Post-Quantum Public-Key Pseudorandom Correlation Functions for OT

Shweta Agrawal[1], Kaartik Bhushan[2], Geoffroy Couteau[2], and Mahshid Riahinia*[3]

[1]IIT Madras, India
[2]Université Paris Cité, CNRS, IRIF, F-75013, Paris, France
[3]DIENS, École normale supérieure, CNRS, Paris, France

Abstract. Public-key pseudorandom correlation functions (PK-PCF) are an exciting recent primitive introduced to enable "non-interactive key exchange for secure computation". Despite significant advances in the group-based setting, success in the post-quantum regime has been much more limited. To the best of our knowledge, there does not exist even a single efficient candidate post-quantum PK-PCF for the standard string oblivious transfer (OT) correlation.

In this work, we address this gap by constructing the first efficient lattice-based public-key PCF for the string OT correlation. Our PK-PCF generates a few hundred OTs per second, and requires a large but manageable public key size (a few hundred megabytes). In contrast, the only previous lattice-based *non-public-key* PCF, proposed in the very recent work of Hasler, Reisert and Küsters (Asiacrypt 2025), can generate up to 9 OT/s and has key sizes of several gigabytes.

At the heart of our result lie several technical contributions that might be of independent interest. In particular, we introduce the first efficient lattice-based constrained pseudorandom functions for low-degree polynomials, from a new but natural "secret-power" variant of ring learning with errors. Our assumption is non-interactive and falsifiable, and we carefully analyze it for attacks. Additionally, we introduce a new packing mechanism compatible with local rounding of noisy shares from a "truncated" variant of our previous assumption, which allows further efficiency.

We remark that in the pre-quantum regime, the state of art for PK-PCF only two years ago was 1 OT/s, while they now clock at 30k OT/s. We are optimistic that our construction will follow a similar trajectory.

*This work was done while Mahshid Riahinia was visiting IRIF, Paris.

Contents

1 Introduction **3**
- 1.1 Our Results . . . 4
- 1.2 Comparison with Prior Work . . . 4

2 Technical Overview **6**
- 2.1 PCFs for OT from Pseudorandomly-Constrained PRFs . . . 6
- 2.2 Packing from Truncated Secret-Power Ring-LWE . . . 11
- 2.3 Efficiency Analysis . . . 14
- 2.4 Directions for Future Work . . . 15

3 Preliminaries **15**
- 3.1 Pseudorandom Functions . . . 15
- 3.2 Constrained Pseudorandom Functions . . . 16
- 3.3 Hardness Assumptions . . . 18
- 3.4 (Public-Key) Pseudorandom Correlation Functions . . . 18

4 Our New Assumption: Secret-Power Ring-LWE **19**

5 Pseudorandom Correlation Function for OT from Secret-Power Ring-LWE **22**
- 5.1 Efficient Pseudorandomly Constrained PRFs from Secret-Power Ring-LWE . . . 22
- 5.2 PCFs for OT Correlations from CPRFs . . . 25

6 Goldreich-Applebaum-Raykov Weak PRF **28**

7 Public Key PCF from Secret-Power and Public-Power Ring-LWE **30**

8 Faster PK-PCF from Truncated SP-RLWE **37**
- 8.1 Truncated SP-RLWE . . . 37
- 8.2 Sparse Coefficient Packing . . . 37
- 8.3 Packed CPRF from TSP-RLWE . . . 38
- 8.4 Packed PK-PCF . . . 42

9 Performance Evaluation **42**
- 9.1 Selecting the candidate . . . 42
- 9.2 Modulus size . . . 43
- 9.3 Selecting the packing subset . . . 43
- 9.4 Counting operations . . . 44
- 9.5 Key Sizes of Our PK-PCF . . . 47

10 Malicious Security of PK-PCF **49**

1 Introduction

Oblivious transfer (OT) is one of the most fundamental primitives in cryptography. An OT protocol involves two parties, a *sender* and a *receiver*, where the sender has a pair of messages (m_0, m_1) and the receiver has a choice bit b. The protocol allows the receiver to learn m_{b} with the guarantees that (i) the receiver learns nothing about $m_{1-\mathsf{b}}$ and (ii) the sender learns nothing about b.

OT lies at the heart of secure computation, and efficient OT generation is often at the heart of efficient multi-party computation (MPC), since protocols involving OT typically require them to be generated in the millions or even billions. One clever idea to mitigate the cost of generating OTs, popularly used in protocol design, is to perform OT generation offline in a preprocessing phase which is independent of the inputs. Later, when the inputs become available in the online phase, the OTs are used in lightweight and input-independent protocols.

Starting with the pioneering work of Beaver [12] and the efficient follow-up by Ishai, Kilian, Nissim, and Petrank [55], a long line of work has resulted in very efficient protocols for computing OTs in the preprocessing model. These works have relied on the so-called *OT extension* paradigm, where a small number of seed OTs are generated using any black-box, possibly expensive OT protocol, which are subsequently extended by the parties to generate a large number of OTs. Importantly, the extension step only uses lightweight symmetric key cryptography, amortizing the cost of the expensive base step over millions of OTs. However, the runtime as well as concrete communication overhead of these protocols in the offline phase was frequently high [57, 58].

Pseudorandom Correlation Generators and Functions. A drastic reduction in the communication costs was achieved by Boyle et al. [22, 25], who introduced pseudorandom correlation generators (PCGs) and functions (PCFs). These primitives enable the compression of large quantities of correlated randomness (for instance millions of Beaver triples), into small secret seeds or keys, which can be generated with low communication costs in the initial interactive phase. Analogously to their standard pseudorandom generator (PRG) counterparts, PCGs allow the parties to generate only a fixed polynomial amount of correlated randomness—this limitation is overcome by PCFs analogously to pseudorandom functions (PRFs).

A PCF for the OT correlation consists of two algorithms $\mathsf{PCF.KeyGen}$ and $\mathsf{PCF.Eval}$ that work as follows. First, the key generation algorithm $\mathsf{PCF.KeyGen}$ generates two keys k_0 and k_1, held by the sender and receiver, respectively. The two parties can subsequently use their PCF keys locally and without any interaction to generate random OT-correlated pairs[1], on the fly, by running the evaluation algorithm $\mathsf{PCF.Eval}(k_\sigma, x)$ over some common input x. Security requires that (i) the sender learns (computationally) no information about the bit b, and (ii) the receiver learns (computationally) no information about $r_{1-\mathsf{b}}$.

Public Key PCF or "NIKE for Secure Computation". In this work, as in [33, 37, 74], we are interested in the non-interactive setting. Consider a large network with N parties, where any pair of parties might potentially participate in some secure computation protocol. In general, each pair must run the pre-processing phase independently to exchange OT correlations, resulting in $O(N^2)$ communication and simultaneous interactions. It is well-known that for secure *communication*, there exist non-interactive key-exchange (NIKE) protocols that allow N parties to exchange N^2 symmetric keys using only $O(N)$ public keys. Can we do the same for secure *computation*?

In more detail, we desire that in the network of N parties, each pair of parties can efficiently generate an arbitrary number of pseudorandom OT instances, given *only the public key* of the other party. The question of achieving oblivious transfer with a public-key setup was first explicitly put forth by Orlandi, Scholl and Yakoubov (OSY) [74]. OSY provided the first public-key PCFs under the quadratic residuosity (QR) assumption and the existence of correlation-robust hash functions. However, the efficiency of this construction at 1 OT-correlated pair per second, is still quite slow compared to state-of-the-art PCFs for OTs. Moreover, the reliance on QR makes it post-quantum insecure.

State of the Art. The recent work of Bui, Couteau, Meyer, Passelègue, and Riahinia (BCMPR) [33] achieved the first practically efficient candidate construction of public-key PCF for OT from the decisional

[1] Recall that random OT-correlated pairs are of the form $((r_0, r_1), (\mathsf{b}, r_{\mathsf{b}}))$, where r_0 and r_1 are random strings and b is a random bit – these can be converted to chosen-bit OT-correlations at the cost of an additional 3 bits of communication.

composite residuosity (DCR) assumption, the sparse power-decisional Diffie-Hellman assumption (DDH), correlation robust hash functions, and a suitable low-complexity weak pseudorandom function (wPRF). The performance of BCMPR is orders of magnitude better at 30K OTs per second but the reliance on quantum-unsafe assumptions remains a problem.

In the post-quantum regime, a significant advancement was obtained by QuietOT [37]—a public key protocol where, given only the other party's public key, two parties can generate an arbitrary amount of pseudorandom "ListOTs", a variant of OT which can be converted into regular OT in one round of communication. QuietOT is based on ring-LWE, and is thus plausibly post-quantum secure. The performance of QuietOT is also excellent; it achieves millions of OTs per second.

However, QuietOT comes with a significant drawback: it requires additional communication to convert each ListOT into a regular OT. While the added overhead is not much compared to a PK-PCF for OT (7 bits/OT versus 3 bits/OT), this additional communication makes QuietOT inherently insecure against a malicious adversary: any malicious adversary can completely destroy security and recover all secrets after a few hundred OTs by sending inconsistent messages and mounting selective failure attacks, even if the public key setup is maliciously secure. Natural approaches to prevent such attacks incur the usage of expensive zero-knowledge, which creates serious hurdles for efficiency.

In contrast, it is not hard to see that any PK-PCF that directly generates standard OT correlations becomes automatically maliciously secure as soon as the public key setup is maliciously secure. We show this formally in Section 10. Asymptotically, enhancing the setup to malicious security adds a constant overhead, independent of the number of OTs, even when using generic techniques. At a high level, this stems from the fact that when communication is optimal (3 bits/OT), there is no room to cheat: every sequence of messages corresponds to a valid honest input. Therefore, an important open question is:

Can we construct efficient public-key PCFs for OT from lattice-based assumptions?

1.1 Our Results

In this work, we answer the above question in the affirmative and put forth the first efficient candidate post-quantum public-key PCF for the OT correlation. At the heart of our result lie several technical contributions that might be of independent interest. In particular, we introduce the first efficient lattice-based constrained pseudorandom functions (CPRFs) for low-degree polynomials from a new but natural "secret-power" variant of ring-LWE (RLWE). Our assumption is non-interactive and falsifiable and builds in a certain natural correlations in the RLWE secrets across samples. A related but incomporable assumption which incorporates similar dependencies into the public RLWE labels has been recently used in the work of [1]. We analyze our assumption carefully for attacks. We additionally design a new packing mechanism compatible with local rounding of noisy shares from a "truncated" variant of our previous assumption. While more aggressive, this assumption leads to our most efficient construction. While we do not see any new vulnerabilities introduced by our stronger assumption, we caution that more cryptanalytics effort by the community will be needed to build confidence.

1.2 Comparison with Prior Work

We provide a comparison of our results with the state-of-the-art PK-PCFs, summarized in Table 1. As shown in the table, our PK-PCF generates a few hundred OTs per second, which we view as a promising performance. It requires a large but manageable public key size (a few hundred megabytes). For comparison, the only previous lattice-based *non-public-key* PCF, proposed in the very recent work of [54], can generate up to 9 OT/s (on lower-end hardware compared to the one of the benchmark we used) and has key sizes of several Gigabytes.

Key sizes. We briefly comment on our key sizes, that are significantly bigger as compared to pre-quantum variants. However, this is to be expected since the notion of a PK-PCF is strictly harder to achieve than that of NIKE, which is notoriously much heavier in the post-quantum regime. Our current performance leaves room for improvement, however it is already much better as compared to the pre-quantum state-of-the-art of OSY. We are optimistic that our techniques can pave the way to much faster post-quantum solutions in the future; we discuss some approaches towards this in Section 2.4.

Table 1: Comparison to existing public-key PCFs.

Work	Post-Quantum Secure	OT Variant	Communication[a] (bits/OT)	Maliciously[b] Secure	Performance (OT/sec)	Key Size
[74]	No	string OT	3	Yes	1	small
[33]	No	string OT	3	Yes	30k	small
[37]	Yes	ListOT	7	No	1.2M	5.5MB
This (Fast)	Yes	string OT	3	Yes	540∼1k[c]	576MB
This (Small)	Yes	string OT	3	Yes	190∼450	200MB

[a] Our PCF generates *random* OTs. To transform these to chosen-bit OT, the two parties need to communicate to de-randomize their shares. Transforming a random OT to a chosen-bit OT requires 3 bits of communication [10,11]. This is larger for ListOT; in particular, for the ListOT realized by QuietOT [37], the parties need to communicate 7 bits per ListOT correlation to derive chosen-bit OT correlations.

[b] This column indicates whether the PK-PCF produces maliciously secure OT as soon as the setup is maliciously secure.

[c] About 540 OT/s for the receiver and 1000 OT/s for the sender using our fast variant with aggressive parameters. The conservative parameters yield a slightly slower rate of 460 OT/s for the receiver and 860 OT/s for the sender. All numbers are estimated from the low-level benchmark of ring operations from the Intel HEXL reference paper [15] (benchmarked on a 3rd Gen Intel Xeon® Scalable Processors Platinum 8360Y 2.4GHz processor, single-threaded, single core).

Our Approach. Our most efficient instantiation requires two assumptions (either in the random oracle model (ROM), or additionally assuming a correlation-robust hash function): pseudorandomness of the Goldreich-Applebaum-Raykov wPRF candidate (as in previous works [33,37]), and security of a variant of "secret-power" ring-LWE, which assumes that ring-LWE samples of the form $(M \cdot \mu^i + e_i)_i$, for a public M, secret μ, and small e_i's, look random, when the public element μ comes from a *truncated* distribution (with r leading and trailing coefficients, for some parameter r). We analyze our assumptions in detail, and believe they will have further applications—we hope that our work will motivate their study.

1.2.1 Perspective.

When designing a new cryptographic primitive for real-world applications, cryptographers can pursue two approaches: the *top-down* approach seeks the most efficient construction under a standard, well-established assumption. The *bottom-up* approach seeks instead to maximize concrete efficiency using the minimal design that appears to resist attacks. We believe that for real world systems that demand high concrete efficiency, the bottom-up approach has been significantly more successful—as a case in point, though we have post-quantum non-interactive zero-knowledge proofs (NIZKs) from standard assumptions (LWE) [79], all real-world uses of post-quantum NIZKs that we are aware of rely on the Fiat-Shamir heuristic. For example, consider

- ML-DSA / Crystal Dilithium [46], standardized by NIST in August 2024 (FIPS 204) [70], currently being integrated into the TLS 1.3 stack by providers like Cloudflare and Google,
- FRI [13], a zkSNARK used by many companies like Starknet/StarkWare, RISC Zero, Polygon Miden, ZKsync, and probably dozens of others, and
- VOLE-in-the-Head [9], used in zkPass / zkTLS and by other companies such as Chainlink.

For PCG/PCF constructions, it is widely believed that truly efficient constructions require new assumptions, as evidenced by the fact that an overwhelming fraction of new PCG/PCF constructions have introduced a new assumption, or a new parametrization of an existing assumption [17,18,22–27,33,42,60,61,80]. Our work follows this approach. We observe that these assumptions have attracted a significant cryptanalytic attention [21,31,40,52,62,63,71,80] which caused many of them to be revised,

or their parameters updated. We believe this is a healthy research dynamic that, not only yields efficient constructions for practice but also leads to progress in the understanding of assumptions.

2 Technical Overview

We provide a high level overview of our techniques.

Folklore Approach via Spooky Encryption. A folklore approach to building a post-quantum PCF, outlined in [26], is to rely on function secret sharing (FSS) for a class of functions that contains products of standard pseudorandom functions (PRFs). At a high level, an FSS scheme for a class of functions $\mathcal{F}$ enables sharing a function $f \in \mathcal{F}$ into two succinct shares f_0, f_1 such that (1) given any input x, the parties with respective shares f_0, f_1 can locally compute shares y_0, y_1 of $f(x)$, and (2) each individual share leaks no information about f (beyond its membership to $\mathcal{F}$).

Given an FSS scheme for the function $f : x \mapsto \mathsf{PRF}_{k_0}(x) \cdot \mathsf{PRF}_{k_1}(x)$, where PRF is a PRF and k_0, k_1 are PRF keys known to party P_0 and party P_1 respectively, the two parties can compute a virtually unbounded number of shares of products of pseudorandom bits, which is equivalent to random OT correlations. To see why, let $a = \mathsf{PRF}_{k_0}(x)$ and $b = \mathsf{PRF}_{k_1}(x)$ be bits. Therefore, $a, (ab)_0$ are held by P_0 and $b, (ab)_1$ are held by P_1. From these, the two parties can respectively derive the pairs $((ab)_0, (ab)_0 \oplus a)$ and $(b, (ab)_1)$ which are OT-correlated. To obtain a *public key* PCF via this approach, the underlying FSS should be replaced with *multi-key* FSS [39], which is known to be implied by spooky encryption [45], and can be based on LWE.

A natural approach is to start from this generic construction and optimize it to make it concretely efficient. A step in this direction was recently made in [54], who introduced the first concretely efficient lattice-based PCF for oblivious linear evaluation (OLE) correlations over large fields under the ring-LWE assumption. Their approach follows the template above, introducing a significant amount of optimizations and a clever choice of the underlying PRF, partially inspired by the work of [74]. However, the resulting PCF has some shortcomings: (i) it produces around 9 OLEs/sec [2], (ii) its key size appears to be several gigabytes at the very least by a back of the envelope calculation [53], (iii) it uses FHE schemes that are not multi-key, and is therefore not a PK-PCF. Turning it into a PK-PCF would require replacing at least some of its FHE components with multi-key FHE, which is likely to cause a significant slowdown.

In this work, we explore an alternative pathway suggested by [33] which trades off the standard-ness of the assumption for efficiency.

2.1 PCFs for OT from Pseudorandomly-Constrained PRFs

The work of [33] proposes a generic transformation for building PCFs for the OT correlation from *pseudorandomly-constrained PRFs*. A constrained PRF (CPRF) [20, 28, 59] is a pseudorandom function that additionally supports the generation of a *constrained key*. Given a constrained key ck_C for a predicate C, one can locally evaluate the PRF on all inputs x where $C(x) = 0$, while the output of the PRF looks random on all other inputs. We recall this compiler below.

CPRF-to-PCF Compiler of [33]. Let F be a weak pseudorandom function (wPRF) with range $\{0,1\}$. Let CPRF be a CPRF that supports $\{F_k(\cdot) \colon k\} \cup \{\overline{F}_k(\cdot) \colon k\}$ as constraints, where $\overline{F}_k(\cdot) := 1 - F_k(\cdot)$ is the complement of F. Such a CPRF is called a pseudorandomly-constrained PRF [33]. The compiler of [33] works as follows:

- Sender's key: consists of two random master secret keys $\mathsf{msk}_0, \mathsf{msk}_1$ of CPRF.
- Receiver's key: consists of a random key k of the wPRF F, and two constrained keys $\mathsf{ck}_0, \mathsf{ck}_1$ that are, respectively, msk_0 constrained at $\{x : \ F_k(x) = 1\}$, and msk_1 constrained at $\{x : \ F_k(x) = 0\}$. Note that ck_b can evaluate the output of msk_b on any input x such that $F_k(x) = \mathsf{b}$, for $\mathsf{b} \in \{0,1\}$.
- Sender's evaluation: on input x, the sender outputs (r_0, r_1), where r_b is the evaluation of CPRF on x using the master secret key msk_b, for $\mathsf{b} \in \{0,1\}$.

[2] For the informed reader, we remark that producing OTs instead of OLEs does not result in any significant speedup due to the constraints in modulus size imposed by ring-LWE.

- Receiver's evaluation: on input x, the receiver first computes $\mathsf{b} \leftarrow F_k(x)$, and then outputs (b, r), where r is the output of ck_b on x.

Correctness and security follow from inspecting the structure of the two PCF keys. For any input x, it holds that $F_k(x) = \mathsf{b}$ for some $\mathsf{b} \in \{0,1\}$. Therefore, the output of ck_b on x is equal to the output of msk_b on this input, that is r_b. The two parties therefore correctly output OT-correlated pairs. The security against the sender is implied by the pseudorandomness of F, as the key k is hidden from the the sender. The security against the receiver, on the other hand, requires invoking the security of the underlying CPRF; for any input x such that $F_k(x) = \mathsf{b}$, the constrained key $\mathsf{ck}_{1-\mathsf{b}}$ fails in computing the output of $\mathsf{msk}_{1-\mathsf{b}}$ on x, that is $r_{1-\mathsf{b}}$. Due to the security of the CPRF, this output looks random to the receiver.

To instantiate this transformation in the post-quantum setting, we need to select a candidate post-quantum CPRF, and a post-quantum wPRF in the class of constraints supported by the CPRF. We describe this next.

An Efficient Pseudorandomly-Constrained PRF from Lattices. A first idea is to plug in lattice-based CPRFs that support general circuits as constraints [29, 30, 34, 78]. These constructions, while more than powerful enough to be used in the CPRF-to-PCF transformation, rely heavily on advanced lattice machinery developed in the context of attribute-based encryption (ABE) [19] for which no practical constructions are known. Therefore, similarly to spooky encryption, we do not expect to obtain a practically efficient PCF via this route. Below, we develop new ideas to leverage the algebraic structure of the constraint we wish to support to provide a concretely efficient candidate. We are optimistic that our techniques may find application in other contexts.

Fix a wPRF $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, where $F_\mathbf{z}(\mathbf{x})$ denotes the evaluation of F on input $\mathbf{x}$ using key $\mathbf{z}$. Our idea for constructing a CPRF that supports F as a constraint is to instead handle a more general circuit over a ring R_q; one that correctly encodes $F_\mathbf{z}(\mathbf{x}) = 0$ while outputting arbitrary values when $F_\mathbf{z}(\mathbf{x}) = 1$.

More precisely, consider a function $P : \{0,1\}^n \times R_q^n \to R_q$ that for any $\mathbf{x}, \mathbf{z} \in \{0,1\}^n$ represents $F_\mathbf{z}(\mathbf{x})$ over R_q. Concretely, for all $\mathbf{x}, \mathbf{z} \in \{0,1\}^n$, letting $P_\mathbf{x}(\cdot) := P(\mathbf{x}, \cdot)$, it should hold that $P_\mathbf{x}(\mathbf{z}) = 0$ (in R_q) $\iff F_\mathbf{z}(\mathbf{x}) = 0$. Finally let t denote a bound such that for any input $\mathbf{x}$, $P_\mathbf{x}(\cdot)$ is a polynomial of degree at most t. We define a CPRF that supports P as a constraint as follows:

- $\mathsf{CPRF.KeyGen}(1^\lambda)$: Sample $\mathbf{k} = (k_1, \ldots, k_n) \xleftarrow{\$} R_q^n$. Sample an invertible element $\mu \xleftarrow{\$} R_q$, and $M \xleftarrow{\$} R_q$. Output $\mathsf{msk} := (\mathbf{k}, M, \mu)$.
- $\mathsf{CPRF.Eval}(\mathsf{msk}, \mathbf{x})$: Output $y := \lfloor M\mu \cdot P_\mathbf{x}(\mathbf{k}/\mu) \pmod q \rceil_2$.

