an additive share of $Q_x(k) = \mathcal{C}(F_k(x))$, i.e., of the target correlation.[1]

The HSS scheme can be instantiated using a suitable threshold fully homomorphic encryption (FHE) scheme $(\mathsf{Enc}, \mathsf{Dec})$ with secret key s via $\mathsf{share}_\sigma(k) := (\mathsf{ct}_k, s_\sigma)$, where ct_k is an encryption of k and s_σ an additive share of s, and $\mathsf{Eval}(\mathsf{share}_\sigma(k), Q_x) := \mathsf{Dec}_{s_\sigma}(\mathcal{C}(F_{\mathsf{ct}_k}(x)))$. It is not directly obvious that decryption with a key share s_σ yields a useful result, but in the two-party setting and if the ciphertext's noise is small enough, this decryption yields an additive share of $\mathsf{Dec}_s(\mathcal{C}(F_{\mathsf{ct}_k}(x))) = \mathcal{C}(F_k(x))$ (cf. [DHRW16] or our Corollary 7.1), i.e., an additive share of the target correlation. Hence, in the two-party setting, it is enough to secret-share the key s and distribute an encryption of k to generate additively shared correlations. For instance, in the case of Beaver triples, the parties first get from the homomorphic evaluation of $\mathcal{C}(F_{\mathsf{ct}_k})$ ciphertexts $\mathsf{ct}_a, \mathsf{ct}_b, \mathsf{ct}_{ab}$ with underlying plaintexts a, b, ab for secret random a, b. Then, by applying Dec_{s_σ}, party P_σ obtains shares $[a]_\sigma, [b]_\sigma, [ab]_\sigma$ such that $[a]_0 + [a]_1 = a, [b]_0 + [b]_1 = b, [ab]_0 + [ab]_1 = ab$.

We instantiate this generic construction with a threshold variant of BFV. In an initial interactive phase the parties use standard MPC protocols, e.g., [DPSZ12, Kel20], to securely generate the key pair $(s, \mathsf{pk}, \mathsf{evk})$ for the BFV encryption scheme (where evk is the evaluation key required for homomorphic operations) and to provide each party P_σ for $\sigma = 0, 1$ with an additive share s_σ of s. The generic approach outlined above then provides shares of BFV plaintexts. To use them with MPC protocols like SPDZ, a plaintext should correspond to a tuple of ring elements in $\mathbb{Z}_t$ (or field elements in $\mathbb{F}_t$) for large modulus, often $t \approx 128$ bits [Kel20]. We therefore choose standard cyclotomic rings with modulus t as plaintext space. However, as discussed below and in Section 6, the large modulus t makes certain BFV techniques (e.g., established bootstrapping algorithms) impractical.

We next want to explain our choice of a WPRF F_k and how it allows us to improve over the generic approach described above. Recall that, in the generic PCF approach, the homomorphic evaluation of $\mathcal{C}(F_{\mathsf{ct}_k}(x))$ required an encryption of a secret WPRF key k. In particular, k and its encryption have to be generated securely first. We show that by using a special Ring-LWR-based WPRF, the BFV decryption key and the WPRF key coincide. Since an encryption of the decryption key is contained in evk, our construction does not require a separate encryption of the WPRF key. A party only needs a share of the secret key (plus public inputs like pk or evk) to compute a share of the target correlation.

In the following, we illustrate our construction in more detail for a Beaver triple (a, b, ab). Firstly, each party takes the same random r, r', e.g., from the random oracle or other source of public randomness. A party then locally generates BFV-type ciphertexts $\mathsf{ct} = (0, r), \mathsf{ct}' = (0, r')$. While $\mathsf{ct}, \mathsf{ct}'$ are not created by the BFV encryption algorithm, they nevertheless share many of the properties of BFV ciphertexts. We therefore call $\mathsf{ct}, \mathsf{ct}'$ *pseudo-ciphertexts*. Our pseudo-ciphertexts can be decrypted to define a function $F_s(r) := \mathsf{BFV.Dec}_s((0, r))$ for each secret key s. We show that $F_s(r)$ corresponds to a Ring-LWR sample and, from the Ring-LWR assumption [BPR12], we then deduce that $F_s(r)$ is a WPRF with key s (cf. Section 5). Hence, the parties have constructed pseudo-ciphertexts $\mathsf{ct}, \mathsf{ct}'$ with underlying (pseudo)random plaintexts $a = F_s(r)$, $b = F_s(r')$, without access to s.

[1] Note that this HSS scheme naturally corresponds to an FSS scheme for the class $\{x \mapsto \mathcal{C}(F_k(x)) \mid k\}$ [BGI16a].

Unfortunately, pseudo-ciphertext noise can be too large for homomorphic operations. In particular, the product $\mathsf{ct} \cdot \mathsf{ct}'$ is generally not a ciphertext of ab for our pseudo-ciphertexts, in contrast to fresh BFV ciphertexts (and suitably chosen BFV parameters). To nevertheless use the homomorphic properties of BFV, we therefore have to reduce the noise in ct and ct' such that homomorphic multiplication no longer leads to a noise overflow. For noise reduction in BFV (and CKKS) ciphertexts, there are well-established bootstrapping protocols [HS15, CH18, GV23, GIKV23, OPP23, KSS24]. However, these protocols are either not efficient for the large plaintext sizes used in secret-sharing-based MPC protocols [HS15, CH18, GV23, GIKV23, OPP23], or not applicable to the large noise of our pseudo-ciphertexts [KSS24]. We, therefore, construct a new bootstrapping for BFV, which combines bootstrapping techniques from BFV and CKKS to overcome both of these limitations.

After bootstrapping, we can use the standard homomorphic properties of the BFV scheme and multiply $\mathsf{ct}_{\mathsf{out}} \cdot \mathsf{ct}'_{\mathsf{out}}$ to get a ciphertext $\mathsf{ct}''_{\mathsf{out}}$ that encrypts ab. Both parties perform the ciphertext operations locally in parallel and receive the same $\mathsf{ct}_{\mathsf{out}}, \mathsf{ct}'_{\mathsf{out}}, \mathsf{ct}''_{\mathsf{out}}$. Finally, both parties use their share of the decryption key to obtain shares $[a]_i, [b]_i, [ab]_i$ using the result from [DHRW16]. Analogously, the parties can construct other correlations. They can also generate authenticated shares.

Overall, we get a new PCF construction in the random oracle model (ROM) based on Ring-LWR and standard FHE schemes, namely BFV and CKKS. Our PCF allows two MPC parties (after the interactive key generation) to locally generate all correlated randomness needed for the intended MPC online computation without further communication. The security of our PCF follows from the security of the underlying primitives, which can all be reduced to the hardness of Ring-LWE. For the reduction from Ring-LWR to Ring-LWE we present an optimization of a result from [LW20], which allows us to deduce secure and efficient parameters for our PCF (see Table 2 in Section 4).

Contributions.

- We construct a new two-party PCF to generate shares of arbitrarily correlated elements in $\mathbb{Z}_t$ or $\mathbb{F}_{p^d}$ such as authenticated Beaver triples, matrix triples, and other forms of correlated random data used in MPC protocols like SPDZ (Section 7). Our PCF outperforms a generic AES-128-based PCF by 55–116×.
- We show that the Ring-LWR-based WPRF [BPR12] can be evaluated under FHE with $\tilde{\mathcal{O}}(\lambda^{4/3})$ ciphertext operations – compared to $\tilde{\mathcal{O}}(\lambda^2)$ for other WPRF candidates based on standard assumptions [NR04, MV12] (Section 6). Our evaluation strategy outperforms the naive way to evaluate this WPRF by 543–808×.
- We present a new reduction from Ring-LWE to Ring-LWR compatible with our generation of correlated randomness with BFV (Section 4).
- We give a new *key switching* algorithm that replaces the key of a Ring-LWR sample with a BFV key to make it compatible with BFV bootstrapping and further homomorphic operations on ciphertexts. In contrast to the conventional key switching of BFV, our algorithm does not require an a priori bound on the noise (Section 5).
- We construct a new BFV bootstrapping algorithm, which might be of independent interest. In contrast to prior work, our bootstrapping works for arbitrary plaintext modulus *and* does not require an a priori bound on the noise (Section 6).

2 Related Work

This paper presents a new method to generate correlated randomness for state-of-the-art two-phase MPC protocols like SPDZ [DPSZ12] and related protocols, e.g., [KOS16, KPR18,

BCS19, RRKK23]. Therefore, we briefly recall different forms of correlated randomness used in these protocols before discussing their generation. We then focus on PCFs.

Classically, two-phase MPC protocols like SPDZ use Beaver triples [Bea92] to multiply secret values. For squares of a secret number [DKL+13] or polynomials [ON20, RRK+24] there are specialized correlated tuples that either come with a smaller size or allow for an online evaluation with less communication. Similarly, there are matrix triples [MZ17, CKR+20, MG23, RRKK23] that speed up online matrix multiplications and convolution triples [CKR+20, RRHK23] to compute tensor convolutions. Furthermore, there are variants of these correlated randomnesses that result from function-dependent preprocessing, e.g., [BNO19, PSSY21].

Standard constructions for Beaver triples in the offline phase use somewhat homomorphic encryption [DPSZ12], oblivious transfer (OT) [KOS16] or linear homomorphic encryption [KPR18]. Many improvements to these protocols have been suggested, e.g., [BCS19, BNO19, PSSY21, RRKK23]. For other correlations, like matrix multiplication or tensor convolutions, optimized generation protocols exist, see, e.g., [CKR+20, RRKK23, RRHK23]. For a base ring $\mathbb{Z}_{2^k}$ (instead of a field $\mathbb{F}_t$), variants of these protocols exist, e.g., [CDE+18, OSV20, HRRK24]. We also present variants of our protocol over these base rings in Appendix C.

As mentioned in Section 1, previous construction methods usually require the parties to exchange a large amount of data while generating correlated randomness. Two different approaches in the two-party setup, which significantly reduce the communication cost, are the PCGs and PCFs already introduced in the introduction. We want to discuss the variants of these constructions in more detail. The PCGs of Boyle et al. [BCGI18, BCG+19b, BCG+20c] use the LPN assumption and FSS for point functions [BGI15, BGI16b] to generate different forms of correlated randomness. There are very efficient constructions for OT and vector oblivious linear evaluation (VOLE) correlations [BCG+19a, SGRR19, BCG+20c, YWL+20, BCG+23]. Many of these constructions use the natural compatibility of FSS and LPN with linear operations. Unfortunately, they do not directly transfer to the generation of (authenticated) Beaver triples needed for actively secure MPC in the arithmetic setting. Other recent constructions, like [BBC+24], only address small base fields like $\mathbb{F}_2$. For Beaver triples over large fields, the actively secure PCGs [BCG+20c, BCCD23, LXYY25] are all based on variants of the Ring-LPN assumption, which currently do not have a reduction from a standard (ideal) lattice problem.

Boyle et al. additionally introduce in [BCG+20a] PCFs as an extension of PCGs. PCFs can output a virtually unbounded number of correlated samples after a one-time seed/key generation (see [BCG+20b, Definition 4.3] or Definition 10 for details). Many existing PCF constructions, including ours, have a *universal setup*, meaning that the correlation doesn't need to be known at the time of key generation, and the same key can be used for arbitrary correlations in the future. Boyle et al. also present a PCF construction based on FSS and a variable-density LPN (VDLPN) assumption. However, that construction mainly applies to correlations over $\mathbb{F}_2$ and doesn't generalize to larger rings due to two problems: Firstly, the security analysis in [BCG+20a, CD23] of VDLPN doesn't generalize to rings other than $\mathbb{F}_2$.[2] Secondly, even if VDLPN over general rings is secure, only the PCF for VOLE from [BCG+20a] generalizes. In particular, the PCF from [BCG+20a, Theorem 6.7] for arbitrary degree-d ($d \geq 2$) correlations (such as Beaver triples) is exclusively for $\mathbb{F}_2$ and doesn't seem to generalize. A different PCF approach is introduced in [OSY21] and builds on the homomorphic properties of the Paillier cryptosystem, but it only provides a PCF for OT and VOLE relations, not for (authenticated) Beaver triples. Finally, [BCM+24, CDD+24]

[2] The authors of [BCG+20a] claim that VDLPN over general rings is at least resistant against "linear attacks", but the proof was flawed and the fix in [CD23] only handles $\mathbb{F}_2$.

Table 1: Comparison of WPRF candidates for the generic PCF approach. Here $k := \log t$ denotes the number of bits of the MPC plaintext space.

	Output range	Circuit size	Ciphertext ops per $\mathcal{R}_t$-output	
SPN candidate [MV12, Cand. 2]	$\{0,1\}^\lambda$	$\tilde{\mathcal{O}}(\lambda)$	$\tilde{\mathcal{O}}(k)$	
Mixed linear functions [BIP+18]	$\mathbb{Z}_3$	$\tilde{\mathcal{O}}(\lambda)$	$\tilde{\mathcal{O}}(\lambda k)$	
VDLPN-based candidate [BCG+20a]	$\{0,1\}$	$\tilde{\mathcal{O}}(\lambda^3)$	$\tilde{\mathcal{O}}(\lambda^3 k)$	
Number-theoretic [NR04]	$\mathbb{F}_t^*$	$\tilde{\mathcal{O}}(\lambda k)$	$\tilde{\mathcal{O}}(\lambda k)$	standard assumptions
AES generalization [MV12, Cand. 5]	$\{0,1\}^\lambda$	$\tilde{\mathcal{O}}(\lambda^2)$	$\tilde{\mathcal{O}}(\lambda k)$	
Ring-LWR WPRF [BPR12] (naive)	$\mathcal{R}_t$	$\tilde{\mathcal{O}}(\lambda^2 k)$	$\tilde{\mathcal{O}}(\lambda k)$	
Ours (Section 5.1)	$\mathcal{R}_t$	—	$\tilde{\mathcal{O}}(\lambda^{4/3})$	

construct PCFs for (List)OT based on constrained pseudorandom functions (CPRFs). This technique so far doesn't yield PCFs over large rings/fields.

PCFs can also be generically constructed from group-based HSS schemes [BGI16a,BGI+18, BKS19,ILM21,CMPR23,ARS24,CDH+25], instead of the FHE-based HSS scheme we use. To make the circuit Q_x from the introduction usable with these group-based HSS schemes, one first needs to convert Q_x into a branching program and then into a restricted-multiplication straight-line (RMS) program (cf. [BGI16a]). However, this conversion increases the number of ciphertext operations by a polynomial factor[3] – this blowup cannot be compensated by the faster ciphertext operations available in group-based schemes.

Generic PCF Construction. Due to the abovementioned shortcomings of specialized PCF schemes, we revisit the generic PCF approach based on evaluating a suitable WPRF under threshold FHE [DHRW16, BCG+20a, COS+22]. That approach requires a low-complexity circuit that takes as input a WPRF key (and public randomness) and computes pseudorandomness following the target correlation. WPRFs based on standard assumptions such as the number-theoretic construction from [NR04], an AES generalization [MV12, Candidate 5] or the Ring-LWR-based WPRF [BPR12] have a circuit of size $\tilde{\mathcal{O}}(\lambda^2)$. Furthermore, we are aware of the three WPRF candidates with quasi-optimal circuit size (i.e., $\tilde{\mathcal{O}}(\lambda)$ in the security parameter λ) listed in Table 1 and discussed below:

(i) *SPN candidate [MV12, Candidate 2].* This candidate is based on Substitution Permutation Networks (SPNs) and is restricted to binary outputs.

(ii) *Mixed linear functions candidate [BIP+18].* The WPRF of [BIP+18] mixes linear functions over $\mathbb{F}_2$ and $\mathbb{F}_3$. By [BIP+18, Remark 6.7], it can be plausibly instantiated with linear size and constant depth. This translates to *quasi*-linear size and logarithmic depth under FHE due to bounded fan-in of FHE operations. However, its security relies on several non-standard conjectures (cf. [BIP+18]). Moreover, a generalization to prime fields larger than $\mathbb{F}_3$ has been suggested [BIP+18, Remark 3.2] but not studied. In particular, it is unclear whether efficient instantiations are possible.

(iii) *VDLPN-based candidate [BCG+20a].* The WPRF of [BCG+20a] is based on VDLPN. However, as mentioned, the existing security analysis doesn't generalize to rings other

[3] A boolean circuit of depth d with fan-in 2 turns into a branching program of length 4^d [Bar86,HDW21]. For our circuit $Q_x(k) = \mathcal{C}(F_k(x))$, both the WPRF ($F_k$) and the correlation's circuit ($\mathcal{C}$) contribute to d. Note that, as the conversion requires a boolean circuit, every sequential ring/field operation in $\mathcal{C}$ contributes with a logarithmic term (in the ring/field size) to d.

than $\mathbb{F}_2$. Furthermore, even in the $\mathbb{F}_2$ case, the circuit size of this candidate is $\mathcal{O}(\lambda^3)$ when instantiated with parameters that plausibly lead to λ bits of security. Note that the circuit size of $\tilde{\mathcal{O}}(\lambda)$ mentioned in [BCG+20a] is based on weaker parameter choices whose security has not been analyzed.

We see that the low-complexity WPRF candidates in (i)–(iii) do not generalize well to large rings.[4] Moreover, they are not based on established standard assumptions. Instead of the WPRFs in (i)–(iii), we use the Ring-LWR-based WPRF from [BPR12] that works natively over a large ring with its security based on Ring-LWE. However, instead of evaluating its boolean circuit of size $\tilde{\mathcal{O}}(\lambda^2 \log t)$ directly, we show that this WPRF can be evaluated using $\tilde{\mathcal{O}}(\lambda^{4/3})$ ciphertext operations. Our circuit utilizes non-blackbox techniques on FHE, such as our pseudo-ciphertexts and reinterpreting ciphertexts between different FHE schemes.

3 Preliminaries

Notation. We denote vectors by bold letters ($\boldsymbol{v}$) and matrices by capital letters (A). For $a \in \mathbb{R}/q\mathbb{Z}$, we denote by $[a]_q$ the unique representative $[a]_q \in (a + q\mathbb{Z}) \cap [-q/2, q/2)$. We denote the uniform distribution over a finite set Ω as $U(\Omega)$.

3.1 LWE and LWR

The well-known (search) learning with errors (LWE) problem is to distinguish a pair $(A, A\boldsymbol{s} + \boldsymbol{e}) \in \mathbb{Z}_q^{\ell \times n} \times \mathbb{Z}_q^{\ell}$ for secret vector $\boldsymbol{s}$, short random error vector $\boldsymbol{e}$ and uniformly random matrix A, from the uniform distribution over the same space. For details, we refer to [Reg05]. In [BPR12], the learning with rounding (LWR) problem was introduced, which can be thought of as a variant of LWE with deterministically generated error $\boldsymbol{e}$. Namely, instead of adding an error $\boldsymbol{e}$ drawn from some distribution, the entries of $A\boldsymbol{s}$ get deterministically rounded to the nearest multiple of q/t for some $t < q$. The idea is that this rounding hides the exact value of $A\boldsymbol{s}$, similar to the role of the error in LWE. As there are t possible values in $\mathbb{Z}_q$ to round to, the result can alternatively be seen as a vector with entries in $\mathbb{Z}_t$, computed as $\lfloor (t/q)A\boldsymbol{s} \rceil \bmod t$.

Reduction from LWE to LWR. In [BPR12], it was shown that when q/t is super-polynomial in the norm of the LWE errors $\boldsymbol{e} \leftarrow \chi^N$, then LWR is at least as hard as LWE for the same q. This is a relatively simple observation, as adding noise sampled from χ^N to $A\boldsymbol{s}$ would make LWR at least as hard as LWE, and this added noise would only change the rounding result with negligible probability if q/t is super-polynomial. Later works showed that the ratio q/t can be kept polynomial, both for plain LWR [AKPW13,BLR+18,BGM+16] as well as for its ring variant [LW20], which is defined (in Section 3.3) over the ring of integers of a cyclotomic field.

3.2 Cyclotomic Number Fields

As this work is based on Ring-LWR, we first need to introduce some number theoretic preliminaries. We refer to [LPR13] for details.

For $m \in \mathbb{N}$, $\zeta_m \in \mathbb{C}$ denotes a fixed primitive complex m-th root of unity and $\Phi_m \in \mathbb{Z}[X]$ denotes the m-th cyclotomic polynomial. The corresponding m-th cyclotomic field is denoted as $\mathcal{K} := \mathbb{Q}[\zeta_m] \cong \mathbb{Q}[X]/(\Phi_m)$ and contains the ring of integers $\mathcal{R} := \mathbb{Z}[\zeta_m] \cong \mathbb{Z}[X]/(\Phi_m)$. Also recall that $\mathcal{K}$ has degree $\varphi(m) = \deg(\Phi_m)$ over $\mathbb{Q}$ and φ is the Euler's totient function.

[4] While it is possible to compose multiple binary outputs into a pseudorandom element over $\mathbb{Z}_{2^k}$, this blows up the circuit size by a factor of k where $k \approx \lambda$ in arithmetic MPC. This is reflected in the last column of Table 1.

Definition 1 (Canonical Embedding). *For each $i \in \mathbb{Z}_m^*$, there is a ring automorphism $\sigma_i : \mathcal{K} \to \mathcal{K}$ that maps ζ_m to ζ_m^i and fixes the base field $\mathbb{Q}$. The canonical embedding $\sigma : \mathcal{K} \to \mathbb{C}^{\mathbb{Z}_m^*}$ is defined by $\sigma(a) := (\sigma_i(a))_{i \in \mathbb{Z}_m^*}$. The* trace *$Tr := Tr_{\mathcal{K}/\mathbb{Q}} : \mathcal{K} \to \mathbb{Q}, a \mapsto Tr(m_a)$ for the linear maps $m_a : \mathcal{K} \to \mathcal{K}, x \mapsto ax$ can be written in terms of the embedding maps as $Tr(a) = \sum_{i \in \mathbb{Z}_m^*} \sigma_i(a)$.*

Fractional Ideals. If $d\mathcal{I} \subset \mathcal{R}$ is an ideal for some $d \in \mathcal{R} \setminus \{0\}$ then we call $\mathcal{I} \subset \mathcal{K}$ a *fractional ideal* of $\mathcal{R}$. Every fractional ideal $\mathcal{I} \subset \mathcal{K}$ has full rank, i.e., it has a $\mathbb{Z}$-basis of size $\varphi(m)$. The dual of a fractional ideal $\mathcal{I} \subset \mathcal{K}$ is defined as $\mathcal{I}^\vee := \{a \in \mathcal{K} \mid \mathrm{Tr}(a\mathcal{I}) \subseteq \mathbb{Z}\}$ where Tr is the trace on $\mathcal{K}$. It holds that $(\mathcal{I}^\vee)^\vee = \mathcal{I}$.