Let us now focus on generating a constrained key for a vector $\mathbf{z} \in \{0,1\}^n$ that can evaluate the CPRF on all inputs $\mathbf{x} \in \{0,1\}^n$ *iff* $F_\mathbf{z}(\mathbf{x}) = 0$ or equivalently $P_\mathbf{x}(\mathbf{z}) = 0$. Let $\boldsymbol{\ell} := \mathbf{k} - \mu \cdot \mathbf{z}$. We can re-write $P_\mathbf{x}(\mathbf{k}/\mu)$ in terms of $\boldsymbol{\ell}$, and expand the polynomial as follows:

$$P_\mathbf{x}(\mathbf{k}/\mu) = P_\mathbf{x}(\boldsymbol{\ell}/\mu + \mathbf{z}) = \sum_{i=0}^{t} p_i(\boldsymbol{\ell}/\mu + \mathbf{z})^i = P_\mathbf{x}(\mathbf{z}) + \frac{1}{\mu} \cdot Q(\mathbf{x}, \mathbf{z}, 1/\mu, \boldsymbol{\ell}) \ ,$$

The last equality is obtained by simply isolating all the terms $p_i \mathbf{z}^i$ for $i = 0, \ldots, t$. Since the remaining terms all include non-zero powers of $\boldsymbol{\ell}/\mu$, we can write their summation as $(1/\mu) \cdot Q(\mathbf{x}, \mathbf{z}, 1/\mu, \boldsymbol{\ell})$, where $Q(\mathbf{x}, \mathbf{z}, 1/\mu, \boldsymbol{\ell}) = \sum_{i=0}^{t-1} q_i (1/\mu)^i$. It therefore holds that

$$\mathsf{CPRF.Eval}(\mathsf{msk}, \mathbf{x}) = \lfloor M\mu \cdot P_\mathbf{x}(\mathbf{k}/\mu) \rceil_2 = \lfloor M\mu \cdot P_\mathbf{x}(\mathbf{z}) + M \cdot \sum_{i=0}^{t-1} q_i (1/\mu)^i \rceil_2 \ . \tag{1}$$

Recall that we want $\mathsf{ck}_\mathbf{z}$ to evaluate $\mathsf{Eval}(\mathsf{msk}, \mathbf{x})$ on all inputs $\mathbf{x}$ satisfying $P_\mathbf{x}(\mathbf{z}) = 0$. We therefore set $\mathsf{ck}_\mathbf{z} := (\mathbf{z}, \boldsymbol{\ell}, (\mathcal{M}_i)_{i \in [t]})$, where $\boldsymbol{\ell} = \mathbf{k} - \mu \cdot \mathbf{z}$ and each $\mathcal{M}_i = M \cdot (1/\mu)^{i-1} + e_i$, for some small-norm

error $e_i \xleftarrow{\$} R_q$. Consequently, the constrained evaluation on input $\mathbf{x} \in \{0,1\}^n$ proceeds by first solving the following symbolic equation (in the symbolic variable Y)

$$P(\mathbf{x}, \boldsymbol{\ell} \cdot Y + \mathbf{z}) = P_{\mathbf{x}}(\mathbf{z}) + \sum_{i=1}^{t} c_i \cdot Y^i \pmod q \ .$$

to learn the coefficients $\{c_i\}_{i\in[t]}$. It then outputs $y' := \lfloor \sum_{i=1}^{t} c_i \cdot \mathcal{M}_i \pmod q \rceil_2$.

Correctness and Security. These properties follow by inspection of Equation 1. Regarding correctness, when $P_{\mathbf{x}}(\mathbf{z}) = 0$, the only difference between the evaluation and the constrained evaluation on $\mathbf{x}$, is the error term $\lfloor \sum_{i=1}^{t} c_i \cdot e_i \rceil_2$ that occurs in the constrained evaluation. Recall from above that each coefficient can be computed from $\mathbf{x}, \mathbf{z} \in \{0,1\}^n$, and $\boldsymbol{\ell} \in R_q^n$. To make this error disappear when rounding modulo 2, we need to sample each k_i and μ as small-norm elements in order to keep the norm of $\boldsymbol{\ell}$ small. Finally, setting q to be super-polynomially larger than the magnitude of this term, the error disappears when rounding modulo 2. [3]

To prove the security, we need to argue that an adversary, given a constrained key $\mathsf{ck}_{\mathbf{z}}$, cannot tell the output of $\mathsf{CPRF.Eval}(\mathsf{msk}, \mathbf{x}^*)$ from random on a challenge input $\mathbf{x}^*$ satisfying $P_{\mathbf{x}^*}(\mathbf{z}) \neq 0$ (equivalently, when $F_{\mathbf{z}}(\mathbf{x}^*) = 1$). We first argue the pseudorandomness when the adversary makes no queries to the evaluation algorithm (*no-evaluation* security). Firstly, note that $\mathsf{CPRF.Eval}(\mathsf{msk}, \mathbf{x}^*)$ can be written as $\lfloor M\mu \cdot P_{\mathbf{x}^*}(\mathbf{z}) + \ \mathsf{CEval}_{\mathsf{pre\text{-}round}}(\mathsf{ck}_{\mathbf{z}}, \mathbf{x}) \rceil_2$, where $\mathsf{CEval}_{\mathsf{pre\text{-}round}}(\mathsf{ck}_{\mathbf{z}}, \mathbf{x}) = \sum_{i=1}^{t} c_i \cdot \mathcal{M}_i$ can be computed by the adversary.

Roughly speaking, as long as $\mathsf{ck}_{\mathbf{z}}$ hides μ, we can claim that it makes this output look random [4]. More precisely, we prove this by (1) setting the bound on the magnitude of each k_i to be superpolynomially bigger than the bound on the magnitude of μ, so that $\boldsymbol{\ell} = \mathbf{k} - \mu \cdot \mathbf{z}$ statistically hides μ, and (2) assuming the hardness of a variant of the ring-LWE problem, that we call *secret-power ring-LWE*, in order to argue that the extra elements $(\mathcal{M}_i = M \cdot (1/\mu)^{i-1} + e_i)_{i\in[t]}$ of the constrained key hide μ.We discuss our assumption and its credibility later in this section.

Insecurity in case of evaluation queries. The security of the CPRF presented above is compromised if an adversary has access to the evaluation oracle. For some constraint $\mathbf{z}$, consider two inputs $\mathbf{x} \neq \mathbf{x}'$ that satisfy $P_{\mathbf{x}}(\mathbf{z}) = P_{\mathbf{x}'}(\mathbf{z})$. Re-write the evaluation as $\mathsf{Eval}(\mathsf{msk}, \mathbf{x}) = y_{\mathbf{x}} = \lfloor a + b_{\mathbf{x}} \rceil_2$ and $\mathsf{Eval}(\mathsf{msk}, \mathbf{x}') = y_{\mathbf{x}'} = \lfloor a + b_{\mathbf{x}'} \rceil_2$, where $a = M\mu \cdot P_{\mathbf{x}}(\mathbf{z}) = M\mu \cdot P_{\mathbf{x}'}(\mathbf{z})$, $b_{\mathbf{x}} = \mathsf{CEval}_{\mathsf{pre\text{-}round}}(\mathsf{ck}_{\mathbf{z}}, \mathbf{x})$, and $b_{\mathbf{x}'} = \mathsf{CEval}_{\mathsf{pre\text{-}round}}(\mathsf{ck}_{\mathbf{z}}, \mathbf{x}')$. Note that $y_{\mathbf{x}'}$ can be re-written as
$y_{\mathbf{x}'} = \lfloor a + b_{\mathbf{x}} + (b_{\mathbf{x}'} - b_{\mathbf{x}}) \rceil_2$. Therefore, if $b_{\mathbf{x}'} - b_{\mathbf{x}}$ is "small", e.g., $|b_{\mathbf{x}'} - b_{\mathbf{x}}| \ll q/4$, then $y_{\mathbf{x}'} = y_{\mathbf{x}}$ with high probability, which can be used to distinguish the output of the function from random.

Achieving Full Security. We follow the approach suggested by [81] to achieve full security: we apply a suitable hash to the output of the CPRF concatenated with the input, namely $X||\mathbf{x}$, where $X = \mathsf{Eval}(\mathsf{msk}, \mathbf{x})$. The same hash is applied to the constrained evaluation. This modification has two goals: (1) concatenating with $\mathbf{x}$ prevents collisions since $X||\mathbf{x} \neq X'||\mathbf{x}'$ as long as $\mathbf{x} \neq \mathbf{x}'$, and (2) hashing in the ROM ensures that the output of the CPRF is indistinguishable from random as long as the probability of collisions inside the hash is negligible—that is the case as explained in (1).[5] As observed by prior works [8, 33, 81], the random oracle can be replaced by a correlation-robust hash function [51] in order to hide the linear correlations between the outputs and achieve selective security.

We note that our CPRF is inspired, though substantially different from the work of Attrapadung *et al.* [8], which supports log-depth circuits as constraints and is secure under a group-based assumption called decisional Diffie-Hellman inversion.

[3] For simplicity of this overview, we ignore several subtleties regarding the rounding of this error. We refer the reader to Section 5.1 for more details.

[4] We additionally have to assume that the range of P over $\{0,1\}^n \times \{0,1\}^n$ is invertible over R_q. We discuss these details in Section 5.1.

[5] We note that it is actually enough to hash the first 128 bits of X, concatenated with $\mathbf{x}$. This gives enough entropy to get security in the ROM and (depending on d) increases the concrete correctness probability since equality of rounding of fewer coefficients of two ring elements is required.

GAR-wPRF as a Constraint. We now turn our attention to finding a suitable wPRF candidate to evaluate with our CPRF. Following the approach of [33], we select the Goldreich-Applebaum-Raykov wPRF, we describe this in detail next.

Let κ, ι be positive integer. The predicate $\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota : \{0,1\}^{\kappa+\iota} \to \{0,1\}$ is defined as the following Boolean function

$$\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota(b_1, \ldots, b_{\kappa+\iota}) = \mathsf{XOR}(b_1, \ldots, b_\kappa) \oplus \mathsf{MAJ}(b_{\kappa+1}, \ldots, b_{\kappa+\iota}),$$

where $\mathsf{XOR}(b_1, \ldots, b_\kappa) = \bigoplus_{i=1}^{\kappa} b_i$, and $\mathsf{MAJ}(b_{\kappa+1}, \ldots, b_{\kappa+\iota})$ outputs 1 if $\sum_{i=1}^{\iota} b_{\kappa+i} \geq \iota/2$, and 0, otherwise.

Let $\ell = (\kappa + \iota) \cdot \lceil \log n \rceil$. The *Goldreich-Applebaum-Raykov wPRF with XOR-MAJ predicate* (GAR-wPRF) is the following function $F : \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}$, proposed by [7, 49] as a candidate weak PRF:

$$F(\mathbf{z}, \mathbf{x}) = \mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota(\mathbf{z}[x_1], \ldots, \mathbf{z}[x_{\kappa+\iota}]),$$

where $\mathbf{x} = \{x_1, \ldots, x_{\kappa+\iota}\}$ denotes the input and is viewed a set of $\kappa + \iota$ many bit-strings mapping to locations inside $\mathbf{z}$ (which denotes the key), with each bit-string $x_i \in \{0,1\}^{\lceil \log n \rceil}$. We refer to Section 9 for an analysis on how to set the parameters n, κ, ι.

Plugging GAR-wPRF into our CPRF. In order to use the GAR-wPRF as a constraint in the CPRF of Construction 1, a naive approach would be to find a polynomial that interpolates $F(\mathbf{z}, \mathbf{x})$ over the integers. However, this would result in a polynomial of degree $\kappa + \iota$ in general. To improve efficiency, we make the following observation: for our CPRF application, where the output is hashed with a hash function modelled as a random oracle, it suffices to interpolate $F(\mathbf{z}, \mathbf{x})$ with a polynomial P that maps $P(\mathbf{z}, \mathbf{x}) = 0$ whenever $F(\mathbf{z}, \mathbf{x}) = 0$, and $P(\mathbf{z}, \mathbf{x}) \neq 0$ whenever $F(\mathbf{z}, \mathbf{x}) = 1$—that is, it suffices to map $\{0,1\}$ to $\{0, \neq 0\}$.

Then, we observe that this relaxed interpolation allows for important efficiency gain: $F(\mathbf{z}, \mathbf{x})$ can be relaxed-interpolated with a polynomial of degree only $\max(\kappa, \iota)$. As the efficiency of our scheme degrades significantly with the degree due to the exponential noise growth, this yields major efficiency improvements.

Let us elaborate on how we interpolate the predicate. We need to find a polynomial $P_{\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota}$ such that for all $\mathbf{x} \in \{0,1\}^\ell, \mathbf{z} \in \{0,1\}^n$ it holds that:

$$P_{\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota}(\mathbf{x}, \mathbf{z}) = 0 \iff \mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota(\mathbf{z}[x_1], \ldots, \mathbf{z}[x_{\kappa+\iota}]) = 0$$

We observe that $P_{\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota}$ should evaluate to 0 *iff* (i) $\mathsf{XOR}_\kappa = 0$ and $\mathsf{MAJ}_\iota = 0$, or (ii) $\mathsf{XOR}_\kappa = 1$ and $\mathsf{MAJ}_\iota = 1$. We therefore define the following four predicates that represent $\mathsf{XOR}(b_1, \ldots, b_\kappa) \stackrel{?}{=} \sigma$ and $\mathsf{MAJ}(b'_1, \ldots, b'_\iota) \stackrel{?}{=} \sigma$, for $\sigma \in \{0,1\}$:

- $P_{\mathsf{XOR}_\kappa}(b_1, \ldots, b_\kappa) := \prod_{j \in \mathsf{Even}_\kappa} \left(\sum_{i=1}^{\kappa} b_i - j\right)$, where $\mathsf{Even}_\kappa = \{0, 2, \ldots, 2 \cdot \lfloor \kappa/2 \rfloor\}$ is the set of all even integers less than or equal to κ. The function P_{XOR_κ} therefore evaluates to 0 if an even number of input bits are 1, i.e., xoring inputs yields 0, and to a non-zero value, otherwise.

- $P_{\mathsf{MAJ}_\iota}(b'_1, \ldots, b'_\iota) := \prod_{j \in \mathsf{Small}_\iota} \left(\sum_{i=1}^{\iota} b'_i - j\right)$, where $\mathsf{Small}_\iota = \{0, 1, \ldots, \lfloor \iota/2 \rfloor\}$. Consequently, P_{MAJ_ι} evaluates to 0 if the majority of the input bits are 0.

- $P_{\neg\mathsf{XOR}_\kappa}(b_1, \ldots, b_\kappa) := \prod_{j \in \mathsf{Odd}_\kappa} \left(\sum_{i=1}^{\kappa} b_i - j\right)$, where $\mathsf{Odd}_\kappa = \{1, 3, \ldots, \kappa_{\text{odd}}\}$ denotes the set of all odd integers less than or equal to κ, and $\kappa_{\text{odd}} = \kappa$ if κ is odd and $\kappa_{\text{odd}} = \kappa - 1$ if κ is even. Therefore, $P_{\neg\mathsf{XOR}_\kappa}$ evaluates to 0 if xoring inputs yields 1, and to a non-zero value, otherwise.

- $P_{\neg\mathsf{MAJ}_\iota}(b'_1, \ldots, b'_\iota) := \prod_{j \in \mathsf{Big}_\iota} \left(\sum_{i=1}^{\iota} b'_i - j\right)$, where $\mathsf{Big}_\iota = \{\lfloor \iota/2 \rfloor + 1, \ldots, \iota\}$. Consequently, $P_{\neg\mathsf{MAJ}_\iota}$ evaluates to 0 if the majority of the input bits are 1.

Using the polynomials above, we define $P_{\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota}$ as

$$\begin{aligned}P_{\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota}(\mathbf{x},\mathbf{z}) :=& (P_{\mathsf{XOR}_\kappa}(\mathbf{z}[x_1],\ldots,\mathbf{z}[x_\kappa]) + P_{\mathsf{MAJ}_\iota}(\mathbf{z}[x_{\kappa+1}],\ldots,\mathbf{z}[x_{\kappa+\iota}]))\times \\ & (P_{\neg\mathsf{XOR}_\kappa}(\mathbf{z}[x_1],\ldots,\mathbf{z}[x_\kappa]) + P_{\neg\mathsf{MAJ}_\iota}(\mathbf{z}[x_{\kappa+1}],\ldots,\mathbf{z}[x_{\kappa+\iota}])),\end{aligned}$$

which can be further simplified to

$$\begin{aligned}P_{\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota}(\mathbf{x},\mathbf{z}) :=& P_{\mathsf{XOR}_\kappa}(\mathbf{z}[x_1],\ldots,\mathbf{z}[x_\kappa]) \times P_{\neg\mathsf{MAJ}_\iota}(\mathbf{z}[x_{\kappa+1}],\ldots,\mathbf{z}[x_{\kappa+\iota}]) + \\ & P_{\mathsf{MAJ}_\iota}(\mathbf{z}[x_{\kappa+1}],\ldots,\mathbf{z}[x_{\kappa+\iota}]) \times P_{\neg\mathsf{XOR}_\kappa}(\mathbf{z}[x_1],\ldots,\mathbf{z}[x_\kappa]),\end{aligned}$$

as the terms $P_{\mathsf{XOR}_\kappa}(\mathbf{b}) \times P_{\neg\mathsf{XOR}_\kappa}(\mathbf{b})$ and $P_{\mathsf{MAJ}_\iota}(\mathbf{b}) \times P_{\neg\mathsf{MAJ}_\iota}(\mathbf{b})$ are always 0.

Finally, note that the degree of this polynomial can be computed as $\max\left(|\mathsf{Even}_\kappa| + |\mathsf{Big}_\iota|, |\mathsf{Odd}_\kappa| + |\mathsf{Small}_\iota|\right)$, where $|\mathsf{Even}_\kappa|, |\mathsf{Odd}_\kappa| \leq \lfloor \kappa/2 \rfloor + 1$, and $|\mathsf{Small}_\iota|, |\mathsf{Big}_\iota| \leq \lfloor \iota/2 \rfloor + 1$.

The complement polynomial $P_{\neg\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota}$ is defined and analyzed analogously. Further analysis on this polynomial can be found in Section 6.

On the Secret-Power Ring-LWE Assumption. As discussed above, we introduce a new assumption, namely the secret-power ring-LWE (SP-RLWE) assumption, which is inspired by the *power ring-LWE* assumption introduced in [1]. Power ring-LWE assumes pseudorandomness of a set of ring-LWE samples of the form $(N^i, N^i\nu + e_i)$ for $i \in [\mathsf{poly}]$ where N is a fixed random ring element. We make a similar assumption, except that our correlations apply to the ring-LWE secret rather than its public label. We refer to the assumption introduced by [1] as *public-power ring-LWE* (PP-RLWE), to highlight the distinction between their variant and ours. A variant of our assumption was also considered in [56], with the difference being that it only considered up to power 2 of the secret μ, while we state it more generally. Our assumption is described next.

Definition 1 ((Decision) Secret-Power Ring-LWE (SP-RLWE) Assumption)**.** *Let $\lambda \in \mathbb{N}$ be a security parameter, and let $m = m(\lambda) \in \mathbb{N}$, $q = q(\lambda) \in \mathbb{N}$. Let $R = R(\lambda)$ be a ring defined over $\mathbb{Z}$, and $\chi_\mu = \chi_\mu(\lambda)$ and $\chi_e = \chi_e(\lambda)$ be distributions over R. The $\mathsf{SP\text{-}RLWE}(q, \chi_\mu, \chi_e, m)$ assumption states that*

$$\left(M,\ \ M \cdot \mu + e,\ \{M \cdot (1/\mu)^i + e_i\}_{i\in[m]}\right) \approx_c \left(M,\ \ \mathcal{M},\ \{\mathcal{M}_i\}_{i\in[m]}\right),$$

where $\mu \xleftarrow{\$} \chi_\mu$ is an invertible element, and $M, \mathcal{M}, \mathcal{M}_i \xleftarrow{\$} R_q$, and $e, e_i \xleftarrow{\$} \chi_e$, for all $i \in [m]$.

We conjecture that the decision SP-RLWE assumption holds for all parameter choices admitted by plain ring-LWE. While this assumption requires time and cryptanalytic effort to build confidence, we are not aware of any attacks that can exploit this correlation. Please see Section 4 for more details.

Compiling to Public-Key PCFs. We finally build our public-key PCF for OT by incorporating techniques from *succinct homomorphic secret sharing* (succinct HSS) [1]. Succinct HSS described a way to achieve balanced communication between two parties engaging in certain non-interactive MPC protocols, like vector OLE (or VOLE). They describe their techniques in the lattice setting based on the public-power ring-LWE assumption (PP-RLWE), where different samples are computed using different powers of public ring elements. We refer to their version as *public-power ring-LWE* (PP-RLWE).

The reason that techniques from succinct HSS work for our public-key setup is because we want a non-interactive protocol (for our CPRF) after which the party having secret input μ learns the vector $\mathbf{k}$, while the party having secret input vector $\mathbf{z}$ learns the output $\boldsymbol{\ell}$, such that $\boldsymbol{\ell} = \mathbf{k} - \mu \cdot \mathbf{z}$. This can be seen as a VOLE protocol, and while any such protocol would have sufficed for our purpose, an advantage of using Succinct HSS is that we get smaller key sizes which are moreover balanced between the two parties.

The PK-PCF Protocol. Now we describe our PK-PCF protocol. We work over a ring R of dimension d. We use polynomial representations $P : \{0,1\}^\ell \times R \to R$, $\overline{P} : \{0,1\}^\ell \times R \to R$ associated with a wPRF $F : \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}$ such that

$$P(\mathbf{x},\mathbf{z}) = 0 \iff F(\mathbf{z},\mathbf{x}) = 0 \text{ and } \overline{P}(\mathbf{x},\mathbf{z}) = 0 \iff 1 - F(\mathbf{z},\mathbf{x}) = 0,$$

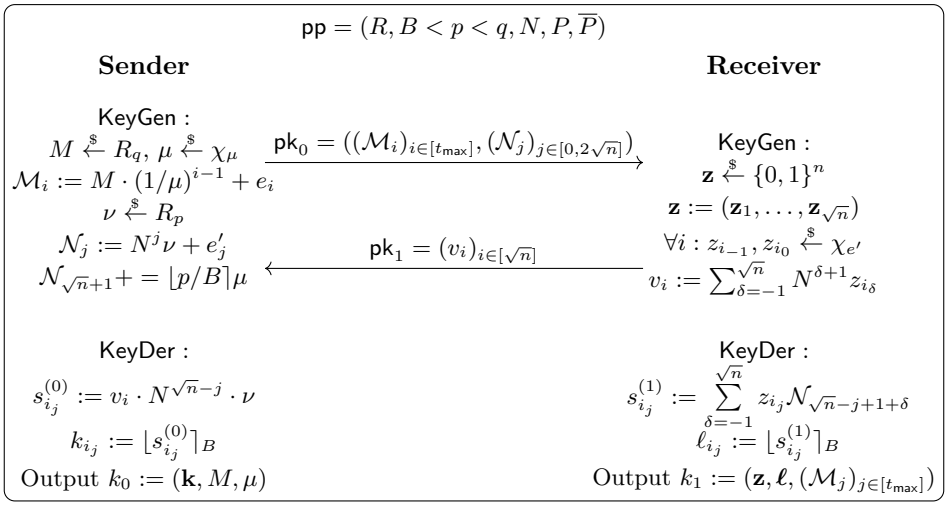

Figure 1: Public-Key Setup for our CPRF and PCF, inspired from Succinct HSS [1]. Here, R is a ring, $B < p < q$ are positive integers and $N \xleftarrow{\$} R_p$. Also P and $\overline{P}$ are polynomials that represent a wPRF and its complement. $t_{\max}$ is the maximum degree of P and $\overline{P}$. All computations are done modulo p, except for computing $\mathcal{M}_i$'s that is modulo q. Some details, in particular regarding setting q, are skipped. We refer to Section 7 for more details.

for any $\mathbf{x} \in \{0,1\}^\ell, \mathbf{z} \in \{0,1\}^n$. We need two positive integers $p < q$ such that PP-RLWE holds in R_p while SP-RLWE holds in R_q, in accordance with the respective distributions of public, secret and error elements. Let $N \xleftarrow{\$} \mathcal{D}_N$ be a public element, where $\mathcal{D}_N$ is a distribution over R_p. [6] Looking ahead, N acts as the public element for the PP-RLWE assumption.

The sender and the receiver start by generating their public keys and deriving their evaluation keys as shown in Figure 1. Essentially, pk_0 consists of (1) $t_{\max}$ SP-RLWE samples, where $t_{\max}$ denotes the maximum of the degrees of polynomials P and $\overline{P}$, and (2) $2\sqrt{n}+1$ PP-RLWE samples. Thus, assuming the hardness of the SP-RLWE and PP-RLWE problems, pk_0 hides μ. On the other hand, pk_1 contains $\sqrt{n}$ elements of the form $z_{i_0} + z_{i_{-1}} \cdot N + S_i$, where z_{i_0} and $z_{i_{-1}}$ are freshly-sampled random elements of $\chi_{e'}$, and $S_i = \sum_{j=1}^{\sqrt{n}} N^{j+1} z_{i_j}$. Thus, assuming the hardness of the ring-LWE problem (with small secrets), each v_i hides S_i and therefore pk_1 hides $\mathbf{z}$.

Evaluation for the sender on input $\mathbf{x} \in \{0,1\}^\ell$ involves performing two different CPRF evaluations, using the same master secret key $(\mathbf{k}, M, \mu)$ but different polynomials P and $\overline{P}$, to get values r_0 and r_1 respectively. It then outputs (r_0, r_1). The receiver's evaluation proceeds by first computing the bit $\mathsf{b} = F(\mathbf{z}, \mathbf{x})$ and then performing constrained evaluation for the CPRF, using the constrained key $(\mathbf{z}, \boldsymbol{\ell}, (\mathcal{M}_i)_{i\in[t_{\max}]})$ and the polynomial P^{b}, that is P if $\mathsf{b} = 0$ and $\overline{P}$ if $\mathsf{b} = 1$, to get the value r'. It then outputs (b, r').