In the special case $\mathcal{I} = \mathcal{R}$, we have $\mathcal{R}^\vee$ is generated by $\frac{1}{m}\prod_{p|m,p \text{ prime}}(1 - \zeta_p)$. Since $\zeta_2 = -1$ we can also rewrite $\mathcal{R}^\vee = \langle g/\hat{m} \rangle$ for $\hat{m} = m/2$ if m is even or $\hat{m} = m$ if m is odd, and $g = \prod_p(1 - \zeta_p) \in \mathcal{R}$ where the product runs over all *odd* primes that divide m (cf. [LPR13, Corollary 2.18]).

Bases. Let $\boldsymbol{b}$ be a $\mathbb{Q}$-basis of $\mathcal{K}$. Then each element $x \in \mathcal{K}$ can be written as $x = \langle \boldsymbol{z}, \boldsymbol{b} \rangle$ for some unique $\boldsymbol{z} \in \mathbb{Q}^{\varphi(m)}$. We denote this $\boldsymbol{z}$ by $\boldsymbol{\lambda}_x^{\boldsymbol{b}}$. The dual basis $\boldsymbol{b}^\vee$ is defined as the unique basis such that $\mathrm{Tr}(b_i b_j^\vee)$ is 1 if $i = j$ and 0 otherwise. If $\boldsymbol{b}$ is a $\mathbb{Z}$-basis of a fractional ideal $\mathcal{I} \subset \mathcal{K}$, then $\boldsymbol{b}^\vee$ is a $\mathbb{Z}$-basis of $\mathcal{I}^\vee$.

In $\mathcal{K}$, rounding and centered modular reduction need to be defined w.r.t. some basis $\boldsymbol{b}$. Let $\mathcal{I} = \mathrm{span}(\boldsymbol{b})$. The rounding function is defined as $\lfloor \cdot \rceil^{\boldsymbol{b}} : \mathcal{K} \to \mathcal{I}, x \mapsto \langle \lfloor \boldsymbol{\lambda}_x^{\boldsymbol{b}} \rceil, \boldsymbol{b} \rangle$. Next, we define the following notation (used in our Ring-LWR definition) that combines downscaling with rounding:

$$\lfloor \cdot \rceil_{q \to t}^{\boldsymbol{b}} : \mathcal{I}_q \to \mathcal{I}_t, \quad x + q\mathcal{I} \mapsto \langle \lfloor (t/q)\boldsymbol{\lambda}_x^{\boldsymbol{b}} \rceil, \boldsymbol{b} \rangle + t\mathcal{I}.$$

We define the centered modular reduction by $[\cdot]_q^{\boldsymbol{b}} : \mathcal{K}/q\mathcal{I} \to \mathcal{K}, x \mapsto \langle [\boldsymbol{\lambda}_x^{\boldsymbol{b}}]_q, \boldsymbol{b} \rangle$.

In the following, we define important bases of $\mathcal{K}$, $\mathcal{R}$, and $\mathcal{R}^\vee$.

Definition 2 (Powerful Basis). *As in [LPR13, Definition 4.1], define the powerful basis $\boldsymbol{p}$ of $\mathcal{K} = \mathbb{Q}(\zeta_m)$ and $\mathcal{R} = \mathbb{Z}[\zeta_m]$ as follows:*

- *For prime power $m = p^e$, $\boldsymbol{p}$ is the power basis $(\zeta_m^i)_{[i \in \varphi(m)]}$. Alternatively this can be written as $\boldsymbol{p} = (\zeta_p^{i_0}\zeta_m^{i_1})_{(i_0,i_1) \in [\varphi(p)] \times [m/p]}$.*
- *for m with prime power factorization $m = \prod_\ell m_\ell$, $\boldsymbol{p} = \bigotimes_\ell \boldsymbol{p}_{m_\ell}$ where p_{m_ℓ} is the powerful basis of $\mathbb{Z}[\zeta_{m_\ell}]$.*

Define the powerful basis $\hat{\boldsymbol{p}}$ of $\mathcal{R}^\vee$ to be $(g/\hat{m})\boldsymbol{p}$. (Here, we don't use the $^\vee$ notation since $\hat{\boldsymbol{p}}$ is not the dual basis of $\boldsymbol{p}$). Each element p_i of the powerful basis $\boldsymbol{p}$ has length $||\sigma(p_i)||_2 = \sqrt{\varphi(m)}$ under the canonical embedding [LPR13].

For certain tasks, having a basis of $\mathcal{R}^\vee$ whose dual is optimally short is useful. This is the so-called decoding basis [LPR13]:

Definition 3 (Decoding Basis). *The decoding basis of $\mathcal{R}^\vee$ is $\boldsymbol{d} := \tau(\boldsymbol{p})^\vee$ where the automorphism $\tau = \sigma_{-1}$ (that maps ζ_m to ζ_m^{-1}) is applied entrywise.*

The decoding basis is defined as $\tau(\boldsymbol{p})^\vee$ instead of $\boldsymbol{p}^\vee$ following the convention in [LPR13, Definition 6.1]. We then get for the decoding basis $\boldsymbol{d} = (g/\hat{m})L_m^T\boldsymbol{p}$ [LPR13, Section 6.1] with $L_m = \bigotimes_p(L_p \otimes I_{[m/p]})$, where the tensor product runs over all primes p that divide m and, for prime p, $L_p \in \{0,1\}^{\varphi(p) \times \varphi(p)}$ is the lower triangular matrix filled with 1s, i.e., the (i,j)th entry is 1 iff $i \geq j$.

Remark 1. When m is a power of two, both bases $\hat{\boldsymbol{p}} = \boldsymbol{d} = (g/\hat{m})L_m^T\boldsymbol{p}$ are the same, since L_m collapses to the identity matrix.

Norms. We use the powerful basis to endow $\mathcal{K}$ with the usual norms. That is, for $a \in \mathcal{K}$ and $l \in \{1, \infty\}$, we define $||a||_l = ||\boldsymbol{\lambda}_a^{\boldsymbol{p}}||_l$. Furthermore, we define the *expansion factor* $\gamma_l := \max\{||ab||_\infty / (||a||_\infty \cdot ||b||_l) \mid a, b \in \mathcal{R}\}$. Note that this differs from prior work (e.g., [Gen09]) where the expansion factor is defined based on the 2-norm instead. In our work, we deal with saturated noise (see Section 5.1), which is better captured by a bound on the ∞-norm, while the 2-norm is more suitable for spherical noise. By Lemma B.6 in the appendix, the factor γ_1 is 2^ω where ω is the number of *odd* primes that divide m. For instance, $\gamma_1 = 1$ for power-of-two m. For γ_∞, we have $\gamma_\infty \leq \varphi(m)\gamma_1$.

3.3 Ring-LWE and Ring-LWR

The $\mathcal{R}$-LWE problem [LPR10] is defined as follows. Let $\mathcal{K}_\mathbb{R} := \mathbb{R}[\zeta_m]$.

Definition 4 ($\mathcal{R}$-LWE). *Let $s \in \mathcal{R}_q$, let $q \in \mathbb{N}$, and let ϕ be a distribution over $\mathcal{R}_q^\vee$. The $\mathcal{R}$-LWE$_{s,q,\phi}$ distribution is*

$$\{(a, as + e + q\mathcal{R}^\vee) \in \mathcal{R}_q \times (\mathcal{K}_\mathbb{R}/q\mathcal{R}^\vee) : a \xleftarrow{\$} \mathcal{R}_q, e \leftarrow \phi\}.$$

- *The decision problem D-$\mathcal{R}$-LWE$_{q,\phi,\ell}$ is to distinguish between ℓ independent samples from $\mathcal{R}$-LWE$_{s,q,\phi}$ for $s \xleftarrow{\$} \mathcal{R}_q$, and the same number of independent samples from $U(\mathcal{R}_q \times (\mathcal{K}_\mathbb{R}/q\mathcal{R}^\vee))$.*
- *The search problem S-$\mathcal{R}$-LWE$_{q,\phi,\ell}$ is, given ℓ independent samples from $\mathcal{R}$-LWE$_{s,q,\phi}$ for $s \xleftarrow{\$} \mathcal{R}_q$, to find s.*

Cryptographic schemes based on $\mathcal{R}$-LWE are more efficient compared to the previously known LWE-based schemes because $\mathcal{R}_q$ decomposes nicely into copies of $\mathbb{Z}_q$ or $\mathbb{F}_{q^d}$ depending on q. In particular, one instance of $\mathcal{R}$-LWE provides pseudorandomness in each of these factors in a SIMD fashion. Similarly to LWE, the LWR problem can be transferred to the ring setting (as already proposed in [BPR12]) to achieve similar efficiency gains for LWR-based cryptography. In our work, we need this ring variant of LWR (called $\mathcal{R}$-LWR) to exploit (in Section 5.1) a similarity between $\mathcal{R}$-LWR and the decryption function of the BFV scheme.

Recall that, in LWR, the term $A\boldsymbol{s}$ gets downscaled by t/q and rounded to the nearest integer vector. To transfer this to the ring setting, we use our notation $\lfloor\cdot\rceil_{q\to t}^{\boldsymbol{b}}$ (from Section 3.2) where rounding is defined w.r.t. some basis of $\mathcal{K}$.

Definition 5 ($\mathcal{R}$-LWR). *Let $\boldsymbol{b}$ be a $\mathbb{Z}$-basis of some fractional ideal $\mathcal{I} \subset \mathcal{K}$, let $s \in \mathcal{I}_q$, and let $q, t \in \mathbb{N}$ with $q > t$. The $\mathcal{R}$-LWR$_{s,\boldsymbol{b},q,t}$ distribution is*

$$\{(a, \lfloor as + q\mathcal{I}\rceil_{q\to t}^{\boldsymbol{b}}) \in \mathcal{R}_q \times \mathcal{I}_t : a \xleftarrow{\$} \mathcal{R}_q\}.$$

- *The decision problem D-$\mathcal{R}$-LWR$_{\boldsymbol{b},q,t,\ell}$ is to distinguish between ℓ independent samples from $\mathcal{R}$-LWR$_{s,\boldsymbol{b},q,t}$ for $s \xleftarrow{\$} \mathcal{I}_q$, and the same number of independent samples from $U(\mathcal{R}_q \times \mathcal{I}_t)$.*
- *The search problem S-$\mathcal{R}$-LWR$_{\boldsymbol{b},q,t,\ell}$ is, given ℓ independent samples from $\mathcal{R}$-LWR$_{s,\boldsymbol{b},q,t}$ for $s \xleftarrow{\$} \mathcal{I}_q$, to find s.*

Note that our $\mathcal{R}$-LWR definition is more general than prior works [BPR12,AA16,BGM+16] which usually have $\mathcal{I} = \mathcal{R}$ with $\boldsymbol{b}$ being the power basis $(\zeta_m^i)_{i\in[\varphi(m)]}$ of $\mathcal{R}$. By being more general, our $\mathcal{R}$-LWR definition also encompasses the definition from [LW20] where the secret is sampled from the dual fractional ideal $\mathcal{I} = \mathcal{R}^\vee$ instead. This is required since we use the following state-of-the-art hardness result for $\mathcal{R}$-LWR from [LW20, Theorem 4.11], which shows that the ratio q/t can be kept polynomial in the security parameter (under common ideal lattice assumptions).

Theorem 1. *Let ϕ be a distribution such that $\forall e \leftarrow \phi : ||\sigma(e)||_2 \leq B_e$, let $\boldsymbol{b}$ be a basis of $\mathcal{R}^\vee$ with dual basis $\boldsymbol{b}^\vee = (b_j^\vee)_j$ such that $\forall j : ||\sigma(b_j^\vee)||_2 \leq B_d$, and let $q > 18tB_dB_e\ell\varphi(m)$. Then there exists a polynomial time reduction from S-$\mathcal{R}$-LWE$_{q,\phi,\ell}$ to S-$\mathcal{R}$-LWR$_{\boldsymbol{b},q,t,\ell}$.*

Remark. The paper [LW20] refers for a proof of Theorem 4.11 to its full version. Unfortunately, this full version does not seem to exist or is not available, and we could not check whether the dual basis can be replaced by another basis of $\mathcal{R}^\vee$ and how a change affects the parameters. We will see in Section 4 that this choice of the basis of $\mathcal{R}^\vee$ also affects the security parameters for our approach.

3.4 BFV and CKKS

This section describes the BFV and CKKS encryption schemes. The BFV scheme was first proposed based on LWE in [Bra12] and then ported to the $\mathcal{R}$-LWE setting in [FV12]. It is an FHE scheme for exact arithmetic over integer numbers. On the other hand, the CKKS scheme from [CKKS17] is an FHE scheme for approximate arithmetic over fixed-point numbers. For this work, it is not necessary to know the details of key generation and encryption, so we skip directly to the ciphertext format and refer to [FV12, CKKS17] for further details.

Ciphertext Format. In both schemes, the secret key is some $s \in \mathcal{R}_q$, and a ciphertext has the form $\mathsf{ct} = (c_0, c_1) \in \mathcal{R}_q^2$. The parameter q is called the *ciphertext modulus*. The ciphertext satisfies

$$c_0 + c_1 \cdot s = \begin{cases} (q/t)m + \mathsf{noise} & \text{in BFV} \\ m + \mathsf{noise} & \text{in CKKS} \end{cases} \tag{1}$$

where m is the plaintext and $\mathsf{noise} \in \mathcal{K}$ is called the noise. The noise is small just after encryption, but homomorphic operations increase the noise. In BFV, the parameter t is called *plaintext modulus*, as the plaintext m is from $\mathcal{R}_t$.

We view a ciphertext as a polynomial $\mathsf{ct}(Y) = c_0 + c_1 \cdot Y$ that gets evaluated on the secret key for decryption, so we typically use the notation $\mathsf{ct}(s)$ instead of $c_0 + c_1 \cdot s$. The decryption procedure is defined as

$$\mathsf{BFV.Dec}_s(\mathsf{ct}) = \lfloor \mathsf{ct}(s) \rceil_{q\to t}^{\boldsymbol{p}} \in \mathcal{R}_t, \quad \mathsf{CKKS.Dec}_s(\mathsf{ct}) = \mathsf{ct}(s) \in \mathcal{R}_q.$$

That is, in BFV, decryption removes the noise via rounding. In this work, we round w.r.t. the powerful basis $\boldsymbol{p}$. Hence, decryption is correct if noise has small enough powerful basis coefficients, namely, within the interval $[-q/(2t), q/(2t))$. We call this bound of $q/(2t)$ the *noise ceiling*. Note that in CKKS, decryption doesn't remove the noise, so the noise introduces an error to the plaintext.

Remark. A CKKS plaintext m is in $\mathcal{R}_q$ and, therefore, represented by integer coefficients. When doing computations using CKKS, multiple fixed-point numbers are encoded in a plaintext, and homomorphic multiplication is defined such that the result corresponds to the product of the corresponding fixed-point numbers, up to some error. Our usage of CKKS is independent of this encoding technique.

Choice of Secret Key. As in [DKL+13, KPR18, OSV20, HRRK24], we have a parameter $h \in \mathbb{N}$ and assume the BFV/CKKS secret key s to be sampled as a random polynomial from $\mathcal{R}$ with (powerful basis) coefficients in $\{-1, 0, 1\}$ and with only h non-zero coefficients, i.e., with Hamming weight h. This way, for any $a \in \mathcal{R}$, we have $||as||_\infty \leq \gamma_1 h||a||_\infty$, which yields smaller rounding errors (e.g., in modulus switching, see below) than having s uniform over the whole $\mathcal{R}_q$.

Homomorphic Operations.

- *Addition:* Homomorphic addition takes two ciphertexts and outputs a ciphertext, which encrypts the sum of the underlying plaintexts. This operation is cheap in terms of computation and noise growth. The noise terms simply add up. Alternatively, homomorphic addition can take one plaintext and one ciphertext, in which case the noise of the ciphertext is preserved.
- *Multiplication:* Homomorphic multiplication takes two ciphertexts and outputs a ciphertext which encrypts the product of the underlying plaintexts. This requires a so-called relinearization step [FV12]. Multiplication is significantly more expensive than addition in terms of noise growth.
- *Exact Division:* In BFV, a ciphertext encrypting some message $t_1 m \in \mathcal{R}_{t_1 t_0}$ can be reinterpreted as encrypting the message $m \in \mathcal{R}_{t_0}$, cf. Equation (1) above in the BFV case. In CKKS, this operation is not possible (but approximate division is possible via modulus switching, as described below).
- *Modulus Switching:* The ciphertext modulus of a ciphertext $\mathsf{ct} \in \mathcal{R}_q^2$ can be changed from q to some q' (which can be smaller or larger than q) by computing $\mathsf{ct}' := \lfloor \mathsf{ct} \rceil_{q \to q'}^{\boldsymbol{p}}$, which satisfies
$$\mathsf{ct}'(s) = \begin{cases} (q'/t)m + (q'/q)\mathsf{noise} + e_{\mathsf{Rnd}} & \text{(BFV interpretation)}, \\ (q'/q)(m + \mathsf{noise}) + e_{\mathsf{Rnd}} & \text{(CKKS interpretation)}, \end{cases}$$
where e_{Rnd} is the rounding error with $||e_{\mathsf{Rnd}}||_\infty \leq (\gamma_1 h + 1)/2$. If $q \mid q'$, then $e_{\mathsf{Rnd}} = 0$. In BFV, this operation doesn't change the plaintext, but it changes the noise to $(q'/q)\mathsf{noise} + e_{\mathsf{Rnd}}$. In CKKS, this operation approximately multiplies the plaintext (including the noise/error) by q'/q (which can alternatively be viewed as approximate division by q/q').
- *Key Switching:* The secret key to a ciphertext ct can also be switched out. This operation requires possession of a *key switching key*, essentially an encryption of the old key under the new key. This operation increases the noise slightly more than the rounding error of the modulus switching.
- *Automorphisms:* For $i \in \mathbb{Z}_m^*$, one can homomorphically compute the automorphism σ_i (recall Section 3.2). Namely, given a ciphertext ct that encrypts some message m under secret key s, the ciphertext $\mathsf{ct}' := \sigma_i(\mathsf{ct})$ then encrypts $\sigma_i(m)$, but under the secret key $\sigma_i(s)$. This secret key change can be reverted by switching from $\sigma_i(s)$ back to s via key switching, i.e., we then get an encryption of $\sigma_i(m)$ that can be decrypted with the original key s.

As homomorphic operations increase the ciphertexts' noise, the ciphertext modulus q needs to be chosen suitably such that the final noise after all homomorphic operations doesn't overflow the noise ceiling $q/(2t)$.

3.5 BFV Plaintext Slots

When the plaintext modulus is a prime power $t = p^k$ with $\gcd(p, m) = 1$, then $\Phi_m(x) \bmod p^k$ splits into distinct irreducible polynomials of degree d, i.e., $\Phi_m(X) \equiv f_0(X) \cdots f_{r-1}(X) \pmod{p^k}$. The degree d is the order of p in $\mathbb{Z}_m^*$, and the number of factors is $r = \varphi(m)/d$. Therefore, by the Chinese Remainder Theorem (CRT), the plaintext space $\mathcal{R}_{p^k}$ decomposes into a direct product $\mathcal{R}_{p^k} \cong \bigtimes_{i=0}^{r-1} \mathbb{Z}_{p^k}[X]/(f_i)$ where we have element-wise addition and multiplication on the right-hand side. We call these factors *slots*. It can be shown that all factors $\mathbb{Z}_{p^k}[X]/(f_i)$ are isomorphic to each other [GHS12], so we fix f_0, define $\zeta_E = [X] \in \mathbb{Z}_{p^k}[X]/(f_0)$ (which is a root of f_0), and identify each slot with the $\mathbb{Z}_{p^k}$-algebra $E := \mathbb{Z}_{p^k}[\zeta_E] = \mathbb{Z}_{p^k}[X]/(f_0)$. We call E the *slot algebra*, as in [GV23].

Packing values into slots allows operations to be performed on these values in parallel. When $p \equiv 1 \pmod{m}$, we have $d = 1$ and $\mathcal{R}_{p^k} \simeq \mathbb{Z}_{p^k}^{\varphi(m)}$.

Linear Maps. Let $\sigma_E : E \to E$ be the Frobenius homomorphism that maps ζ_E to ζ_E^p and fixes the base ring $\mathbb{Z}_{p^k}$. As described in [HS15], every $\mathbb{Z}_{p^k}$-linear map $M : E \to E$ can be written as a sum $M(\eta) = \sum_{f=0}^{d-1} \theta_f \sigma_E^f(\eta)$ where $\theta_0, \ldots, \theta_{d-1} \in E$ are unique constants depending on M. Furthermore, if the image of M lies in $\mathbb{Z}_{p^k}$, then $\theta_f = \sigma_E^f(\theta_0)$ [HS15]. We use this in Appendix C to convert between different packing methods within plaintext slots.

Coefficients-to-Slots Map. Define the map $\mathsf{coeffsToSlots} : \mathcal{R}_{p^k} \to \mathcal{R}_{p^k}^d$ which takes some polynomial x and stores its powerful basis coefficients (i.e., entries of $\boldsymbol{\lambda}_x^{\boldsymbol{p}}$) in the constant coefficients of the slots of d polynomials $y_0, \ldots, y_{d-1} \in \mathcal{R}_{p^k}$. Similarly, define $\mathsf{slotsToCoeffs} : \mathcal{R}_{p^k}^d \to \mathcal{R}_{p^k}$ as the inverse map. As described in [HS15], these maps can be homomorphically evaluated on, e.g., BFV ciphertexts with plaintext space $\mathcal{R}_{p^k}$. This is used in Section 6.2 to move the plaintext coefficients into slots and vice versa. When working with ciphertexts, we also use the notations $\mathsf{coeffsToSlots}(\mathsf{ct})$ (which outputs d ciphertexts) and $\mathsf{slotsToCoeffs}(\mathsf{ct}_0, \ldots, \mathsf{ct}_{d-1})$ (which outputs a single ciphertext), for the homomorphic evaluation of these maps.

4 Reduction from Dual to Primal Ring-LWR

In Theorem 1, the authors of [LW20] sample the $\mathcal{R}$-LWR secret s from the dual fractional ideal $\mathcal{I} = \mathcal{R}^\vee$ instead of $\mathcal{R}$, as this provided them with several analytical advantages. We call this form *dual* $\mathcal{R}$-LWR, while we call the form with $\mathcal{I} = \mathcal{R}$ *primal* $\mathcal{R}$-LWR. We cannot base our $\mathcal{R}$-LWR-based PRF (Section 5.1) on dual $\mathcal{R}$-LWR directly, since we want to output randomness over $\mathcal{R}_t$ and not over $\mathcal{R}_t^\vee$. However, this can be fixed via the following reduction:

Theorem 2. *Let $f \in \mathcal{K}^*$, let $\boldsymbol{b}_0$ be a $\mathbb{Z}$-basis of some fractional ideal $\mathcal{I} \subset \mathcal{K}$, and let $\boldsymbol{b}_1 = f\boldsymbol{b}_0$ (so $\boldsymbol{b}_1$ is a $\mathbb{Z}$-basis of $f\mathcal{I}$). Then there exists a polynomial time reduction from decisional (resp. search) $\mathcal{R}$-LWR$_{\boldsymbol{b}_0,q,t,\ell}$ to decisional (resp. search) $\mathcal{R}$-LWR$_{\boldsymbol{b}_1,q,t,\ell}$.*

Proof. For the decision variant, the reduction function is $\mathsf{red} : (a, b) \mapsto (a, fb + tf\mathcal{I})$. Evidently, red maps $U(\mathcal{R}_q \times \mathcal{I}_t)$ to $U(\mathcal{R}_q \times (f\mathcal{I})_t)$, so it remains to prove that red maps $\mathcal{R}\text{-LWR}_{s,\boldsymbol{b}_0,q,t}$ to $\mathcal{R}\text{-LWR}_{s',\boldsymbol{b}_1,q,t}$ where s and s' are both uniformly random (but could be dependent). For this, let $s \xleftarrow{\$} \mathcal{I}_q$ and $s' := fs + qf\mathcal{I}$, and observe that s' is distributed uniformly over $(f\mathcal{I})_q$. Now for any $a \in \mathcal{R}_q$, observe that $\boldsymbol{\lambda}_{as}^{\boldsymbol{b}_0} \equiv \boldsymbol{\lambda}_{as'}^{\boldsymbol{b}_1} \pmod q$ and therefore

$$f\lfloor as \rceil_{q\to t}^{\boldsymbol{b}_0} = f(\langle \lfloor (t/q)\boldsymbol{\lambda}_{as}^{\boldsymbol{b}_0} \rceil, \boldsymbol{b}_0 \rangle + t\mathcal{I}) = \langle \lfloor (t/q)\boldsymbol{\lambda}_{as}^{\boldsymbol{b}_0} \rceil, \boldsymbol{b}_1 \rangle + tf\mathcal{I} = \lfloor as' \rceil_{q\to t}^{\boldsymbol{b}_1}.$$

This concludes the proof for the decision variant. For the search variant, we apply the same reduction, then find s', and then recover s via $s = f^{-1}s' + q\mathcal{I}$. □

By choosing $f = \hat{m}/g$ and $\mathcal{I} = \mathcal{R}^\vee$, we obtain a reduction from dual to primal $\mathcal{R}$-LWR, since $(\hat{m}/g)\mathcal{R}^\vee = \mathcal{R}$ (see Section 3.2). However, note that this reduction only tells us something about primal $\mathcal{R}$-LWR with basis $\boldsymbol{b}_1 = (\hat{m}/g)\boldsymbol{b}_0$, which depends on $\boldsymbol{b}_0$ and therefore might differ from the power basis that was traditionally [BPR12, AA16, BGM+16] used with primal $\mathcal{R}$-LWR. Concretely, we propose two possible ways to choose $\boldsymbol{b}_0$ (and thereby $\boldsymbol{b}_1$):

(1) Choose $\boldsymbol{b}_0$, such that $\boldsymbol{b}_1$ is the powerful basis, to obtain an $\mathcal{R}$-LWR instance that can be used nicely (i.e., without a possibly expensive change of basis) in contexts where ring elements are represented in the powerful basis. This is achieved by $\boldsymbol{b}_0 = (g/\hat{m})\boldsymbol{p} = \hat{\boldsymbol{p}}$, which is the powerful basis of $\mathcal{R}^\vee$ by Definition 2. This choice of $\boldsymbol{b}_0$ yields less tight parameters than the following choice 2, as the dual basis of $\hat{\boldsymbol{p}}$ is not optimally short with a bound of up to $B_d = \sqrt{\varphi(m)} \prod_{\text{prime } p \mid m} \varphi(p)$, as per Lemma 4.1.

(2) Choose $\boldsymbol{b}_0$ with optimally short dual basis to obtain the most tight parameters from Theorem 1. That is, choose $\boldsymbol{b}_0$ as the decoding basis $\boldsymbol{d}$ of $\mathcal{R}^\vee$ (cf. [LPR13] or Definition 3). Then $B_d = \max\{||\sigma(d_i^\vee)||_2\} = \sqrt{\varphi(m)}$.

Lemma 4.1. *Let $\hat{\boldsymbol{p}}^\vee$ be the dual basis of the powerful basis $\hat{\boldsymbol{p}}$ of $\mathcal{R}^\vee$. The length of each element $\hat{p}_i^\vee$ of $\hat{\boldsymbol{p}}^\vee$ under the canonical embedding is bounded by*

$$||\sigma(\hat{p}_i^\vee)||_2 \leq \sqrt{\varphi(m)} \prod_p \varphi(p)$$

where the product runs over all primes p that divide m.

Lemma 4.1 is proven in Appendix B. Note that, when m is a power of two, both choices ($\boldsymbol{b}_0 = \hat{\boldsymbol{p}}$ vs. $\boldsymbol{b}_0 = \boldsymbol{d}$) are the same (cf. Remark 1). If m is not a power of two, case (1) results in an overhead factor $\prod_p \varphi(p)$ in reduction tightness. Case 2, on the other hand, requires a change of basis to switch between $\boldsymbol{b}_1 = (\hat{m}/g)\boldsymbol{d}$ and the powerful basis that we usually use, e.g., in the BFV decryption in Section 5. It turns out that homomorphically computing this change of basis increases the ciphertext modulus in our application in Section 5 by factor $\varphi(m)$, i.e., more than the overhead factor $\prod_p \varphi(p)$ resulting from case (1). For details we refer to Appendix B. In the following, we stick to the conceptually simpler case (1) (i.e., $\boldsymbol{b}_0 = \hat{\boldsymbol{p}}$) for our construction.

Parameter Choices. For ($\mathcal{R}$-)LWE, it is common practice to estimate the concrete security of a parameter set by estimating the runtime (in terms of the number of operations required) of the best-known attacks from the literature. This can be done automatically by tools such as lattice estimator [APS15] and yields a reasonable estimate, since attacks on ($\mathcal{R}$-)LWE are well studied. Unfortunately, attacks on $\mathcal{R}$-LWR are less studied and we cannot reasonably base the security of our pseudorandom function (PRF) on resistance against known attacks. Instead, we base security on the security of $\mathcal{R}$-LWE together with a reduction from $\mathcal{R}$-LWE to $\mathcal{R}$-LWR. That is, we choose an $\mathcal{R}$-LWR instance that has a reduction from an $\mathcal{R}$-LWE instance with λ bits of computational security. An attack on our PRF with runtime $T(\lambda)$ would then imply an attack on the $\mathcal{R}$-LWE instance with runtime $\mathsf{poly}(\lambda) \cdot T(\lambda)$. Hence, we conclude $T(\lambda) \geq 2^\lambda / \mathsf{poly}(\lambda) \geq 2^{\lambda - \mathcal{O}(\log \lambda)}$.

By chaining Theorem 1 and Theorem 2 (with basis $\boldsymbol{b} = \boldsymbol{b}_0 = (g/\hat{m})\boldsymbol{p}$ as motivated in Section 4), $\mathcal{R}\text{-LWE}_{q,\phi,\ell}$ reduces to $\mathcal{R}\text{-LWR}_{\boldsymbol{p},q,t,\ell}$ if $q > 18tB_dB_e\ell\varphi(m)$. The bound B_d is given by Lemma 4.1. We always assume the $\mathcal{R}$-LWR error to be sampled coefficient-wise from the centered binomial distribution of variance $\sigma^2 = 10$, which approximates the Gaussian distribution of the same variance and is a standard choice, e.g., in [DPSZ12]. Hence, the LWE error's coefficients are bounded by $B_e = 10$. Note that the reduction from Theorem 1 uses search variants, while we require the decision variant of $\mathcal{R}$-LWR to be intractable. A search-to-decision reduction for $\mathcal{R}$-LWR exists for so-called *normal integral bases* [LW20]. Unfortunately, the powerful basis is, in general, not a normal integral basis. However, for important special cases, e.g., for the case $m = 2\varphi(m)$ used in standard multiparty computation (MPC) implementations like MP-SPDZ [Kel20], the proof in [LW20] nevertheless works (with slightly modified rounding) with the powerful basis (cf. [LW20, Section 4.3]). We remark that the minor changes to the rounding function are compatible with our construction.

In Table 2 we present the $\mathcal{R}$-LWR parameters for plaintext moduli with 64 and 128 bits induced by the $\mathcal{R}$-LWE to $\mathcal{R}$-LWR reduction. The plaintext modulus t can be a power of two or any other smaller number (such as a prime) without changing the remaining parameters. The cyclotomic index is $m = 2^{16} - 1 = 3{\cdot}5{\cdot}17{\cdot}257$. The column d is the degree of each plaintext slot when using a power-of-two plaintext modulus. q (resp. q') is the ciphertext modulus before (resp. after) switching to a sparse key with Hamming weight $h = 192$ (see Section 5.1), and

sec is the estimated computational security (according to the lattice estimator [APS15]) of the underlying $\mathcal{R}$-LWE instance. We give all parameters for two different values of the product $\varphi(m)\ell$, which measures how many encrypted $\mathbb{Z}_t$-values the parties can securely sample since each sample contains $\varphi(m)$ coefficients.

In the prime plaintext modulus setting, the concrete prime t should satisfy $t \equiv 1 \pmod{m}$, such that each sample encodes $\varphi(m)$ random values, as explained in Appendix C. For power-of-two plaintext modulus t, our choice of $m = 2^{16} - 1$ yields a relatively low slot degree $d = 16$. In Appendix C.2, we show how this leads to good packing efficiency.

Table 2: $\mathcal{R}$-LWR parameters induced by the reduction.

	$\varphi(m)\ell = 2^{32}$					$\varphi(m)\ell = 2^{64}$				
t	$\varphi(m)$	d	$\log q$	$\log q'$	sec	$\varphi(m)$	d	$\log q$	$\log q'$	sec
$\leq 2^{64}$	32768	16	141	154	1044	32768	16	173	186	820
$\leq 2^{128}$	32768	16	205	218	671	32768	16	237	250	565

5 Encrypted Randomness From Ring-LWR

Recall from Section 1 that for our construction, we require a sequence of public homomorphic encryption (HE) ciphertexts that encode secret pseudorandom values unknown to both parties. Such a ciphertext can be obtained by homomorphically evaluating a pseudorandom function (PRF), given that the parties already have an encryption of the PRF key. Namely, let F be a PRF and let ct_k be a publicly known encryption of an honestly generated PRF key k. Then, for all x in the domain of F_k, $F_k(x)$ is pseudorandom, and one can compute an encryption of $F_k(x)$ by homomorphically evaluating the circuit for $F_k(x)$. By iterating over different x (chosen according to some publicly known rule), this yields a sequence of encrypted random values.

The same can be achieved using a weak pseudorandom function (WPRF) (Definition 8). In a WPRF F the input x needs to be chosen according to some predefined sampling algorithm for the output $F_k(x)$ to be pseudorandom. When F is only a WPRF, we can use the random oracle to sample x and then proceed as above, i.e., homomorphically evaluate $F_k(x)$.

Remark. This requires the WPRF to be *public-coin* [PS08], as otherwise the randomness used to sample x must be kept secret and cannot be taken from the random oracle. The WPRF considered in this work is public-coin.

5.1 HE-friendly Weak PRF

To generate encrypted randomness as described above, we require a public-coin WPRF, which is *HE-friendly*. That is, its evaluation circuit should have a low multiplicative depth in order to make it practical to evaluate it in the HE domain (given an encryption of the key). In this work, we choose a WPRF based on the $\mathcal{R}$-LWR assumption (Definition 5).

Assuming D-$\mathcal{R}$-LWR$_{\boldsymbol{b},q,t,\ell}$ is hard, $F_s(a) \coloneqq \lfloor as \rceil^{\boldsymbol{b}}_{q\to t}$ is a public-coin WPRF with key generation algorithm $s \xleftarrow{\$} \mathcal{R}_q$, input sampling algorithm $a \xleftarrow{\$} \mathcal{R}_q$, and range $\mathcal{R}_t$, as long as we restrict the number of PRF samples to $\leq \ell$.

As observed by [AA17], in the case where $\boldsymbol{b} = \boldsymbol{p}$ is the powerful basis, the PRF evaluation $F_s(a)$ corresponds exactly to the decryption of the BFV [FV12] ciphertext $\mathsf{ct} = (0, a)$ under secret key s, since

$$\mathsf{ct}(s) = as = (q/t) \underbrace{\lfloor as \rceil^{\boldsymbol{b}}_{q\to t}}_{=F_s(a)} + \underbrace{[as]^{\boldsymbol{b}}_{q/t}}_{=:\,\mathsf{noise}} \pmod q.$$

In other words, the pair $\mathsf{ct} = (0, a)$ behaves similarly to a BFV ciphertext to the message $F_s(a)$. We therefore call $\mathsf{ct} = (0, a)$ a *pseudo-ciphertext* to the message $F_s(a)$. Please note that a party does obviously not need the secret key s to generate the pseudo-ciphertext to $F_s(a)$, only the randomness a.

Unfortunately, these pseudo-ciphertexts cannot be used for further homomorphic operations, e.g., to generate correlated randomness, as the noise is too large. Namely, the noise is saturated, i.e., it is only bounded by the noise ceiling $||\mathsf{noise}||_\infty \leq q/(2t)$. In order to support homomorphic operations, we, therefore, need to reduce the noise. A natural choice to reduce noise in BFV ciphertexts are bootstrapping techniques. However, existing bootstrapping techniques cannot successfully be applied to our pseudo-ciphertexts since:

1. They assume a sparse secret key with small coefficients, while the definition of D-$\mathcal{R}$-LWR (Definition 5) requires the key s to be dense with large coefficients.
2. They either cannot handle saturated noise or are limited for efficiency reasons to small plaintext and ciphertext moduli, as we explain in Section 6.

We solve the first problem by switching to a sparse key s' before bootstrapping, which we explain next. Afterward, in Section 6.5, we construct a new large-ratio BFV bootstrapping algorithm which solves the second problem.

Switching to a Sparse Key. We want to key switch the pseudo-ciphertext $\mathsf{ct} = (0, a)$ from the dense key $s \xleftarrow{\$} \mathcal{R}_q$ to a sparse key $s' \leftarrow \chi$ that is sampled with Hamming weight h as motivated in Section 3.4. As ct has saturated noise, the conventional BFV key switching procedure does not work as the introduced error could overflow the noise ceiling. For that reason, we need to make a change to the key switching procedure, namely, we switch from ciphertext modulus q to a slightly larger modulus $q' \geq 2(\gamma_1 h + 1)q$ where h is the Hamming weight of s'. For our key switching procedure, we require $t \mid q$ and a key switching key

$$\mathsf{ksk} := ((Q/q)s + \tilde{a}s' + \tilde{e}, -\tilde{a}) \pmod Q$$

where $\tilde{a} \xleftarrow{\$} R_Q$, $\tilde{e} \xleftarrow{\$} \chi$. This key switching key is computed over some larger modulus $Q > 2q^2\gamma_\infty\sigma^2$ with $q \mid Q$. Note that $\tilde{a}s' + \tilde{e}$ is an $\mathcal{R}$-LWE sample that masks $(Q/q)s$, so distributing ksk doesn't leak information about s. Note that ksk is a BFV encryption (with ciphertext modulus Q and plaintext modulus q) of s under s' since $\langle \mathsf{ksk}, (1, s')\rangle = (Q/q)s + \tilde{e} \pmod Q$. The switched ciphertext is then computed as

$$\mathsf{ct}' := \begin{cases} \lfloor a \cdot \mathsf{ksk} \rceil_{Q\to q'} & \text{if } q/t \text{ is odd,} \\ \lfloor a \cdot \mathsf{ksk} + (\frac{Q}{2q}, 0) \rceil_{Q\to q'} & \text{if } q/t \text{ is even.} \end{cases}$$

Decrypting ct' with plaintext space $\mathcal{R}_t$ then yields $\lfloor as \rceil_{q\to t}$, i.e., the same message as in ct. We will prove this statement for odd and even q/t separately.

Proof. First, for odd q/t, observe that

$$\mathsf{ct}'(s') = \langle \lfloor a \cdot \mathsf{ksk} \rceil_{Q\to q'}, (1, s')\rangle = (q'/q)as + (q'/Q)a\tilde{e} + e_{\mathsf{Rnd}} \pmod{q'}$$

with $||e_{\mathsf{Rnd}}||_\infty \leq (\gamma_1 h + 1)/2 \leq q'/(4q)$. Therefore decryption of ct' under secret key s' with plaintext space $\mathcal{R}_t$ yields

$$\lfloor (t/q')(q'/q)as + \underbrace{(t/q')((q'/Q)a\tilde{e} + e_{\mathsf{Rnd}})}_{=:\,\mathsf{error}}\rceil \bmod t = \lfloor (t/q)as + \mathsf{error}\rceil \bmod t$$
$$\overset{(*)}{=} \lfloor (t/q)as\rceil \bmod t = \lfloor as\rceil_{q\to t}$$

where $(*)$ holds because q/t is odd and the error term satisfies

$$||\mathsf{error}||_\infty \leq (t/q') \cdot ((q'/Q)\gamma_\infty \underbrace{||a||_\infty}_{\leq q/2} \underbrace{||\tilde{e}||_\infty}_{\leq \sigma^2} + \underbrace{||e_{\mathsf{Rnd}}||_\infty}_{\leq q'/(4q)}) < t/(2q),$$

i.e., error is too small to influence the rounding result.

Second, for even q/t, decryption of ct' instead yields

$$\lfloor (t/q)as + (t/2q) + \mathsf{error}\rceil \bmod t = \lfloor (t/q)as\rceil \bmod t = \lfloor as\rceil_{q\to t}.$$

Here, the summand $t/2q$ ensures that $(t/2q) + \mathsf{error}$ has positive coefficients $< t/q$ and hence doesn't influence the rounding result, since $\lfloor (t/q)as\rceil$ gets rounded upward in case of a tie. □

Remark. As mentioned, we use our key-switching procedure before performing a new large-ratio BFV bootstrapping. As we will see, the latter requires q' to be of the form $q' = p^V t$ for some small prime p. However, this choice of q' does not impose any new restrictions on q. In particular, q only needs to satisfy $q' \geq 2(\gamma_1 h + 1)q$ and must guarantee $\mathcal{R}$-LWR hardness.

6 BFV Bootstrapping

Below, we present established digit extraction and bootstrapping procedures for BFV/CKKS ciphertexts and use these techniques to build a new BFV bootstrapping algorithm that also works with the pseudo-ciphertexts from Section 5.1.

6.1 Digit Extraction Approach

Let the plaintext modulus t be some prime power $t := p^r$. The *digit extraction* bootstrapping algorithm [HS15, CH18, GV23, GIKV23, OPP23] then performs the following procedure for some $e > r$:

$$1.\quad \mathsf{ct}' := \lfloor \mathsf{ct}\rceil_{q\to p^e} \qquad 2.\quad w := [\mathsf{ct}'(s)]_{p^e}, \qquad 3.\quad x := \lfloor w\rceil_{p^e\to p^r}.$$

The three operations can be done directly on ciphertexts given an encryption of s under plaintext modulus p^e. For the modulus switching, we have seen this in Section 3.4. For the second step, we can use the encryption of s under plaintext modulus p^e. The third step is done by homomorphically evaluating a *digit extraction procedure* [GV23] on the ciphertext for w. As shown in [CH18], this digit extraction procedure can be implemented as a circuit of depth $\leq (e-r)\log(p) + \log(e)$ that contains $\approx (e-r)\sqrt{2pe}$ ciphertext-ciphertext multiplications.

When using this approach directly to bootstrap our pseudo-ciphertexts from Section 5.1, we have $v := e - r = \log_p(q/t)$ and $r = \log_p(t)$. With plaintext modulus $t \in 2^{\mathcal{O}(\lambda)}$, we have both $v, r \in \mathcal{O}(\lambda)$ and thus need to evaluate a circuit of size $\mathcal{O}(\lambda^{3/2})$ and multiplicative depth $\mathcal{O}(\lambda)$. We improve this by presenting two bootstrapping algorithms that use digit extraction as a subroutine that gets called with reduced parameters. The first algorithm (Section 6.4) reduces the parameter r to $\Theta(v)$. On its own, this doesn't improve the asymptotic circuit size. The second algorithm (Section 6.5) additionally reduces the parameter v by using an iterative approach.

6.2 CKKS Bootstrapping via Digit Extraction

For a CKKS ciphertext $\mathsf{ct}_{\mathsf{in}} \in \mathcal{R}_q^2$ with $\mathsf{ct}_{\mathsf{in}}(s) = x \pmod q$, the functionality of bootstrapping is to increase the ciphertext modulus while keeping approximately the same plaintext x. In this subsection, we present a new (although not surprising) bootstrapping algorithm $\mathsf{CKKS.DexBts}_{v,r,\alpha}$ based on digit extraction that works for prime power $q = p^v$ and increases the ciphertext modulus to αp^{r+v} for some $\alpha, r > 0$. The parameter α makes the bootstrapping scale up the plaintext by a factor α. This is useful if we need to reduce the bootstrapping error relative to the plaintext while maintaining the same complexity in number of ciphertext operations. In contrast to previous CKKS bootstrapping algorithms [CHK+18, LLL+21, LLK+22, JM22, BCC+22], ours does not require x to be much smaller than the ciphertext modulus. However, $\mathsf{CKKS.DexBts}_{v,r}$ is only efficient for relatively small values of p and $e := r + v$, as the depth of the digit extraction circuit grows with these parameters, as described in Section 6.1. $\mathsf{CKKS.DexBts}_{v,r}(\mathsf{ct}_{\mathsf{in}})$ performs the following three steps.

1. Switch to BFV. We first raise the ciphertext modulus to $p^{r+v} = p^e$ (without rescaling the ciphertext), obtaining $\mathsf{ct}_1 := [\mathsf{ct}_{\mathsf{in}}]_{p^v}^{\boldsymbol{p}} \bmod p^e$ with $\mathsf{ct}_1(s) = p^v w + x \pmod{p^e}$ where w is some "garbage" contained in the upper bits. The goal is to eliminate this garbage via digit extraction. As digit extraction is very sensitive to approximation errors, this is better done using exact arithmetic, i.e., using BFV instead of CKKS. Therefore we choose some $\Delta \in \mathbb{N}$ and compute the modulus-switched ciphertext $\mathsf{ct}_2 := \lfloor \mathsf{ct}_1 \rceil_{p^e \to \Delta p^e}$ that satisfies $\mathsf{ct}_2(s) = \Delta \cdot (p^v w + x) \pmod{\Delta p^e}$ which we reinterpret as a BFV ciphertext with plaintext $p^v w + x$, plaintext modulus p^e, and no noise (yet). The factor Δ must be chosen large enough to absorb the noise induced by the upcoming ciphertext operations.