We analyze the protocol in detail in Section 7. We show that the evaluation keys, in particular the vectors $\mathbf{k}$ and $\boldsymbol{\ell}$, indeed form subtractive shares of $\mu \cdot \mathbf{z}$. We also show that it achieves the property of pseudorandom correlated outputs by assuming PP-RLWE, the pseudorandomness of the wPRF F and the adaptive security of our CPRF. In our protocol, *sender security* (i.e., security against the receiver) relies on the PP-RLWE assumption while *receiver security* relies on the variant of ring-LWE that is in turn implied by PP-RLWE, as described previously.

2.2 Packing from Truncated Secret-Power Ring-LWE

Our efficiency estimates indicate that for a suitable choice of parameters, the PK-PCF outlined in the previous sections can achieve a throughput of a few dozen OTs per second. While this is promising, the aim of our work is to investigate how far we can push efficiency, even if this requires making new (plausible) assumptions. In this spirit, we introduce a variant of the SP-RLWE assumption that modifies it

[6] This is the same distribution that the public matrix for the PP-RLWE assumption comes from, as defined in [1].

slightly. Concretely, we let the public ring element M be sampled from a slighly non-standard distribution: M is *truncated on both ends* except for the constant coefficient (that is, there exists a value $r \ll d$ such that the coefficients of $X, \cdots, X^r$ and $X^{d-1}, \cdots, X^{d-r}$ are all 0), and uniformly random otherwise.

Concretely, our assumption states that

$$\left(M' = M + \Delta,\ \{M' \cdot (1/\mu)^i + e_i\}_{i\in[m]}\right) \approx_c \left(M' = M + \Delta,\ \{\mathcal{M}_i\}_{i\in[m]}\right),$$

where $\mu \xleftarrow{\$} \chi_\mu$ is an invertible element, M is a random r-*truncated* element of R (it starts and ends with r zero-coefficients), $\Delta \xleftarrow{\$} \mathbb{Z}_q$ (hence M' re-adds the constant coefficient), and $e, e_i \xleftarrow{\$} \chi_e$, for all $i \in [m]$.

Our assumption captures the intuition that even when M is slightly truncated on both ends, the security of the the corresponding ring-LWE instance should not decrease significantly more than the security loss resulting from reducing the dimension by the truncated amount. That is, we expect r-Truncated Secret-Power Ring-LWE to be at least as hard as (Secret-Power) Ring-LWE with dimension $d - 2r$.

2.2.1 Discussion on the assumption.

For simplicity, let us ignore the secret power in this discussion, since the truncation of M appears to be an orthogonal modification of the ring-LWE assumption, and focus on a term of the form $(M', M' \cdot s + e)$ for small s, e. It is easy to observe that our assumption is invariant with respect to the position of the $2r$ zero coefficients as multiplication by X^{-i} is an automorphism, and the error distribution is stable with respect to this automorphism: anyone can publicly compute $X^{-i} \cdot (M', M' \cdot s + e) = (M'', M'' \cdot s + X^{-i} \cdot e')$ where $e' = X^{-i} \cdot e$ follows the same distribution as e, and M'' has its $2r$ zero coefficients shifted by i. Let us also focus on the seemingly stronger version of the assumption where the constant coefficient is also zeroed-out.[7]

Observe that using the shift X^{-r} on this variant translates the assumption to ring-LWE *with a degree* $d - 2r$ *polynomial*. We can then ask how much lowering the degree of M reduces security. While obvious issues appear if the secret s also has low degree, this is not the case when s has full degree. Since the entropy of M is reduced from $d \log q$ to $(d - 2r) \log q$, one can anticipate that the "effective dimension" of the truncated problem can be reduced to $d - 2r$. We note, however, that it is not entirely obvious whether state-of-the-art lattice attacks such as BKZ on the ideal lattice, or the more recent dual attacks, can fully benefit from the entropy reduction, as the cost of these attacks typically depends on the ambient lattice dimension, and truncating does not obviously reduce the ambient lattice dimension of the (ring-)LWE instance. We view it as an interesting problem to adapt lattice attacks to benefit from truncation, and conservatively assume that they can take full benefit of it in the meantime.

To give the reader a sense of scale, here is a typical parameter setting that we consider: set $\log q = 698$ and $d = 32768$. Here, standard methods to estimate ring-LWE security [14] predict that $d = 25961$ would have sufficed to get 128 bits of security (the real number is probably much lower, as the rough estimator tends to be overly conservative). Then, we sample a random M whose first and last 2886 coefficients are 0, leaving 26996 random nonzero coefficients, which can reasonably be expected to be above 128 bits of security.

Equipped with this new assumption, we introduce a variant of our baseline PK-PCF candidate that achieves a faster throughput (by a factor 8 to 16 with our parameters) by *packing* multiple CPRF evaluations in parallel. The idea is natural: the parties' keys form additive shares $(\mathbf{k}, \boldsymbol{\ell})$ of $\mu \cdot \mathbf{z}$, where $\mathbf{z}$ is a vector of bits. That is, a pair (k_i, ℓ_i) of *full ring elements* is used to share each bit of $\mathbf{z}$, which is very wasteful. Instead, one could attempt to pack multiple wPRF keys $(\mathbf{z}_1, \mathbf{z}_2, \cdots)$ into a vector of ring elements $\mathbf{Z}$, which would not increase the size of the shares $(\mathbf{k}, \boldsymbol{\ell})$ of $\mu \cdot \mathbf{Z}$. Unfortunately, the baseline construction from Section 7 is not compatible with standard packing techniques, that rely on Chinese remainder theorem (CRT) decomposition to map the ring to many copies of a small ring and fits one value in each copy. We explain the issue below.

[7] There is a straightforward reduction to $\Delta = 0$ with a loss of $1/q$ which shows that this version is strictly stronger when q is polynomial. In the setting where q is exponential, we are not aware of any formal reduction, but it is reasonable to expect attacks to perform similarly well or better when additional coefficients are zeroed-out.

2.2.2 Infeasibility of CRT Packing.

In our construction, after evaluating the CPRF, the two parties obtain noisy shares of $M_c \cdot P^{\kappa,\iota}_{\mathbf{x}}(\mathbf{z})$, where $P^{\kappa,\iota}_{\mathbf{x}}(\mathbf{z})$ is viewed as an element of the ring. Now, if multiple wPRF keys $(\mathbf{z}_1, \mathbf{z}_2, \cdots)$ are CRT-packed into a single ring element $Z = \mathsf{Pack}(\mathbf{z}_1, \mathbf{z}_2, \cdots)$, the homomorphic properties of CRT decomposition guarantee that $M_c \cdot P^{\kappa,\iota}_{\mathbf{x}}(Z) = M_c \cdot \mathsf{Pack}(P^{\kappa,\iota}_{\mathbf{x}}(\mathbf{z}_1), P^{\kappa,\iota}_{\mathbf{x}}(\mathbf{z}_2), \cdots)$. At this stage, it remains for the parties to round away the noise via local rounding, after what they should obtain the same value whenever $P^{\kappa,\iota}_{\mathbf{x}}(\mathbf{z}_i) = 0$. This leaves the parties with two options:

1. Either they round the polynomial coefficient-wise, but then the coefficients of $\mathsf{Pack}(P^{\kappa,\iota}_{\mathbf{x}}(\mathbf{z}_1), P^{\kappa,\iota}_{\mathbf{x}}(\mathbf{z}_2), \cdots)$ have no reason to be zero even when some of the $P^{\kappa,\iota}_{\mathbf{x}}(\mathbf{z}_i)$ evaluate to 0.

2. Or they round "slot-wise" in the CRT domain (where some of the slots contain a value $P^{\kappa,\iota}_{\mathbf{x}}(\mathbf{z}_i) = 0$). However, the noise is only small coefficient-wise: it does not remain small in the CRT domain, and correctness fails once more.[8]

A similar issue shows up if one attempts to pack in the evaluation of the polynomial instead. To round away the noise, it seems necessary to store the values $P^{\kappa,\iota}_{\mathbf{x}}(\mathbf{z}_i)$ as *coefficients* of the polynomial. Unfortunately, coefficient packing does not enjoy the same homomorphic properties as CRT packing, or interpolation-based packing.

2.2.3 Sparse Coefficient Packing.

Our starting point is the observation from [75] that coefficient-wise packing enjoys *some* amount of homomorphism if a suitable subset of coefficients is used to store the values. Consider two ring elements U, V with coefficients u_i, v_i respectively. Fix i such that $2i < d$ and assume that U, V have degree at most $d/2$. Then, the coefficient of X^{2i} in $U \cdot V$ is $\sum_{k+\ell=2i} u_k v_\ell$. Therefore, denoting S as the subset of indices of all nonzero coefficients of U, V, if S does not contain any pair $(k, \ell) \neq (i, i)$ such that $k + \ell = 2i$, it holds that the coefficient of X^{2i} in $U \cdot V$ is exactly $u_i v_i$. More generally, if S is a subset of $[d/2]$ that does not contain any 3-term arithmetic progression, it holds that for all $i \in S$, the $2i$-th coefficient of $U \cdot V$ is $u_i v_i$. Unfortunately, no closed-form solution is known for the classical problem of finding progression-free subsets of an interval. However, one can always construct such a set greedily, by adding indices to the set as long as they do not contradict the conditions for any of the previous indices. In this work, we generalize this partial homomorphism to arbitrary *homogeneous multivariate polynomials* and prove the following:

Lemma 1 (Coefficient Packing Homomorphism)**.** *Let C denote an n-input, one-output arithmetic circuit that computes a homogeneous polynomial of total degree t. Let $R = \mathbb{Z}_q[X]/(F(X))$ be a polynomial ring over $\mathbb{Z}_q$ with $\deg(F) = d$, and let $S \subseteq \{0, \ldots, \lfloor (d-1)/t \rfloor\}$ be a set such that for every $i \in S$ and every $(i_1, \ldots, i_t) \in S^t \setminus \{(i, \ldots, i)\}$, it holds that $\sum_{j=1}^{t} i_j \neq t \cdot i$. For $j = 1, \ldots, n$, let $U_j(X) = \sum_{i \in S} u_{j,i} X^i \in R$ (i.e. each U_j has nonzero coefficients only at exponents in S). Then for every $i \in S$, the coefficient of $X^{t \cdot i}$ in the polynomial $C(U_1, \ldots, U_n)$ is equal to $C(u_{1,i}, \ldots, u_{n,i})$.*

Packed CPRF via Sparse Coefficient Packing. We now explain why our new truncated assumption is crucial to improving the efficiency of our alternative CPRF. Our starting point is a variant of our baseline CPRF with the following modifications:

- In $\mathsf{CPRF.Eval}(\mathsf{msk}, \mathbf{x} \in \{0,1\}^\ell)$, output $y := \lfloor (M + \Delta) \cdot P_{\mathbf{x}}(\mathbf{k}/\mu) \pmod q \rceil_p$ (instead of $y := \lfloor M\mu \cdot P_{\mathbf{x}}(\mathbf{k}/\mu) \pmod q \rceil_2$) where p is a λ-bit modulus, and keep only the first rounded coefficient (the constant term).
- In $\mathsf{CPRF.Constrain}(\mathsf{msk}, \mathbf{z} \in \{0,1\}^n)$, set $\mathcal{M}_i := (M + \Delta) \cdot (1/\mu)^i + e_i$ (instead of $\mathcal{M}_i := M \cdot (1/\mu)^{i-1} + e_i$), where $e_i \xleftarrow{\$} \chi_e$.
- $\mathsf{CPRF.CEval}(\mathsf{ck}_{\mathbf{z}}, \mathbf{x} \in \{0,1\}^\ell)$ proceeds identically, but only the first coefficient of the rounded polynomial is kept.

[8] One could attempt to use a different noise distribution that is small in the CRT domain instead of having small coefficients; however, ring-LWE is well-known to be insecure with this noise distribution.

We give a high level sketch of correctness and security: one can check that $\mathsf{Eval}(\mathsf{msk}, \mathbf{x}) = \lfloor (M + \Delta) \cdot P_{\mathbf{x}}(\mathbf{z}) + T \rceil_p$ and $\mathsf{CEval}(\mathsf{ck}_{\mathbf{z}}, \mathbf{x}) = \lfloor T + e \rceil_2$, where T is some polynomial and e is a small error term. Then, after rounding, both parties obtain the same value w.h.p. whenever $P_{\mathbf{x}}(\mathbf{z}) = 0$. When $P_{\mathbf{x}}(\mathbf{z}) \neq 0$, however, the evaluation and constrained evaluation differ by an affine function $u\Delta + v = P_{\mathbf{x}}(\mathbf{z}) \cdot \Delta + P_{\mathbf{x}}(\mathbf{z}) \cdot M$ of Δ in their constant coefficient. Invoking SP-RLWE allows to replace Δ with a fresh random scalar in the security analysis, which lets us obtain full security after hashing with a hash modeled as a random oracle.

This alternative construction does not improve over the CPRF construction from Section 5.1 (it achieves comparable efficiency), but serves as a stepping stone for our improved scheme. Starting from this variant, we explain how to enhance it with our sparse coefficient packing method, and explain why our new truncated assumption is crucial to improving the efficiency of our alternative CPRF.

Recall that $\mathsf{Eval}(\mathsf{msk}, \mathbf{x}) = \lfloor (M + \Delta) \cdot P_{\mathbf{x}}(\mathbf{z}) + T \rceil_p$ and $\mathsf{CEval}(\mathsf{ck}_{\mathbf{z}}, \mathbf{x}) = \lfloor T + e \rceil_p$, where T is some polynomial and e is a small error term. Looking ahead, we will want to replace $P_{\mathbf{x}}(\mathbf{z})$ (which is a scalar) with a larger degree polynomial $P_{\mathbf{x}}(\mathbf{Z})$ that *packs* multiple evaluations $P_{\mathbf{x}}(\mathbf{z}_i)$. However, as we will see shortly, our packing mechanism has very limited homomorphic properties, that are in particular incompatible with the multiplication by M. However, if $\deg(P_{\mathbf{x}}(\mathbf{Z}) < r$ and when $M \in \mathsf{Trunc}_r(R_q)$ instead, we have $(M + \Delta) \cdot P_{\mathbf{x}}(\mathbf{Z}) = M \cdot P_{\mathbf{x}}(\mathbf{Z}) + \Delta \cdot P_{\mathbf{x}}(\mathbf{Z})$, where $M \cdot P_{\mathbf{x}}(\mathbf{Z}) = X^r \cdot T'$ for some polynomial T' (since $\deg(M) \leq d - 1 - r$, the product $M \cdot P_{\mathbf{x}}(\mathbf{Z})$ causes no wraparounds, and M's lowest nonzero coefficient is in front of X^r), and $\deg(\Delta \cdot P_{\mathbf{x}}(\mathbf{Z})) < r$. Therefore, the first r coefficients of $\lfloor (M + \Delta) \cdot P_{\mathbf{x}}(\mathbf{z}) + T \rceil_p$ are those of $\lfloor \Delta \cdot P_{\mathbf{x}}(\mathbf{z}) + T \rceil_p$, resolving the incompatibility between our packing mechanism and the multiplication with M. From there, the parties will be able to extract multiple OT-correlated pairs at once ("for the price of one") from the "slots" of $P_{\mathbf{x}}(\mathbf{Z})$ where the $P_{\mathbf{x}}(\mathbf{z}_i)$ are packed, yielding a speedup factor proportional to the number of packed $\mathbf{z}_i$.

Using this approach and skipping several details, we obtain a packed CPRF that we use as a drop-in replacement in our PK-PCF construction. The new PK-PCF retains essentially the same efficiency features as our initial construction, but now outputs a batch of $|S|$ OTs for the price of one, upon each evaluation, yielding a factor $|S|$ speedup. For more details, please refer to Section 8.

2.3 Efficiency Analysis

We rely on the parameter selection method of previous works [33, 37] to select concrete parameters for the Goldreich-Applebaum-Raykov wPRF. We select two candidates that we conjecture to achieve 128 bits of security. Having selected the candidate and setting the statistical security parameter to $\sigma = 64$, we measure the noise growth of our CPRF evaluation algorithm, and select a modulus q large enough to guarantee correctness of the result after rounding. We summarize the choices below:

- **(Fast)** XOR_5-MAJ_5. wPRF key size: $2^{17.5}$. Maximum number of OTs: $2^{43.75}$. $\log q = 698$.
- **(Small)** XOR_6-MAJ_8. wPRF key size: 2^{14} Maximum number of OTs: 2^{42}. $\log q = 956$.

For both settings, we use a ring of dimension $d = 32768$. Then, to select the sparse subset in which to pack wPRF keys, we implement and run a Python script that greedily searches for the largest possible S satisfing our condition (for every $s \in S$, the only way to write $t \cdot s$ as a sum of t elements of s is by summing s with itself t times) within an interval, where the size of the interval is determined by the amount of truncation of the coefficients of M we are willing to assume. Our script finds usable sets of size 9 to 16 (where larger sets require a more aggressive parametrization of the assumption) for the fast candidate, and of size 6 to 10 for the small candidate (see Table 3).

We then provide a detailed efficiency estimation. We rely on benchmarks for low-level ring operations: the forward NTT, the inverse NTT, and pointwise multiplications, using the numbers from the Intel HEXL C++ Library reference paper [32] (see Table 4), and describe an implementation strategy for our (packed) PK-PCF that minimizes calls to expensive routines (such as NTT and inverse NTT), significantly improving over a naive evaluation. Our final results are summarized in Table 1 and feature throughputs in the range of 173 to 1000 OT/s, depending on the party (receiver is about twice slower), wPRF candidate (*fast* is about twice faster), and aggressiveness of the assumption, with a maximum bound on the number of OT produced at $2^{45} \sim 2^{48}$ depending on the setting. For more details, please refer to Section 9.

Table 2: Estimated runtimes (in milliseconds) for computing a pack of $|S|$ OTs, and estimated throughput (OT produced per seconds) for the sender and the receiver, for two values of t_{max}. "cons." denotes conservative and "aggr." denotes aggressive. The quantity #OTs denotes a bound on the maximum number of pseudorandom OTs that can be produced in each parameter setting.

	Sender		
	Runtime	Throughput (cons.)	Throughput (aggr.)
$t_{\mathsf{max}} = 6$ (Fast, 128 bits)	14ms	857 OT/s	1000 OT/s
$t_{\mathsf{max}} = 9$ (Small, 100 bits)	22ms	409 OT/s	454 OT/s
	Receiver		
	Runtime	Throughput (cons.)	Throughput (aggr.)
$t_{\mathsf{max}} = 6$ (Fast, 128 bits)	26ms	462 OT/s	538 OT/s
$t_{\mathsf{max}} = 9$ (Small, 100 bits)	52ms	173 OT/s	192 OT/s
		#OTs	
		Conservative	Aggressive
$t_{\mathsf{max}} = 6$ (Fast, 128 bits)		$2^{47.56}$	$2^{47.75}$
$t_{\mathsf{max}} = 9$ (Small, 100 bits)		$2^{44.58}$	2^{45}

2.4 Directions for Future Work

We are optimistic that our results can serve as a stepping stone to efficient post-quantum PK-PCFs. We mention two open questions. The first is whether it is possible to build similarly efficient post-quantum PK-PCFs from a more standard flavor of ring-LWE or other post-quantum assumptions. The second question is regarding the packing techniques we used to optimize efficiency. Note that due to technical complications, we could only pack 8-12 parallel executions, while with our parameters, we theoretically have 32768 available slots. We envision that future work could possibly make better use of these slots and gain several orders of magnitude of improvements following our template.

3 Preliminaries

We provide some definitions and preliminaries below.

We use λ to denote the security parameter. For $n \in \mathbb{N}$, we denote by $[n]$ the set $\{1, \ldots, n\}$. For integers $m < n$, we denote by $[m, n]$ the set $\{m, m+1, \ldots, n\}$. For any positive integers x, y such that $x \geq y$, let $\binom{x}{y}$ denote the combinatorial expression $\frac{x!}{(x-y)!y!}$. We mostly use bold lowercase letters (e.g., $\mathbf{r}$) to denote vectors. For any bit-string $\mathbf{x} \in \{0,1\}^*$, let $\mathbf{x}[i]$ denote the ith bit in the description of $\mathbf{x}$. We write $\mathsf{poly}(\lambda)$ to denote an arbitrary polynomial function. We denote by $\mathsf{negl}(\lambda)$ a negligible function in λ, and PPT stands for probabilistic polynomial-time. For a finite set S, we write $x \xleftarrow{\$} S$ to denote that x is sampled uniformly at random from S. For an algorithm $\mathcal{A}$, we denote by $y \leftarrow \mathcal{A}(x)$ the output y after running $\mathcal{A}$ on input x. For two distributions $\mathcal{D}$ and $\mathcal{D}'$ over the same domain, we write $\mathcal{D} \approx_c \mathcal{D}'$ to denote that they are computationally indistinguishable, and $\mathcal{D} \approx_s \mathcal{D}'$ to denote that they are statistically indistinguishable.

3.1 Pseudorandom Functions

Definition 2 ((Weak) Pseudorandom Function (wPRF, PRF), [50,69])**.** *Let $\lambda \in \mathbb{N}$ be a security parameter. A (weak) pseudorandom function with domain $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$, key space $\mathcal{K} = \mathcal{K}(\lambda)$, and range $\mathcal{Y} = \mathcal{Y}(\lambda)$, consists of the following two polynomial-time algorithms:*

- $\mathsf{KeyGen}(1^\lambda) \to k$*: A probabilistic algorithm that on input the security parameter λ, outputs a master secret key $k \in \mathcal{K}$.*
- $\mathsf{Eval}(k, x) \to y$*: A deterministic algorithm that on input the master secret key k, and an input value $x \in \mathcal{X}$, outputs a value $y \in \mathcal{Y}$.*

We say that the pair $(\mathsf{KeyGen}, \mathsf{Eval})$ *is a*

- ***pseudorandom function (PRF)*** *if for all* $\lambda \in \mathbb{N}$ *and any PPT adversary* Adv*, it holds that*

$$\left|\Pr\left[\mathsf{Adv}^{\mathsf{Eval}(k,\cdot)}(1^\lambda)=1 \middle| k \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)\right] - \Pr\left[\mathsf{Adv}^{RF(\cdot)}(1^\lambda)=1 \middle| RF \xleftarrow{\$} \mathcal{F}\right]\right| = \mathsf{negl}(\lambda),$$

where $\mathcal{F}$ *is the set of all functions with domain* $\mathcal{X}$ *and range* $\mathcal{Y}$*.*

- ***weak pseudorandom function (wPRF)*** *if for all* $\lambda \in \mathbb{N}$*, any PPT adversary* Adv *and any polynomially bounded number* $Q \in \mathbb{N}$*, it holds that*

$$\left\{ \left((x_i, \mathsf{Eval}(k,x_i))_{i\in[Q]}\right) \middle| \begin{array}{c} k \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda) \\ \forall i \in [Q] : x_i \xleftarrow{\$} \mathcal{X} \end{array} \right\} \approx_c \left\{ \left((x_i, y_i)_{i\in[Q]}\right) \middle| \begin{array}{c} \forall i \in [Q] : \\ x_i \xleftarrow{\$} \mathcal{X},\ y_i \xleftarrow{\$} \mathcal{Y} \end{array} \right\}.$$

Definition 2 captures the PRF security notion that is referred to as the *real-or-random* security. We also recall the find-then-guess security of a pseudorandom function which is equivalent to the real-or-random security up to a multiplicative gap of $\mathcal{O}(Q)$ between the advantage functions, where Q is the number of evaluation queries.