2. Homomorphic Digit Extraction. Digit extraction needs to be performed component-wise on the coefficient vector of the plaintext. As we can only perform component-wise computations on the plaintext slots, we first need to homomorphically apply the $\mathsf{coeffsToSlots}$ linear transform to move the plaintext coefficients into plaintext slots (see Section 3.5). As each BFV ciphertext only has $\varphi(m)/d$ plaintext slots, this requires creating d BFV ciphertexts. For simplicity of presentation, we assume here that $d = 1$. The analogous construction, however, works for all $d \geq 1$. Let $\mathsf{ct}_3 := \mathsf{coeffsToSlots}(\mathsf{ct}_2)$. Then

$$\mathsf{ct}_3(s) = \Delta \cdot \mathsf{coeffsToSlots}(p^v w + x) + e_3 \pmod{\Delta p^e}$$

where e_3 is the noise introduced by $\mathsf{coeffsToSlots}$. We now perform homomorphic digit extraction (cf. Section 6.1) on ct_3 to obtain the r most-significant base-p digits of the plaintext slots. Thereby we obtain the ciphertext ct_4 that encrypts $\lfloor p^v w + x \rceil_{p^e \to p^r}^{\boldsymbol{p}} = w$, i.e., $\mathsf{ct}_4(s) = \Delta \cdot \mathsf{coeffsToSlots}(w) + e_4 \pmod{\Delta p^r}$, where e_4 is the noise after homomorphically evaluating the digit extraction circuit. This allows us to zero out the "garbage" by computing $\mathsf{ct}_5 := \mathsf{ct}_3 - p^v \mathsf{ct}_4 \pmod{\Delta p^e}$. By moving the plaintext slots back into the plaintext coefficients, i.e., by computing $\mathsf{ct}_6 := \mathsf{slotsToCoeffs}(\mathsf{ct}_5)$, we have $\mathsf{ct}_6(s) = \Delta \cdot x + e_6 \pmod{\Delta p^e}$ where e_6 is the noise after homomorphically evaluating $\mathsf{slotsToCoeffs}$.

3. Switch Back to CKKS. Recall that we wanted to bootstrap a CKKS ciphertext with plaintext x, so we need to eliminate the factor Δ. We do so by modulus switching downward, i.e., we compute $\mathsf{ct}_{\mathsf{out}} := \lfloor \mathsf{ct}_6 \rceil_{\Delta p^e \to \alpha p^e}^{\boldsymbol{p}}$ which satisfies $\mathsf{ct}_{\mathsf{out}}(s) = \alpha x + e_{\mathsf{DexBts}} \pmod{p^e}$ with $e_{\mathsf{DexBts}} := \alpha e_6 / \Delta + e_{\mathsf{Rnd}}$ and e_{Rnd} the rounding error. By choosing Δ large enough, the bootstrapping error e_{DexBts} is dominated by e_{Rnd}, which has a small bound of $||e_{\mathsf{Rnd}}||_\infty \leq (\gamma_1 h + 1)/2$.

We compactly present this algorithm in Figure 1. In summary, we have transformed our input CKKS ciphertext $\mathsf{ct}_{\mathsf{in}}$ modulo p^v, which encrypts x, into an output CKKS ciphertext $\mathsf{ct}_{\mathsf{out}}$ modulo p^e that encrypts the slightly modified plaintext $x + e_{\mathsf{DexBts}}$.

$\mathsf{CKKS.DexBts}_{v,r,\alpha}(\mathsf{ct}_{\mathsf{in}})$

1. $\mathsf{ct}_1 \coloneqq [\mathsf{ct}_{\mathsf{in}}]^{\boldsymbol{p}}_{p^v} \bmod p^e$, $\mathsf{ct}_2 \coloneqq \lfloor \mathsf{ct}_1 \rceil_{p^e \to \Delta p^e}$ *(switch to BFV)*
2. $\mathsf{ct}_3 \coloneqq \mathsf{coeffsToSlots}(\mathsf{ct}_2)$
   Obtain ct_4 by evaluating $x \mapsto \lfloor x \rceil^{\boldsymbol{p}}_{p^e \to p^r}$ on ct_3 *(homomorphic digit extraction)*
   $\mathsf{ct}_5 \coloneqq \mathsf{ct}_3 - p^v \mathsf{ct}_4 \pmod{\Delta p^e}$
   $\mathsf{ct}_6 \coloneqq \mathsf{slotsToCoeffs}(\mathsf{ct}_5)$
3. **return** $\mathsf{ct}_{\mathsf{out}} \coloneqq \lfloor \mathsf{ct}_6 \rceil^{\boldsymbol{p}}_{\Delta p^e \to \alpha p^e}$ *(switch back to CKKS)*

Fig. 1: CKKS bootstrapping via digit extraction

Complexity. The linear transformations $\mathsf{coeffsToSlots}$ and $\mathsf{slotsToCoeffs}$ can be implemented in $\mathcal{O}(\lambda)$ ciphertext operations and constant multiplicative depth [HS21]. In total, also accounting for the digit extraction step (see Section 6.1), $\mathsf{CKKS.DexBts}_{v,r}$ needs $\mathcal{O}(\lambda + v\sqrt{r+v})$ ciphertext operations and multiplicative depth $\mathcal{O}(v + \log r)$.

6.3 BFV Bootstrapping via Approximate Lifting

In [KSS24], the authors construct a BFV bootstrapping algorithm that works for arbitrary plaintext modulus t with performance that only weakly depends on t. To bootstrap a BFV ciphertext, the authors use the following strategy, which uses a CKKS bootstrapping algorithm as a subroutine.

1. *Noise Extraction:* The input ciphertext $\mathsf{ct}_{\mathsf{in}}$ with $\mathsf{ct}_{\mathsf{in}}(s) = (q/t)x + \mathsf{noise} \pmod q$ is reduced modulo q/t, in order to obtain ct_1 with $\mathsf{ct}_1(s) = \mathsf{noise} \pmod{q/t}$. When reinterpreted as a CKKS ciphertext, ct_1 is an encryption of the noise of $\mathsf{ct}_{\mathsf{in}}$.
2. *Approximate Lifting:* A CKKS bootstrapping algorithm is applied to ct_1 to increase the ciphertext modulus back to q. This outputs a ciphertext ct_2 with $\mathsf{ct}_2(s) = \mathsf{noise} + e_{\mathsf{CkksBts}} \pmod q$ and e_{CkksBts} the noise introduced by the bootstrapping.
3. *Subtraction:* Compute the output ciphertext $\mathsf{ct}_{\mathsf{out}} \coloneqq \mathsf{ct}_1 - \mathsf{ct}_2 \pmod q$. Observe that the subtraction cancels out the noise.

Remark. While so far, CKKS bootstrapping has been a separate line of research [CHK+18, LLL+21, LLK+22, JM22, BCC+22], the approximate lifting approach means that future optimizations to CKKS bootstrapping transfer directly to BFV as well.

For our pseudo-ciphertexts from Section 5.1, we cannot apply approximate lifting directly since it requires the CKKS plaintext before bootstrapping (i.e., the noise) to be significantly smaller than the ciphertext modulus. Hence, to bootstrap BFV noise, the noise needs to meet an upper bound significantly smaller than the noise ceiling $q/(2t)$. Recall that a in our pseudo-ciphertext $(0, a)$ is chosen uniformly at random and that the noise is therefore saturated, i.e., the noise might get arbitrarily close to $q/(2t)$.

6.4 Small-Ratio BFV Bootstrapping

As seen above, we cannot directly apply the approximate lifting approach to our application. We, therefore, construct a new algorithm $\mathsf{BFV.SrBts}$ that combines the approximate lifting and digit extraction approaches. While $\mathsf{BFV.SrBts}$ already works for saturated noise and arbitrary plaintext modulus, it only achieves a complexity improvement for $\log(q/t) \in o(\log t)$,

i.e. for *small ratio*. In many applications, this is not a severe restriction, as the ciphertext modulus (and thereby the absolute magnitude of the noise) can be decreased via modulus switching downward before bootstrapping. However, this is not an option for our pseudo-ciphertexts from Section 5.1, as the rounding error incurred by modulus switching can cause an overflow of the saturated noise. We, therefore, refine our approach in Section 6.5 to also work with BFV ciphertexts with *large ratio* q/t.

However, for now we focus on $\mathsf{BFV.SrBts}$. Similar to the approximate lifting approach in Section 6.3, we extract and bootstrap the noise as a CKKS ciphertext. Since conventional CKKS bootstrapping algorithms cannot be used to bootstrap saturated noise, we preprocess the extracted noise with our approach based on digit extraction from Section 6.2. Afterward, the extracted noise can be bootstrapped using a conventional algorithm, and the bootstrapped ciphertext can then be subtracted from our original ciphertext to cancel out the noise. We again have a parameter $\alpha \in \mathbb{N}$ that is useful to decrease the relative noise after bootstrapping. In contrast to $\mathsf{CKKS.DexBts}$, here α scales up the ciphertext modulus during bootstrapping without affecting the plaintext. In the following, we present this algorithm in more detail.

1. Noise Extraction. Let $\mathsf{ct}_{\mathsf{in}}$ be the BFV ciphertext we like to bootstrap. Let q be the ciphertext modulus, t the plaintext modulus, and let $b \in \mathbb{N}$ be the desired noise budget (in bits) after bootstrapping, i.e., the output ciphertext's noise magnitude is $\leq \alpha q/(t2^{b+1})$. We require $q = p^v t$, where p is a small prime (e.g., $p = 2$) and $\alpha p^v/2^{b+1} \geq \gamma_1 h + 1$, as otherwise the final ciphertext modulus $\alpha p^v t$ cannot comprise the desired noise budget. With this, we have $\mathsf{ct}_{\mathsf{in}}(s) = p^v x + \mathsf{noise} \pmod q$ where $x \in \mathcal{R}_t$ is the plaintext and $||\mathsf{noise}||_\infty \leq p^v/2$. By computing $\mathsf{ct}_1 \coloneqq \mathsf{ct}_{\mathsf{in}} \bmod p^v$, we have $\mathsf{ct}_1(s) = \mathsf{noise} \pmod{p^v}$.

2. Digit Extraction and Approximate Lifting. We now reinterpret ct_1 as a CKKS ciphertext and apply our digit extraction-based bootstrapping algorithm from Section 6.2 for some $r \in \Theta(v)$, obtaining the ciphertext $\mathsf{ct}_2 \coloneqq \mathsf{CKKS.DexBts}_{v,r,\alpha}(\mathsf{ct}_1)$ with $\mathsf{ct}_2(s) = \alpha \cdot \mathsf{noise} + e_{\mathsf{DexBts}} \pmod{\alpha p^{r+v}}$ where e_{DexBts} is defined in Section 6.2.

Note that ct_2 satisfies a plaintext bound $B_{\mathsf{pt}} = p^v$ and therefore has a ratio of $\approx p^r$ between plaintext size and ciphertext modulus. By $r \in \Theta(v)$, this ratio is large enough to make ct_2 appropriate for CKKS bootstrapping using conventional techniques such as [BCC+22] (similar to the approximate lifting approach from Section 6.3). We bootstrap ct_2 with input modulus $q_{\mathsf{in}} = \alpha p^{r+v}$ and output modulus $q_{\mathsf{out}} = \alpha q$, obtaining a bootstrapped ciphertext ct_3 with $\mathsf{ct}_3(s) = \alpha \cdot \mathsf{noise} + e_{\mathsf{DexBts}} + e_{\mathsf{CkksBts}} \pmod{\alpha q}$ where e_{CkksBts} is the noise introduced by bootstrapping.

3. Subtraction. We can interpret ct_3 as a BFV encryption of 0 with noise $\alpha \cdot \mathsf{noise} + e_{\mathsf{DexBts}} + e_{\mathsf{CkksBts}}$, i.e., it has (except for the factor α) approximately the same noise as our input ciphertext $\mathsf{ct}_{\mathsf{in}}$. We can exploit this to cancel out the noise of $\mathsf{ct}_{\mathsf{in}}$ by computing the output ciphertext $\mathsf{ct}_{\mathsf{out}} \coloneqq \lfloor \mathsf{ct}_{\mathsf{in}} \rceil_{q \to \alpha q} - \mathsf{ct}_3 \pmod{\alpha q}$. Let $e_{\mathsf{SrBts}} \coloneqq -e_{\mathsf{DexBts}} - e_{\mathsf{CkksBts}}$. It is easy to see that $\mathsf{ct}_{\mathsf{out}}(s) = \alpha p^v x + e_{\mathsf{SrBts}} \pmod{\alpha q}$ is a BFV ciphertext that encrypts the same plaintext as $\mathsf{ct}_{\mathsf{in}}$ (namely x), but with smaller noise if we choose parameters such that the total error e_{SrBts} is not too large.

We summarize $\mathsf{BFV.SrBts}_{v,b,\alpha}$ in Figure 2. Note that, even though we used CKKS as an intermediate encryption scheme in our bootstrapping algorithm, the original plaintext is still preserved exactly. The approximation error introduced by CKKS only influences the noise of $\mathsf{ct}_{\mathsf{out}}$. This noise is dominated by e_{CkksBts} (i.e., the error introduced by the conventional CKKS bootstrapping subroutine, e.g., [BCC+22]), since the other error term e_{DexBts} has a small magnitude $\leq (\gamma_1 h + 1)/2$ (see Section 6.2).

$\mathsf{BFV.SrBts}_{v,b,\alpha}(\mathsf{ct}_{\mathsf{in}})$

1. $\mathsf{ct}_1 \coloneqq \mathsf{ct}_{\mathsf{in}} \bmod p^v$
2. $\mathsf{ct}_2 \coloneqq \mathsf{CKKS.DexBts}_{v,r,\alpha}(\mathsf{ct}_1)$
   $q_{\mathsf{out}} \coloneqq \alpha q$
   $\mathsf{ct}_3 \coloneqq \mathsf{CKKS.Bts}_{q_{\mathsf{out}},b+2}(\mathsf{ct}_2)$ *(CKKS bootstrapping with precision factor $b+2$)*
3. **return** $\mathsf{ct}_{\mathsf{out}} \coloneqq \lfloor \mathsf{ct}_{\mathsf{in}} \rceil_{q \to \alpha q} - \mathsf{ct}_3 \pmod{\alpha q}$

Fig. 2: Small-ratio BFV bootstrapping

Theorem 3. $\mathsf{BFV.SrBts}_{v,b,\alpha}$ *requires* $\tilde{\mathcal{O}}(\lambda + v^{3/2})$ *ciphertext operations and multiplicative depth* $\tilde{\mathcal{O}}(v)$.

Proof. Due to $r \in \Theta(v)$, the digit extraction subroutine requires $\mathcal{O}(\lambda + v^{3/2})$ ciphertext operations and multiplicative depth $\mathcal{O}(v)$ (see Section 6.2). To achieve b bits of noise budget after bootstrapping, we require the CKKS bootstrapping subroutine to have a precision factor [KSS24] of $\log_2(B_{\mathsf{pt}}/B_{\mathsf{err}}) = b+2$. This is sufficient since $||e_{\mathsf{DexBts}}||_\infty \leq (\gamma_1 h + 1)/2 \leq \alpha p^v/2^{b+2}$ and $||e_{\mathsf{CkksBts}}||_\infty \leq B_{\mathsf{err}} = B_{\mathsf{pt}}/2^{b+2} = \alpha p^v/(2^{b+2})$ add up to a total error of $\leq \alpha p^v/2^{b+1} = \alpha q/(t2^{b+1})$. CKKS bootstrapping with precision factor $b+2$ can be done in a roughly linear number of ciphertext operations using META-BTS [BCC+22]. This is asymptotically cheaper than the digit extraction step. □

6.5 Large-Ratio BFV Bootstrapping

Our small-ratio bootstrapping algorithm from Section 6.4 requires $\tilde{\mathcal{O}}(\log^{3/2}(q/t))$ ciphertext operations. In this subsection we show how to reduce this to $\tilde{\mathcal{O}}(\log^{4/3}(q/t))$. Note that we cannot simply modulus-switch to a smaller ciphertext modulus (and hence a smaller ratio q/t) because we want to support saturated noise. Therefore, the rounding error incurred by modulus switching could overflow the noise ceiling and introduce an error to the underlying plaintext. Instead, we change the interpretation of the input ciphertext and then use the small-ratio bootstrapping algorithm from Section 6.4 as a subroutine to decrease the ciphertext modulus. By iteratively applying this, the ciphertext modulus can be decreased further and further until it is small enough to apply the small-ratio bootstrapping algorithm without a change in interpretation. In the following, we present this algorithm, called $\mathsf{BFV.LrBts}$, in detail.

Choose a number $c \in \mathbb{N}_{>0}$ that determines by how many digits we decrease the ciphertext modulus in each iteration. We again require $q = p^V t$, where p is a small prime and $\alpha p^V/2^{b+1} \geq \gamma_1 h + 1$, where b is the desired noise budget (in bits) after bootstrapping. With this, the BFV ciphertext to be bootstrapped is $\mathsf{ct}_{\mathsf{in}}$ with $\mathsf{ct}_{\mathsf{in}}(s) = p^V x + \mathsf{noise} \pmod q$ where noise has coefficients in $[-\lfloor p^V/2 \rfloor, \lceil p^V/2 \rceil)$. Let $v \leq V$ be the smallest natural number such that $p^{v-c}/2 \geq \gamma_1 h + 1$. We can decompose the noise into

$$\mathsf{noise} = \begin{cases} p^v n_1 + n_0 & \text{if } p \text{ is odd} \\ p^v n_1 + n_0 + p^v/2 & \text{if } p = 2 \end{cases}$$

for some $n_1 \in \mathcal{R}_q$ with coefficients in $[-\lfloor p^{V-v}/2 \rfloor, \lceil p^{V-v}/2 \rceil)$ and $n_0 \in \mathcal{R}_q$ with coefficients in $[-\lfloor p^v/2 \rfloor, \lceil p^v/2 \rceil)$. For $\mathsf{ct}_1 \coloneqq \mathsf{ct}_{\mathsf{in}} - \mathbb{1}_{p=2}(p^v/2, 0)$ we get

$$\mathsf{ct}_1(s) = p^V x + p^v n_1 + n_0 = p^v(p^{V-v}x + n_1) + n_0 \pmod q.$$

We can reinterpret ct_1 under plaintext modulus q/p^v where it decrypts to $p^{V-v}x + n_1$ with noise n_0. Let h be the Hamming weight of the secret key. Observe that v is large enough such that $p^v/2^{c\log(p)+1} \geq \gamma_1 h + 1$. Hence, we can apply $\mathsf{BFV.SrBts}_{v,c\log(p),1}$ (from Section 6.4)

with target noise budget $c\log(p)$ bits to ct_1, obtaining a bootstrapped ciphertext $\mathsf{ct}_2(s) = p^v(p^{V-v}x + n_1) + e_{\mathsf{SrBts}} \pmod q$ where $||e_{\mathsf{SrBts}}||_\infty < p^v/2^{c\log(p)} = p^{v-c}$. Our main insight is that the new noise e_{SrBts} of ct_2 is small enough such that we can now switch to a smaller ciphertext modulus $p^{V-c}t = p^{-c}q$ without having the rounding error overflow the noise ceiling, i.e., without introducing an error to the message. Specifically, this is achieved by computing $\mathsf{ct}_3 \coloneqq \lfloor \mathsf{ct}_2 \rceil_{q\to p^{-c}q} + \mathbb{1}_{p=2}(p^{v-c}/2, 0)$ which fulfills

$$\begin{aligned}\mathsf{ct}_3(s) &= p^{v-c}(p^{V-v}x + n_1) + p^{-c}e_{\mathsf{SrBts}} + e_{\mathsf{Rnd}} + \mathbb{1}_{p=2}p^{v-c}/2\\ &= p^{V-c}x + \mathsf{noise}' \pmod{p^{-c}q}\end{aligned}$$

with $\mathsf{noise}' \coloneqq p^{v-c}(n_1 + \mathbb{1}_{p=2}/2) + p^{-c}e_{\mathsf{SrBts}} + e_{\mathsf{Rnd}}$ and $||e_{\mathsf{Rnd}}||_\infty \le (\gamma_1 h + 1)/2 \le p^{v-c}/4$. That is, ct_3 is a ciphertext with the same message as $\mathsf{ct}_{\mathsf{in}}$ (namely x) but smaller ciphertext modulus $p^{-c}q$ and noise noise'. Note that the noise ceiling is now at $p^{-c}q/(2t) = p^{V-c}/2$. And indeed, the new noise can be bounded by

$$||\mathsf{noise}'||_\infty = ||p^{v-c}\underbrace{(n_1 + \mathbb{1}_{p=2}/2)}_{\le (p^{V-v}-1)/2} + p^{-c}\underbrace{e_{\mathsf{SrBts}}}_{<p^v/4} + \underbrace{e_{\mathsf{Rnd}}}_{\le p^{v-c}/4}||_\infty < p^{V-c}/2,$$

so it doesn't overflow the noise ceiling, and the message x is preserved error-free.

By repeating this process, the ciphertext modulus can be iteratively reduced to a smaller and smaller modulus. We iterate until we arrive after i iterations at a ciphertext $\mathsf{ct}^{(i)}$ under modulus $q_i \coloneqq p^{V-ic}t$ that no longer meets the requirement $v \le V - ic$, i.e., it's no longer possible to reduce the modulus by another c digits. We finish the bootstrapping with a final pass of $\mathsf{BFV.SrBts}_{V-ic,b,\alpha p^{ic}}$ which scales the ciphertext modulus up to αp^V. This final call to $\mathsf{BFV.SrBts}$ requires $\alpha p^V/2^{b+1} \ge \gamma_1 h + 1$, as otherwise the final ciphertext modulus cannot comprise b bits noise budget. We summarize this algorithm in Figure 3.

$\mathsf{BFV.LrBts}_{V,c,b,\alpha}(\mathsf{ct}_{\mathsf{in}}^{(0)})$

for $i = 0$ **to** $\lfloor (V-v)/c \rfloor$ **do**:
- $\mathsf{ct}_1^{(i)} \coloneqq \mathsf{ct}_{\mathsf{in}}^{(i)} - \mathbb{1}_{p=2}(p^v/2, 0)$
- $\mathsf{ct}_2^{(i)} \coloneqq \mathsf{BFV.SrBts}_{v,c\log(p),1}(\mathsf{ct}_1^{(i)})$
- $\mathsf{ct}_3^{(i)} \coloneqq \lfloor \mathsf{ct}_2^{(i)} \rceil_{q_i \to q_{i+1}} + \mathbb{1}_{p=2}(p^{v-c}/2, 0)$ where $q_i \coloneqq p^{V-ic}t$
- $\mathsf{ct}_{\mathsf{in}}^{(i+1)} \coloneqq \mathsf{ct}_3^{(i)}$

return $\mathsf{ct}_{\mathsf{out}} \coloneqq \mathsf{BFV.SrBts}_{V-ic,b,\alpha p^{ic}}(\mathsf{ct}_{\mathsf{in}}^{(i)})$ *(final pass)*

Fig. 3: Large-ratio BFV bootstrapping

Theorem 4. $\mathsf{BFV.LrBts}_{V,c,b,\alpha}$ *requires* $\tilde{\mathcal{O}}(\lambda V/c + Vc^{1/2})$ *ciphertext operations and multiplicative depth* $\tilde{\mathcal{O}}(V)$.

Proof. The algorithm terminates after $\mathcal{O}(V/c)$ iterations. In all iterations except for the final pass, the small-ratio bootstrapping is called with the same $v \in \tilde{\mathcal{O}}(c)$, requiring $\tilde{\mathcal{O}}(\lambda + c^{3/2})$ ciphertext operations. The last iteration uses an even smaller v (here called $V - ic$). Hence, $\mathsf{BFV.LrBts}_{V,c,b}$ requires in total $\tilde{\mathcal{O}}(V/c(\lambda + c^{3/2}))$ ciphertext operations. The multiplicative depth adds up to $\tilde{\mathcal{O}}(V)$. □

By instantiating with $c \in \Theta(V^{2/3})$, the number of ciphertext operations simplifies to $\tilde{\mathcal{O}}(\lambda V^{1/3} + V^{4/3})$. When bootstrapping our pseudo-ciphertexts from Section 5.1, we have $V \in \mathcal{O}(\lambda)$ and hence require $\tilde{\mathcal{O}}(\lambda^{4/3})$ ciphertext operations.

Table 3: Estimated runtime for generating one encrypted $\mathcal{R}_t$ element via (1) our WPRF evaluation strategy, (2) naive evaluation and (3) evaluation of AES-128.

	Ours, i.e., $\mathcal{R}$-LWR via BFV.LrBts					Naive $\mathcal{R}$-LWR	AES-128
t	V	i	DexBts	Bts	time	time	time
2^{64}	90	5	1765 s	74 s	**1839 s**	278 h (543×)	28 h (55×)
2^{128}	90	5	1765 s	92 s	**1857 s**	417 h (808×)	60 h (116×)

6.6 Runtime Evaluation

In our PCF we apply BFV.LrBts to evaluate the $\mathcal{R}$-LWR-based WPRF from Section 5.1 under threshold BFV. Table 3 presents runtime estimates for the BFV.LrBts evaluation of the $\mathcal{R}$-LWR-based WPRF and for naively evaluating the same WPRF, i.e., using its boolean circuit directly. Additionally, the table compares our approach to a baseline PRF, namely AES-128. We chose AES-128 as the baseline due to its widespread adoption and relatively small circuit size.

To instantiate the $\mathcal{R}$-LWR-based WPRF, we use the parameters from Table 2 with $\varphi(m)\ell = 2^{32}$. V denotes the number of bits ($p = 2$) of the initial noise after switching to a sparse key, i.e., $V = \log(q'/t)$, and i denotes the number of iterations, including the final pass. The DexBts (resp. Bts) column gives the total time spent in the CKKS.DexBts (resp. conventional CKKS.Bts) subroutines over all iterations. The CKKS.DexBts runtime was estimated with microbenchmarks conducted single-threaded using Microsoft SEAL [SEA23] on a laptop (Intel Core i7-8565U @ 1.80 GHz). For the Bts runtime, we extrapolated the runtimes of [KSS24].

For the naive $\mathcal{R}$-LWR evaluation, we used the same $\mathcal{R}$-LWR parameters and even included our optimization of switching to a sparse key (Section 5.1), as this yields a much smaller key and circuit size. We also include the batching optimization, which reduces the required number of ciphertext operations by the factor $\varphi(m)/d$. With both of these optimizations, the circuit has 2,310,400 ($t = 2^{64}$) or 3,283,200 ($t = 2^{128}$) fully-packed ciphertext multiplications.

For AES-128, we used the circuit from [Sma]. As each AES-128 evaluation only outputs 128 bits of pseudorandomness, we evaluate it $\varphi(m)\log(t)/128$ times (in parallel) to obtain pseudorandomness comparable to one $\mathcal{R}$-LWR sample. We again include the batching optimization. This way, the final circuit has 276,608 ($t = 2^{64}$) or 553,216 ($t = 2^{128}$) fully-packed ciphertext multiplications.

In all cases, the final noise budget b is chosen large enough to allow for one more multiplication (e.g., to compute a Beaver triple) and afterwards have 128 bits of noise budget left, so the final ciphertexts can be correctly converted to secret shares with overwhelming probability (see Section 7.1).

Our construction outperforms the naive evaluation strategy by 543–808× and AES-128 by 55–116×. This performance boost translates directly to the throughput (e.g., number of triples per second) of the generic PCF in Section 7. The times in Table 3 correspond to the generation of $\varphi(m)/2$ Beaver triples, since each triple requires two random coefficients and all $\varphi(m)$ random coefficients contained in a ciphertext can be extracted as explained in Appendix C.1. Therefore, the throughput of our PCF from Section 7 is ≈ 9 triples per second.

7 Our Pseudorandom Correlation Function

In this section, we construct our weak pseudorandom correlation function (PCF) in the standard model. A weak PCF (see Definition 10 in Appendix A) is built from two keyed functions Eval_{k_0}, Eval_{k_1}, such that $(\mathsf{Eval}_{k_0}(\cdot), \mathsf{Eval}_{k_1}(\cdot))$ follows a *target correlation* $\mathcal{Y} = (Y_0, Y_1)$. In the random oracle model (ROM), a weak PCF can be turned into a strong PCF by taking the inputs from the random oracle [BCG+20a].

Our construction works for arbitrary correlations that output an additive sharing of the outputs of a polynomially-sized arithmetic circuit evaluated on random inputs:

Definition 6 (Additive Correlation). *Let $n_i, n_o \in \mathbb{N}$ and let $\mathcal{C} : \mathcal{D}^{n_i} \to \mathcal{D}^{n_0}$ be an arithmetic circuit over some ring $\mathcal{D}$. The additive correlation for $\mathcal{C}$ is $\mathcal{Y}_\mathcal{C} := (Y_0, Y_1)$ where $Y_0, Y_1 \sim U(\mathcal{D}^{n_o})$ and $Y_0 + Y_1 \sim \mathcal{C}(U(\mathcal{D}^{n_i}))$.*

For instance, the additive Beaver triple correlation $\mathcal{C} : (a, b) \mapsto (a, b, ab)$ becomes $((a_0, b_0, c_0), (a_1, b_1, c_1))$ where all entries are uniformly random, except for c_1, which is uniquely determined by $(a_0 + a_1) \cdot (b_0 + b_1) = c_0 + c_1$.

When we have a family $\mathcal{C} = \{\mathcal{C}_\lambda\}_\lambda$ of arithmetic circuits where the concrete circuit depends on the security parameter λ, there is a corresponding family $\mathcal{Y}_\mathcal{C} = \{\mathcal{Y}_{\mathcal{C}_\lambda}\}_\lambda$ of additive correlations. We omit the index λ when it's obvious from the context. For all families of polynomially-sized arithmetic circuits, the corresponding family of additive correlations satisfies the *reverse-sampleability* property (Definition 9), which allows us to define a PCF.

Our (weak) PCF uses a fixed plaintext modulus t and parameters m, q, q' that depend on the security parameter. Concrete instantiations of these parameters were already presented in Table 2. Furthermore, our we use an auxiliary weak pseudorandom function (WPRF) $\mathsf{W} = (\mathsf{W.KeyGen}, \mathsf{W.InputGen}, \mathsf{W.Eval})$ to generate masks which party P_0 adds to all output shares and party P_1 subtracts from all output shares. This doesn't change the sum of the shares. Still, it makes sure that the individual shares follow the uniform distribution $U(\mathcal{D}^{n_o})$ (like the target correlation, see Definition 6) for outside distinguishers. We use this in the proof of the *pseudorandom $\mathcal{Y}$-correlated outputs* property of Definition 10. The auxiliary WPRF W doesn't need to be HE-friendly and can be instantiated, e.g., based on AES.

Definition 7. *For a family $\mathcal{C}$ of polynomially-sized arithmetic circuits over $\mathcal{R}_t$, we define our weak PCF for $\mathcal{Y}_\mathcal{C}$ as follows.*

- $\mathsf{KeyGen}(1^\lambda)$ *generates BFV key material $s, \mathsf{pk}, \mathsf{evk}$ (with evk the evaluation key needed for homomorphic operations) and $\mathsf{wk} \leftarrow \mathsf{W.KeyGen}(1^\lambda)$, then samples $s_0 \xleftarrow{\$} \mathcal{R}_q$, $s_1 := s - s_0 \pmod q$, and outputs (k_0, k_1) with $k_\sigma = (s_\sigma, \mathsf{pk}, \mathsf{evk}, \mathsf{wk})$.*
- $\mathsf{InputGen}(1^\lambda)$ *outputs $\big((a_i)_{i\in[n_i]}, (r_i)_{i\in[n_o]}\big)$ where $(a_i)_{i\in[n_i]} \xleftarrow{\$} \mathcal{R}_q^{n_i}$ and each r_i is sampled via $r_i \leftarrow \mathsf{W.InputGen}(1^\lambda)$.*
- $\mathsf{Eval}_{k_\sigma}((a_i)_{i\in[n_i]})$*: On input from $\mathsf{InputGen}$, first use the pseudo-ciphertexts $((0, a_i))_{i\in[n_i]}$ and use bootstrapping as in Sections 5.1 and 6 to receive n_i pseudo-ciphertexts ct_i with low noise. Then homomorphically evaluate $\mathcal{C}$ on these and decrypt the n_o output ciphertexts using secret key s_σ*[5]*, obtaining shares $\boldsymbol{d}_\sigma \in \mathcal{R}_t^{n_o}$. For all $i \in [n_o]$, compute the mask $\mu_i := \mathsf{W.Eval}_{\mathsf{wk}}(r_i)$, and let $\boldsymbol{\mu} := (\mu_i)_{i\in[n_o]}$. Party P_0 outputs $\boldsymbol{d}_0 + \boldsymbol{\mu}$ and party P_1 outputs $\boldsymbol{d}_1 - \boldsymbol{\mu}$.*

Note that, in Eval_{s_σ}, each party P_σ decrypts the computed ciphertexts using her own share of the secret key s_σ. In Section 7.1 we show that the Eval_{s_σ} are then additive shares of the underlying plaintexts with high probability, if the noise is not too large. As proven in

[5] Before decryption, one of the two parties needs to set the 0th component of each ciphertext to 0, in order to conform to Corollary 7.1.

Section 7.2, it follows that our weak PCF is a *secure* weak PCF for the additive correlation $\mathcal{Y}_{\mathcal{C}}$.

For multiparty computation (MPC) based on secret sharing, we typically need correlations over the domain $\mathcal{D} = \mathbb{Z}_t$. Our (weak) PCF can be modified to work over this domain using packing methods that pack many $\mathbb{Z}_t$-values in a single $\mathcal{R}_t$ polynomial and allow for element-wise operations. We describe such packing methods in Appendix C. For a target correlation over $\mathbb{Z}_t$ with n_i inputs, we only need to bootstrap $\lceil n_i/\varphi(m) \rceil$ pseudo-ciphertexts in the evaluation of F_{k_σ}, as this yields random polynomials with sufficiently many $\mathbb{Z}_t$-coefficients to compute the correlation. After the element-wise homomorphic operations are done, one can apply the inverse map (see Appendices C.2 and C.3) of the utilized packing method to reduce the number of ciphertexts to be decrypted to $\lceil n_o/\varphi(m) \rceil$.

To efficiently use the available slots, n_i and n_o should be large enough. This is achieved by defining a circuit $\mathcal{C}$ that computes a large batch of, e.g., Beaver triples in parallel. The technique is, of course, not limited to Beaver triples, but can be applied to all kinds of additive correlations, such as matrix triples [MZ17, CKR+20, MG23, RRKK23], convolution triples [CKR+20, RRHK23], polynomial encodings [RRK+24], and many more.

Authentication. For many actively secure protocols, such as SPDZ [DPSZ12], the (correlated) randomness needs to be authenticated via multiplication by a global and secret MAC key $\alpha \in \mathcal{D}$. To do this, we can distribute a ciphertext ct_α containing α in each slot as part of the key generation. In Eval_{s_σ}, after producing homomorphically encrypted unauthenticated correlated randomness and before decryption, the parties homomorphically multiply all ciphertexts by ct_α to obtain the homomorphically encrypted MAC tags and then also decrypt these with s_σ.

7.1 From Encrypted Randomness to Secret-Shares

Our main goal is to produce correlated randomness in additively secret-shared form used in MPC protocols like SPDZ [DPSZ12]. In Sections 5 and 6, we have seen how we can generate encryptions of the correlated randomness. Now, we need to convert these encryptions into additive secret shares of the underlying plaintext.

When the number of parties is two, this conversion can be performed non-interactively as follows. Assume the secret key s is additively shared between the parties, i.e., party P_0 has $s_0 \xleftarrow{\$} \mathcal{R}_q$ and party P_1 has $s_1 = s - s_0 \pmod q$. Further, assume that $\mathsf{ct} = (c_0, c_1)$ has negligibly small noise relative to the noise ceiling. Define $\mathsf{ct}' \coloneqq (0, c_1)$. Then, with overwhelming probability, $\mathsf{Dec}_{s_0}(\mathsf{ct}) + \mathsf{Dec}_{s_1}(\mathsf{ct}') = \mathsf{Dec}_s(\mathsf{ct}) \pmod t$, i.e., the parties can simply decrypt using their own key share to obtain additive shares of the plaintext (with overwhelming probability). In [DHRW16, Lemma 1], this technique was introduced for certain LWE-based encryption schemes with plaintext space $\{0, 1\}$. In the following Lemma 7.1 and Corollary 7.1 (with proofs in Appendix B), we extend this result to BFV with arbitrary plaintext space.

Lemma 7.1. *Let $t, q \in \mathbb{N}$ with $t < q$, let $x \in \mathcal{R}_q$ with $||[x]^{\boldsymbol{p}}_{q/t}||_\infty < \varepsilon\frac{q}{2t}$, and let x_0, x_1 be random variables over $\mathcal{R}_q$ with $x_0 + x_1 = x$ (i.e., (x_0, x_1) is an additive sharing of x) and $\Pr\left[||[x_0]^{\boldsymbol{p}}_{q/t}||_\infty + ||[x_1]^{\boldsymbol{p}}_{q/t}||_\infty > (1 - \frac{\varepsilon}{2})\frac{q}{t}\right] \leq \delta$. Then $\lfloor x \rceil^{\boldsymbol{p}}_{q\to t} = \lfloor x_0 \rceil^{\boldsymbol{p}}_{q\to t} + \lfloor x_1 \rceil^{\boldsymbol{p}}_{q\to t} \pmod t$ with probability $\geq 1 - \delta$.*

Corollary 7.1. *Let $\mathsf{ct} = (c_0, c_1)$ be a ciphertext under secret key s with relative noise bound negligible in the security parameter, let $s_0 \xleftarrow{\$} \mathcal{R}_q$, $s_1 \coloneqq s - s_0 \in \mathcal{R}_q$, and $\mathsf{ct}' \coloneqq (0, c_1)$. Then $\mathsf{Dec}_s(\mathsf{ct}) = \mathsf{Dec}_{s_0}(\mathsf{ct}) + \mathsf{Dec}_{s_1}(\mathsf{ct}')$ with overwhelming probability.*

7.2 Security of Our PCF

Finally, we can prove the security of our (weak) PCF:

Theorem 5. *The weak PCF from Definition 7 is a secure weak PCF (in the sense of Definition 10) for $\mathcal{Y}_\mathcal{C}$.*

Proof. First, observe that, with overwhelming probability over the random coins used by $x \leftarrow \mathsf{InputGen}(1^\lambda)$, it holds that

$$\mathsf{Eval}_{k_0}(x) + \mathsf{Eval}_{k_1}(x) \overset{(1)}{=} \boldsymbol{d}_0 + \boldsymbol{d}_1 \overset{(2)}{=} \mathsf{Dec}_s((\mathsf{ct}_j)_j) = \mathcal{C}\big((F_s(a_i))_{i\in[n_i]}\big) \tag{2}$$

where (1) is because the masks $\boldsymbol{\mu}$ cancel out and (2) is by Corollary 7.1. To a distinguisher who has no information about the secret key s, the right-hand side is indistinguishable from $\mathcal{C}(U(\mathcal{R}_t^{n_i}))$, since F is a WPRF.

– *Pseudorandom $\mathcal{Y}$-correlated outputs:* We have to prove that

$$\big(1^\lambda, \big(x^{(i)}, \mathsf{Eval}_{k_0}(x^{(i)}), \mathsf{Eval}_{k_1}(x^{(i)})\big)_i\big) \approx_c \big(1^\lambda, \big(x^{(i)}, y_0^{(i)}, y_1^{(i)}\big)_i\big)$$

where $(y_0^{(i)}, y_1^{(i)}) \sim \mathcal{Y}_\mathcal{C}$. Note that both parties, during $\mathsf{Eval}_{k_\sigma}(x^{(i)})$, use the same output masks $\boldsymbol{\mu}^{(i)} \in \mathcal{R}_t^{n_o}$. By the security of the auxiliary WPRF W, we can replace $\boldsymbol{\mu}^{(i)}$ by independent uniformly random masks $\tilde{\boldsymbol{\mu}}^{(i)} \xleftarrow{\$} \mathcal{R}_t^{n_o}$. Further, let $\boldsymbol{r}^{(i)} \coloneqq \boldsymbol{d}_0^{(i)} + \tilde{\boldsymbol{\mu}}^{(i)}$. It follows that

$$\begin{aligned}\big(1^\lambda, \big(x^{(i)}, \mathsf{Eval}_{k_0}(x^{(i)}), \mathsf{Eval}_{k_1}(x^{(i)})\big)_i\big) &\approx_c \big(1^\lambda, \big(x^{(i)}, \boldsymbol{d}_0^{(i)} + \tilde{\boldsymbol{\mu}}^{(i)}, \boldsymbol{d}_1^{(i)} - \tilde{\boldsymbol{\mu}}^{(i)}\big)_i\big)\\ &\approx_c \big(1^\lambda, \big(x^{(i)}, \underbrace{\boldsymbol{r}^{(i)}, \mathcal{C}(U(\mathcal{R}_t^{n_i})) - \boldsymbol{r}^{(i)}}_{\sim\mathcal{Y}_\mathcal{C}}\big)_i\big)\end{aligned}$$

where the latter indistinguishability follows as in Equation (2).

– *Security:* For security, we first define $\mathsf{RSample}(1^\lambda, \sigma, y_\sigma)$ as the algorithm that samples $y \leftarrow \mathcal{C}(U(\mathcal{R}_t^{n_i}))$ and returns $y_{1-\sigma} \coloneqq y - y_\sigma$. Let $y_\sigma^{(i)} \coloneqq \mathsf{Eval}_{k_\sigma}(x^{(i)})$. Again, by above observation about Equation (2), it immediately follows that

$$\big(1^\lambda, \sigma, k_\sigma, \big(x^{(i)}, \mathsf{Eval}_{k_{1-\sigma}}(x^{(i)})\big)_i\big) \approx_c \big(1^\lambda, \sigma, k_\sigma, \big(x^{(i)}, \underbrace{\mathcal{C}(U(\mathcal{R}_t^{n_i})) - y_\sigma^{(i)}}_{=\,\mathsf{RSample}(1^\lambda, \sigma, y_\sigma^{(i)})}\big)_i\big).$$

□

This final theorem concludes our construction of a new pseudorandom correlation function. Our (weak) PCF from Definition 7 can output an additive secret sharing in the two-party setup for any additive correlation over our base rings $\mathbb{Z}_t$ or $\mathbb{F}_{p^d}$, including VOLE correlations, Beaver triples, matrix triples, and convolution triples.

Acknowledgments. This research was supported by the CRYPTECS project (funded by the German Federal Ministry of Research, Technology and Space under Grant Agreement No. 16KIS1441 and by the French National Research Agency under Grant Agreement No. ANR-20-CYAL-0006) and by Advantest as part of the Graduate School "Intelligent Methods for Test and Reliability" (GS-IMTR) at the University of Stuttgart. Additionally, this research was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under grants 411720488 and 548713845. Furthermore, we thank Vincent Rieder for proofreading and for his helpful feedback on related literature.

References

AA16. Jacob Alperin-Sheriff and Daniel Apon. Dimension-preserving reductions from LWE to LWR. Cryptology ePrint Archive, Report 2016/589, 2016.

AA17. Jacob Alperin-Sheriff and Daniel Apon. Weak is better: Tightly secure short signatures from weak PRFs. Cryptology ePrint Archive, Report 2017/563, 2017.

AKPW13. Joël Alwen, Stephan Krenn, Krzysztof Pietrzak, and Daniel Wichs. Learning with rounding, revisited - new reduction, properties and applications. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 57–74, Santa Barbara, CA, USA, August 18–22, 2013. Springer Berlin Heidelberg, Germany.

APS15. Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *J. Math. Cryptol.*, 9(3):169–203, 2015.

ARS24. Damiano Abram, Lawrence Roy, and Peter Scholl. Succinct homomorphic secret sharing. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024, Part VI*, volume 14656 of *Lecture Notes in Computer Science*, pages 301–330, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland.

Bar86. David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. In *18th Annual ACM Symposium on Theory of Computing*, pages 1–5, Berkeley, CA, USA, May 28–30, 1986. ACM Press.

BBC+24. Maxime Bombar, Dung Bui, Geoffroy Couteau, Alain Couvreur, Clément Ducros, and Sacha Servan-Schreiber. FOLEAGE: $\mathbb{F}_4$OLE-based multi-party computation for boolean circuits. In Kai-Min Chung and Yu Sasaki, editors, *Advances in Cryptology – ASIACRYPT 2024, Part VI*, volume 15489 of *Lecture Notes in Computer Science*, pages 69–101, Kolkata, India, December 9–13, 2024. Springer, Singapore, Singapore.

BCC+22. Youngjin Bae, Jung Hee Cheon, Wonhee Cho, Jaehyung Kim, and Taekyung Kim. META-BTS: Bootstrapping precision beyond the limit. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022: 29th Conference on Computer and Communications Security*, pages 223–234, Los Angeles, CA, USA, November 7–11, 2022. ACM Press.

BCCD23. Maxime Bombar, Geoffroy Couteau, Alain Couvreur, and Clément Ducros. Correlated pseudorandomness from the hardness of quasi-abelian decoding. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part IV*, volume 14084 of *Lecture Notes in Computer Science*, pages 567–601, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland.

BCG+19a. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. Efficient two-round OT extension and silent non-interactive secure computation. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019: 26th Conference on Computer and Communications Security*, pages 291–308, London, UK, November 11–15, 2019. ACM Press.

BCG+19b. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 489–518, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Cham, Switzerland.

BCG+20a. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Correlated pseudorandom functions from variable-density LPN. In *61st Annual Symposium on Foundations of Computer Science*, pages 1069–1080, Durham, NC, USA, November 16–19, 2020. IEEE Computer Society Press.