Definition 3 (Find-then-Guess Security of (weak) PRFs)**.** *Let* λ *be a security parameter. A function* $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ *is called a secure pseudorandom function if it is efficiently computable and the advantage of any PPT adversary* Adv *in the following game is negligible:*

- ***Setup.*** *The challenger chooses a random key* $k \xleftarrow{\$} \mathcal{K}$ *and a random bit* $b \xleftarrow{\$} \{0,1\}$*, and initializes a set* $S = \varnothing$*.*
- ***Pre-Challenge Evaluation Queries.*** Adv *adaptively sends arbitrary inputs* $x \in \mathcal{X}$ *to the challenger. The challenger computes and returns* $F_k(x)$ *to* Adv*. It also updates* $S \leftarrow S \cup \{x\}$*.*
- ***Challenge Phase.*** Adv *sends an input* $x^* \in \mathcal{X}$ *as its challenge query to the challenger with the restriction that* $x^* \notin S$*. If* $b = 0$*, then the challenger computes* $y^* \leftarrow F_k(x^*)$*. If* $b = 1$*, then the challenger samples a random element* $y^* \xleftarrow{\$} \mathcal{Y}$*. It then returns* y^* *to* Adv*.*
- ***Post-Challenge Evaluation Queries.*** Adv *continues sending arbitrary inputs* $x \in \mathcal{X}$ *to the challenger with the restriction that* $x \neq x^*$*, and receives* $F_k(x)$*.*
- ***Guess.*** Adv *outputs a bit* $b' \in \{0,1\}$*.*

Adv *wins if* $b' = b$*. We define the advantage of* Adv *in winning the game as* $|2 \cdot \Pr[\mathsf{Adv}\ \textit{wins}] - 1|$*, where the probability is over the internal coins of* Adv *and the challenger.*

3.2 Constrained Pseudorandom Functions

Definition 4 (Constrained Pseudorandom Functions)**.** *Let* λ *be a security parameter. A constrained pseudorandom function (CPRF) with domain* $\mathcal{X} = \mathcal{X}(\lambda)$*, key space* $\mathcal{K} = \mathcal{K}(\lambda)$*, and range* $\mathcal{Y} = \mathcal{Y}(\lambda)$*, that supports a class of circuits* $\mathcal{C} = \mathcal{C}(\lambda)$*, where each* $C_\lambda \in \mathcal{C}_\lambda$ *has domain* $\mathcal{X}_\lambda$ *and range* $\{0,1\}$*, consists of the following four polynomial-time algorithms:*[9]

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pp}, \mathsf{msk})$*: A probabilistic algorithm that on input the security parameter* λ*, outputs a public parameter* pp *and a master secret key* $\mathsf{msk} \in \mathcal{K}$*. We assume that all algorithms have access to* pp*.*

[9] In the rest of the paper, we drop the subscript λ when it is clear from context.

- $\mathsf{Eval}(\mathsf{msk}, x) \to y$: *A deterministic algorithm that on input the public parameter* pp*, the master secret key* msk*, and an input* $x \in \mathcal{X}$*, outputs a value* $y \in \mathcal{Y}$*.*
- $\mathsf{Constrain}(\mathsf{msk}, C) \to \mathsf{ck}_C$: *A probabilistic algorithm that on input the master secret key* msk*, and a circuit* $C \in \mathcal{C}$*, outputs a constrained key* ck_C*.*
- $\mathsf{CEval}(\mathsf{ck}_C, x) \to y$: *A deterministic algorithm that on input the public parameter* pp*, a constrained key* ck_C*, and an input* $x \in \mathcal{X}$*, outputs a value* $y \in \mathcal{Y}$*.*

Correctness. *For all security parameter* $\lambda \in \mathbb{N}$*, any constrain* $C \in \mathcal{C}$*, and any input* $x \in \mathcal{X}$ *such that* $C(x) = 0$*, we have:*

$$\Pr\left[\mathsf{Eval}(\mathsf{msk}, x) \neq \mathsf{CEval}(\mathsf{ck}_C, x) : \begin{array}{l}(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ \mathsf{ck}_C \leftarrow \mathsf{Constrain}(\mathsf{msk}, C)\end{array}\right] \leq \mathsf{negl}(\lambda).$$

1-Key Adaptive (Real-Or-Random) Security. *We say that a CPRF is 1-key adaptively secure if for all* $\lambda \in \mathbb{N}$*, the advantage of any PPT adversary* Adv *in the following game is negligible in* λ*:*

- ***Setup:*** *The challenger runs* $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$*, initializes a set* $S_{\mathsf{Eval}} = \varnothing$*, and chooses a random bit* $b \xleftarrow{\$} \{0,1\}$*. It then (lazily) samples a random function* $\mathcal{RF} : \mathcal{X} \to \mathcal{Y}$*. It then sends* pp *to* Adv*.*
- ***Pre-Constrained-Key Queries:*** Adv *can adaptively send arbitrary input values* $x \in \mathcal{X}$ *to the challenger. If* $b = 0$*, the challenger computes* $y \leftarrow \mathsf{Eval}(\mathsf{msk}, x)$*. Otherwise, if* $b = 1$*, the challenger computes* $y \leftarrow \mathcal{RF}(x)$*. It then returns* y *to* Adv*, and updates* $S_{\mathsf{Eval}} \leftarrow S_{\mathsf{Eval}} \cup \{x\}$*.*
- ***Constrained-Key Query:*** Adv *can choose a circuit* $C \in \mathcal{C}$ *and sends it to the challenger. If there exists* $x \in S_{\mathsf{Eval}}$ *such that* $C(x) = 0$*, the challenger aborts and outputs* $\perp$*. Otherwise, the challenger computes* $\mathsf{ck}_C \leftarrow \mathsf{Constrain}(\mathsf{msk}, C)$ *and returns the constrained key* ck_C *to* Adv*.*
- ***Post-Constrained-Key Evaluation Queries:*** Adv *can continue submitting evaluation queries as before, with the restriction that it cannot query any* x *such that* $C(x) = 0$*.*[10]
- ***Guess:*** Adv *outputs a bit* $b' \in \{0,1\}$*.*

1-Key Selective No-Evaluation Security. *We say that a CPRF is 1-key selectively no-evaluation secure if for all* $\lambda \in \mathbb{N}$*, the advantage of any PPT adversary* Adv *in the following game is negligible in* λ*:*

- ***Setup:*** *The challenger runs* $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ *and chooses a random bit* $b \xleftarrow{\$} \{0,1\}$*. It then sends* pp *to* Adv*.*
- ***Constrained-Key Query:*** *The adversary chooses a circuit* $C \in \mathcal{C}$ *and sends it to the challenger. The challenger computes* $\mathsf{ck}_C \leftarrow \mathsf{Constrain}(\mathsf{msk}, C)$ *and returns the constrained key* ck_C *to* Adv*.*
- ***Challenge:*** Adv *sends a single input* $x^* \in \mathcal{X}$ *as its challenge query to the challenger with the restriction that* $C(x^*) \neq 0$*. If* $b = 0$*, then the challenger computes* $y^* \leftarrow \mathsf{Eval}(\mathsf{msk}, x^*)$*. Otherwise, if* $b = 1$*, the challenger samples a random value* $y^* \xleftarrow{\$} \mathcal{Y}$*. Finally, the challenger returns* y^* *to* Adv*.*
- ***Guess:*** Adv *outputs a bit* $b' \in \{0,1\}$*.*

In all games described above, Adv *wins if* $b' = b$*. We also define the advantage of* Adv *in winning a game as* $|2 \cdot \Pr[\mathsf{Adv}\ \textit{wins}] - 1|$*, where the probability is over the internal coins of* Adv *and the challenger.*

Definition 5 (Statistically Collision-Resistant Constrained PRF, [8])**.** *A constrained pseudorandom function* $\mathsf{CPRF} = (\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{CEval})$ *is said to be statistically collision-resistant if*

$$\Pr\left[\exists x \neq x' \ \textit{s.t.}\ \mathsf{Eval}(\mathsf{msk}, x) = \mathsf{Eval}(\mathsf{msk}, x')\right] \leq \mathsf{negl}(\lambda)\ ,$$

where $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$*.*

[10] Equivalently, upon receiving an x such that $C(x) = 0$, the challenger, instead of aborting, can return $y \leftarrow \mathsf{CEval}(\mathsf{pp}, \mathsf{ck}_C, x)$

Remark 1. *The work of [8] shows that a 1-key selectively no-evaluation secure constrained PRF that is also statistically collision resistant (per Definition 5), can be upgraded to 1-key adaptively-secure constrained PRF by applying a hash function, modeled as a random oracle, over the output of the* Eval *and* CEval *algorithms.*

Definition 6 (Pseudorandomly-Constrained PRF, [33])**.** *A weakly/strongly pseudorandomly constrained PRF (PC-PRF) is a constrained pseudorandom function (per Definition 4) that supports a family of weak/strong pseudorandom predicates as constraints. If a constrained pseudorandom function admits* both *a family of weak/strong pseudorandom predicates* $\{F(k,\cdot) \ : \ k \in \mathcal{K}_\lambda\}$ *and their complement* $\{1 - F(k,\cdot) \ : \ k \in \mathcal{K}_\lambda\}$*, we refer to it as a full-domain pseudorandomly constrained PRF.*

3.3 Hardness Assumptions

For a ring R and a positive integer q, define $R_q = R/qR$ to be the quotient ring modulo q.

Definition 7 ((Decision) Ring-LWE Assumption, [76])**.** *Let* $\lambda \in \mathbb{N}$ *be a security parameter, and let* $m = m(\lambda) \in \mathbb{N}$, $q = q(\lambda) \in \mathbb{N}$. *Let* $R = R(\lambda)$ *be a ring defined over* $\mathbb{Z}$, *and* $\chi_e = \chi_e(\lambda)$ *be a distribution over* R. *The* $\mathsf{RLWE}(q, \chi_e, m)$ *assumption states that*

$$(T_i,\ T_i \cdot \tau + e_i \pmod q)_{i\in[m]} \approx_c (T_i,\ \mathcal{T}_i)_{i\in[m]}\,,$$

where $\tau \xleftarrow{\$} R_q$, *and* $T_i, \mathcal{T}_i \xleftarrow{\$} R_q$, *and* $e_i \xleftarrow{\$} \chi_e$, *for all* $i \in [m]$.

We also denote by $\mathsf{RLWE}(q, \mathcal{D}_T, \chi_\tau, \chi_e, m)$ *the same assumption when additionally* $\tau \xleftarrow{\$} \chi_\tau$ *and* $T_i \xleftarrow{\$} \mathcal{D}_T$ *for* $i \in [m]$.

Definition 8 ((Decision) Public-Power Ring-LWE Assumption, [1])**.** *Let* $\lambda \in \mathbb{N}$ *be a security parameter, and let* $m = m(\lambda) \in \mathbb{N}$, $p = p(\lambda) \in \mathbb{N}$. *Let* $R = R(\lambda)$ *be a ring defined over* $\mathbb{Z}$, *and* $\mathcal{D}_N = \mathcal{D}_N(\lambda)$ *and* $\chi_{e'} = \chi_{e'}(\lambda)$ *be distributions over* R. *The* $\mathsf{PP\text{-}RLWE}(p, \mathcal{D}_N, \chi_{e'}, m)$ *assumption states that*

$$\left(N,\ \{N^i \cdot \nu + e_i\}_{i\in[0,m]}\right) \approx_c \left(N,\ \{\mathcal{N}_i\}_{i\in[0,m]}\right),$$

where $N \xleftarrow{\$} \mathcal{D}_N$, $\nu \xleftarrow{\$} R_p$, *and* $e_i \xleftarrow{\$} \chi_{e'}$ *and* $\mathcal{N}_i \xleftarrow{\$} R_p$ *for all* $i \in [0, m]$.

We also denote by $\mathsf{PP\text{-}RLWE}(p, \mathcal{D}_N, \chi_\nu, \chi_{e'}, m)$ *the same assumption when additionally* $\nu \xleftarrow{\$} \chi_\nu$.

Remark 2. *We note that the assuming the hardness of* $\mathsf{PP\text{-}RLWE}(p, \mathcal{D}_N, \chi_\nu, \chi_{e'}, m)$ *implies assuming the hardness of* $\mathsf{PP\text{-}RLWE}(p, \mathcal{D}_N, \chi_{e'}, \chi_{e'}, m-1)$, *i.e., when the distribution of the secret is that of the error. The reduction is as follows: Let* Adv *be an adversary of* $\mathsf{PP\text{-}RLWE}(p, \mathcal{D}_N, \chi_{e'}, \chi_{e'}, m-1)$. *We build an adversary* Adv′ *attacking* $\mathsf{PP\text{-}RLWE}(p, \mathcal{D}_N, \chi_{e'}, m)$. *Upon receiving a tuple* $(N,\ \{\mathcal{N}_i\}_{i\in[0,m]})$, Adv′ *computes* $\mathcal{N}'_i := N^i \cdot \mathcal{N}_1 - \mathcal{N}_{i+1}$ *for* $i \in [m-1]$. *It additionally sets* $\mathcal{N}'_0 := \mathcal{N}_0$. Adv′ *then forwards* $(N, \{\mathcal{N}'_i\}_{i\in[0,m-1]})$ *to* Adv. *If* $\mathcal{N}_i \xleftarrow{\$} R_p$, *then each* $\mathcal{N}'_i$ *is a random element of* R_p. *Otherwise, if* $\mathcal{N}_i = N^i \cdot \nu + e_i$, *it holds that* $\mathcal{N}'_i = N^i \cdot e_1 + e_{i+1}$. *Therefore if* Adv *can attack* $\mathsf{PP\text{-}RLWE}(p, \mathcal{D}_N, \chi_{e'}, \chi_{e'}, m-1)$, *then* Adv′ *can attack* $\mathsf{PP\text{-}RLWE}(p, \mathcal{D}_N, \chi_{e'}, m)$.

Finally, note that assuming $\mathsf{PP\text{-}RLWE}(p, \mathcal{D}_N, \chi_{e'}, \chi_{e'}, m-1)$ *implies assuming* $\mathsf{RLWE}(p, \mathcal{D}_N, \chi_{e'}, \chi_{e'}, m-1)$.

3.4 (Public-Key) Pseudorandom Correlation Functions

Definition 9 (Reverse-Sampleable Correlation)**.** *Let* $\lambda \in \mathbb{N}$ *and* $1 \le \ell_0(\lambda), \ell_1(\lambda) \le \mathsf{poly}(\lambda)$ *be output-length functions. Let* $\mathcal{Y}$ *be a probabilistic algorithm that, on input* 1^λ, *returns a pair of outputs* $(y_0, y_1) \in \{0,1\}^{\ell_0(\lambda)} \times \{0,1\}^{\ell_1(\lambda)}$, *defining a correlation on the outputs. We say that* $\mathcal{Y}$ *defines a* reverse-sampleable *correlation if there exists a probabilistic polynomial time algorithm* RSample *which takes as input* 1^λ, $\sigma \in \{0,1\}$, *and* $y_\sigma \in \{0,1\}^{\ell_\sigma(\lambda)}$, *and outputs* $y_{1-\lambda}^{\ell_{1-\sigma}(\lambda)}$, *such that for all* $\sigma \in \{0,1\}$ *the following distributions are statistically close:*

$$\{(y_0, y_1) \colon (y_0, y_1) \xleftarrow{\$} \mathcal{Y}(1^\lambda)\}, \text{ and}$$
$$\{(y_0, y_1) \colon (y'_0, y'_1) \xleftarrow{\$} \mathcal{Y}(1^\lambda), y_\sigma \leftarrow y'_\sigma, y_{1-\sigma} \leftarrow \mathsf{RSample}(1^\lambda, \sigma, y_\sigma)\}\ .$$

Definition 10 (OT Correlation)**.** *A* (1-out-of-2, n-bit) OT correlation *is a pair* $((r_0, r_1), (\mathsf{b}, r_{\mathsf{b}}))$*, where* $r_0, r_1 \xleftarrow{\$} \{0,1\}^n$*, and* $\mathsf{b} \xleftarrow{\$} \{0,1\}$*. We denote by* $\mathsf{OT}(1^\lambda, n)$ *the functionality that outputs a random (1-out-of-2, n-bit) OT-correlated pair.*

Remark 3 (OT Correlation is Reverse-Sampleable)**.** A (1-out-of-2, n-bit) OT correlation is reverse-sampleable. $\mathsf{RSample}_{\mathsf{OT}_n}(1^\lambda, \sigma, y_\sigma)$: *If* $\sigma = 0$*, parse* y_σ *as* $y_\sigma = (r_0, r_1)$*, sample* $\mathsf{b} \xleftarrow{\$} \{0,1\}$*, and output* $(\mathsf{b}, r_{\mathsf{b}})$*; Otherwise if* $\sigma = 1$*, parse* y_σ *as* $y_\sigma = (\mathsf{b}, r)$*, sample* $r' \xleftarrow{\$} \{0,1\}^n$*, and output* $((1-\mathsf{b}) \cdot r + \mathsf{b} \cdot r', \mathsf{b} \cdot r + (1-\mathsf{b}) \cdot r')$.

Definition 11 (Public-Key Pseudorandom Correlation Function, [33])**.** *Let* $\lambda \in \mathbb{N}$ *be a security parameter, and* $\mathcal{Y}$ *be a reverse-sampleable correlation with output length functions* $\ell_0(\lambda), \ell_1(\lambda)$*. Let* $\lambda \leq n(\lambda) \leq \mathsf{poly}(\lambda)$ *be an input length function. A public-key pseudorandom correlation function (PK-PCF) consists of the following PT algorithms:*

- $\mathsf{PKPCF.Setup}(1^\lambda) \to \mathsf{pp}$*: A probabilistic algorithm that takes as input the security parameter* λ *and outputs the public parameters* pp*. We assume that all algorithms take* pp *as input.*
- $\mathsf{PKPCF.KeyGen}(\sigma) \to (\mathsf{pk}_\sigma, \mathsf{sk}_\sigma)$*: A probabilistic algorithm that takes as input a bit* $\sigma \in \{0,1\}$*, and outputs a public key* pk_σ *and a secret key* sk_σ.
- $\mathsf{PKPCF.KeyDer}(\sigma, \mathsf{sk}_\sigma, \mathsf{pk}_{1-\sigma}) \to k_\sigma$*: A deterministic algorithm that takes as input a bit* $\sigma \in \{0,1\}$*, a secret key* sk_σ *and a public key* $\mathsf{pk}_{1-\sigma}$*, and outputs an evaluation key* k_σ.
- $\mathsf{PKPCF.Eval}(\sigma, k_\sigma, x) \to y_\sigma$*: A deterministic algorithm that takes as input a bit* $\sigma \in \{0,1\}$*, an evaluation key* k_σ *and an input value* $x \in \{0,1\}^{n(\lambda)}$*, and outputs* $y_\sigma \in \{0,1\}^{\ell_\sigma(\lambda)}$.

We say that a PK-PCF for a correlation $\mathcal{Y}$ *is* (N, B, ϵ)*-secure, if the following two conditions hold:*

Pseudorandom* $\mathcal{Y}$*-Correlated Outputs. *For every non-uniform adversary* $\mathcal{A}$ *of size* $B(\lambda)$*, it holds that for all sufficiently large* λ*,*

$$\left|\Pr[\mathsf{Exp}^{\mathsf{pr}}_{\mathcal{A},0}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{pr}}_{\mathcal{A},1}(\lambda) = 1]\right| \leq \epsilon(\lambda),$$

where Adv *makes at most* $N(\lambda)$ *queries to the oracle* $\mathcal{O}^{\mathsf{pr}}_b$ *in* $\mathsf{Exp}^{\mathsf{pr}}_{\mathcal{A},b}$*, that is defined in Figure 2 for* $b \in \{0,1\}$.

Security. *For every* $\sigma \in \{0,1\}$ *and every non-uniform adversary* $\mathcal{A}$ *of size* $B(\lambda)$*, it holds that for all sufficiently large* λ*,*

$$\left|\Pr[\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},\sigma,0}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},\sigma,1}(\lambda) = 1]\right| \leq \epsilon(\lambda)$$

where Adv *makes at most* $N(\lambda)$ *queries to the oracle* $\mathcal{O}^{\mathsf{sec}}_b$ *in* $\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},\sigma,b}$*, that is defined in Figure 3 for* $b \in \{0,1\}$.

4 Our New Assumption: Secret-Power Ring-LWE

In this section, we describe out new assumption secret-power ring-LWE (SP-RLWE). At a high level, the assumption strengthens the famous ring-LWE assumption by allowing the secrets across samples to be correlated.

Definition 12 ((Decision) Secret-Power Ring-LWE (SP-RLWE) Assumption)**.** *Let* $\lambda \in \mathbb{N}$ *be a security parameter, and let* $m = m(\lambda) \in \mathbb{N}$*,* $q = q(\lambda) \in \mathbb{N}$*. Let* $R = R(\lambda)$ *be a ring defined over* $\mathbb{Z}$*, and* $\chi_\mu = \chi_\mu(\lambda)$ *and* $\chi_e = \chi_e(\lambda)$ *be distributions over* R*. The* $\mathsf{SP\text{-}RLWE}(q, \chi_\mu, \chi_e, m)$ *assumption states that*

$$\left(M,\ \ M \cdot \mu + e,\ \{M \cdot (1/\mu)^i + e_i\}_{i \in [m]}\right) \approx_c \left(M,\ \ \mathcal{M},\ \{\mathcal{M}_i\}_{i \in [m]}\right),$$

where $\mu \xleftarrow{\$} \chi_\mu$ *is an invertible element, and* $M, \mathcal{M}, \mathcal{M}_i \xleftarrow{\$} R_q$*, and* $e, e_i \xleftarrow{\$} \chi_e$*, for all* $i \in [m]$.

The assumption is inspired by the power ring-LWE assumption (Definition 8) introduced in [1], which allows similar correlations between the public "labels" in ring-LWE. Our assumption can be seen as the secret-power counterpart of their assumption.

We conjecture that the decision SP-RLWE assumption holds for all parameter choices admitted by plain ring-LWE. While this assumption requires time and cryptanalytic effort to build confidence, we are not aware of any attacks that can exploit this correlation.

Experiment Pseudorandom $\mathcal{Y}$-Correlated Outputs

$\underline{\mathsf{Exp}^{\mathsf{pr}}_{\mathcal{A},b}(\lambda)}$:

$\mathcal{Q} := \varnothing$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$

For $\sigma \in \{0,1\}$**:**

$(\mathsf{sk}_\sigma, \mathsf{pk}_\sigma) \leftarrow \mathsf{KeyGen}(\sigma)$

For $\sigma \in \{0,1\}$**:**

$k_\sigma \leftarrow \mathsf{KeyDer}(\sigma, \mathsf{sk}_\sigma, \mathsf{pk}_{1-\sigma})$

$b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{pr}}_b}(1^\lambda, \mathsf{pk}_0, \mathsf{pk}_1)$

Output b'

$\underline{\mathcal{O}^{\mathsf{pr}}_0}$:

$x \xleftarrow{\$} \{0,1\}^{n(\lambda)}$

If $(x, y_0, y_1) \in \mathcal{Q}$**,**

Return (y_0, y_1)

Else:

$(y_0, y_1) \xleftarrow{\$} \mathcal{Y}(1^\lambda)$

$\mathcal{Q} = \mathcal{Q} \cup \{(x, y_0, y_1)\}$

Return (x, y_0, y_1)

$\underline{\mathcal{O}^{\mathsf{pr}}_1}$:

$x \xleftarrow{\$} \{0,1\}^{n(\lambda)}$

For $\sigma \in \{0,1\}$**:**

$y_\sigma \leftarrow \mathsf{Eval}(\sigma, k_\sigma, x)$

Return (x, y_0, y_1)

Figure 2: Pseudorandom $\mathcal{Y}$-correlated outputs of a PK-PCF.

Experiment Public-Key PCF Security

$\underline{\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},\sigma,b}(\lambda)}$:

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$

For $\sigma \in \{0,1\}$**:**

$(\mathsf{sk}_\sigma, \mathsf{pk}_\sigma) \leftarrow \mathsf{KeyGen}(\sigma)$

For $\sigma \in \{0,1\}$**:**

$k_\sigma \leftarrow \mathsf{KeyDer}(\sigma, \mathsf{sk}_\sigma, \mathsf{pk}_{1-\sigma})$

$b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{sec}}_b}(1^\lambda, \mathsf{pk}_0, \mathsf{pk}_1, \sigma, \mathsf{sk}_\sigma)$

Output b'

$\underline{\mathcal{O}^{\mathsf{sec}}_0}$:

$x \xleftarrow{\$} \{0,1\}^{n(\lambda)}$

$y_{1-\sigma} \leftarrow \mathsf{Eval}(1-\sigma, k_{1-\sigma}, x)$

Return $(x, y_{1-\sigma})$

$\underline{\mathcal{O}^{\mathsf{sec}}_1}$:

$x \xleftarrow{\$} \{0,1\}^{n(\lambda)}$

$y_\sigma \leftarrow \mathsf{Eval}(\sigma, k_\sigma, x)$

$y_{1-\sigma} \leftarrow \mathsf{RS}(1^\lambda, \sigma, y_\sigma)$

Return $(x, y_{1-\sigma})$

Figure 3: Security of a PK-PCF. RS is the algorithm for reverse sampling $\mathcal{Y}$.

Discussion about Hardness. As seen above, the difference between our assumption and the standard ring-LWE assumption is that we provide multiple ring-LWE samples where the secrets are correlated in non-linear ways. Our conjecture for its security essentially relies on the fact that these samples seem very difficult to combine algebraically in any non-trivial way. Let us examine some basic approaches to attack this new assumption.