BCG+20b. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Correlated pseudorandom functions from variable-density LPN. Cryptology ePrint Archive, Report 2020/1417, 2020.

BCG+20c. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators from ring-LPN. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part II*, volume 12171 of *Lecture Notes in Computer Science*, pages 387–416, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Cham, Switzerland.

BCG+23. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Nicolas Resch, and Peter Scholl. Oblivious transfer with constant computational overhead. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part I*, volume 14004 of *Lecture Notes in Computer Science*, pages 271–302, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland.

BCGI18. Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018: 25th Conference on Computer and Communications Security*, pages 896–912, Toronto, ON, Canada, October 15–19, 2018. ACM Press.

BCM+24. Dung Bui, Geoffroy Couteau, Pierre Meyer, Alain Passelègue, and Mahshid Riahinia. Fast public-key silent OT and more from constrained Naor-Reingold. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024, Part VI*, volume 14656 of *Lecture Notes in Computer Science*, pages 88–118, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland.

BCS19. Carsten Baum, Daniele Cozzo, and Nigel P. Smart. Using TopGear in overdrive: A more efficient ZKPoK for SPDZ. In Kenneth G. Paterson and Douglas Stebila, editors, *SAC 2019: 26th Annual International Workshop on Selected Areas in Cryptography*, volume 11959 of *Lecture Notes in Computer Science*, pages 274–302, Waterloo, ON, Canada, August 12–16, 2019. Springer, Cham, Switzerland.

Bea92. Donald Beaver. Efficient multiparty protocols using circuit randomization. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 420–432, Santa Barbara, CA, USA, August 11–15, 1992. Springer Berlin Heidelberg, Germany.

BGI15. Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 337–367, Sofia, Bulgaria, April 26–30, 2015. Springer Berlin Heidelberg, Germany.

BGI16a. Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under DDH. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 509–539, Santa Barbara, CA, USA, August 14–18, 2016. Springer Berlin Heidelberg, Germany.

BGI16b. Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, pages 1292–1303, Vienna, Austria, October 24–28, 2016. ACM Press.

BGI+18. Elette Boyle, Niv Gilboa, Yuval Ishai, Huijia Lin, and Stefano Tessaro. Foundations of homomorphic secret sharing. In Anna R. Karlin, editor, *ITCS 2018: 9th Innovations in Theoretical Computer Science Conference*, volume 94, pages 21:1–21:21, Cambridge, MA, USA, January 11–14, 2018. Leibniz International Proceedings in Informatics (LIPIcs).

BGM+16. Andrej Bogdanov, Siyao Guo, Daniel Masny, Silas Richelson, and Alon Rosen. On the hardness of learning with rounding over small modulus. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part I*, volume 9562 of *Lecture Notes in Computer Science*, pages 209–224, Tel Aviv, Israel, January 10–13, 2016. Springer Berlin Heidelberg, Germany.

BIP+18. Dan Boneh, Yuval Ishai, Alain Passelègue, Amit Sahai, and David J. Wu. Exploring crypto dark matter: New simple PRF candidates and their applications. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018: 16th Theory of Cryptography Conference, Part II*, volume 11240 of *Lecture Notes in Computer Science*, pages 699–729, Panaji, India, November 11–14, 2018. Springer, Cham, Switzerland.

BKS19. Elette Boyle, Lisa Kohl, and Peter Scholl. Homomorphic secret sharing from lattices without FHE. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part II*, volume 11477 of *Lecture Notes in Computer Science*, pages 3–33, Darmstadt, Germany, May 19–23, 2019. Springer, Cham, Switzerland.

BLR+18. Shi Bai, Tancrède Lepoint, Adeline Roux-Langlois, Amin Sakzad, Damien Stehlé, and Ron Steinfeld. Improved security proofs in lattice-based cryptography: Using the Rényi divergence rather than the statistical distance. *Journal of Cryptology*, 31(2):610–640, April 2018.

BNO19. Aner Ben-Efraim, Michael Nielsen, and Eran Omri. Turbospeedz: Double your online SPDZ! Improving SPDZ using function dependent preprocessing. In Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung, editors, *ACNS 2019: 17th International Conference on Applied Cryptography and Network Security*, volume 11464 of *Lecture Notes in Computer Science*, pages 530–549, Bogota, Colombia, June 5–7, 2019. Springer, Cham, Switzerland.

BPR12. Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 719–737, Cambridge, UK, April 15–19, 2012. Springer Berlin Heidelberg, Germany.

Bra12. Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 868–886, Santa Barbara, CA, USA, August 19–23, 2012. Springer Berlin Heidelberg, Germany.

CD23. Geoffroy Couteau and Clément Ducros. Pseudorandom correlation functions from variable-density LPN, revisited. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023: 26th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 13941 of *Lecture Notes in Computer Science*, pages 221–250, Atlanta, GA, USA, May 7–10, 2023. Springer, Cham, Switzerland.

CDD+24. Geoffroy Couteau, Lalita Devadas, Srinivas Devadas, Alexander Koch, and Sacha Servan-Schreiber. QuietOT: Lightweight oblivious transfer with a public-key setup. In Kai-Min Chung and Yu Sasaki, editors, *Advances in Cryptology – ASIACRYPT 2024, Part II*, volume 15485 of *Lecture Notes in Computer Science*, pages 197–231, Kolkata, India, December 9–13, 2024. Springer, Singapore, Singapore.

CDE+18. Ronald Cramer, Ivan Damgård, Daniel Escudero, Peter Scholl, and Chaoping Xing. SPD $\mathbb{Z}_{2^k}$: Efficient MPC mod 2^k for dishonest majority. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 769–798, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Cham, Switzerland.

CDH+25. Geoffroy Couteau, Lalita Devadas, Aditya Hegde, Abhishek Jain, and Sacha Servan-Schreiber. Multi-key homomorphic secret sharing. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology – EUROCRYPT 2025, Part V*, volume 15605 of *Lecture Notes in Computer Science*, pages 3–33, Madrid, Spain, May 4–8, 2025. Springer, Cham, Switzerland.

CH18. Hao Chen and Kyoohyung Han. Homomorphic lower digits removal and improved FHE bootstrapping. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 315–337, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Cham, Switzerland.

CHK+18. Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. Bootstrapping for approximate homomorphic encryption. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 360–384, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Cham, Switzerland.

CKKS17. Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. Homomorphic encryption for arithmetic of approximate numbers. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 409–437, Hong Kong, China, December 3–7, 2017. Springer, Cham, Switzerland.

CKL21. Jung Hee Cheon, Dongwoo Kim, and Keewoo Lee. MHz2k: MPC from HE over $\mathbb{Z}_{2^k}$ with new packing, simpler reshare, and better ZKP. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part II*, volume 12826 of *Lecture Notes in Computer Science*, pages 426–456, Virtual Event, August 16–20, 2021. Springer, Cham, Switzerland.

CKR+20. Hao Chen, Miran Kim, Ilya P. Razenshteyn, Dragos Rotaru, Yongsoo Song, and Sameer Wagh. Maliciously secure matrix multiplication with applications to private deep learning. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020, Part III*, volume 12493 of *Lecture Notes in Computer Science*, pages 31–59, Daejeon, South Korea, December 7–11, 2020. Springer, Cham, Switzerland.

CMPR23. Geoffroy Couteau, Pierre Meyer, Alain Passelègue, and Mahshid Riahinia. Constrained pseudorandom functions from homomorphic secret sharing. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part III*, volume 14006 of *Lecture Notes in Computer Science*, pages 194–224, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland.

COS+22. Ilaria Chillotti, Emmanuela Orsini, Peter Scholl, Nigel P. Smart, and Barry van Leeuwen. Scooby: Improved multi-party homomorphic secret sharing based on FHE. In Clemente Galdi and Stanislaw Jarecki, editors, *SCN 22: 13th International Conference on Security in Communication Networks*, volume 13409 of *Lecture Notes in Computer Science*, pages 540–563, Amalfi, Italy, September 12–14, 2022. Springer, Cham, Switzerland.

DHRW16. Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. Spooky encryption and its applications. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part III*, volume 9816 of *Lecture Notes in Computer Science*, pages 93–122, Santa Barbara, CA, USA, August 14–18, 2016. Springer Berlin Heidelberg, Germany.

DKL+13. Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P. Smart. Practical covertly secure MPC for dishonest majority - or: Breaking the SPDZ limits. In Jason Crampton, Sushil Jajodia, and Keith Mayes, editors, *ESORICS 2013: 18th European Symposium on Research in Computer Security*, volume 8134 of *Lecture Notes in Computer Science*, pages 1–18, Egham, UK, September 9–13, 2013. Springer Berlin Heidelberg, Germany.

DPSZ12. Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 643–662, Santa Barbara, CA, USA, August 19–23, 2012. Springer Berlin Heidelberg, Germany.

FV12. Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012.

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

GHS12. Craig Gentry, Shai Halevi, and Nigel P. Smart. Better bootstrapping in fully homomorphic encryption. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012: 15th International Conference on Theory and Practice of Public Key Cryptography*, volume 7293 of *Lecture Notes in Computer Science*, pages 1–16, Darmstadt, Germany, May 21–23, 2012. Springer Berlin Heidelberg, Germany.

GIKV23. Robin Geelen, Ilia Iliashenko, Jiayi Kang, and Frederik Vercauteren. On polynomial functions modulo p^e and faster bootstrapping for homomorphic encryption. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part III*, volume 14006 of *Lecture Notes in Computer Science*, pages 257–286, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland.

GV23. Robin Geelen and Frederik Vercauteren. Bootstrapping for BGV and BFV revisited. *J. Cryptol.*, 36(2):12, 2023.

HDW21. Yupu Hu, Xingting Dong, and Baocang Wang. Size of IK00 branching program. Cryptology ePrint Archive, Report 2021/400, 2021.

HRK25. Sebastian Hasler, Pascal Reisert, and Ralf Küsters. Pseudorandom correlation functions from ring-LWR. In *Advances in Cryptology – ASIACRYPT 2025*, Lecture Notes in Computer Science, Melbourne, Australia, December 2025. Springer.

HRRK24. Sebastian Hasler, Pascal Reisert, Marc Rivinius, and Ralf Küsters. Multipars: Reduced-communication MPC over Z2k. *Proceedings on Privacy Enhancing Technologies*, 2024(2):5–28, April 2024.

HS15. Shai Halevi and Victor Shoup. Bootstrapping for HElib. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 641–670, Sofia, Bulgaria, April 26–30, 2015. Springer Berlin Heidelberg, Germany.

HS21. Shai Halevi and Victor Shoup. Bootstrapping for HElib. *Journal of Cryptology*, 34(1):7, January 2021.

ILM21. Yuval Ishai, Russell W. F. Lai, and Giulio Malavolta. A geometric approach to homomorphic secret sharing. In Juan Garay, editor, *PKC 2021: 24th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 12711 of *Lecture Notes in Computer Science*, pages 92–119, Virtual Event, May 10–13, 2021. Springer, Cham, Switzerland.

JM22. Charanjit S. Jutla and Nathan Manohar. Sine series approximation of the mod function for bootstrapping of approximate HE. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part I*, volume 13275 of *Lecture Notes in Computer Science*, pages 491–520, Trondheim, Norway, May 30 – June 3, 2022. Springer, Cham, Switzerland.

Kel20. Marcel Keller. MP-SPDZ: A versatile framework for multi-party computation. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020: 27th Conference on Computer and Communications Security*, pages 1575–1590, Virtual Event, USA, November 9–13, 2020. ACM Press.

KOS16. Marcel Keller, Emmanuela Orsini, and Peter Scholl. MASCOT: Faster malicious arithmetic secure computation with oblivious transfer. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, pages 830–842, Vienna, Austria, October 24–28, 2016. ACM Press.

KPR18. Marcel Keller, Valerio Pastro, and Dragos Rotaru. Overdrive: Making SPDZ great again. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018,*

Part III, volume 10822 of *Lecture Notes in Computer Science*, pages 158–189, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Cham, Switzerland.

KSS24. Jaehyung Kim, Jinyeong Seo, and Yongsoo Song. Simpler and faster BFV bootstrapping for arbitrary plaintext modulus from CKKS. Cryptology ePrint Archive, Report 2024/109, 2024.

LLK+22. Yongwoo Lee, Joon-Woo Lee, Young-Sik Kim, Yongjune Kim, Jong-Seon No, and HyungChul Kang. High-precision bootstrapping for approximate homomorphic encryption by error variance minimization. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part I*, volume 13275 of *Lecture Notes in Computer Science*, pages 551–580, Trondheim, Norway, May 30 – June 3, 2022. Springer, Cham, Switzerland.

LLL+21. Joon-Woo Lee, Eunsang Lee, Yongwoo Lee, Young-Sik Kim, and Jong-Seon No. High-precision bootstrapping of RNS-CKKS homomorphic encryption using optimal minimax polynomial approximation and inverse sine function. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part I*, volume 12696 of *Lecture Notes in Computer Science*, pages 618–647, Zagreb, Croatia, October 17–21, 2021. Springer, Cham, Switzerland.

LPR10. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23, French Riviera, May 30 – June 3, 2010. Springer Berlin Heidelberg, Germany.

LPR13. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-LWE cryptography. Cryptology ePrint Archive, Report 2013/293, 2013.

LW20. Feng-Hao Liu and Zhedong Wang. Rounding in the rings. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part II*, volume 12171 of *Lecture Notes in Computer Science*, pages 296–326, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Cham, Switzerland.

LXYY25. Zhe Li, Chaoping Xing, Yizhou Yao, and Chen Yuan. Efficient pseudorandom correlation generators for any finite field. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology – EUROCRYPT 2025, Part V*, volume 15605 of *Lecture Notes in Computer Science*, pages 145–175, Madrid, Spain, May 4–8, 2025. Springer, Cham, Switzerland.

MG23. Johannes Mono and Tim Güneysu. Implementing and optimizing matrix triples with homomorphic encryption. In Joseph K. Liu, Yang Xiang, Surya Nepal, and Gene Tsudik, editors, *ASIACCS 23: 18th ACM Symposium on Information, Computer and Communications Security*, pages 29–40, Melbourne, VIC, Australia, July 10–14, 2023. ACM Press.

MV12. Eric Miles and Emanuele Viola. Substitution-permutation networks, pseudorandom functions, and natural proofs. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 68–85, Santa Barbara, CA, USA, August 19–23, 2012. Springer Berlin Heidelberg, Germany.

MZ17. Payman Mohassel and Yupeng Zhang. SecureML: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy*, pages 19–38, San Jose, CA, USA, May 22–26, 2017. IEEE Computer Society Press.

NR04. Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *Journal of the ACM*, 51(2):231–262, March 2004.

ON20. Satsuya Ohata and Koji Nuida. Communication-efficient (client-aided) secure two-party protocols and its application. In Joseph Bonneau and Nadia Heninger, editors, *FC 2020: 24th International Conference on Financial Cryptography and Data Security*, volume 12059 of *Lecture Notes in Computer Science*, pages 369–385, Kota Kinabalu, Malaysia, February 10–14, 2020. Springer, Cham, Switzerland.

OPP23. Hiroki Okada, Rachel Player, and Simon Pohmann. Homomorphic polynomial evaluation using Galois structure and applications to BFV bootstrapping. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023, Part VI*, volume 14443 of *Lecture Notes in Computer Science*, pages 69–100, Guangzhou, China, December 4–8, 2023. Springer, Singapore, Singapore.

OSV20. Emmanuela Orsini, Nigel P. Smart, and Frederik Vercauteren. Overdrive2k: Efficient secure MPC over $\mathbb{Z}_{2^k}$ from somewhat homomorphic encryption. In Stanislaw Jarecki, editor, *Topics in Cryptology – CT-RSA 2020*, volume 12006 of *Lecture Notes in Computer Science*, pages 254–283, San Francisco, CA, USA, February 24–28, 2020. Springer, Cham, Switzerland.

OSY21. Claudio Orlandi, Peter Scholl, and Sophia Yakoubov. The rise of paillier: Homomorphic secret sharing and public-key silent OT. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part I*, volume 12696 of *Lecture Notes in Computer Science*, pages 678–708, Zagreb, Croatia, October 17–21, 2021. Springer, Cham, Switzerland.

PS08. Krzysztof Pietrzak and Johan Sjödin. Weak pseudorandom functions in minicrypt. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP 2008: 35th International Colloquium on Automata, Languages*

and Programming, Part II, volume 5126 of *Lecture Notes in Computer Science*, pages 423–436, Reykjavik, Iceland, July 7–11, 2008. Springer Berlin Heidelberg, Germany.

PSSY21. Arpita Patra, Thomas Schneider, Ajith Suresh, and Hossein Yalame. ABY2.0: Improved mixed-protocol secure two-party computation. In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021: 30th USENIX Security Symposium*, pages 2165–2182. USENIX Association, August 11–13, 2021.

Reg05. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press.

RRHK23. Marc Rivinius, Pascal Reisert, Sebastian Hasler, and Ralf Küsters. Convolutions in overdrive: Maliciously secure convolutions for MPC. *Proceedings on Privacy Enhancing Technologies*, 2023(3):321–353, July 2023.

RRK+24. Pascal Reisert, Marc Rivinius, Toomas Krips, Sebastian Hasler, and Ralf Küsters. Actively secure polynomial evaluation from shared polynomial encodings. In Kai-Min Chung and Yu Sasaki, editors, *Advances in Cryptology – ASIACRYPT 2024, Part VI*, volume 15489 of *Lecture Notes in Computer Science*, pages 3–35, Kolkata, India, December 9–13, 2024. Springer, Singapore, Singapore.

RRKK23. Pascal Reisert, Marc Rivinius, Toomas Krips, and Ralf Küsters. Overdrive LowGear 2.0: Reduced-bandwidth MPC without sacrifice. In Joseph K. Liu, Yang Xiang, Surya Nepal, and Gene Tsudik, editors, *ASIACCS 23: 18th ACM Symposium on Information, Computer and Communications Security*, pages 372–386, Melbourne, VIC, Australia, July 10–14, 2023. ACM Press.

SEA23. Microsoft SEAL (release 4.1). `https://github.com/Microsoft/SEAL`, January 2023. Microsoft Research, Redmond, WA.

SGRR19. Phillipp Schoppmann, Adrià Gascón, Leonie Reichert, and Mariana Raykova. Distributed vector-OLE: Improved constructions and implementation. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019: 26th Conference on Computer and Communications Security*, pages 1055–1072, London, UK, November 11–15, 2019. ACM Press.

Sma. Nigel Smart. 'Bristol Fashion' MPC Circuits. `https://nigelsmart.github.io/MPC-Circuits/`. Last accessed May 15th, 2025.

YWL+20. Kang Yang, Chenkai Weng, Xiao Lan, Jiang Zhang, and Xiao Wang. Ferret: Fast extension for correlated OT with small communication. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020: 27th Conference on Computer and Communications Security*, pages 1607–1626, Virtual Event, USA, November 9–13, 2020. ACM Press.

A Definitions for Weak PRF and PCF

In this appendix we formally define weak pseudorandom functions (WPRFs) and pseudorandom correlation functions (PCFs).

A WPRF is a PRF that requires the inputs to be chosen according to some prescribed input sampling algorithm. Formally WPRFs are defined as follows:

Definition 8 (Weak Pseudorandom Function). *A WPRF consists of a key generation algorithm* KeyGen*, an input sampling algorithm* $\mathsf{InputGen}$*, and a polynomial-time computable deterministic function* $F_k(x)$ *with range* $\mathcal{D}$*, such that:*

$$\left\{ \left(1^\lambda, \left(x^{(i)}, F_k(x^{(i)})\right)_{i\in[\mathsf{poly}(\lambda)]}\right) : x^{(i)} \leftarrow \mathsf{InputGen}(1^\lambda), k \leftarrow \mathsf{KeyGen}(1^\lambda) \right\}$$
$$\approx_c \left\{ \left(1^\lambda, \left(x^{(i)}, y^{(i)}\right)_{i\in[\mathsf{poly}(\lambda)]}\right) : x^{(i)} \leftarrow \mathsf{InputGen}(1^\lambda), y^{(i)} \xleftarrow{\$} \mathcal{D} \right\}.$$

where we denoted computational distinguishability (w.r.t. security parameter λ*) by* $\approx_c$*.*

A weak PCF is built from two keyed functions Eval_{k_σ} (for $\sigma = 0, 1$), such that $(\mathsf{Eval}_{k_0}(x^{(i)}), \mathsf{Eval}_{k_1}(x^{(i)}))$ follows a *target correlation* $\mathcal{Y} = (Y_0, Y_1)$. A simple example for such a target correlation is the Beaver triple correlation $((a_0, b_0, c_0), (a_1, b_1, c_1))$ where all entries are uniformly random except for c_1, which is uniquely determined as $c_1 = (a_0 + a_1) \cdot (b_0 + b_1) - c_0$.

To ensure security of PCFs in useful applications like MPC, Boyle et al. [BCG+20a] require $Y_\sigma \approx \mathsf{Eval}_{k_\sigma}$ to be private to party P_σ, i.e., $(k_\sigma, \mathsf{Eval}_{k_\sigma}(x^{(i)}))$ should leak nothing about $\mathsf{Eval}_{k_{1-\sigma}}(x^{(i)})$, except what can be inferred from the fact that $(\mathsf{Eval}_{k_0}(x^{(i)}), \mathsf{Eval}_{k_1}(x^{(i)}))$ follows $\mathcal{Y}$. This is captured by demanding (in the *security* part of Definition 10) that $\mathsf{Eval}_{k_{1-\sigma}}$ is indistinguishable from some distribution which party P_σ could have sampled herself. To formally capture this property, we recall the concept of a reverse-sampleable correlation from Boyle et al. [BCG+19b]:

Definition 9 (Reverse-Sampleable Correlation). *Let* l_0, l_1 *be polynomials and let* $\mathcal{Y}(1^\lambda)$ *be a probabilistic algorithm that outputs a pair* $(y_0, y_1) \in \{0,1\}^{l_0(\lambda)} \times \{0,1\}^{l_1(\lambda)}$*.* $\mathcal{Y}$ *is a reverse-sampleable correlation, if there exists a probabilistic and polynomial time (ppt.) algorithm* $\mathsf{RSample}(1^\lambda, \sigma, y_\sigma)$ *such that* $\mathcal{Y}(1^\lambda)$ *is statistically close to*

$$\left\{ (y_0, y_1) : (y_0', y_1') \xleftarrow{\$} \mathcal{Y}(1^\lambda), y_\sigma := y_\sigma', y_{1-\sigma} \leftarrow \mathsf{RSample}(1^\lambda, \sigma, y_\sigma) \right\}.$$

Remark. For our concrete PCF constructions, all correlations $\mathcal{Y}$ are additive correlations (Definition 6) which are always reverse-sampleable.

Finally, we restate the notion of a weak PCF from [BCG+20b, Definition 4.3].