We begin with the observation that arbitrary correlations between the secrets would be problematic. For instance, suppose we had samples of the form

$$M, \quad M \cdot \mu + e, \ \{ \ M \cdot k_i \cdot \mu + e_i\}_i$$

where k_i is a small, public scalar, then clearly the attacker can recover equations on secret error terms over the integers, which would lead to a complete breach of security. In more detail, one can compute $k_i(M \cdot \mu + e) - (M \cdot k_i \cdot \mu + e_i)$, cancelling out the large masking term $M \cdot \mu$, to obtain $k_i \cdot e - e_i$ which is clearly insecure. Indeed, even if k_i is not explicitly given, an attacker could brute force try all values so

long as k_i is polynomially bounded in value.

Next, we observe that aside from low norm linear dependencies, we do not know any other attacks from the literature that exploit correlations between secrets. In particular, multiplicative dependencies between the secrets as used above seem unexploitable because there is no way to generate samples that "hit" the same masking term, using samples from our distribution. So long as the secrets are hidden from the adversary, which is clearly necessary, the only operations that an adversary can perform are generic operations which do not yield anything useful. For instance, multiplying two samples does not yield any meaningful term since the errors would blow up by such an operation. Analogously, one can try dividing out one sample with another but this leads to the same problem. Another point of consideration is the multiplicative order of our secret μ – since we give samples having successive powers of $\frac{1}{\mu}$ as the RLWE secrets for some polynomial number of samples, it is crucial that we do not hit the multiplicative order of the secret – if that happens then evidently the secret simply becomes identity and security is completely lost. However, since we are sampling μ as an invertible low-norm element, the multiplicative order is large[11] ($q^{\Omega(d)}$ where q denotes the modulus of the ring and d denotes its dimension) and this is not a problem.

In our attempts at cryptanalyzing the assumption, we did indeed try to attack by cancelling μ^{-1} terms via polynomial evaluation, but we could not achieve anything non-trivial this way. We assumed uniform distributions everywhere (up to the infinity-norm bound). In our concrete setting q is exponentially large, but we believe the assumption to plausibly hold in any parameter setting where the standard ring-LWE assumption holds (perhaps up to a tiny speedup of $\sqrt{m}$, m being the maximum degree of μ^{-1}, by adapting the generic attack of [35] on power-DDH to the ring-LWE setting, but in our context m is too small for this to make a difference: the largest value we use is $m = 9$.

We also observe that prior works have made philosophically similar assumptions, by using correlated labels as in the PP-RLWE assumption [1] or using correlated secrets as in the public-coin evasive LWE assumption [84] based on exactly the same reasoning. These assumptions have so far remained sound, although they are admittedly recent.

While we do not know of any meaningful attack strategies, we also do not know of any meaningful reduction establishing the hardness of our problem. It is also easy to see that a straightforward attempt at a reduction from a standard assumption like ring-LWE does not work. Evidently, given a set of ring-LWE samples, it appears infeasible to generate samples with correlated secrets without already knowing the secret in the first place. Placing our assumption on a more solid theoretical footing, either via non-trivial algorithms or via a reduction from a more standard problem is a fascinating direction for future research.

Weaker Version. Our construction in fact relies on a weaker form of the above assumption, which we state next.

Definition 13 ((Weak) Secret-Power Ring-LWE (SP-RLWE) Assumption)**.** *Let $\lambda \in \mathbb{N}$ be a security parameter, and let $m = m(\lambda) \in \mathbb{N}$, $q = q(\lambda) \in \mathbb{N}$. Let R be a ring defined over $\mathbb{Z}$, and $\chi_\mu = \chi_\mu(\lambda)$ and $\chi_e = \chi_e(\lambda)$ be distributions over R. The* SP-RLWE(q, χ_μ, χ_e, m) *assumption states that*

$$M \cdot \mu + e \pmod q \approx_c \mathcal{M} ,$$

given $(M \cdot (1/\mu)^i + e_i \pmod q)_{i\in[m]}$, where $\mu \xleftarrow{\$} \chi_\mu$ is an invertible element, and $M, \mathcal{M} \xleftarrow{\$} R_q$, and $e, e_i \xleftarrow{\$} \chi_e$, for all $i \in [m]$.

Again, we conjecture that the weak SP-RLWE assumption holds for all parameter choices admitted by plain ring-LWE. For our construction, we require q to be superpolynomially larger than $d \cdot B_e \cdot B_\mu$, where d is the degree of R, and B_e, B_μ are bounds on the magnitudes of the error and secret respectively.

[11] While there is no formal theorem stating this, we think this statement holds true given the arguments provided in the context of CRT decomposition and randomness heuristics in [65, 77].

5 Pseudorandom Correlation Function for OT from Secret-Power Ring-LWE

We now detail the description of our PCF for OT correlations from SP-RLWE.[12] We use the following theorem by [33].

Theorem 1 (PCFs for OT from Pseudorandomly-Constrained PRFs (PC-PRFs), [33])**.** *Let $F\colon \mathcal{K}_\lambda \times \mathcal{X}_\lambda \to \{0,1\}$ be a wPRF. Let $\overline{F}\colon \mathcal{K}_\lambda \times \mathcal{X}_\lambda \to \{0,1\}$ denote the function defined by $\overline{F}_k(\cdot) := 1 - F_k(\cdot)$. Let $\mathsf{CPRF} = (\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{CEval})$ be an adaptively-secure (weak) constrained PRF supporting as constraints $\{F_k(\cdot)\colon k \in \mathcal{K}_\lambda\} \cup \{\overline{F}_k(\cdot)\colon k \in \mathcal{K}_\lambda\}$ (this is called a pseudorandomly-constrained PRF, as per Definition 6). The construction presented in Section 5.2 is a (weak) PCF for oblivious transfer correlations.*

Therefore, to build a PCF for OT, it suffices to construct a CPRF supporting constraints $\{F_k(\cdot)\colon k \in \mathcal{K}_\lambda\} \cup \{\overline{F}_k(\cdot)\colon k \in \mathcal{K}_\lambda\}$, which we do next.

5.1 Efficient Pseudorandomly Constrained PRFs from Secret-Power Ring-LWE

Construction 1 (Efficient CPRF from Secret-Power Ring-LWE). Denote the security parameter by $\lambda \in \mathbb{N}$. Let $n = n(\lambda), \ell = \ell(\lambda)$, and $P : \{0,1\}^\ell \times \{0,1\}^* \to \{0,1\}^*$ be a circuit such that for all $\mathbf{x} \in \{0,1\}^\ell$, $P_{\mathbf{x}}(\cdot) := P(\mathbf{x}, \cdot)$ is computable as a polynomial of degree $t = \mathsf{poly}(\lambda)$, with each coefficient being bounded by $B_P = \mathsf{poly}(\lambda)$. Let $T_{P,s}$ denote the number of terms of degree s in the polynomial P, for $s \in [t]$. Define $\mathsf{BI}_P := \{P_{\mathbf{x}}(\mathbf{z}) \mid \mathbf{x} \in \{0,1\}^\ell, \mathbf{z} \in \{0,1\}^n\} \setminus \{0\}$ as the image set of P when computed over binary inputs of length n, excluding 0.

The following construction proposes a constrained PRF for the class of constraints $\mathcal{C}_P = \{C_{\mathbf{z}} \mid \mathbf{z} \in \{0,1\}^n\}$, where the circuit $C_{\mathbf{z}} : \{0,1\}^\ell \to \{0,1\}$ is defined as $C_{\mathbf{z}}(\mathbf{x}) = 0$ if $P_{\mathbf{x}}(\mathbf{z}) = 0$, and $C_{\mathbf{z}}(\mathbf{x}) = 1$ otherwise.

Parameters. Let R be a ring of degree $d = d(\lambda)$. Let $\chi_\mu = \chi_\mu(\lambda)$ and $\chi_e = \chi_e(\lambda)$ be distributions over R with magnitude bounded by $B_\mu = B_\mu(\lambda)$ and $B_e = B_e(\lambda)$, respectively. [13] Let $B = B(\lambda) = 2^{\omega(\log \lambda)} \cdot B_\mu$. Finally, choose $q = q(\lambda) = 2^{\omega(\log \lambda)} \cdot d \cdot B_e \cdot B_c$, where

$$B_c = \max_{j \in [t-1]} \left(B_P \cdot d^{j+1} \cdot B^{j+2} \sum_{s=j+1}^{t} T_{P,s} \cdot \binom{s}{j+1} \right),$$

such that $\mathsf{BI}_P \subseteq R_q^*$, where R_q^* is the set of all invertible elements of $R_q = R/qR$.

In the following, for any $x \in R_q$, denote by $\lfloor x \rceil_2$ the rounding of $x \cdot \frac{2}{q}$ to the nearest integer modulo 2. The public parameters pp is accessible to all algorithms.

- $\mathsf{CPRF.KeyGen}(1^\lambda, P)$:
  1. Sample $\mathbf{k} = (k_1, \ldots, k_n) \xleftarrow{\$} R_q^n$ such that $|k_i|_\infty \le B(\lambda)$ for all $i \in [n]$.
  2. Sample $\mu \xleftarrow{\$} \chi_\mu$, and $M \xleftarrow{\$} R_q$. // μ should be invertible.
  3. Set and output $\mathsf{pp} := P$, and $\mathsf{msk} := (\mathbf{k}, M, \mu)$.
- $\mathsf{CPRF.Eval}(\mathsf{msk}, \mathbf{x} \in \{0,1\}^\ell)$:
  1. Parse $\mathsf{msk} = (\mathbf{k}, M, \mu)$.
  2. Compute and output $y := \lfloor M\mu \cdot P_{\mathbf{x}}(\mathbf{k}/\mu) \pmod q \rceil_2$.
- $\mathsf{CPRF.Constrain}(\mathsf{msk}, \mathbf{z} \in \{0,1\}^n)$:

[12] PCFs are defined in Definition 14.

[13] In other words, for all $e \xleftarrow{\$} \chi_e$, it holds that $|e|_\infty \le B_e$. Similarly for χ_μ.

1. Parse $\mathsf{msk} = (\mathbf{k}, M, \mu)$.
2. Compute $\boldsymbol{\ell} := \mathbf{k} - \mu \cdot \mathbf{z}$.
3. For $i \in [t]$, compute $\mathcal{M}_i := M \cdot (1/\mu)^{i-1} + e_i$, where $e_i \xleftarrow{\$} \chi_e$.
4. Output $\mathsf{ck}_{\mathbf{z}} := (\mathbf{z}, \boldsymbol{\ell}, (\mathcal{M}_i)_{i\in[t]})$.

- $\mathsf{CPRF.CEval}(\mathsf{ck}_{\mathbf{z}}, \mathbf{x} \in \{0,1\}^{\ell})$:
  1. Parse $\mathsf{ck}_{\mathbf{z}} = (\mathbf{z}, \boldsymbol{\ell}, (\mathcal{M}_i)_{i\in[t]})$.
  2. Use $\mathsf{ck}_{\mathbf{z}}$ and $\mathbf{x}$ to compute $(c_1, \ldots, c_t)$ such that (see Lemma 2):

$$P_{\mathbf{x}}(\boldsymbol{\ell}/\mu + \mathbf{z}) = P_{\mathbf{x}}(\mathbf{z}) + \sum_{i=1}^{t} c_i (1/\mu)^i \pmod q \ . \tag{2}$$

  3. Compute and output $y' := \lfloor \sum_{i=1}^{t} c_i \cdot \mathcal{M}_i \pmod q \rceil_2$.

We first provide a lemma that will be handy in the analysis.

Lemma 2. *Let* CPRF *be the constrained PRF presented in Construction 1. Let* $\mathsf{ck}_{\mathbf{z}} \leftarrow \mathsf{CPRF.Constrain}(\mathsf{msk}, \mathbf{z})$*, where* $\mathsf{msk} \leftarrow \mathsf{CPRF.KeyGen}(1^{\lambda}, P)$ *and* $\mathbf{z} \in \{0,1\}^n$*. For any* $\mathbf{x} \in \{0,1\}^{\ell}$*, one can use* $\mathsf{ck}_{\mathbf{z}}$ *to efficiently compute* $(c_i)_{i\in[t]}$ *that satisfies Equation 4.*

Proof. To compute the coefficients, it is enough to write the symbolic version of the equation in terms of $Y = 1/\mu$ as follows, and solve for $(c_i)_{i\in[t]}$:

$$P_{\mathbf{x}}(\boldsymbol{\ell} \cdot Y + \mathbf{z}) = P_{\mathbf{x}}(\mathbf{z}) + \sum_{i=1}^{t} c_i \cdot Y^i \ .$$

□

Theorem 2 (CPRF from Secret-Power Ring-LWE)**.** *Assuming the hardness of decision secret-power ring-LWE* $\mathsf{SP\text{-}RLWE}_{q,\chi,t}$ *(Definition 13), construction 1 is a no-evaluation secure constrained pseudorandom function for the class of constraints* $\mathcal{C}_P$*.*

Proof. Note that for all $\mathbf{x} \in \{0,1\}^{\ell}$, we have

$$P_{\mathbf{x}}(\mathbf{k}/\mu) = P_{\mathbf{x}}(\boldsymbol{\ell}/\mu + \mathbf{z}) = P_{\mathbf{x}}(\mathbf{z}) + \frac{1}{\mu} \cdot Q(\mathbf{x}, \mathbf{z}, 1/\mu, \boldsymbol{\ell}) \ .$$

Let $Q(\mathbf{x}, \mathbf{z}, 1/\mu, \boldsymbol{\ell}) = \sum_{i=0}^{t-1} q_i (1/\mu)^i$. Renaming $c_i := q_{i-1}$ for $i \in [t]$, we have

$$M\mu \cdot P_{\mathbf{x}}(\mathbf{k}/\mu) = M\mu \cdot P_{\mathbf{x}}(\mathbf{z}) + M \cdot \sum_{i=1}^{t} c_i (1/\mu)^{i-1}$$

Therefore, for all $\mathbf{x} \in \{0,1\}^{\ell}, \mathbf{z} \in \{0,1\}^n$ and $(c_i)_{i\in[t]}$ that satisfy Equation 2, it holds that

$$\mathsf{Eval}(\mathsf{msk}, \mathbf{x}) = \lfloor M\mu \cdot P_{\mathbf{x}}(\mathbf{k}/\mu) \rceil_2 = \lfloor M\mu \cdot P_{\mathbf{x}}(\mathbf{z}) + M \cdot \sum_{i=1}^{t} c_i (1/\mu)^{i-1} \rceil_2 \ . \tag{3}$$

Correctness. In Equation 3, for any $\mathbf{x} \in \{0,1\}^{\ell}, \mathbf{z} \in \{0,1\}^n$ such that $P_{\mathbf{x}}(\mathbf{z}) = 0$, it holds that $\mathsf{Eval}(\mathsf{msk}, \mathbf{x}) = \lfloor M \cdot \sum_{i=1}^{t} c_i (1/\mu)^{i-1} \rceil_2$, where $(c_i)_{i\in[t]}$ are coefficients computed in CEval algorithm. On the other hand, we have

$$\mathsf{CEval}(\mathsf{ck}_{\mathbf{z}}, \mathbf{x}) = \lfloor \sum_{i=1}^{t} c_i \cdot \mathcal{M}_i \rceil_2 = \lfloor \sum_{i=1}^{t} c_i \cdot (M \cdot (1/\mu)^{i-1} + e_i) \rceil_2$$

$$= \lfloor M \cdot \sum_{i=1}^{t} c_i (1/\mu)^{i-1} + \sum_{i=1}^{t} c_i e_i \rceil_2 = \lfloor M \cdot \sum_{i=1}^{t} c_i (1/\mu)^{i-1} \rceil_2$$

where the last equality holds since $|\sum_{i=1}^{t} c_i e_i|_\infty \leq t \cdot d \cdot B_e \cdot B_c \ll q$ from the definition of q, where B_c denotes a bound on c_i for every $i \in [t]$. Note that one can compute a rough upper-bound of

$$B_c \leq \max_{j \in [t]} \left(B_P \cdot d^{j+1} \cdot B^{j+2} \sum_{s=j+1}^{t} T_{P,s} \cdot \binom{s}{j+1} \right)$$

by opening up Equation 4 and then using standard combinatorial analysis.

Pseudorandomness. We prove the no-evaluation selective security (as per Definition 4) of our CPRF in the following.

Let $\mathsf{CEval}_{\mathsf{pre\text{-}round}}$ denote the variant of CEval that omits the final rounding step. More precisely define

$$\mathsf{CEval}_{\mathsf{pre\text{-}round}}(\mathsf{ck}_{\mathbf{z}}, \mathbf{x}) := \sum_{i=1}^{t} c_i \cdot \mathcal{M}_i \pmod q,$$

where $(c_i)_{i \in [t]}$ are computed as in CEval. Consider the following sequence of hybrid games:

Hyb_0: This game is identical to the CPRF no-evaluation selective security game where the challenge query $\mathbf{x}^*$ is answered by returning the real PRF evaluation $y^* = \mathsf{Eval}(\mathsf{msk}, \mathbf{x}^*)$. The view of the adversary is $\mathsf{View}_0^{\mathsf{Adv}} = (\mathsf{pp}, \mathsf{ck}_{\mathbf{z}}, \mathbf{x}^*, y^*)$.

Hyb_1: In this game, we change the definition of the evaluation oracle that we use to answer the challenge query. After receiving the constraint $\mathbf{z}$, the challenger first computes $\mathsf{ck}_{\mathbf{z}}$ honestly as in Hyb_0 and returns it to the adversary. Upon receiving the challenge query $\mathbf{x}^*$, the challenger computes $y^* = \lfloor \mathsf{CEval}_{\mathsf{pre\text{-}round}}(\mathsf{ck}_{\mathbf{z}}, \mathbf{x}^*) + M\mu \cdot P_{\mathbf{x}^*}(\mathbf{z}) \rceil_2$ and returns this to the adversary.

Claim 1. Hyb_1 *remains statistically indistinguishable to* Hyb_0.

Proof. The claim follows by noting that for all $\mathbf{x} \in \{0,1\}^\ell, \mathbf{z} \in \{0,1\}^n$ and $(c_i)_{i \in [t]}$ that satisfy Equation 2, it holds that

$$= \mathsf{Eval}(\mathsf{msk}, \mathbf{x}) = \lfloor M\mu \cdot P_{\mathbf{x}}(\mathbf{k}/\mu) \rceil_2 = \lfloor M\mu \cdot P_{\mathbf{x}}(\mathbf{z}) + M \cdot \sum_{i=1}^{t} c_i (1/\mu)^{i-1} \rceil_2 \ .$$

□

Hyb_2: In this game, we aim to remove information about μ from the vector $\boldsymbol{\ell}$ in the constrained key. In order to do so, the challenger, after receiving the constraint $\mathbf{z}$ from the adversary, it first samples $\boldsymbol{\ell} \xleftarrow{\$} R_q^n$ such that $|\boldsymbol{\ell}_i|_\infty \leq B$ for $i \in [n]$. It then samples an invertible element $\mu \xleftarrow{\$} \chi_\mu$, and computes $\mathbf{k} = \boldsymbol{\ell} + \mu \cdot \mathbf{z}$. The challenger then sets $\mathsf{msk} = (\mathbf{k}, M, \mu)$ and $\mathsf{ck}_{\mathbf{z}} = (\mathbf{z}, \boldsymbol{\ell}, (\mathcal{M}_i)_{i \in [t]})$, where $M \xleftarrow{\$} R_q$ and $(\mathcal{M}_i = M \cdot (1/\mu)^{i-1} + e_i)_{i \in [t]}$. The rest of the game proceeds as in Hyb_1.

Claim 2. Hyb_2 *remains statistically indistinguishable to* Hyb_1.

Proof. The view of the adversary in Hyb_1 is $\mathsf{View}_1^{\mathsf{Adv}} = (\mathsf{pp}, \mathsf{ck}_{\mathbf{z}}, \mathbf{x}^*, y^*)$, where y^* is computed using the constrained key $\mathsf{ck}_{\mathbf{z}} = (\mathbf{z}, \boldsymbol{\ell} = \mathbf{k} - \mu \cdot \mathbf{z}, (\mathcal{M}_i)_{i \in [t]})$. Note that $\mathbf{k}$ is only used to compute the constrained key. Moreover, since for $i \in [n]$ it holds that $k_i \xleftarrow{\$} R_q$ s.t. $|k_i|_\infty \leq B$, and $|\mu|_\infty \leq B_\mu$ where $B = 2^{\omega(\log \lambda)} \cdot B_\mu$, the statistical distance between the distribution of $k_i - \mu \cdot z_i$, where $z_i \in \{0,1\}$, and that of k_i is negligible in λ. Therefore, $\mathbf{k}$ statistically hides $\mu \cdot \mathbf{z}$. Consequently, Hyb_2 remains statistically indistinguishable to Hyb_1.

□

Hyb_3: This game the challenger generates the constrained key as in Hyb_2, but differently, in order to answer the challenge query, it samples $r \xleftarrow{\$} R_q$, and returns $y^* = \lfloor \mathsf{CEval}_{\mathsf{pre\text{-}round}}(\mathsf{ck}_{\mathbf{z}}, \mathbf{x}^*) + r \cdot P_{\mathbf{x}^*}(\mathbf{z}) \rceil_2$.

Claim 3. *Assuming the hardness of secret-power ring-LWE* $\mathsf{SP\text{-}RLWE}_{q,\chi,t}$ *(Definition 13),* Hyb_3 *and* Hyb_2 *are computationally indistinguishable.*

Proof. Assume that a PPT adversary Adv succeeds in distinguishing between these two hybrids. We construct Adv' that breaks the $\mathsf{SP\text{-}RLWE}(q, \chi_\mu, \chi_e, t-1)$ assumption as follows. Given $(M, r) \in R_q$ and $t-1$ structured elements $(\mathcal{M}_i = M \cdot (1/\mu)^i + e_i)_{i\in[t-1]}$, the goal of Adv' is to decide whether $r \xleftarrow{\$} R_q$ or $r = M \cdot \mu + e$, for some $\mu \xleftarrow{\$} \chi_\mu$ and $e \xleftarrow{\$} \chi_e$. Adv' sets $\mathsf{pp} = (P, R_q)$. Upon receiving a constraint $\mathbf{z}$ from Adv, it sets $\mathsf{ck}_\mathbf{z} = (\mathbf{z}, \boldsymbol{\ell}, (\mathcal{M}_i)_{i\in[0,t-1]})$, where $\boldsymbol{\ell} \xleftarrow{\$} R_q^n$ such that $|\boldsymbol{\ell}_i|_\infty \le B$ for $i \in [n]$ and $\mathcal{M}_0 := M + e$ for some $e \xleftarrow{\$} \chi_e$. It then returns $\mathsf{ck}_\mathbf{z}$ to Adv. Finally, to answer the challenge query $\mathbf{x}^*$, it computes $y^* = \lfloor \mathsf{CEval}_{\mathsf{pre\text{-}round}}(\mathsf{ck}_\mathbf{z}, \mathbf{x}^*) + r \cdot P_{\mathbf{x}^*}(\mathbf{z}) \rceil_2$. Note that $|P_{\mathbf{x}^*}(\mathbf{z}) \cdot e|_\infty \le d \cdot B_e \cdot B_P \cdot \sum_{s=0}^{t} T_{P,s} \ll q$. [14] Therefore, if r is of the form $r = M \cdot \mu + e$, with overwhelming probability it holds that $y^* = \lfloor \mathsf{CEval}_{\mathsf{pre\text{-}round}}(\mathsf{ck}_\mathbf{z}, \mathbf{x}^*) + M\mu \cdot P_{\mathbf{x}^*}(\mathbf{z}) \rceil_2$ in which case Adv' simulates Hyb_2, and otherwise, if r is a random ring element, Adv' simulates Hyb_3. Therefore, distinguishing these two hybrids implies breaking the $SP - RLWE_{q,\chi_\mu,\chi_e,t}$ assumption. □

Hyb_4: This game is similar to Hyb_3, except that here the challenger answers the challenge query $\mathbf{x}^*$ by returning a random value $y^* \xleftarrow{\$} \{0,1\}$.

Claim 4. Hyb_4 *remains perfectly indistinguishable to* Hyb_3.