Definition 10 (Weak Pseudorandom Correlation Function). *A weak PCF for a reverse-sampleable correlation* $\mathcal{Y}$*, with respect to an input sampling algorithm* $\mathsf{InputGen}$*, consists of*

- *a ppt. key generation algorithm* $\mathsf{KeyGen}(1^\lambda)$ *that outputs a pair of private keys* (k_0, k_1) *(where* λ *can be inferred from a key) and*
- *a polynomial-time computable deterministic function* $\mathsf{Eval}(\sigma, k_\sigma, \cdot)$*, or* Eval_{k_σ} *for short, with outputs in the range of* $(\mathcal{Y}(1^\lambda))_\sigma$*.*

A weak PCF is secure if the following two properties hold:

– ***Pseudorandom $\mathcal{Y}$-correlated outputs.***

$$\left\{ \left(1^\lambda, \left(x^{(i)}, \mathsf{Eval}_{k_0}(x^{(i)}), \mathsf{Eval}_{k_1}(x^{(i)})\right)_{i\in[\mathsf{poly}(\lambda)]}\right) : \begin{smallmatrix}(k_0,k_1)\leftarrow\mathsf{KeyGen}(1^\lambda),\\ x^{(i)}\leftarrow\mathsf{InputGen}(1^\lambda)\end{smallmatrix} \right\}$$
$$\approx_c \left\{ \left(1^\lambda, \left(x^{(i)}, y_0^{(i)}, y_1^{(i)}\right)_{i\in[\mathsf{poly}(\lambda)]}\right) : \begin{smallmatrix}x^{(i)}\leftarrow\mathsf{InputGen}(1^\lambda),\\ (y_0^{(i)},y_1^{(i)})\leftarrow\mathcal{Y}(1^\lambda)\end{smallmatrix} \right\}.$$

– ***Security.*** $\forall\sigma\in\{0,1\}$*:*

$$\left\{ \left(1^\lambda, \sigma, k_\sigma, \left(x^{(i)}, \mathsf{Eval}_{k_{1-\sigma}}(x^{(i)})\right)_{i\in[\mathsf{poly}(\lambda)]}\right) : \begin{smallmatrix}(k_0,k_1)\leftarrow\mathsf{KeyGen}(1^\lambda),\\ x^{(i)}\leftarrow\mathsf{InputGen}(1^\lambda)\end{smallmatrix} \right\}$$
$$\approx_c \left\{ \left(1^\lambda, \sigma, k_\sigma, \left(x^{(i)}, y_{1-\sigma}^{(i)}\right)_{i\in[\mathsf{poly}(\lambda)]}\right) : \begin{smallmatrix}(k_0,k_1)\leftarrow\mathsf{KeyGen}(1^\lambda),\\ x^{(i)}\leftarrow\mathsf{InputGen}(1^\lambda),\\ y_\sigma^{(i)}=\mathsf{Eval}_{k_\sigma}(x^{(i)}),\\ y_{1-\sigma}^{(i)}\leftarrow\mathsf{RSample}(1^\lambda,\sigma,y_\sigma^{(i)})\end{smallmatrix} \right\}.$$

B Proofs

As a preparation for the proof of Lemma 4.1, we need the following lemma:

Lemma B.1. *The dual basis of $\boldsymbol{p}$ is $\boldsymbol{p}^\vee = (g/\hat{m})L_m\tau(\boldsymbol{p})$ where $\tau = \sigma_{-1}$ and $L_m \in \{0,1\}^{\varphi(m)\times\varphi(m)}$ is the matrix*

$$L_m = \bigotimes\nolimits_p (L_p \otimes I_{[m/p]})$$

where the tensor product runs over all primes p that divide m and, for prime p, $L_p \in \{0,1\}^{\varphi(p)\times\varphi(p)}$ is the lower triangular matrix filled with 1s, i.e., the (i,j)th entry is 1 iff. $i \geq j$.

Before we prove this lemma, note that it is very closely related to [LPR13, Lemma 6.3] on how to obtain the decoding basis $\boldsymbol{d} = \tau(\boldsymbol{p})^\vee$. We want to obtain $\boldsymbol{p}^\vee$ instead, which could be computed as $\boldsymbol{p}^\vee = \tau(\boldsymbol{d}) = \tau((g/\hat{m})L_m^T\boldsymbol{p})$, but the term $\tau(g/\hat{m})$ would be troublesome in our upcoming proof of Lemma 4.1 as it doesn't cancel out with $\hat{m}/g$. The insight of Lemma B.1 is that this can be solved by replacing L_m^T with L_m and our following proof works largely analogously to [LPR13, Lemma 6.3].

Proof. We first show the statement for prime power $m = p^e$. First note that $g/\hat{m} = (1-\zeta_p)/m$, even in the case of $p = 2$, since $1 - \zeta_2 = 2$. In the prime power case, the equation $\boldsymbol{p}^\vee = (g/\hat{m})\tau(L_m\boldsymbol{p})$ can be restated as follows: For all row indices $(i_0, i_1) \in [\varphi(p)] \times [m/p]$, it must hold that

$$\tau(p^\vee_{(i_0,i_1)}) = \tau(g/\hat{m}) \cdot \left((\zeta_p^0 + \cdots + \zeta_p^{i_0}) \cdot \zeta_m^{i_1}\right) = \frac{1}{m}\left(\zeta_p^{i_0} - \zeta_p^{-1}\right) \cdot \zeta_m^{i_1} \tag{3}$$

where the latter equality is due to $\tau(g/\hat{m}) = \tau((1-\zeta_p)/m) = (1-\zeta_p^{-1})/m$. To verify Equation (3), we have to show that the right-hand side multiplied by $\tau(p_{(i'_0,i'_1)})$ for any $(i'_0, i'_1) \in [\varphi(p)]\times[m/p]$ has a trace of 1 iff $(i_0, i_1) = (i'_0, i'_1)$. This verification and the extension to non-prime-powers m is entirely analogous to the proof of [LPR13, Lemma 6.3]. □

Lemma 4.1. *Let $\hat{\boldsymbol{p}}^\vee$ be the dual basis of the powerful basis $\hat{\boldsymbol{p}}$ of $\mathcal{R}^\vee$. The length of each element $\hat{p}_i^\vee$ of $\hat{\boldsymbol{p}}^\vee$ under the canonical embedding is bounded by*

$$||\sigma(\hat{p}_i^\vee)||_2 \leq \sqrt{\varphi(m)} \prod\nolimits_p \varphi(p)$$

where the product runs over all primes p that divide m.

Proof. Using the definition of $\hat{\boldsymbol{p}}$ and Lemma B.1, it follows that $\hat{\boldsymbol{p}}^\vee = (\hat{m}/g)\boldsymbol{p}^\vee = L_m\tau(\boldsymbol{p})$. Using $||\sigma(p_j)||_2 = \sqrt{\varphi(m)}$ for any $j \in [\varphi(m)]$, it follows that $||\sigma(\hat{p}_i^\vee)||_2 \leq \sqrt{\varphi(m)} \cdot ||(L_m)_{i\cdot}||_1$ where $(L_m)_{i\cdot}$ is the ith row of L_m. The statement to be shown follows since L_m is a tensor product of matrices $L_p \otimes I_{[m/p]}$ that contain at most $\varphi(p)$ 1s per row and otherwise 0s. □

Next, we will briefly explain why the basis change leads to the overhead factor $\varphi(m)$ in case (2) of Section 4.

We start with the following Lemma B.2, which shows how rounding works if a change of basis is required.

Lemma B.2. *Let $\boldsymbol{b}_2, \boldsymbol{b}_1$ with $\boldsymbol{b}_2 = M\boldsymbol{b}_1$ (for transition matrix $M \in \mathbb{Z}^{n\times n}$) be $\mathbb{Z}$-bases of a fractional ideal $\mathcal{I} \subset \mathcal{K}$ and let $x \in \mathcal{K}$. Then $\lfloor x \rceil^{\boldsymbol{b}_1} = \langle M^{-T}\lfloor M^T\boldsymbol{\lambda}_x^{\boldsymbol{b}_2}\rceil, \boldsymbol{b}_2\rangle$. Furthermore, if $\lfloor x \rceil^{\boldsymbol{b}_1} + t\mathcal{I}$ (for some $t > 1$) is uniformly distributed over $\mathcal{I}_t$, then so is $\langle \lfloor M^T\boldsymbol{\lambda}_x^{\boldsymbol{b}_2}\rceil, \boldsymbol{b}_2\rangle + t\mathcal{I}$.*

Proof. For the first statement, observe that

$$\begin{aligned}\lfloor x \rceil^{\boldsymbol{b}_1} = \lfloor \langle \boldsymbol{\lambda}_x^{\boldsymbol{b}_2}, \boldsymbol{b}_2\rangle \rceil^{\boldsymbol{b}_1} = \lfloor \langle \boldsymbol{\lambda}_x^{\boldsymbol{b}_2}, M\boldsymbol{b}_1\rangle \rceil^{\boldsymbol{b}_1} = \lfloor \langle M^T\boldsymbol{\lambda}_x^{\boldsymbol{b}_2}, \boldsymbol{b}_1\rangle \rceil^{\boldsymbol{b}_1} = \langle \lfloor M^T\boldsymbol{\lambda}_x^{\boldsymbol{b}_2}\rceil, \boldsymbol{b}_1\rangle \\ = \langle \lfloor M^T\boldsymbol{\lambda}_x^{\boldsymbol{b}_2}\rceil, M^{-1}\boldsymbol{b}_2\rangle = \langle M^{-T}\lfloor M^T\boldsymbol{\lambda}_x^{\boldsymbol{b}_2}\rceil, \boldsymbol{b}_2\rangle.\end{aligned}$$

The final multiplication by M^{-T} is bijective over $\mathbb{Z}_t^n$ since $\det(M^{-T}) = \pm 1$, so it can be omitted while preserving the uniform distribution over $\mathcal{I}_t$. □

By Lemma B.2, to round some element $x \in \mathcal{K}$ w.r.t. $\boldsymbol{b}_1$ while working in some different basis $\boldsymbol{b}_2$, one needs to apply M^T to the coefficient vector $\boldsymbol{\lambda}_x^{\boldsymbol{b}_2}$, then round the coefficients, and then apply the inverse M^{-T}. In the $\mathcal{R}$-LWR context (where $x = (t/q)as$), we are interested in a uniformly distributed output, so we can omit to apply M^{-T} after rounding, i.e.:

Corollary B.1. *Let $\boldsymbol{b}_2, \boldsymbol{b}_1, M, \mathcal{I}$ as in Lemma B.2. In $\mathcal{R}$-LWR$_{\boldsymbol{b}_1,q,t,\ell}$, the second component $\lfloor as + q\mathcal{I}\rceil_{q\to t}^{\boldsymbol{b}_1}$ can be replaced by $\lfloor \langle M^T\boldsymbol{\lambda}_{as}^{\boldsymbol{b}_2}, \boldsymbol{b}_2\rangle + q\mathcal{I}\rceil_{q\to t}^{\boldsymbol{b}_2}$ while preserving hardness of the corresponding search/decision problems.*

For instance, when we generally work in the powerful basis $\boldsymbol{b}_2 = \boldsymbol{p}$ and use primal $\mathcal{R}$-LWR w.r.t. basis $\boldsymbol{b}_1 = (\hat{m}/g)\boldsymbol{d}$ (as suggested above in choice 2), we have $\boldsymbol{b}_2 = M\boldsymbol{b}_1$ for $M = L_m^{-T}$. Still, applying the initial linear transformation M^T to $\boldsymbol{\lambda}_{as}^{\boldsymbol{p}}$ is challenging in some applications, including ours, since we need to do this under homomorphic encryption. However, for the case where the cyclotomic index m is prime, the following Lemma B.3 shows that computing $M^T\boldsymbol{\lambda}_x^{\boldsymbol{p}}$ reduces to computing a single coefficient extraction.

Lemma B.3. *Let m be prime, let $M = L_m^{-T}$, let $x \in \mathcal{K}$, and let $z \in \mathbb{Z}$ be the last powerful basis coefficient of x (i.e., z is the last entry of $\boldsymbol{\lambda}_x^{\boldsymbol{p}}$). Then $\langle M^T\boldsymbol{\lambda}_x^{\boldsymbol{p}}, \boldsymbol{p}\rangle = (1-\zeta_m)x + \zeta_m^{-1}z$.*

Proof. Observe $\langle M^T\boldsymbol{\lambda}_x^{\boldsymbol{p}}, \boldsymbol{p}\rangle = \langle \boldsymbol{\lambda}_x^{\boldsymbol{p}}, L_m^{-T}\boldsymbol{p}\rangle$, so it suffices to show that $L_m^{-T}\boldsymbol{p} = (1-\zeta_m)\boldsymbol{p} + (0,\ldots,0,\zeta_m^{-1})$. This follows directly from the following Lemma B.4. □

Lemma B.4. *For m with prime power factorization $m = \prod_\ell m_\ell = \prod_\ell p_\ell^{e_\ell}$, it holds that*

$$L_m^{-T}\boldsymbol{p} = \bigotimes\nolimits_\ell ((1-\zeta_{p_\ell})\boldsymbol{p}_{m_\ell} + \boldsymbol{e}_{m_\ell})$$

where $\boldsymbol{e}_{m_\ell} \coloneqq (0,\ldots,0,\zeta_{p_\ell}^{-1}) \otimes (\zeta_{m_\ell}^{i_1})_{i_1\in[m_\ell/p_\ell]} \in \mathcal{R}^{[\varphi(p_\ell)\times[m_\ell/p_\ell]]}$.

Proof. Recall the definition of L_m from Section 3.2. We have $L_m^{-T} = \bigotimes_p L_p^{-T} \otimes I_{[m/p]}$ where the tensor product runs over all primes p that divide m and, for prime p, L_p^{-T} is the bidiagonal

matrix that contains 1s in its major diagonal and −1s in its upper minor diagonal. For an odd prime power $m = p^e$, observe that

$$\begin{aligned}(L_m^{-T}\boldsymbol{p})_{(i_0,i_1)} &= (\zeta_p^{i_0} - \mathbb{1}_{(i_0<p-2)}\zeta_p^{i_0+1})\zeta_m^{i_1} \\ &= (1-\zeta_p)\zeta_p^{i_0}\zeta_m^{i_1} + \mathbb{1}_{(i_0=p-2)}\zeta_p^{p-1}\zeta_m^{i_1} = (1-\zeta_p)\boldsymbol{p} + \boldsymbol{e}_m.\end{aligned} \tag{4}$$

For an even prime power $m = 2^e$, L_m is the identity matrix and $(1-\zeta_p)\boldsymbol{p} + \boldsymbol{e}_m = (1-(-1))\boldsymbol{p} + (-1)\boldsymbol{p} = \boldsymbol{p}$, so Equation (4) holds in this case, too. Now more generally, for m with arbitrary prime power factorization $m = \prod_\ell m_\ell = \prod_\ell p_\ell^{e_\ell}$, it follows that

$$L_m^{-T}\boldsymbol{p} = \bigotimes\nolimits_\ell L_{m_\ell}^{-T}\boldsymbol{p}_{m_\ell} = \bigotimes\nolimits_\ell((1-\zeta_{p_\ell})\boldsymbol{p}_{m_\ell} + \boldsymbol{e}_{m_\ell}).$$

□

That is, computing $\langle M^T\boldsymbol{\lambda}_x^{\boldsymbol{p}}, \boldsymbol{p}\rangle$ is as expensive as a single homomorphic coefficient extraction (since the subsequent multiplications by small constants and homomorphic addition are cheap). For arbitrary m, the number of required coefficient extractions is $\approx \sum_p m/p$ where p runs over all odd primes dividing m. For the prime m case, the following Lemma B.5 shows how to extract the last coefficient.

Lemma B.5. *For prime power $m = p^e$, the last powerful basis coefficient of any $x \in \mathcal{K}$ is $z = Tr((\zeta_p - 1)\zeta_m m^{-1}x)$.*

Proof. We look for the $\varphi(m) - 1$ index. By linearity of the trace, it suffices to prove that

$$\mathrm{Tr}((\zeta_p - 1)\zeta_m\zeta_m^{\varphi(m)-1-k}) = \begin{cases} m & \text{for } k = 0, \\ 0 & \text{for } 0 < k < \varphi(m). \end{cases} \tag{5}$$

Using $\zeta_m\zeta_m^{\varphi(m)-1-k} = \zeta_m^{\frac{m}{p}(p-1)-k} = \zeta_p^{-1}\zeta_m^{-k}$, the left-hand side of Equation (5) equals $\sum_{i\in\mathbb{Z}_m^*}\sigma_i((1-\zeta_p^{-1})\zeta_m^{-k})$. Now if $k = 0$, we get the sum $\varphi(m) - \sum_{j\in\mathbb{Z}_m^*}\zeta_p^{-j} = \varphi(m) + \sum_{j\in p\mathbb{Z}\cap[m]}\zeta_p^{-j} = \varphi(m) + p^{e-1} = m$, since the sum over all roots ζ_p^{-j} (over $j \in [m]$) is zero. In the other case $0 < k < \varphi(m)$, we get 0, since $\sum_{i\in\mathbb{Z}_m^*}\zeta_m^{-ik} = 0 = \sum_{i\in\mathbb{Z}_m^*}\zeta_m^{-i(k+m/p)}$ for $k \not\equiv 0 \not\equiv k + m/p \pmod m$. □

Recall that the trace can be written as $\mathrm{Tr}(a) = \sum_{i\in\mathbb{Z}_m^*}\sigma_i(a)$ and the automorphisms σ_i can be computed homomorphically as described in Section 3.4. Lemma B.5 is just an example (motivated by Lemma B.3) and other coefficient extractions are possible, too.

In Section 5 we use a key switching procedure in case (1) of Section 4. This key switching can be extended to case (2) using Lemma B.3 combined with Lemma B.5. However, the sum over all $i \in \mathbb{Z}_m^*$ in the trace then leads to $\varphi(m)$-times higher final modulus. As mentioned, this is more than the additional factor $\prod_p \varphi(p)$ induced by case (1).

The following lemma is used in Section 3.

Lemma B.6. *Let ω be the number of* odd *primes that divide m. Then $\gamma_1 = 2^\omega$.*

Proof. We first show $||ab'||_\infty \le 2^\omega||a||_\infty$ for any element b' of the powerful basis. Observe that

$$ab' = a\cdot\prod\nolimits_\ell\zeta_{p_\ell}^{j_\ell} = \left\langle\boldsymbol{\lambda}_a^{\boldsymbol{p}}, \bigotimes\nolimits_\ell\boldsymbol{p}_{m_\ell}\right\rangle\cdot\prod\nolimits_\ell\zeta_{p_\ell}^{j_\ell} = \left\langle\boldsymbol{\lambda}_a^{\boldsymbol{p}}, \bigotimes\nolimits_\ell(\zeta_{p_\ell}^{i_\ell+j_\ell})_{i_\ell}\right\rangle$$

and $(\zeta_{p_\ell}^{i_\ell+j_\ell})_{i_\ell} = M_\ell\boldsymbol{p}_{m_\ell}$ for a matrix $M_\ell \in \{-1,0,1\}^{\varphi(m_\ell)\times\varphi(m_\ell)}$ that has at most two non-zero entries per column (and only one if $p_{m_\ell} = 2$). Observe that $ab' = \langle(\bigotimes_\ell M_\ell)^T\boldsymbol{\lambda}_a^{\boldsymbol{p}}, \boldsymbol{p}\rangle$,

hence $||ab'||_\infty = ||(\bigotimes_\ell M_\ell)^T \boldsymbol{\lambda}_a^{\boldsymbol{p}}||_\infty \leq 2^\omega ||a||_\infty$. For general $b \in \mathcal{R}$, it follows $||ab||_\infty = ||\langle \boldsymbol{\lambda}_b^{\boldsymbol{p}}, a\boldsymbol{p}\rangle||_\infty \leq ||b||_1 \cdot 2^\omega ||a||_\infty$, hence $\gamma_1 \leq 2^\omega$.

To show that this bound is tight, we choose $b' = \prod_\ell \zeta_{p_\ell}$, so M_ℓ has a column with exactly two non-zero entries, except when $p_{m_\ell} = 2$ where every column has exactly one non-zero entry. Therefore $\bigotimes_\ell M_\ell$ has a column with exactly 2^ω non-zero entries. We further choose a as the elements with powerful basis coefficients given by that column, so $||ab'||_\infty = 2^\omega = 2^\omega ||a||_\infty ||b'||_\infty$. □

In the following, we include further proofs for statements from Section 7.