Proof. As q is chosen such that $P_{\mathbf{x}^*}(\mathbf{z}) \in \mathsf{BI}_P$ is invertible in R_q, the random element r randomizes y^* in Hyb_3, and therefore, the distribution of y^* in both hybrids is identical and equal to uniform. □

The rest of the proof proceeds by reversing the sequence of hybrid games while leaving the challenge query answered by a uniformly random value. □

Achieving full security. Similarly to prior works [8, 33, 81], our no-evaluation-secure CPRF can be made adaptively secure by applying a suitable correlation-robust hash function to the output of the Eval and CEval algorithms. More precisely, for any input $\mathbf{x}$, the new evaluation function outputs the hash of $X||\mathbf{x}$, where $X = \mathsf{Eval}(\mathsf{msk}, \mathbf{x})$ is the evaluation of the no-evaluation-secure CPRF above. We similarly modify the constrained evaluation by outputting the hash of $X||\mathbf{x}$, where $X = \mathsf{CEval}(\mathsf{ck}, \mathbf{x})$ is the constrained evaluation of the no-evaluation-secure CPRF above. The correctness remains intact. Furthermore, modeling the hash as a random oracle ensures that the outputs of the evaluation algorithm look perfectly random as long as an adversary cannot find collisions in the input of the hash, i.e., $\mathbf{x} \neq \mathbf{x}'$ such that $X'||\mathbf{x}' = X||\mathbf{x}$, which never happens. The proof is exactly the same as Theorem 4.21 of [8].

5.2 PCFs for OT Correlations from CPRFs

Definition 14 (Pseudorandom Correlation Functions (PCFs) [26])**.** *Let $\lambda \in \mathbb{N}$ be a security parameter, and $\mathcal{Y}$ be a reverse-sampleable correlation with output length functions $\ell_0(\lambda), \ell_1(\lambda)$. Let $\lambda \le n(\lambda) \le \mathsf{poly}(\lambda)$ be an input length function. A pseudorandom correlation function (PCF) consists of the following PT algorithms:*

- $\mathsf{PCF.KeyGen}(1^\lambda) \to (k_0, k_1)$*: A probabilistic algorithm that on input a security parameter λ outputs a pair of keys (k_0, k_1).*
- $\mathsf{PCF.Eval}(\sigma, k_\sigma, x) \to y_\sigma$*: A deterministic algorithm that takes as input a bit $\sigma \in \{0,1\}$, an evaluation key k_σ and an input value $x \in \{0,1\}^{n(\lambda)}$, and outputs $y_\sigma \in \{0,1\}^{\ell_\sigma(\lambda)}$.*

We say that a PCF for a correlation $\mathcal{Y}$ is (N, B, ϵ)-secure, if the following two conditions hold:

Pseudorandom $\mathcal{Y}$-Correlated Outputs. *For every non-uniform adversary $\mathcal{A}$ of size $B(\lambda)$, it holds that for all sufficiently large λ,*

$$\left|\Pr[\mathsf{Exp}^{\mathsf{pr}}_{\mathcal{A},0}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{pr}}_{\mathcal{A},1}(\lambda) = 1]\right| \le \epsilon(\lambda),$$

[14] This is because $P_{\mathbf{x}^*}(\mathbf{z})$ can be viewed as a scalar in R_q.

where Adv *makes at most* $N(\lambda)$ *queries to the oracle* $\mathcal{O}_b^{\mathsf{pr}}$ *in* $\mathsf{Exp}^{\mathsf{pr}}_{\mathcal{A},b}$*, that is defined in Figure 4 for* $b \in \{0,1\}$*.*

Security. *For every* $\sigma \in \{0,1\}$ *and every non-uniform adversary* $\mathcal{A}$ *of size* $B(\lambda)$*, it holds that for all sufficiently large* λ*,*

$$\left|\Pr[\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},\sigma,0}(\lambda)=1] - \Pr[\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},\sigma,1}(\lambda)=1]\right| \leq \epsilon(\lambda)$$

where Adv *makes at most* $N(\lambda)$ *queries to the oracle* $\mathcal{O}_b^{\mathsf{sec}}$ *in* $\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},\sigma,b}$*, that is defined in Figure 5 for* $b \in \{0,1\}$*.*

Experiment Pseudorandom $\mathcal{Y}$-Correlated Outputs

$\underline{\mathsf{Exp}^{\mathsf{pr}}_{\mathcal{A},b}(\lambda)}$:

$\mathcal{Q} := \varnothing$

$(k_0, k_1) \leftarrow \mathsf{KeyGen}(1^\lambda)$

$b' \leftarrow \mathcal{A}^{\mathcal{O}_b^{\mathsf{pr}}}(1^\lambda)$

Output b'

$\underline{\mathcal{O}_0^{\mathsf{pr}}}$:

$x \xleftarrow{\$} \{0,1\}^{n(\lambda)}$

If $(x, y_0, y_1) \in \mathcal{Q}$**,**

Return (y_0, y_1)

Else:

$(y_0, y_1) \xleftarrow{\$} \mathcal{Y}(1^\lambda)$

$\mathcal{Q} = \mathcal{Q} \cup \{(x, y_0, y_1)\}$

Return (x, y_0, y_1)

$\underline{\mathcal{O}_1^{\mathsf{pr}}}$:

$x \xleftarrow{\$} \{0,1\}^{n(\lambda)}$

For $\sigma \in \{0,1\}$**:**

$y_\sigma \leftarrow \mathsf{Eval}(\sigma, k_\sigma, x)$

Return (x, y_0, y_1)

Figure 4: Pseudorandom $\mathcal{Y}$-correlated outputs of a PCF.

Experiment PCF Security

$\underline{\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},\sigma,b}(\lambda)}$:

$(k_0, k_1) \leftarrow \mathsf{KeyGen}(\sigma)$

$b' \leftarrow \mathcal{A}^{\mathcal{O}_b^{\mathsf{sec}}}(1^\lambda, \sigma, k_\sigma)$

Output b'

$\underline{\mathcal{O}_0^{\mathsf{sec}}}$:

$x \xleftarrow{\$} \{0,1\}^{n(\lambda)}$

$y_{1-\sigma} \leftarrow \mathsf{Eval}(1-\sigma, k_{1-\sigma}, x)$

Return $(x, y_{1-\sigma})$

$\underline{\mathcal{O}_1^{\mathsf{sec}}}$:

$x \xleftarrow{\$} \{0,1\}^{n(\lambda)}$

$y_\sigma \leftarrow \mathsf{Eval}(\sigma, k_\sigma, x)$

$y_{1-\sigma} \leftarrow \mathsf{RS}(1^\lambda, \sigma, y_\sigma)$

Return $(x, y_{1-\sigma})$

Figure 5: Security of a PCF. RS is the algorithm for reverse sampling $\mathcal{Y}$.

Theorem 3 (PCFs for OT from Pseudorandomly-Constrained PRFs, [33])**.** *Let* $F\colon \mathcal{K}_\lambda \times \mathcal{X}_\lambda \to \{0,1\}$ *be a wPRF. Let* $\overline{F}\colon \mathcal{K}_\lambda \times \mathcal{X}_\lambda \to \{0,1\}$ *denote the function defined by* $\overline{F}_k(\cdot) := 1 - F_k(\cdot)$*. Let* $\mathsf{CPRF} = (\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{CEval})$ *be an adaptively-secure (weak) constrained PRF supporting as constraints* $\{F_k(\cdot)\colon k \in \mathcal{K}_\lambda\} \cup \{\overline{F}_k(\cdot)\colon k \in \mathcal{K}_\lambda\}$*. The construction presented in Section 5.2 is a (weak) PCF for oblivious transfer correlations:*

The following construction by [33] is a (weak) PCF for oblivious transfer correlations:

$\underline{\mathsf{PCF.KeyGen}(1^\lambda)}$:

1. *Run* $(\mathsf{pp}_0, \mathsf{msk}_0) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$.
2. *Run* $(\mathsf{pp}_1, \mathsf{msk}_1) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$.
3. *Set and output* $k_0 \leftarrow (\mathsf{msk}_0, \mathsf{msk}_1)$.
4. *Sample* $k \xleftarrow{\$} \mathcal{K}_\lambda$.
5. $\mathsf{ck}_0 \leftarrow \mathsf{Constrain}(\mathsf{msk}_0, F_k)$.
6. $\mathsf{ck}_1 \leftarrow \mathsf{Constrain}(\mathsf{msk}_1, \overline{F}_k)$.
7. *Set and output* $k_1 \leftarrow (k, \mathsf{ck}_0, \mathsf{ck}_1)$.
8. *Output* $(\mathsf{pp}_0, \mathsf{pp}_1)$.

$\underline{\mathsf{PCF.Eval}(1^\lambda, 0, k_0, x)}$:

1. *Parse* $k_0 = (\mathsf{msk}_0, \mathsf{msk}_1)$.
2. *Set* $r_0 \leftarrow \mathsf{Eval}(\mathsf{msk}_0, x)$.
3. *Set* $r_1 \leftarrow \mathsf{Eval}(\mathsf{msk}_1, x)$.
4. *Set and output* (r_0, r_1).

$\underline{\mathsf{PCF.Eval}(1^\lambda, 1, k_1, x)}$:

1. *Parse* $k_1 = (k, \mathsf{ck}_0, \mathsf{ck}_1)$.
2. *Compute* $\mathsf{b} \leftarrow F_k(x)$.
3. *Compute* $r \leftarrow \mathsf{CEval}(\mathsf{pp}_\mathsf{b}, \mathsf{ck}_\mathsf{b}, x)$.
4. *Set and output* (b, r).

Corollary 1 (PCFs for OT from secret-power ring-LWE)**.** *Let* $F\colon \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}$ *be a wPRF. Let* P *and* $\overline{P}$ *be two circuits as follows:*

- $P : \{0,1\}^\ell \times \{0,1\}^* \to \{0,1\}^*$*, such that for all* $\mathbf{x} \in \{0,1\}^\ell, \mathbf{z} \in \{0,1\}^n$ *it holds that (i)* $F_\mathbf{z}(\mathbf{x}) = 0 \iff P(\mathbf{x}, \mathbf{z}) = 0$*, and (ii)* $P_\mathbf{x}(\cdot) := P(\mathbf{x}, \cdot)$ *can be computed by a polynomial of degree* $t_P = \mathsf{poly}(\lambda)$*, with each coefficient bounded by* $B_P = \mathsf{poly}(\lambda)$*.*
- $\overline{P} : \{0,1\}^\ell \times \{0,1\}^* \to \{0,1\}^*$ *such that for all* $\mathbf{x} \in \{0,1\}^\ell, \mathbf{z} \in \{0,1\}^n$ *it holds that (i)* $F_\mathbf{z}(\mathbf{x}) = 1 \iff \overline{P}(\mathbf{x}, \mathbf{z}) = 0$*, and (ii)* $\overline{P}_\mathbf{x}(\cdot) := \overline{P}(\mathbf{x}, \cdot)$ *can be computed by a polynomial of degree* $t_{\overline{P}} = \mathsf{poly}(\lambda)$*, with each coefficient bounded by* $B_{\overline{P}} = \mathsf{poly}(\lambda)$*.* [15]

Define $t_\mathsf{max} = \max(t_P, t_{\overline{P}})$*. Let* $\mathsf{CPRF} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{CEval})$ *be the constrained PRF of Construction 1 that supports* P *and* $\overline{P}$ *as constraints. Finally, let* $\mathsf{H} : \{0,1\}^{n+d} \to \{0,1\}^{n'}$ *be a random oracle for some* $n' \in \mathbb{N}$*. The following is a secure PCF for OT correlations assuming*

- F *is a (weak) pseudorandomness function, and*
- CPRF *is adaptively secure, i.e.,* $\mathsf{SP\text{-}RLWE}(q, \chi_\mu, \chi_e, t_\mathsf{max})$ *is hard.*

$\underline{\mathsf{PCF.KeyGen}(1^\lambda)}$:

1. *Sample* $\mathbf{k} = (k_1, \ldots, k_n) \xleftarrow{\$} R_q^n$ *such that* $|k_i|_\infty \le B(\lambda)$ *for all* $i \in [n]$*.*
2. *Sample an invertible element* $\mu \xleftarrow{\$} \chi_\mu$*, and* $M \xleftarrow{\$} R_q$*.*
3. *Sample* $\mathbf{z} \xleftarrow{\$} \{0,1\}^n$*.*
4. *Compute* $\boldsymbol{\ell} := \mathbf{k} - \mu \cdot \mathbf{z}$*.*
5. *For* $i \in [t_\mathsf{max}]$*, compute* $\mathcal{M}_i := M \cdot (1/\mu)^{i-1} + e_i$*, where* $e_i \xleftarrow{\$} \chi_e$*.*
6. *Set* $k_0 := (\mathbf{k}, M, \mu)$*, and* $k_1 := (\mathbf{z}, \boldsymbol{\ell}, (\mathcal{M}_i)_{i \in [t_\mathsf{max}]})$*.*
7. *Set* $\mathsf{pp} := (P, \overline{P}, (\mathcal{M}_i)_{i \in [t_\mathsf{max}]})$*.*
8. *Output* pp *and* (k_0, k_1)*.*

[15] We sometimes denote P and t_P as $P^{(0)}$ and t_0, and $\overline{P}$ and $t_{\overline{P}}$ as $P^{(1)}$ and t_1.

$\mathsf{PCF.Eval}(0, k_0, \mathbf{x})$:

1. *Parse* $k_0 = (\mathbf{k}, M, \mu)$.
2. *For* $\mathsf{b} \in \{0,1\}$:
   (a) $X_\mathsf{b} := \lfloor M\mu \cdot P^{(\mathsf{b})}_{\mathbf{x}}(\mathbf{k}/\mu) \rceil_2$.
   (b) *Compute* $r_\mathsf{b} = \mathsf{H}(X_\mathsf{b} || \mathbf{x}_\mathsf{b})$.
3. *Output* $y_0 := (r_0, r_1)$.

$\mathsf{PCF.Eval}(1, k_1, \mathbf{x})$:

1. *Parse* $k_1 = (\mathbf{z}, \boldsymbol{\ell}, (\mathcal{M}_j)_{j \in [t_{\mathsf{max}}]})$.
2. *Compute* $\mathsf{b} := F_\mathbf{z}(\mathbf{x})$.
3. *Use* $\boldsymbol{\ell}$ *and* $\mathbf{x}$ *to compute* $(c_1, \ldots, c_{t_\mathsf{b}})$ *s.t.*
$$P^{(\mathsf{b})}_{\mathbf{x}}(\boldsymbol{\ell}/\mu + \mathbf{z}) = P^{(\mathsf{b})}_{\mathbf{x}}(\mathbf{z}) + \sum_{i=1}^{t_\mathsf{b}} c_i (1/\mu)^i.$$
4. *Compute* $X' := \lfloor \sum_{i=1}^{t_\mathsf{b}} c_i \cdot \mathcal{M}_i \rceil_2$.
5. *Compute* $y' := \mathsf{H}(X' || \mathbf{x})$.
6. *Output* $r := (\mathsf{b}, y')$.

6 Goldreich-Applebaum-Raykov Weak PRF

In this section, we describe our choice of the weak-PRF to instantiate Construction 1, show how to represent this as a polynomial and calculate some of its required parameters such as degree and the number of terms for each power.

XOR-MAJ Predicate. Let κ, ι be positive integers. The predicate $\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota : \{0,1\}^{\kappa+\iota} \to \{0,1\}$ is defined as the following Boolean function

$$\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota(b_1, \ldots, b_{\kappa+\iota}) = \mathsf{XOR}(b_1, \ldots, b_\kappa) \oplus \mathsf{MAJ}(b_{\kappa+1}, \ldots, b_{\kappa+\iota}),$$

where $\mathsf{XOR}(b_1, \ldots, b_\kappa) = \bigoplus_{i=1}^{\kappa} b_i$, and $\mathsf{MAJ}(b_{\kappa+1}, \ldots, b_{\kappa+\iota})$ outputs 1 if $\sum_{i=1}^{\iota} b_{\kappa+i} \geq \iota/2$, and 0, otherwise.

XOR-MAJ GAR-wPRF. Let κ and ι be some positive integers. Let $\ell = (\kappa + \iota) \cdot \lceil \log n \rceil$. The following function $F : \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}$ is a candidate weak PRF, proposed by [7, 49],

$$F(\mathbf{z}, \mathbf{x}) = \mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota(\mathbf{z}[x_1], \ldots, \mathbf{z}[x_{\kappa+\iota}]),$$

where $\mathbf{x} = \{x_1, \ldots, x_{\kappa+\iota}\}$ denotes the input and is viewed a set of $\kappa + \iota$ many bit-strings mapping to locations inside $\mathbf{z}$ (which denotes the key), with each bit-string $x_i \in \{0,1\}^{\lceil \log n \rceil}$We refer to Section 9 for an analysis on n, κ, ι. This candidate is referred to as the *Goldreich-Applebaum-Raykov wPRF with XOR-MAJ predicate* [33, 38]. In the following, we simply denote it by GAR-wPRF.

GAR-wPRF as a Constraint in Construction 1. In order to use the GAR-wPRF as a constraint in the CPRF of Construction 1 when working over a ring R, we need to find a polynomial $P_{\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota} : \{0,1\}^\ell \times R^n \to R$ such that for all $\mathbf{x} \in \{0,1\}^\ell, \mathbf{z} \in \{0,1\}^n$ it holds that:

$$P_{\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota}(\mathbf{x}, \mathbf{z}) = 0 \iff \mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota(\mathbf{z}[x_1], \ldots, \mathbf{z}[x_{\kappa+\iota}]) = 0$$ [16]

In order to define the predicate $P_{\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota}(\mathbf{x}, \mathbf{z})$, we first define the following four predicates that represent $\mathsf{XOR}(b_1, \ldots, b_\kappa) \stackrel{?}{=} \sigma$ and $\mathsf{MAJ}(b'_1, \ldots, b'_\iota) \stackrel{?}{=} \sigma$, for $\sigma \in \{0,1\}$:

- To represent $\mathsf{XOR}_\kappa(b_1, \ldots, b_\kappa) \stackrel{?}{=} 0$ **:**

$$P_{\mathsf{XOR}_\kappa}(b_1, \ldots, b_\kappa) := \prod_{j \in \mathsf{Even}_\kappa} \left(\sum_{i=1}^{\kappa} b_i - j \right),$$

[16] To evaluate the polynomial $P_{\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota}$ on a bit-vector $\mathbf{z}$, we map each bit $\mathbf{z}[i]$ to the corresponding ring element - the additive identity of R if $\mathbf{z}[i] = 0$ and the multiplicative identity otherwise.

where $\mathsf{Even}_\kappa = \{0, 2, \ldots, 2 \cdot \lfloor \kappa/2 \rfloor\}$ denotes the set of all even integers less than or equal to κ. The function P_{XOR_κ} therefore evaluates to 0 if an even number of input bits are 1, and to a non-zero value, otherwise.

- To represent $\mathsf{MAJ}_\iota(b'_1, \ldots, b'_\iota) \stackrel{?}{=} 0$:

$$P_{\mathsf{MAJ}_\iota}(b'_1, \ldots, b'_\iota) := \prod_{j \in \mathsf{Small}_\iota} \left(\sum_{i=1}^{\iota} b'_i - j \right)$$

where $\mathsf{Small}_\iota = \{0, 1, \ldots, \lfloor \iota/2 \rfloor\}$. Consequently, P_{MAJ_ι} evaluates to 0 if the minority of the the input bits are 1.

- To represent $\mathsf{XOR}_\kappa(b_1, \ldots, b_\kappa) \stackrel{?}{=} 1$:

$$P_{\neg\mathsf{XOR}_\kappa}(b_1, \ldots, b_\kappa) := \prod_{j \in \mathsf{Odd}_\kappa} \left(\sum_{i=1}^{\kappa} b_i - j \right),$$

where $\mathsf{Odd}_\kappa = \{1, 3, \ldots, \kappa_{\text{odd}}\}$ denotes the set of all odd integers less than or equal to κ, and $\kappa_{\text{odd}} = \kappa$ if κ is odd and $\kappa_{\text{odd}} = \kappa - 1$ if κ is even. Therefore, $P_{\neg\mathsf{XOR}_\kappa}$ evaluates to 0 if xoring inputs yields 1, and to a non-zero value, otherwise.

- To represent $\mathsf{MAJ}_\iota(b'_1, \ldots, b'_\iota) \stackrel{?}{=} 1$:

$$P_{\neg\mathsf{MAJ}_\iota}(b'_1, \ldots, b'_\iota) := \prod_{j \in \mathsf{Big}_\iota} \left(\sum_{i=1}^{\iota} b'_i - j \right),$$

where $\mathsf{Big}_\iota = \{\lfloor \iota/2 \rfloor + 1, \ldots, \iota\}$. Consequently, $P_{\neg\mathsf{MAJ}_\iota}$ evaluates to 0 if the majority of the input bits are 1.

Putting everything together, we can finally define the polynomial $P_{\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota}$ as follows

$$\begin{aligned} P_{\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota}(\mathbf{x}, \mathbf{z}) = & P_{\mathsf{XOR}_\kappa}(\mathbf{z}[x_1], \ldots, \mathbf{z}[x_\kappa]) \times P_{\neg\mathsf{MAJ}_\iota}(\mathbf{z}[x_{\kappa+1}], \ldots, \mathbf{z}[x_{\kappa+\iota}]) + \\ & P_{\neg\mathsf{XOR}_\kappa}(\mathbf{z}[x_1], \ldots, \mathbf{z}[x_\kappa]) \times P_{\mathsf{MAJ}_\iota}(\mathbf{z}[x_{\kappa+1}], \ldots, \mathbf{z}[x_{\kappa+\iota}]). \end{aligned}$$

The complement $P_{\neg\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota}$, such that $P_{\neg\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota}(\mathbf{z}, \mathbf{x}) = 0$ iff $P_{\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota}(\mathbf{z}, \mathbf{x}) \neq 0$ (or equivalently, when $F(\mathbf{z}, \mathbf{x}) = 1$), can also be defined as follows:

$$\begin{aligned} P_{\neg\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota}(\mathbf{x}, \mathbf{z}) = & P_{\mathsf{XOR}_\kappa}(\mathbf{z}[x_1], \ldots, \mathbf{z}[x_\kappa]) \times P_{\mathsf{MAJ}_\iota}(\mathbf{z}[x_{\kappa+1}, \ldots, \mathbf{z}[x_{\kappa+\iota}]) + \\ & P_{\neg\mathsf{XOR}_\kappa}(\mathbf{z}[x_1], \ldots, \mathbf{z}[x_\kappa]) \times P_{\neg\mathsf{MAJ}_\iota}(\mathbf{z}[x_{\kappa+1}], \ldots, \mathbf{z}[x_{\kappa+\iota}]). \end{aligned}$$

6.0.1 Analysis.

The parameters given below are needed to estimate the size of the modulus q in the instantiation of our CPRF in Construction 1.

Degree of the Polynomials. The degree of the polynomials are as follows:

- Degree of $P_{\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota}$:
$$\max\left(|\mathsf{Even}_\kappa| + |\mathsf{Big}_\iota|, |\mathsf{Odd}_\kappa| + |\mathsf{Small}_\iota|\right),$$
- Degree of $P_{\neg\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota}$:
$$\max\left(|\mathsf{Even}_\kappa| + |\mathsf{Small}_\iota|, |\mathsf{Odd}_\kappa| + |\mathsf{Big}_\iota|\right),$$

where $|\mathsf{Even}_\kappa|, |\mathsf{Odd}_\kappa| \leq \lfloor \kappa/2 \rfloor + 1$, and $|\mathsf{Small}_\iota|, |\mathsf{Big}_\iota| \leq \lfloor \iota/2 \rfloor + 1$.

Bound on the Coefficients. The upper-bound $B_{P_{\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota}}$ on the magnitude of the coefficients of $P_{\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota}$ can be computed as

$$B_{P_{\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota}} = \max_{s\in\{0,\ldots,|\mathsf{Even}_\kappa|+|\mathsf{Big}_\iota|\}} C_s^{|\mathsf{Even}_\kappa|,|\mathsf{Big}_\iota|} + \max_{s\in\{0,\ldots,|\mathsf{Odd}_\kappa|+|\mathsf{Small}_\iota|\}} C_s^{|\mathsf{Odd}_\kappa|,|\mathsf{Small}_\iota|},$$

where

$$C_s^{S_1,S_2} = \max_{\substack{p_1,\ldots,p_\kappa\geq 0\\ q_1,\ldots,q_\iota\geq 0\\ 0<p_1+\cdots+p_\kappa\leq|S_1|\\ 0<q_1+\cdots+q_\iota\leq|S_2|\\ p_1+\cdots+p_\kappa+q_1+\cdots+q_\iota=s}} \left(E^{S_1}_{p_1,\ldots,p_\kappa}\cdot E^{S_2}_{q_1,\ldots,q_\iota}\right),$$

with

$$E^S_{a_1,\ldots,a_m} = \begin{cases} D^S_{a_1,\ldots,a_m} & \text{if } a_1+\cdots+a_m\neq 0\\ \prod_{u\in S}(-u) & \text{otherwise}\end{cases},$$

where

$$D^S_{a_1,\ldots,a_m} = \sum_{\substack{r_1,\ldots,r_{a_1+\cdots+a_m}\in S\\ r_1\neq\cdots\neq r_{a_1+\cdots+a_m}}} \left(\left(\prod_{u\in S\setminus\{r_1,\ldots,r_{a_1+\cdots+a_m}\}}(-u)\right)\cdot\frac{(a_1+\cdots+a_m)!}{a_1!\cdot\ldots\cdot a_m!}\right).$$

We can also compute an upper-bound on the number of terms $T^s_{P_{\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota}}$ of degree s, for every $s\in\{0,\ldots,t_P\}$ (t_P denotes the degree of $P_{\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota}$) in the polynomial $P_{\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota}$ as

$$T^s_{P_{\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota}} \leq \sum_{u=0}^{s}\left(\binom{|\mathsf{Even}_\kappa|}{u}^*\cdot\binom{|\mathsf{Big}_\iota|}{s-u}^* + \binom{|\mathsf{Odd}_\kappa|}{u}^*\cdot\binom{|\mathsf{Small}_\iota|}{s-u}^*\right)\cdot\kappa\cdot\iota,$$

where the function $\binom{x}{y}^*$ is defined as

$$\binom{x}{y}^* = \begin{cases}\binom{x}{y} & \text{if } x\geq y\\ 0 & \text{otherwise}\end{cases}.$$

Similarly, we can estimate $B_{\neg\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota}$ and $T^s_{P_{\neg\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota}}$ by interchanging the occurrence of Small_ι with Big_ι in the above formulas for $B_{\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota}$ and $T^s_{P_{\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota}}$ respectively.

7 Public Key PCF from Secret-Power and Public-Power Ring-LWE

The following is an application of a method from a previous work [1] and we covered it in the technical overview in the section "Compiling to Public-Key PCFs". In short: the setup must distribute keys k, $\boldsymbol{\ell}$ that form shares of $\mu\cdot\mathbf{z}$, where μ is known to one party and $\mathbf{z}$ to the other. This is a vector-OLE of length $O(n)$. We used the methodology from [1] which shows how, under the public-power ring-LWE assumption, two parties can non-interactively compute such vector-OLEs using public keys of size $O(\sqrt{n})$ (hence these $\sqrt{n}$ terms that show up in the construction). We make an essentially black-box use of their methodology, except that the choice of parameters for their approach influences the ring-LWE parameters (modulus, dimension, noise bounds) of the rest of the construction. In particular, the choice of p is influenced by the previous work, as well as the sizes of the public keys (since they contain $O(\sqrt{n})$ many elements over the ring R_p).

Construction 2 (PK-PCF for OT). Let λ be a security parameter. Let $n=n(\lambda)$. Let R be a ring of degree $d=d(\lambda)$.

wPRF Parameters. Let $F\colon\{0,1\}^n\times\{0,1\}^\ell\to\{0,1\}$ be a wPRF. Let P and $\overline{P}$ be two circuits as follows:

- $P:\{0,1\}^\ell\times\{0,1\}^*\to\{0,1\}^*$, such that for all $\mathbf{x}\in\{0,1\}^\ell,\mathbf{z}\in\{0,1\}^n$ it holds that (i) $F_{\mathbf{z}}(\mathbf{x})=0\iff P(\mathbf{x},\mathbf{z})=0$, and (ii) $P_{\mathbf{x}}(\cdot):=P(\mathbf{x},\cdot)$ can be computed by a polynomial of degree $t_P=\mathsf{poly}(\lambda)$, with each coefficient bounded by $B_P=\mathsf{poly}(\lambda)$.

- $\overline{P} : \{0,1\}^\ell \times \{0,1\}^* \to \{0,1\}^*$ such that for all $\mathbf{x} \in \{0,1\}^\ell, \mathbf{z} \in \{0,1\}^n$ it holds that (i) $F_{\mathbf{z}}(\mathbf{x}) = 1 \iff \overline{P}(\mathbf{x}, \mathbf{z}) = 0$, and (ii) $\overline{P}_{\mathbf{x}}(\cdot) := \overline{P}(\mathbf{x}, \cdot)$ can be computed by a polynomial of degree $t_{\overline{P}} = \mathsf{poly}(\lambda)$, with each coefficient bounded by $B_{\overline{P}} = \mathsf{poly}(\lambda)$.

In the following, we sometimes denote P and t_P as $P^{(0)}$ and t_0, and $\overline{P}$ and $t_{\overline{P}}$ as $P^{(1)}$ and t_1.

Let $T_{P,s}$, and $T_{\overline{P},s}$ denote the number of terms of degree s in the polynomial P for $s \in [t_P]$, and $\overline{P}$ for $s \in [t_{\overline{P}}]$, respectively.

Define $\mathsf{BI}_P := \{P_{\mathbf{x}}(\mathbf{z}) \mid \mathbf{x} \in \{0,1\}^\ell, \mathbf{z} \in \{0,1\}^n\} \setminus \{0\}$, and $\mathsf{BI}_{\overline{P}}$ similarly.

Constrained PRF Parameters. Let $\chi_\mu = \chi_\mu(\lambda)$ and $\chi_e = \chi_e(\lambda)$ be distributions over R with magnitude bounded by $B_\mu = B_\mu(\lambda)$ and $B_e = B_e(\lambda)$, respectively. Let $B = B(\lambda) = 2^{\omega(\log \lambda)} \cdot B_\mu$ be a bound on the magnitude of the keys.

Let $t_{\mathsf{max}} = \max(t_P, t_{\overline{P}})$, and $B_{c_{\mathsf{max}}} = \max(B_{c_P}, B_{c_{\overline{P}}})$, where

$$B_{c_P} = \max_{j \in [t_P - 1]} \left(B_P \cdot d^{j+1} \cdot B^{j+2} \sum_{s=j+1}^{t_P} T_{P,s} \cdot \binom{s}{j+1} \right),$$

and similarly for $B_{c_{\overline{P}}}$.

Choose $q = q(\lambda) = 2^{\omega(\log \lambda)} \cdot t_{\mathsf{max}} \cdot d \cdot B_e \cdot B_{c_{\mathsf{max}}}$, such that $\mathsf{BI}_P \cup \mathsf{BI}_{\overline{P}} \subseteq R_q^*$, where R_q^* is the set of all invertible elements of $R_q = R/qR$.

Let $\mathsf{H} : \{0,1\}^{n+d} \to \{0,1\}^{n'}$ be a random oracle for some $n' \in \mathbb{N}$.

Public-Key Setup Parameters. Let $p = p(\lambda) = 2^{\omega(\log \lambda)} \cdot B \cdot (dB_{e'}^2 + \sqrt{n} B_{e'})$ be an integer. Let $\mathcal{D}_N = \mathcal{D}_N(\lambda)$, and $\chi_{e'} = \chi_{e'}(\lambda)$ be distributions over R_p, where the magnitude of $\chi_{e'}$ is bounded by $B_{e'} = B_{e'}(\lambda)$.

Algorithms:

- $\mathsf{PKPCF.Setup}(1^\lambda)$:
  1. Sample $N \xleftarrow{\$} \mathcal{D}_N$ and set $\mathsf{pp} := (N, P, \overline{P})$.

- $\mathsf{PKPCF.KeyGen}(0)$:
  1. Sample $M \xleftarrow{\$} R_q$, and $\mu \xleftarrow{\$} \chi_\mu$. // Sampling CPRF master secret keys.
  2. Sample $\nu \xleftarrow{\$} R_p$.
  3. For $j \in [0, 2\sqrt{n}] \setminus \{\sqrt{n} + 1\}$:
     Compute $\mathcal{N}_j := N^j \nu + e'_j \pmod p$, where $e'_j \xleftarrow{\$} \chi_{e'}$.
  4. For $j = \sqrt{n} + 1$:
     Compute $\mathcal{N}_j := N^j \nu + e'_j + \lfloor p/B \rceil \mu \pmod p$, where $e'_j \xleftarrow{\$} \chi_{e'}$.
  5. For $i \in [t_{\mathsf{max}}]$:
     Compute $\mathcal{M}_i := M \cdot (1/\mu)^{i-1} + e_i \pmod q$, where $e_i \xleftarrow{\$} \chi_e$.
  6. Set $\mathsf{sk}_0 := (M, \mu, \nu)$.
  7. Set $\mathsf{pk}_0 := ((\mathcal{M}_i)_{i \in [t_{\mathsf{max}}]}, (\mathcal{N}_j)_{j \in [0, 2\sqrt{n}]})$.
  8. Output $(\mathsf{sk}_0, \mathsf{pk}_0)$.

- $\mathsf{PKPCF.KeyGen}(1)$:
  1. Sample $\mathbf{z} \xleftarrow{\$} \{0,1\}^n$. // Sampling a key for the wPRF.
  2. Let $\mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_{\sqrt{n}})$, where $\mathbf{z}_i \in \{0,1\}^{\sqrt{n}}$ for $i \in [\sqrt{n}]$.

3. For $i \in [\sqrt{n}]$:
   (a) $u_0^i, u_1^i \xleftarrow{\$} \chi_{e'}$.
   (b) $v_i := \langle (u_0^i, u_1^i, \mathbf{z}_i) \ , \ (1, N, N^2, \ldots, N^{\sqrt{n}+1}) \rangle \pmod p$.
4. $\mathsf{sk}_1 := (\mathbf{z}, (u_0^i, u_1^i)_{i \in [\sqrt{n}]})$.
5. $\mathsf{pk}_1 := (v_i)_{i \in [\sqrt{n}]}$.
6. Output $(\mathsf{sk}_1, \mathsf{pk}_1)$.

- $\mathsf{PKPCF.KeyDer}(0, \mathsf{sk}_0, \mathsf{pk}_1)$:
  1. Parse $\mathsf{sk}_0 = (M, \mu, \nu)$.
  2. Parse $\mathsf{pk}_1 = (v_i)_{i \in [\sqrt{n}]}$.
  3. For $i \in [\sqrt{n}]$:
     (a) For $j \in [\sqrt{n}]$:
        i. Compute $s_{i_j}^{(0)} := v_i \cdot N^{\sqrt{n}-j} \cdot \nu \pmod p$.
        ii. View $\lfloor s_{i_j}^{(0)} \rceil_B$ as an element k_{i_j} of R_q. [17]
     (b) Let $\mathbf{k}_i := (k_{i_1}, \ldots, k_{i_{\sqrt{n}}})$
  4. Set $\mathbf{k} := (\mathbf{k}_1, \ldots, \mathbf{k}_{\sqrt{n}})$.
  5. Output $k_0 := (\mathbf{k}, M, \mu)$.

- $\mathsf{PKPCF.KeyDer}(1, \mathsf{sk}_1, \mathsf{pk}_0)$:
  1. Parse $\mathsf{sk}_1 = (\mathbf{z}, (u_0^i, u_1^i)_{i \in [\sqrt{n}]})$.
  2. Parse $\mathsf{pk}_0 = ((\mathcal{M}_i)_{i \in [t_{\mathsf{max}}]}, (\mathcal{N}_j)_{j \in [0, 2\sqrt{n}]})$.
  3. For $i \in [\sqrt{n}]$:
     (a) For $j \in [\sqrt{n}]$:
        i. $s_{i_j}^{(1)} := \langle (u_0^i, u_1^i, \mathbf{z}_i) \ , \ (\mathcal{N}_{\sqrt{n}-j}, \mathcal{N}_{\sqrt{n}-j+1}, \ldots, \mathcal{N}_{2\sqrt{n}-j+1}) \rangle \pmod p$.
        ii. View $\lfloor s_{i_j}^{(1)} \rceil_B$ as an element ℓ_{i_j} of R_q.
     (b) Let $\boldsymbol{\ell}_i = (\ell_{i_1}, \ldots, \ell_{i_{\sqrt{n}}})$
  4. Set $\boldsymbol{\ell} := (\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_{\sqrt{n}})$.
  5. Output $k_1 := (\mathbf{z}, \boldsymbol{\ell}, (\mathcal{M}_j)_{j \in [t_{\mathsf{max}}]})$.

- $\mathsf{PKPCF.Eval}(0, k_0, \mathbf{x})$:
  1. Parse $k_0 = (\mathbf{k}, M, \mu)$.
  2. For $\mathsf{b} \in \{0, 1\}$:
     (a) Compute $X_\mathsf{b} := \lfloor M\mu \cdot P_{\mathbf{x}}^{(\mathsf{b})}(\mathbf{k}/\mu) \pmod q \rceil_2$.
     (b) Compute $r_\mathsf{b} = \mathsf{H}(X_\mathsf{b} || \mathbf{x})$.
  3. Output $y_0 := (r_0, r_1)$.

- $\mathsf{PKPCF.Eval}(1, k_1, \mathbf{x})$:
  1. Parse $k_1 = (\mathbf{z}, \boldsymbol{\ell}, (\mathcal{M}_j)_{j \in [t_{\mathsf{max}}]})$.

[17] This is possible since $q > B$.

2. Compute $\mathsf{b} := F_{\mathbf{z}}(\mathbf{x})$.
3. Use $\boldsymbol{\ell}$ and $\mathbf{x}$ to compute $(c_1, \ldots, c_{t_\mathsf{b}})$ such that

$$P^{(\mathsf{b})}_{\mathbf{x}}(\boldsymbol{\ell}/\mu + \mathbf{z}) = P^{(\mathsf{b})}_{\mathbf{x}}(\mathbf{z}) + \sum_{i=1}^{t_\mathsf{b}} c_i (1/\mu)^i \pmod q \ .$$

4. Compute $X' := \lfloor \sum_{i=1}^{t_\mathsf{b}} c_i \cdot \mathcal{M}_i \pmod q \rceil_2$.
5. Compute $r' := \mathsf{H}(X' || \mathbf{x})$.
6. Output $y_1 := (\mathsf{b}, r')$.

Theorem 4 (PK-PCFs for OT from Secret-Power Ring-LWE and Public-Power Ring-LWE)**.** *Construction 2 is a public-key PCF (as per Definition 11) for oblivious transfer correlations, under the following assumptions*

- H *is a random oracle,*
- F *is a (weak) pseudorandom function,*
- $\mathsf{SP\text{-}RLWE}(q, \chi_\mu, \chi_e, t_{\mathsf{max}})$ *is hard, and*
- $\mathsf{PP\text{-}RLWE}(p, \mathcal{D}_N, \chi_{e'}, 2\sqrt{n})$ *is hard.* [18]

Proof. As per Definition 11, the proof of this theorem consists of proving the following two properties: (1) pseudorandom OT-correlated outputs, and (2) security (against the sender, and against the receiver, separately). Before discussing these two properties, we first prove another property concerning the correlation between the derived evaluation keys, which will be used in consequent proofs.

Correlated-Evaluation Keys. For $\sigma \in \{0,1\}$, let $k_\sigma \leftarrow \mathsf{KeyDer}(\sigma, \mathsf{sk}_\sigma, \mathsf{pk}_{1-\sigma})$, where $(\mathsf{sk}_\sigma, \mathsf{pk}_\sigma) \leftarrow \mathsf{KeyGen}(\sigma)$, and $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$. We first prove that $k_0 = (\mathbf{k}, M, \mu)$ and $k_1 = (\mathbf{z}, \boldsymbol{\ell}, (\mathcal{M}_j)_{j \in [t_{\mathsf{max}}]})$ are correctly computed, i.e., that it holds that

- $\mathcal{M}_j = M \cdot (1/\mu)^{i-1} + e_i \pmod q$, for $j \in [t_{\mathsf{max}}]$, and
- $\mathbf{k} - \boldsymbol{\ell} = \mu \cdot \mathbf{z} \pmod q$, where $|k_i|_\infty \leq B$.

Note that $(\mathcal{M}_j)_{j \in [t_{\mathsf{max}}]}$ is computed by the sender and directly contained in pk_0, without any post-processing. We therefore focus on how $\mathbf{k}$ and $\boldsymbol{\ell}$ are derived.

Let $\mathbf{k} = (\mathbf{k}_1, \ldots, \mathbf{k}_{\sqrt{n}})$ and $\boldsymbol{\ell} = (\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_{\sqrt{n}})$, where $\mathbf{k}_i, \boldsymbol{\ell}_i \in \{0,1\}^{\sqrt{n}}$, for $i \in [\sqrt{n}]$. $\mathsf{KeyDer}(0, \mathsf{sk}_0, \mathsf{pk}_1)$ and $\mathsf{KeyDer}(1, \mathsf{sk}_1, \mathsf{pk}_0)$ compute these elements as $k_{i_j} := \lfloor s^{(0)}_{i_j} \rceil_B$ and $\ell_{i_j} := \lfloor s^{(1)}_{i_j} \rceil_B$, for $i, j \in [\sqrt{n}]$, such that:

$$s^{(1)}_{i_j} - s^{(0)}_{i_j} = \langle (u^i_0, u^i_1, \mathbf{z}_i) \ , \ (E_{\sqrt{n}-j}, E_{\sqrt{n}-j+1}, \ldots, E_{2\sqrt{n}-j+1}) \rangle,$$

where $E_{\sqrt{n}-j+1+\delta} = \mathcal{N}_{\sqrt{n}-j+1+\delta} - N^{\sqrt{n}-j+1+\delta} \cdot \nu$, for $\delta \in [-1, \sqrt{n}]$. By inspecting $\mathsf{KeyGen}(0)$ and how each $\mathcal{N}_j$ is computed, we have that:

$$E_{\sqrt{n}-j+1+\delta} = \begin{cases} e'_{\sqrt{n}-j+1+\delta} + \lfloor p/B \rceil \cdot \mu & \text{if } \delta = j \\ e'_{\sqrt{n}-j+1+\delta} & \text{otherwise} \end{cases} .$$

It therefore holds that

$$s^{(1)}_{i_j} - s^{(0)}_{i_j} = \lfloor p/B \rceil \cdot \mu \cdot z_{i_j} + \sum_{\delta \neq j} e''_{\sqrt{n}-j+1+\delta},$$

[18] The public-power ring-LWE assumption ($\mathsf{PP\text{-}RLWE}$) is defined in Definition 8.

where $e''_{\sqrt{n}-j} = u_0^i \cdot e'_{\sqrt{n}-j}$, and $e''_{\sqrt{n}-j+1} = u_1^i \cdot e'_{\sqrt{n}-j+1}$, and $e''_{\sqrt{n}-j+1+\delta} = e'_{\sqrt{n}-j+1+\delta} \cdot z_{i_\delta}$ for $\delta \in [1, \sqrt{n}]$. Let $E'' = \sum_{\delta \neq j} e''_{\sqrt{n}-j+1+\delta}$. It holds that $||E''||_\infty \leq dB_{e'}^2 + \sqrt{n} \cdot B_{e'}$. Setting $p/B \gg ||E''||_\infty$ yields $\lfloor s_{i_j}^{(1)} - s_{i_j}^{(0)} \rceil_B = \mu \cdot z_{i_j} \pmod B$.

On the other hand since $p \gg B$, we have that $\lfloor s_{i_j}^{(1)} - s_{i_j}^{(0)} \rceil_B = \lfloor s_{i_j}^{(1)} \rceil_B - \lfloor s_{i_j}^{(0)} \rceil_B$ with overwhelming probability. This yields that $k_{i_j} - \ell_{i_j} = \mu \cdot z_{i_j} \pmod q$, for all $i, j \in \sqrt{n}$, concluding that the two derived keys correctly form subtractive shares of $\mu \cdot \mathbf{z}$. Moreover, note that for $i \in [t]$ the property $|k_i|_\infty \leq B$ is satisfied through rounding modulo B.

Pseudorandom OT-Correlated Outputs. Let Adv be a PPT adversary and consider the following sequence of hybrid games:

Hyb_0: This game is identical to $\mathsf{Exp}^{\mathsf{pr}}_{\mathsf{Adv},1}(\lambda)$ of Figure 2. Here, the view of the adversary consists of $(1^\lambda, \mathsf{pk}_0, \mathsf{pk}_1, (\mathbf{x}^{(i)}, y_0^{(i)}, y_1^{(i)})_i)$, where each pair $(y_0^{(i)}, y_1^{(i)})$ is generated in $\mathcal{O}_1^{\mathsf{pr}}$ by running the evaluation algorithms of both parties on random inputs $\mathbf{x}^{(i)}$ for polynomially-many index i.

Hyb_1: The goal of this hybrid is to simulate the two public keys such that pk_0 and pk_1 do not depend on μ and $\mathbf{z}$, respectively. We break this hybrid into smaller ones:

- Hyb_1^1: In this game, after running $(\mathsf{sk}_\sigma, \mathsf{pk}_\sigma) \leftarrow \mathsf{PKPCF.KeyGen}(\sigma)$, for $\sigma \in \{0,1\}$, we do the following:
  - Parse $\mathsf{sk}_0 = (M, \mu, \nu)$, $\mathsf{pk}_0 = ((\mathcal{M}_i)_{i\in[t_{\mathsf{max}}]}, (\mathcal{N}_j)_{j\in[0,2\sqrt{n}]})$, and $\mathsf{sk}_1 = (\mathbf{z}, (u_0^i, u_1^i)_{i\in[\sqrt{n}]})$, $\mathsf{pk}_1 = (v_i)_{i\in[\sqrt{n}]}$.
  - Run $k_0 \leftarrow \mathsf{PKPCF.KeyDer}(0, \mathsf{sk}_0, \mathsf{pk}_1)$. Parse $k_0 = (\mathbf{k}, M, \mu)$.
  - Compute $\boldsymbol{\ell} := \mathbf{k} - \mu \cdot \mathbf{z} \pmod q$, and set $k_1 := (\mathbf{z}, \boldsymbol{\ell}, (\mathcal{M}_j)_{j\in[t_{\mathsf{max}}]})$.
    // Instead of running $k_1 \leftarrow \mathsf{PKPCF.KeyDer}(1, \mathsf{sk}_1, \mathsf{pk}_0)$.

  Hybrid Hyb_1^1 *remains perfectly indistinguishable from Hybrid* Hyb_0 *due to the correctness of the key derivation algorithm, proved above.*

- Hyb_1^2: In this hybrid, we replace $(\mathsf{sk}_0, \mathsf{pk}_0) \leftarrow \mathsf{PKPCF.KeyGen}(0)$ by the following:
  - Compute elements of sk_0 and pk_0 as in $\mathsf{KeyGen}(0)$, except for $(\mathcal{N}_j)_{j\in[0,2\sqrt{n}]}$.
  - For $j \in [0, 2\sqrt{n}]$, sample $\mathcal{N}_j \xleftarrow{\$} R_p$.
    // Instead of encrypting μ.
  - Add $(\mathcal{N}_j)_{j\in[0,2\sqrt{n}]}$ to pk_0 and output $(\mathsf{sk}_0, \mathsf{pk}_0)$.

  The rest of the experiment is done as in Hyb_1^1.

  Hybrid Hyb_1^2 *remains indistinguishable to* Hyb_1^1 *under the PP-RLWE assumption (Definition 8), more precisely* $\mathsf{PP\text{-}RLWE}(p, \mathcal{D}_N, \chi_{e'}, 2\sqrt{n})$.

- Hyb_1^3: In this hybrid, we additionally replace $(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow \mathsf{PKPCF.KeyGen}(1)$ by the following:
  - For $i \in [\sqrt{n}]$: sample $v_i \xleftarrow{\$} R_p$.
    // Instead of encrypting $\mathbf{z}_i$.
  - Set $\mathsf{sk}_1 := \mathbf{z}$, and $\mathsf{pk}_1 := (v_i)_{i\in[\sqrt{n}]}$.
  - Output $(\mathsf{sk}_1, \mathsf{pk}_1)$.

  Hybrid Hyb_1^3 *remains indistinguishable from* Hyb_1^2 *under the ring-LWE assumption with short secret (Definition 7)* $\mathsf{RLWE}(p, \mathcal{D}_N, \chi_{e'}, \chi_{e'}, \sqrt{n})$. *Note that this assumption is implied by the PP-RLWE assumption*
  $\mathsf{PP\text{-}RLWE}(p, \mathcal{D}_N, \chi_{e'}, 2\sqrt{n})$ *that we already rely on (See Remark 2).*

Hybrid Hyb_1 therefore remains computationally indistinguishable from Hyb_0. Note that in Hyb_1, we no longer run the key-generation and derivation algorithms. Instead, the adversary's views consists of simulated public keys $\mathsf{pk}_0, \mathsf{pk}_1$ that are generated independently of μ and $\mathbf{z}$.