Lemma 7.1. *Let $t, q \in \mathbb{N}$ with $t < q$, let $x \in \mathcal{R}_q$ with $||[x]_{q/t}^{\boldsymbol{p}}||_\infty < \varepsilon \frac{q}{2t}$, and let x_0, x_1 be random variables over $\mathcal{R}_q$ with $x_0 + x_1 = x$ (i.e., (x_0, x_1) is an additive sharing of x) and $\Pr\left[||[x_0]_{q/t}^{\boldsymbol{p}}||_\infty + ||[x_1]_{q/t}^{\boldsymbol{p}}||_\infty > (1 - \frac{\varepsilon}{2})\frac{q}{t}\right] \leq \delta$. Then $\lfloor x \rceil_{q\to t}^{\boldsymbol{p}} = \lfloor x_0 \rceil_{q\to t}^{\boldsymbol{p}} + \lfloor x_1 \rceil_{q\to t}^{\boldsymbol{p}} \pmod t$ with probability $\geq 1 - \delta$.*

Proof. For $a \in \mathcal{R}_q$, we have $a = (q/t)\lfloor a \rceil_{q\to t}^{\boldsymbol{p}} + [a]_{q/t}^{\boldsymbol{p}}$. It follows that

$$\lfloor x \rceil_{q\to t}^{\boldsymbol{p}} = \frac{t}{q}(x_0 + x_1 - [x]_{q/t}^{\boldsymbol{p}}) = \lfloor x_0 \rceil_{q\to t}^{\boldsymbol{p}} + \lfloor x_1 \rceil_{q\to t}^{\boldsymbol{p}} + \underbrace{\frac{t}{q}([x_0]_{q/t}^{\boldsymbol{p}} + [x_1]_{q/t}^{\boldsymbol{p}} - [x]_{q/t}^{\boldsymbol{p}})}_{=:\, e}.$$

Assuming $||[x_0]_{q/t}^{\boldsymbol{p}}||_\infty + ||[x_1]_{q/t}^{\boldsymbol{p}}||_\infty \leq (1 - \frac{\varepsilon}{2})\frac{q}{t}$ (which is presumed to have probability $\geq 1 - \delta$), it follows that $||e||_\infty < 1$ and therefore $e = 0$, since all other terms in above equation are integral. □

Corollary 7.1. *Let $\mathsf{ct} = (c_0, c_1)$ be a ciphertext under secret key s with relative noise bound negligible in the security parameter, let $s_0 \xleftarrow{\$} \mathcal{R}_q$, $s_1 \coloneqq s - s_0 \in \mathcal{R}_q$, and $\mathsf{ct}' \coloneqq (0, c_1)$. Then $\mathsf{Dec}_s(\mathsf{ct}) = \mathsf{Dec}_{s_0}(\mathsf{ct}) + \mathsf{Dec}_{s_1}(\mathsf{ct}')$ with overwhelming probability.*

Proof. We instantiate Lemma 7.1 with $x = \mathsf{ct}(s)$, $x_0 = \mathsf{ct}(s_0)$, and $x_1 = \mathsf{ct}'(s_1)$. Then $\mathsf{Dec}_s(\mathsf{ct}) = \lfloor x \rceil_{q\to t}^{\boldsymbol{p}}$, $\mathsf{Dec}_{s_0}(\mathsf{ct}) = \lfloor x_0 \rceil_{q\to t}^{\boldsymbol{p}}$, $\mathsf{Dec}_{s_1}(\mathsf{ct}') = \lfloor x_1 \rceil_{q\to t}^{\boldsymbol{p}}$, and $x = c_0 + c_1 s = c_0 + c_1 s_0 + c_1 s_1 = x_0 + x_1$. In this case, the term $[x]_{q/t}^{\boldsymbol{p}}$ is the noise of ct, and ε is the ratio between the norm of the noise and the noise ceiling $\frac{q}{2t}$, which was presumed to be negligible. Note that $||[x_0]_{q/t}^{\boldsymbol{p}}||_\infty + ||[x_1]_{q/t}^{\boldsymbol{p}}||_\infty$ can be at most q/t, so $\Pr\left[||[x_0]_{q/t}^{\boldsymbol{p}}||_\infty + ||[x_1]_{q/t}^{\boldsymbol{p}}||_\infty > (1 - \frac{\varepsilon}{2})\frac{q}{t}\right]$ is negligible for uniform s_0. □

C Packing Techniques for Encrypted Randomness

In this final appendix we present packing techniques compatible with our new generation of encrypted randomness over $\mathbb{Z}_t$. If $t = p$ is a prime which has order d in $\mathbb{Z}_m^*$, then $\mathcal{R}_t$ decomposes nicely into $\varphi(m)/d$ copies of the field $\mathbb{F}_{p^d}$ and addition as well as multiplication in $\mathcal{R}_t$ correspond to entrywise addition and multiplication in the tuple of copies of $\mathbb{F}_{t^d}$. This setup is for example used in state-of-the-art implementations like MP-SPDZ [Kel20] for the special case where $d = 1$.

However, working in $\mathbb{F}_{p^d}$ is not always the most efficient choice. For instance, SPDZ can be also instantiated over $\mathbb{Z}_{2^k}, k > 1$, [CDE+18] which enables a more efficient online phase, as arithmetic modulo 2^k is implemented natively in hardware. Unfortunately, the case of a base ring $\mathbb{Z}_{p^k}$ for some (typically small) prime p instead of a base field $\mathbb{F}_p$ does not lead to a quite as nice decomposition. Namely, the slot algebra E (see Section 3.5) a rank-d $\mathbb{Z}_{p^k}$-algebra, which generally does not decompose further into smaller rank $\mathbb{Z}_{p^k}$-algebra. In particular, we cannot easily produce tuples of correlated randomness in $\mathbb{Z}_{p^k}$. The problem would of course be solved if we could set $d = 1$, but especially for $p = 2, m > 1$ this will not be the case (since the $2 \neq 1 \pmod m$). Hence, we need to consider the case $d > 1$. In the following, we want to show how we can nevertheless produce several correlated tuples for each copy of E.

C.1 Slot-Wise Coefficient Decomposition

Recall that E can be represented as degree $d-1$ polynomials with coefficients in $\mathbb{Z}_{p^k}$. Now if we just consider constant polynomials a, b, then the product $c = ab$ of two constants is again a constant (and similarly for other operations in $\mathbb{Z}_{p^k}$). Hence, we would trivially have element-wise homomorphic operations. However, parallelism would be reduced by a factor of d compared to field case, since we do not use the $d-1$ coefficients corresponding to higher-degree monomials. Additionally, we first have to first prepare polynomials where only the constant coefficient is non-zero, since our $\mathcal{R}$-LWR-based construction always gives us fully pseudorandom plaintexts, i.e., all coefficients within the plaintext slots are pseudorandom.

To solve these problems, we first homomorphically extract the plaintext slot coefficients of the ciphertext $\mathsf{ct}_{\mathsf{out}}$ after bootstrapping. That is, we decompose $\mathsf{ct}_{\mathsf{out}}$ into d ciphertexts $(\mathsf{ct}_i)_{i\in[d]}$ where each slot contains a constant polynomial that represents the ζ_E^i-coefficient from the corresponding slot in $\mathsf{ct}_{\mathsf{out}}$. We call this *slot-wise coefficient decomposition* and it works as follows.

For each $i \in [d]$, the ciphertext ct_i is obtained by homomorphically evaluating the $\mathbb{Z}_{p^k}$-linear map

$$M_i^{\mathsf{CX}} : E \to E, \quad \sum_{j=0}^{d-1} a_j \zeta_E^j \mapsto a_i$$

on each plaintext slot of $\mathsf{ct}_{\mathsf{out}}$. As described in Section 3.5, each M_i^{CX} can be written as

$$M_i^{\mathsf{CX}}(\eta) = \sum_{f=0}^{d-1} \sigma_E^f(\theta_{0,i}^{\mathsf{CX}}) \cdot \sigma_E^f(\eta) = \sum_{f=0}^{d-1} \sigma_E^f(\theta_{0,i}^{\mathsf{CX}} \cdot \eta)$$

for some $\theta_{0,i}^{\mathsf{CX}} \in E$. σ_E^f can be applied to each slot by applying σ^f (as described in Section 3.4) to the plaintext, since σ^f acts on each slot individually as σ_E^f.

Depending on the implementation, the slots might either all use the unique slot algebra E or use different (isomorphic) algebras $\mathbb{Z}_{p^k}[X]/(f_i)$. In the latter case, the decomposition of M_i^{CX} is different for each slot, i.e., uses a different constant $\theta_{0,i}^{\mathsf{CX}}$, so the plaintext encoding $\theta_{0,i}^{\mathsf{CX}}$ needs to contain the correct $\theta_{0,i}^{\mathsf{CX}}$ in each slot. Depth-wise, the slot-wise coefficient decomposition consists only of a plaintext-ciphertext multiplication followed by a key switch and a sum. The correctness of key switching requires a certain bound on the noise, as otherwise the incurred rounding error (see Section 3.4) could overflow the noise ceiling, so the slot-wise coefficient decomposition can not be applied to the initial uniform sample $(0, a)$, but it can be applied after bootstrapping.

C.2 Overdrive2k Packing

Now that we have extracted the different coefficients of a polynomial in E, we want to use as many as possible of them to produce correlated randomness. Fortunately, there are well-established packing techniques for that. We start with the packing from [OSV20], which uses less coefficients than the tweaked interpolation packing discussed afterwards, but behaves better for higher order relations.

In Overdrive2k [OSV20], the authors construct a packing that encodes multiple $\mathbb{Z}_{p^k}$ values per slot and supports a single homomorphic multiplication. While it was originally used with power-of-two modulus (hence the suffix "2k"), it is compatible with any prime power modulus. The Overdrive2k packing encodes $\mathbb{Z}_{p^k}$ values as coefficients of a sparse polynomial in a way such that, after multiplying two such polynomials, some of the product's coefficients are simple multiplications of the factors' coefficients. For instance, $(a\zeta_E + a')(b\zeta_E + b') = ab\zeta_E^2 + (ab' + a'b)\zeta_E + a'b'$ contains the products ab and $a'b'$ as coefficients.

In general, every slot contains a polynomial of degree less than d and the Overdrive2k packing populates a subset $\mathbb{I} \subsetneq [d]$ of the coefficients. Therefore, given a pseudorandom ciphertext ct, we can partition all of its plaintext slots' coefficients into $H \coloneqq \lceil d/|\mathbb{I}| \rceil$ Overdrive2k-packed ciphertexts $(\mathsf{ct}_h)_{h\in[H]}$. Let $\mathbb{I}[j]$ denote the j-th element (starting at $j = 0$) of $\mathbb{I}$, i.e., $\mathbb{I} = \{\mathbb{I}[0], \ldots, \mathbb{I}[|\mathbb{I}| - 1]\}$. The ciphertext ct_h is obtained by homomorphically applying the map

$$M_h^{(\mathbb{I})} : E \to E, \quad \sum\nolimits_{j=0}^{d-1} a_j \zeta_E^j \mapsto \sum\nolimits_{j=0}^{|\mathbb{I}|-1} a_{h\cdot|\mathbb{I}|+j} \zeta_E^{\mathbb{I}[j]}$$

to each plaintext slot of ct. Here, for $i \geq d$, we define $a_i \coloneqq 0$, which is relevant when d is not a multiple of $|\mathbb{I}|$, in which case we waste some of the usable coefficients in the last ciphertext a_{H-1}. Figure 4 exemplarily visualizes the maps $M_h^{(\mathbb{I})}$.

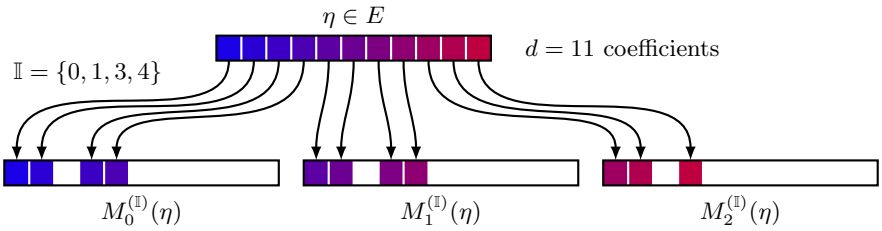

Fig. 4: Visualization of the slot-wise maps $M_h^{(\mathbb{I})}$ for partitioning a ciphertext into Overdrive2k-packed ciphertexts, using the example $d = 11$ and $\mathbb{I} = \{0, 1, 3, 4\}$

Observe that $M_h^{(\mathbb{I})}$ is a $\mathbb{Z}_{p^k}$-linear map, so we can apply the same technique as for the slot-wise coefficient decomposition to evaluate $M_h^{(\mathbb{I})}$ homomorphically on ciphertexts. This decomposition into Overdrive2k-packed ciphertexts improves the number of ciphertexts required, compared to the slot-wise coefficient decomposition, by a factor of $|\mathbb{I}|$. In practice we have $|\mathbb{I}| \approx d^{0.6}$ [OSV20] and therefore $H \approx d^{0.4}$.

In order to obtain the constants $\theta_{f,h}^{(\mathbb{I})}$ required for evaluating $M_h^{(\mathbb{I})}$, observe that $M_h^{(\mathbb{I})}$ can be assembled from $|\mathbb{I}|$ coefficient extractions via

$$\begin{aligned} M_h^{(\mathbb{I})}(\eta) &= \sum\nolimits_{j=0}^{|\mathbb{I}|-1} \zeta_E^{\mathbb{I}[j]} M_{h\cdot|\mathbb{I}|+j}^{\mathsf{CX}}(\eta) = \sum\nolimits_{j=0}^{|\mathbb{I}|-1} \zeta_E^{\mathbb{I}[j]} \sum\nolimits_{f=0}^{d-1} \sigma_E^f(\theta_{0,h\cdot|\mathbb{I}|+j}^{\mathsf{CX}}) \sigma_E^f(\eta) \\ &= \sum\nolimits_{f=0}^{d-1} \left(\sum\nolimits_{j=0}^{|\mathbb{I}|-1} \zeta_E^{\mathbb{I}[j]} \sigma_E^f(\theta_{0,h\cdot|\mathbb{I}|+j}^{\mathsf{CX}}) \right) \sigma_E^f(\eta) = \sum\nolimits_{f=0}^{d-1} \theta_{f,h}^{(\mathbb{I})} \sigma_E^f(\eta) \qquad (6) \\ &\text{with} \quad \theta_{f,h}^{(\mathbb{I})} \coloneqq \sum\nolimits_{j=0}^{|\mathbb{I}|-1} \zeta_E^{\mathbb{I}[j]} \cdot \sigma_E^f(\theta_{0,h\cdot|\mathbb{I}|+j}^{\mathsf{CX}}) \end{aligned}$$

and, in case $i \geq d$, $M_i^{\mathsf{CX}}(\eta) \coloneqq 0$. Again, the constants $\theta_{0,i}^{\mathsf{CX}}$ might differ per slot if the implementation uses different $\mathbb{Z}_{p^k}$-algebras per slot.

Packing Refresh Map. Originally, the Overdrive2k packing is restricted to multiplicative depth one, which suffices to generate many forms of correlated randomness, such as square pairs [DKL+13], Beaver triples [Bea92], matrix triples [MG23, RRKK23], and convolution triples [RRHK23]. However, some forms of correlated randomness (such as, e.g., polynomial encodings [RRK+24]) require larger multiplicative depth. We say that a freshly packed plaintext (i.e., values are packed at coefficients indexed by $\mathbb{I}$) has *packing level one* and a plaintext that has already undergone a multiplication (i.e., values are packed at coefficients indexed by $2\mathbb{I}$) has *packing level zero*. To support larger multiplicative depth, we provide a *packing refresh* procedure that takes an Overdrive2k-packed ciphertext at packing level zero and repacks each slot to packing level one, to allow for further multiplication(s).

The packing refresh procedure simply moves the coefficients indexed by $\mathbb{J} := 2\mathbb{I}$ back to indices $\mathbb{I}$. This is again a slot-wise $\mathbb{Z}_{p^k}$-linear map $M^{(\mathbb{J}\to\mathbb{I})} : E \to E$ that can be assembled similarly to Equation (6), resulting in

$$M^{(\mathbb{J}\to\mathbb{I})}(\eta) = \sum_{f=0}^{d-1} \theta_f^{(\mathbb{J}\to\mathbb{I})} \sigma_E^f(\eta) \quad \text{with} \quad \theta_f^{(\mathbb{J}\to\mathbb{I})} := \sum_{j=0}^{|\mathbb{I}|-1} \zeta_E^{\mathbb{I}[j]} \sigma_E^f(\theta^{\mathsf{CX}}_{0,2\mathbb{I}[j]}).$$

Figure 5 visualizes a multiplication followed by a packing refresh via the slot-wise map $M^{(2\mathbb{I}\to\mathbb{I})}$.

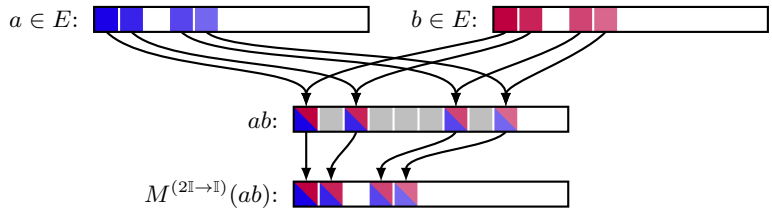

Fig. 5: Visualization of Overdrive2k slot-wise multiplication followed by a packing refresh via the slot-wise map $M^{(2\mathbb{I}\to\mathbb{I})}$, using the example $d = 11$ and $\mathbb{I} = \{0, 1, 3, 4\}$

Inverse Map. Given H Overdrive2k-packed ciphertexts $(\mathsf{ct}_h)_{h\in[H]}$ under packing level zero, we can *unpack* them, i.e., move the packed values back into the plaintext slot coefficients of a single ciphertext ct by homomorphically evaluating the slot-wise inverse map $M^{\mathsf{Inv}(\mathbb{J})} : E^H \to E$ defined by

$$M^{\mathsf{Inv}(\mathbb{J})}(\eta_0, \ldots, \eta_{H-1}) = \sum_{h=0}^{H} \zeta_E^{h\cdot|\mathbb{J}|} M^{(\mathbb{J}\to[|\mathbb{J}|])}(\eta_h) \tag{7}$$

that extracts the coefficients indexed by $\mathbb{J} := 2\mathbb{I}$ from all H arguments and outputs a dense coefficient packing.

In practice, one typically wants to unpack ciphertexts at packing level zero, originating from a homomorphic multiplication. However, in case one wants to unpack ciphertexts at packing level *one* instead, this can also be done by applying $M^{\mathsf{Inv}(\mathbb{I})}$ instead of $M^{\mathsf{Inv}(2\mathbb{I})}$.

C.3 Tweaked Interpolation Packing

The Overdrive2k packing covered in Appendix C.2 is only capable of using a fraction of about $d^{0.6}/d$ of the available coefficients. In [CKL21], the authors construct a more efficient packing method that uses $1/2$ of the available coefficients. The idea of this so-called *tweaked interpolation packing* is to store values not in the coefficients but instead in the evaluation points of a polynomial. To allow interpolation the packing first has to scale up the evaluation points by a factor of p^δ for some relatively small δ. Specifically, a vector $\boldsymbol{a} = (a_j)_{j\in[M]} \in \mathbb{Z}_{p^t}^M$ with $t \le k - 2\delta$ and $M < d/2 + 1$ is packed into a slot by interpolating a polynomial $\mathsf{pack}_E(\boldsymbol{a}) \in E$ of degree less than $d/2$ such that

$$\mathsf{pack}_E(\boldsymbol{a})(j) \equiv p^\delta \cdot a_j \pmod{p^k} \qquad \forall j \in [M]. \tag{8}$$

The product of two such polynomials $\mathsf{pack}_E(\boldsymbol{a})$ and $\mathsf{pack}_E(\boldsymbol{b})$ has degree less than d, so it doesn't overflow modulo f_0. Therefore the evaluation

$$(\mathsf{pack}_E(\boldsymbol{a}) \cdot \mathsf{pack}_E(\boldsymbol{b}))(j) \equiv p^{2\delta} \cdot a_j b_j \pmod{p^k} \qquad \forall j \in [M] \tag{9}$$

yields the component-wise product $\boldsymbol{a} \odot \boldsymbol{b}$ scaled by a factor of $p^{2\delta}$. We say that a freshly packed plaintext (as in Equation (8)) has packing level one and a plaintext that has already

undergone a multiplication (as in Equation (9)) has packing level zero. Note that, from a plaintext at packing level zero, we can only reconstruct the packed values modulo $p^{k-2\delta}$, which is why we require $t \leq k - 2\delta$.

$\mathsf{pack}_E(\boldsymbol{a})$ is computed by multiplying each a_j with a scaled Lagrange polynomial $\tilde{\lambda}_j$, i.e., $\mathsf{pack}_E(\boldsymbol{a}) = \sum_{j=0}^{M-1} a_j \tilde{\lambda}_j$. We refer to [CKL21, Lemma 1] for details. Importantly, this representation immediately implies that pack_E is a $\mathbb{Z}_{p^k}$-linear map. This can be exploited in order to move all of the plaintext slot coefficients of a given ciphertext ct into two tweaked-interpolation-packed ciphertexts $\mathsf{ct}_0, \mathsf{ct}_1$. More specifically, for $i \in \{0, 1\}$, the ciphertext ct_i is obtained by applying the slot-wise $\mathbb{Z}_{p^k}$-linear map

$$M_i^{\mathsf{TIP}} : E \to E, \quad \sum_{j=0}^{d-1} a_j \zeta_E^j \mapsto \mathsf{pack}_E((a_j)_{j \in \mathbb{I}_i})$$

with $\mathbb{I}_0 = \{0, \dots, \lceil d/2 \rceil - 1\}$ and $\mathbb{I}_1 = \{\lceil d/2 \rceil, \dots, d-1\}$. Note that, due to the nature of the tweaked interpolation packing, these maps only preserve the plaintext modulo $p^{k-\delta}$ and, after homomorphic multiplication, this reduces to $p^{k-2\delta}$.

In order to obtain the constants $\theta_f^{(\mathsf{TIP},i)}$ required for evaluating M_i^{TIP}, observe that M_i^{TIP} can be assembled from $|\mathbb{I}_i|$ coefficient extractions similarly to Equation (6), resulting in

$$M_i^{\mathsf{TIP}}(\eta) = \sum_{f=0}^{d-1} \theta_{f,i}^{\mathsf{TIP}} \sigma_E^f(\eta) \quad \text{with} \quad \theta_{f,i}^{\mathsf{TIP}} \coloneqq \sum_{j=0}^{|\mathbb{I}_i|-1} \tilde{\lambda}_j \sigma_E^f(\theta_{0,\mathbb{I}_i[j]}^{\mathsf{CX}}).$$

Inverse Map. Given two tweaked-interpolation-packed ciphertexts $\mathsf{ct}_0, \mathsf{ct}_1$ under packing level zero, we can move the packed values back into the plaintext slot coefficients of a single ciphertext ct by homomorphically evaluating the slot-wise inverse map $M^{\mathsf{InvTIP}} : E^2 \to E$ that evaluates both arguments at $j = 0, \dots, \lceil d/2 \rceil$ and outputs a coefficient packing of these evaluation points. Assuming the second argument is zero, M^{InvTIP} can be assembled from coefficient extractions similarly to Equation (6), resulting in

$$\begin{aligned} M^{\mathsf{InvTIP}}(\eta, 0) &= \sum_{f=0}^{d-1} \theta_f^{\mathsf{InvTIP}} \sigma_E^f(\eta_0) \\ \text{with} \quad \theta_f^{\mathsf{InvTIP}} &\coloneqq \sum_{j=0}^{\lceil d/2 \rceil - 1} \zeta_E^j \sum_{i=0}^{d-1} j^i \sigma_E^f(\theta_{0,i}^{\mathsf{CX}}). \end{aligned}$$

In the general case, where we need to extract evaluation points from both arguments, M^{InvTIP} can be computed via

$$M^{\mathsf{InvTIP}}(\eta_0, \eta_1) = M^{\mathsf{InvTIP}}(\eta_0, 0) + \zeta_E^{\lceil d/2 \rceil} M^{\mathsf{InvTIP}}(\eta_1, 0).$$

If d is odd, we require that $\eta_1(\lceil d/2 \rceil) = 0$ as we don't have sufficient coefficients to store that last evaluation point of η_1.

Note that the plaintext after applying M^{InvTIP} is still scaled by a factor of $p^{2\delta}$. This factor cannot be eliminated by a $\mathbb{Z}_{p^k}$-linear map, but it can be eliminated via *exact division* (see Section 3.4) by $p^{2\delta}$. This also reduces the plaintext modulus by a factor of $p^{2\delta}$.

Packing Refresh. It is possible to refresh the tweaked interpolation packing (i.e., go from packing level zero to level one) by applying M^{InvTIP}, followed by approximate division, followed by $(M_i^{\mathsf{TIP}})_{i \in \{0,1\}}$. As such, the tweaked interpolation packing can support a multiplicative depth greater than one. However, the plaintext modulus reduces by a factor of $p^{2\delta}$ at each level. Hence, the Overdrive2k packing is likely a better choice for larger multiplicative depths, as required, e.g., for polynomial encodings [RRK+24].