Hyb_2: In this hybrid, for any input $\mathbf{x} \xleftarrow{\$} \{0,1\}^\ell$ sampled in $\mathcal{O}_1^{\mathsf{pr}}$, we replace $y_\sigma \leftarrow \mathsf{PKPCF.Eval}(\sigma, k_\sigma, \mathbf{x})$ for $\sigma \in \{0,1\}$, by $(y_0, y_1) \xleftarrow{\$} \mathsf{OT}(1^\lambda, n')$ for all inputs $\mathbf{x}$.

We claim that Hyb_2 remains computationally indistinguishable to Hyb_1 because Hyb_2 essentially evaluates the "secret-key" PCF of Corollary 1, while outputting simulated public keys. Therefore, the output of the evaluation algorithm should be indistinguishable from uniformly sampled OT-correlated pairs.

Claim 5. *Assuming (1) that F is a (weak) pseudorandom function and (2) the constrained PRF* CPRF *presented in Construction 1 is adaptively-secure,* $\mathsf{Hyb}_1 \approx \mathsf{Hyb}_2$.

Proof. Let SKPCF be the PCF presented in Corollary 1. Any adversary Adv that can distinguish between Hyb_2 and Hyb_1 can be used to build an adversary Adv' that can distinguish between the output of $\mathsf{SKPCF.Eval}$ and random OT-correlated pairs. More precisely, Adv' works as follows: it outputs (1) $\mathsf{pk}_0 := ((\mathcal{M}_j)_{j\in[t_{\mathsf{max}}]}, (\mathcal{N}_j)_{j\in[2\sqrt{n}]})$, where $(\mathcal{M}_j)_{j\in[t_{\mathsf{max}}]}$ is set using the public parameters of SKPCF, and $(\mathcal{N}_j)_{j\in[2\sqrt{n}]}$, where each $B_j \xleftarrow{\$} R_p$, and (2) $\mathsf{pk}_1 := (v_i)_{i\in[\sqrt{n}]}$, where each $v_i \xleftarrow{\$} R_p$. It then queries the output of $\mathsf{PKPCF.Eval}$ on random inputs and forwards the answers to Adv. If the outputs are real PCF evaluations, Adv' simulates Hyb_1, and otherwise if they are random OT-correlated pairs, Adv' simulates Hyb_2. Therefore, if Adv can distinguish between these two hybrids, Adv' succeeds in distinguishing between the output of $\mathsf{SKPCF.Eval}$ and random OT-correlated pairs. □

Hyb_3: This is the same as Hybrid Hyb_2 except that here, we undo the changes from hybrids Hyb_1^3 and Hyb_1^2, and replace them with $(\mathsf{sk}_\sigma, \mathsf{pk}_\sigma) \leftarrow \mathsf{PCF.KeyGen}(\sigma)$, for $\sigma \in \{0,1\}$. This hybrid remains indistinguishable to Hybrid Hyb_2 due to the same arguments explained in hybrids Hyb_1^2 and Hyb_1^3.

Hybrid Hyb_3 is essentially $\mathsf{Exp}^{\mathsf{pr}}_{\mathsf{Adv},0}(\lambda)$. This concludes the proof of pseudorandom OT-correlated outputs of the construction.

Receiver Security (Security Against the Sender). Consider the following sequence of hybrid games:

Hyb_0: This is the game $\mathsf{Exp}^{\mathsf{sec}}_{\mathsf{Adv},\sigma=0,0}(\lambda)$ of Figure 3, where the view of the sender Adv consists of $(1^\lambda, \mathsf{pk}_0, \mathsf{pk}_1, \mathsf{sk}_0, (\mathbf{x}^{(i)}, y_0^{(i)}, y_1^{(i)})_i)$, where $y_1^{(i)} \leftarrow \mathsf{PCF.Eval}(1, k_1, \mathbf{x}^{(i)})$ for all i.

Hyb_1: The goal of this hybrid is to simulate the receiver's public key such that pk_1 does not depend on the wPRF key $\mathbf{z}$. We break this hybrid into smaller ones:

- Hyb_1^1: In this game, after running $(\mathsf{sk}_\sigma, \mathsf{pk}_\sigma) \leftarrow \mathsf{PKPCF.KeyGen}(\sigma)$, for $\sigma \in \{0,1\}$, instead of running $k_1 \leftarrow \mathsf{PKPCF.KeyDer}(1, \mathsf{sk}_1, \mathsf{pk}_0)$, we generate k_1 as follows:
  - Parse $\mathsf{sk}_0 = (M, \mu, \nu)$, $\mathsf{pk}_0 = ((\mathcal{M}_i)_{i\in[t_{\mathsf{max}}]}, (\mathcal{N}_j)_{j\in[0,2\sqrt{n}]})$ and $\mathsf{sk}_1 = (\mathbf{z}, (u_0^i, u_1^i)_{i\in[\sqrt{n}]})$, $\mathsf{pk}_1 = (v_i)_{i\in[\sqrt{n}]}$.
  - Run $k_0 \leftarrow \mathsf{PKPCF.KeyDer}(0, \mathsf{sk}_0, \mathsf{pk}_1)$. Parse $k_0 = (\mathbf{k}, M, \mu)$.
  - Compute $\boldsymbol{\ell} := \mathbf{k} - \mu \cdot \mathbf{z} \pmod q$, and set $k_1 := (\mathbf{z}, \boldsymbol{\ell}, (\mathcal{M}_j)_{j\in[t_{\mathsf{max}}]})$.
    // Instead of running $k_1 \leftarrow \mathsf{PKPCF.KeyDer}(1, \mathsf{sk}_1, \mathsf{pk}_0)$.

  Hybrid Hyb_1^1 is indistinguishable from Hyb_0 due to the correctness of the key derivation algorithm, proved above.

- Hyb_1^2: In this hybrid, we additionally replace $(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow \mathsf{PKPCF.KeyGen}(1)$ by the following:
  - For $i \in [\sqrt{n}]$: sample $v_i \xleftarrow{\$} R_p$.
    // Instead of encrypting $\mathbf{z}_i$.
  - Set $\mathsf{sk}_1 := \mathbf{z}$, and $\mathsf{pk}_1 := (v_i)_{i\in[\sqrt{n}]}$.
  - Output $(\mathsf{sk}_1, \mathsf{pk}_1)$.

Hybrid Hyb_1^2 *remains computationally indistinguishable from* Hyb_1^1 *under the ring-LWE assumption (Definition 7), more precisely,* $\mathsf{RLWE}(p, \mathcal{D}_N, \chi_{e'}, \chi_{e'}, \sqrt{n})$. *Note that this assumption is implied by the PP-RLWE assumption* $\mathsf{PP\text{-}RLWE}(p, \mathcal{D}_N, \chi_{e'}, 2\sqrt{n})$ *(see Remark 2).*

Hyb_2: This is the same as Hyb_1 except that we replace $\mathcal{O}_0^{\mathsf{sec}}$ by $\mathcal{O}_1^{\mathsf{sec}}$, i.e., for all i, instead of computing $y_1^{(i)} \leftarrow \mathsf{PKPCF.Eval}(1, k_1, \mathbf{x}^{(i)})$, we compute $y_1^{(i)} \leftarrow \mathsf{RSample}_{\mathsf{OT}_{n'}}(1^\lambda, 0, y_0^{(i)})$.

We claim that Hyb_2 remains indistinguishable to Hyb_1. This is because Hyb_1 essentially evaluates the "secret-key" PCF of Corollary 1, while outputting a simulated public key for the receiver. Therefore, the output of receiver's evaluation should be indistinguishable from a random output that is OT-correlated to the output of the sender; otherwise Corollary 1 is contradicted.

Claim 6. $\mathsf{Hyb}_1 \approx \mathsf{Hyb}_2$.

Proof. Let SKPCF be the PCF presented in Corollary 1, and consider an adversary Adv that can distinguish between Hyb_2 and Hyb_1. We build Adv' that violates the receiver security of SKPCF as follows: Adv' first receives pp, k_0 as the sender in SKPCF. It then generates $(\mathsf{sk}_0, \mathsf{pk}_0)$ by following the steps of $\mathsf{PKPCF.KeyGen}(0)$ (skipping the steps generating elements Adv' already has). It then simulates $\mathsf{pk}_1 := (v_i)_{i\in[\sqrt{n}]}$ by sampling random elements $v_i \xleftarrow{\$} R_p$. It then forwards $\mathsf{pk}_0, \mathsf{pk}_1$ and k_0 to Adv. It then queries the output of the receiver on random inputs and forwards the answers to Adv. If the outputs are real PCF evaluations, Adv' simulates Hyb_1, and otherwise if they are reversely sampled based on the sender's output, Adv' simulates Hyb_2. Therefore, if Adv can distinguish between these two hybrids, Adv' succeeds in distinguishing between the output of SKPCF's receiver and reversely sampled values that are OT-correlated to that of SKPCF's sender. □

Hyb_3: This is the same as Hyb_2 while reversing the change that we made in Hyb_1^2. More precisely, we bring back the algorithm $(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow \mathsf{PKPCF.KeyGen}(1)$. This hybrid remains indistinguishable to hybrid Hyb_2 for the same arguments explained in hybrid Hyb_2.

Note that Hyb_3 is identical to $\mathsf{Exp}^{\mathsf{sec}}_{\mathsf{Adv},\sigma=0,1}(\lambda)$. This concludes the proof of receiver security.

Sender Security (Security Against the Receiver). Consider the following sequence of hybrid games:

Hyb_0: This is the game $\mathsf{Exp}^{\mathsf{sec}}_{\mathsf{Adv},\sigma=1,0}(\lambda)$ of Figure 3, where the view of the sender Adv consists of $(1^\lambda, \mathsf{pk}_0, \mathsf{pk}_1, \mathsf{sk}_1, (\mathbf{x}^{(i)}, y_0^{(i)}, y_1^{(i)})_i)$, where $y_0^{(i)} \leftarrow \mathsf{PCF.Eval}(0, k_0, \mathbf{x}^{(i)})$ for all i.

Hyb_1: The goal of this hybrid is to simulate the sender's public key such that pk_0 does not depend on the CPRF master secret key element μ. We break this hybrid into smaller ones:

- Hyb_1^1: In this game, after running $(\mathsf{sk}_\sigma, \mathsf{pk}_\sigma) \leftarrow \mathsf{PKPCF.KeyGen}(\sigma)$, for $\sigma \in \{0,1\}$, instead of running $k_0 \leftarrow \mathsf{PKPCF.KeyDer}(0, \mathsf{sk}_0, \mathsf{pk}_1)$, we generate k_0 as follows:
  - Parse $\mathsf{sk}_0 = (M, \mu, \nu)$, $\mathsf{pk}_0 = ((\mathcal{M}_i)_{i\in[t_{\mathsf{max}}]}, (\mathcal{N}_j)_{j\in[0,2\sqrt{n}]})$ and $\mathsf{sk}_1 = (\mathbf{z}, (u_0^i, u_1^i)_{i\in[\sqrt{n}]})$, $\mathsf{pk}_1 = (v_i)_{i\in[\sqrt{n}]}$.
  - Run $k_1 \leftarrow \mathsf{PKPCF.KeyDer}(1, \mathsf{sk}_1, \mathsf{pk}_0)$. Parse $k_0 = (\mathbf{z}, \boldsymbol{\ell}, (\mathcal{M}_j)_{j\in[t_{\mathsf{max}}]})$.
  - Compute $\mathbf{k} := \boldsymbol{\ell} + \mu \cdot \mathbf{z} \pmod q$, and set $k_0 := (\mathbf{k}, M, \mu)$.
    // Instead of running $k_0 \leftarrow \mathsf{PKPCF.KeyDer}(0, \mathsf{sk}_0, \mathsf{pk}_1)$.

  Hybrid Hyb_1^1 *is indistinguishable from Hybrid* Hyb_0 *due to the correctness of the key derivation algorithm, proved above.*

- Hyb_1^2: In this hybrid, we additionally replace $(\mathsf{sk}_0, \mathsf{pk}_0) \leftarrow \mathsf{PKPCF.KeyGen}(0)$ by the following:
  - Compute elements of sk_0 and pk_0 as in $\mathsf{KeyGen}(0)$, except for $(\mathcal{N}_j)_{j\in[0,2\sqrt{n}]}$.
  - For $j \in [0, 2\sqrt{n}]$, sample $\mathcal{N}_j \xleftarrow{\$} R_p$.
    // Instead of encrypting μ.
  - Add $(\mathcal{N}_j)_{j\in[0,2\sqrt{n}]}$ to pk_0 and output $(\mathsf{sk}_0, \mathsf{pk}_0)$.

Hybrid Hyb_1^2 *remains indistinguishable to* Hyb_1^1 *under the PP-RLWE assumption (Definition 8), more precisely* $\mathsf{PP\text{-}RLWE}(p, \mathcal{D}_N, \chi_{e'}, 2\sqrt{n})$.

Hyb_2: This is the same as Hyb_1 except that we replace $\mathcal{O}_0^{\mathsf{sec}}$ by $\mathcal{O}_1^{\mathsf{sec}}$, i.e., for all i, instead of computing $y_0^{(i)} \leftarrow \mathsf{PKPCF.Eval}(0, k_0, \mathbf{x}^{(i)})$, we compute $y_0^{(i)} \leftarrow \mathsf{RSample}_{\mathsf{OT}_{n'}}(1^\lambda, 1, y_1^{(i)})$.

We claim that Hyb_2 remains indistinguishable to Hyb_1. This is because Hyb_1 essentially evaluates the "secret-key" PCF of Corollary 1, while outputting a simulated public key for the sender. Therefore, the output of the sender's evaluation should be indistinguishable from a random output that is OT-correlated to the output of the receiver; otherwise Corollary 1 is contradicted.

Claim 7. $\mathsf{Hyb}_1 \approx \mathsf{Hyb}_2$.

Proof. Let SKPCF be the PCF presented in Corollary 1, and consider an adversary Adv that can distinguish between Hyb_2 and Hyb_1. We build Adv' that violates the sender security of SKPCF as follows: Adv' first receives pp, k_1 as the receiver in SKPCF. It then generates $(\mathsf{sk}_1, \mathsf{pk}_1)$ by following the steps of $\mathsf{PKPCF.KeyGen}(1)$ (skipping the step that generate the elements Adv' already has). It then simulates pk_0 by generating it as in Hyb_2 (that is the same as Hyb_1). It then forwards $\mathsf{pk}_0, \mathsf{pk}_1$ and k_1 to Adv. It then queries the output of the sender on random inputs and forwards the answers to Adv. If the outputs are real PCF evaluations, Adv' simulates Hyb_1, and otherwise if they are reversely sampled based on the receiver's outputs, Adv' simulates Hyb_2. Therefore, if Adv can distinguish between these two hybrids, Adv' succeeds in distinguishing between the output of SKPCF's sender and reversely sampled values that are correlated to that of SKPCF's receiver. □

Hyb_3: This is the same as Hyb_2 while reversing the changes made in Hyb_1^2. More precisely, we bring back the algorithm $(\mathsf{sk}_0, \mathsf{pk}_0) \leftarrow \mathsf{PKPCF.KeyGen}(0)$. This hybrid remains indistinguishable to hybrid Hyb_2 for the same arguments explained in indistinguishability of Hyb_2 to Hyb_1^2.

Note that Hyb_3 is identical to $\mathsf{Exp}^{\mathsf{sec}}_{\mathsf{Adv},\sigma=1,1}(\lambda)$. This concludes the proof of receiver security.

This concludes the proof of Theorem 4. □

□

8 Faster PK-PCF from Truncated SP-RLWE

8.1 Truncated SP-RLWE

We consider a variant of the SP-RLWE assumption where M is not uniformly random. Instead, it is *truncated on both ends*: its first and last r coefficients are set to zero (for some parameter r of the assumption), while all other coefficients are picked uniformly at random. To state the assumption formally, we introduce the set $\mathsf{Trunc}_r(R_q) = \{M \in R_q \ : \ M = \sum_{i=r}^{d-1-r} M_i \cdot X^i\}$, where d denotes the dimension of R_q.

Definition 15 ((Decision) Truncated Secret-Power Ring-LWE (TSP-RLWE) Assumption)**.** *Let $\lambda \in \mathbb{N}$ be a security parameter, and let $m = m(\lambda) \in \mathbb{N}$, $q = q(\lambda) \in \mathbb{N}$, $r = r(\lambda) \in \mathbb{N}$. Let R be a ring defined over $\mathbb{Z}$, and $\chi_\mu = \chi_\mu(\lambda)$ and $\chi_e = \chi_e(\lambda)$ be distributions over R. The* $\mathsf{TSPRLWE}(q, \chi_\mu, \chi_e, m, r)$ *assumption states that*

$$\left(M + \Delta,\ \{(M+\Delta)\cdot(1/\mu)^i + e_i\}_{i\in[m]}\right) \approx_c \left(M + \Delta,\ \{\mathcal{M}_i\}_{i\in[m]}\right),$$

where $\mu \xleftarrow{\$} \chi_\mu$ is an invertible element, $M \xleftarrow{\$} \mathsf{Trunc}_r(R_q)$, $\Delta \xleftarrow{\$} \mathbb{Z}_q$, $\mathcal{M}_i \xleftarrow{\$} R_q$, and $e, e_i \xleftarrow{\$} \chi_e$, for all $i \in [m]$.

8.2 Sparse Coefficient Packing

We introduce below a lemma that shows how coefficient packing in t-sum-free subsets of coordinates enjoys homomorphic properties.

Definition 16 (t-Sum-Free Sets)**.** *We say that a subset $S \subset \mathbb{N}$ is* t-sum-free *if for every $i \in S$ and every $(i_1, \ldots, i_t) \in S^t \setminus \{(i, \ldots, i)\}$, it holds that $\sum_{j=1}^t i_j \neq t \cdot i$.*

Lemma 3 (Coefficient Packing Homomorphism)**.** *Let C denote an n-input, one-output arithmetic circuit that computes a homogeneous polynomial of total degree t. Let $R = \mathbb{Z}_q[X]/(F(X))$ be a polynomial ring over $\mathbb{Z}_q$ with $\deg(F) = d$, and let $S \subseteq \{0, \ldots, \lfloor (d-1)/t \rfloor\}$ be a t-sum-free subset. For $j = 1, \ldots, n$ let $U_j(X) = \sum_{i \in S} u_{j,i} X^i \in R$ (i.e. each U_j has nonzero coefficients only at exponents in S). Then for every $i \in S$ the coefficient of $X^{t \cdot i}$ in the polynomial $C(U_1, \ldots, U_n)$ is equal to $C(u_{1,i}, \ldots, u_{n,i})$.*

Proof. Since C computes a homogeneous polynomial of total degree t, we may write $C(Y_1, \ldots, Y_n) = \sum_{|\mu|=t} c_\mu Y^\mu$, where $\mu = (\mu_1, \ldots, \mu_n) \in \mathbb{N}^n$, $|\mu| = \sum_j \mu_j = t$, and $Y^\mu = \prod_{j=1}^n Y_j^{\mu_j}$. Substitute $Y_j = U_j(X) = \sum_{i \in S} u_{j,i} X^i$. Expanding each monomial Y^μ gives

$$\begin{aligned} Y^\mu \Big|_{Y_j = U_j} &= \prod_{j=1}^{n} \Big(\sum_{i \in S} u_{j,i} X^i \Big)^{\mu_j} \\ &= \sum_{\substack{i_{j,1}, \ldots, i_{j,\mu_j} \in S \\ j = 1, \ldots, n}} \prod_{j=1}^{n} \prod_{r=1}^{\mu_j} u_{j, i_{j,r}} \; X^{\sum_{j=1}^{n} \sum_{r=1}^{\mu_j} i_{j,r}}. \end{aligned}$$

Equivalently, each term in the full expansion of $C(U_1, \ldots, U_n)$ contributes a monomial whose X-exponent is $\sum_{j=1}^{n} \sum_{r=1}^{\mu_j} i_{j,r}$, which has exactly t terms $i_{j,r} \in S$, and whose coefficient is $c_\mu \cdot \prod_{j=1}^{n} \prod_{r=1}^{\mu_j} u_{j,i_{j,r}}$; let us call this coefficient the *contribution* of the t-tuple $(i_{j,r})_{j \le n, r \le \mu_j}$ to this monomial. Fix $i \in S$; we want the coefficient of $X^{t \cdot i}$. By the above equation, the coefficient of $X^{t \cdot i}$ equals the sum, over all $(i_1, \ldots, i_t) \in S^t$ with $\sum_{k=1}^{t} i_k = t \cdot i$, of the contributions of $(i_1, \ldots, i_t)$ to $X^{t \cdot i}$. However, by the hypothesis on S the only t-tuple in S^t whose entries sum to $t \cdot i$ is the tuple $(i, \ldots, i)$. This implies that for every μ, the only contribution to $X^{t \cdot i}$ is that of $(i, \cdots, i)$, which contributes $c_\mu \cdot \prod_{j=1}^{n} \prod_{r=1}^{\mu_j} u_{j,i}$. Hence, the sum over all μ of the contributions (that is, the coefficient of $X^{t \cdot i}$ in $C(U_1, \ldots, U_n)$) is

$$\sum_{|\mu|=t} c_\mu \prod_{j=1}^{n} u_{j,i}^{\mu_j} \;=\; \sum_{|\mu|=t} c_\mu \, (u_{1,i})^{\mu_1} \cdots (u_{n,i})^{\mu_n} \;=\; C(u_{1,i}, \ldots, u_{n,i}),$$

which is exactly the claimed identity. □ □

In the following, we will restrict our attention to t-sum-free subsets S of the interval $[0, r]$, where r is a bound whose value will control the strength of the truncated SP-RLWE assumption.

8.3 Packed CPRF from TSP-RLWE

8.3.1 Packed CPRFs.

We consider a natural extension of CRPFs to *packed* CPRFs where given a packing parameter s, $\mathsf{Constrain}$ takes as input an s-tuple of constrains, and $\mathsf{Eval}(\mathsf{msk}, \mathbf{x}), \mathsf{CEval}$ output s-tuples of values. The CPRF security notion is extended in the natural way, and boils down to s parallel executions of the standard CPRF security game.

8.3.2 Notations.

We use the same notations as Section 5.1 for the ring R_q and the distributions χ_μ, χ_e. Parameters are the same as for the CPRF from Section 5.1, except that we set $B = 2^{\omega(\log \lambda)} \cdot B_\mu \cdot |S|$. Let p denote a λ-bit modulus. We let $r < d$ denote a parameter for the TSP-RLWE assumption, and S denote a t-sum-free subset of $\{0, \cdots, \lfloor r/t \rfloor\}$. We let $t \cdot S$ denote the set $\{t \cdot s \;:\; s \in S\}$. For a set T, we let $\sigma_T : T \to \{1, \cdots, |T|\}$ denote an arbitrary indexing of the elements of T. We introduce the following procedures:

- $\mathsf{Pack}(\mathbf{z}_1, \cdots, \mathbf{z}_{|S|})$: on input $|S|$ vectors in $\{0,1\}^n$, define $\mathbf{Z} = \sum_{i \in S} \mathbf{z}_{\sigma_S(i)} \cdot X^i$, viewed as an n-tuple of elements of R_q with binary coefficients.
- $\mathsf{coeffs}_T(u)$: given $u = \sum_i u_i \cdot X^i \in R_q$, returns $(u_{\sigma_T^{-1}(1)}, \cdots, u_{\sigma_T^{-1}(|T|)})$ (the list of coefficients of u indexed by T).

- $\mathsf{Adapt}(P) \mapsto \tilde{P}$: on input a degree-t multivariate polynomial P, returns a modified (homogeneous) polynomial $\tilde{P}$ such that for any $(\mathbf{z}_1, \cdots, \mathbf{z}_{|S|}) \in (\{0,1\}^n)^{|S|}$, denoting $\mathbf{Z} = \mathsf{Pack}(\mathbf{z}_1, \cdots, \mathbf{z}_{|S|})$, it holds that
$$\mathsf{coeffs}_{t \cdot S}\left(\tilde{P}(\mathbf{Z})\right) = (P(\mathbf{z}_1), \cdots, P(\mathbf{z}_{|S|})).$$

Note that we only provide the template for Adapt; in other words, the packed CPRF works for a target polynomial P provided that there is a suitable adaptation of P to a homogoneous polynomial $\tilde{P}$ compatible with sparse coefficient packing. Note also that for our target application, a weaker correctness guarantee would suffice: we only need the coefficients of $\tilde{P}(\mathbf{Z})$ indexed by T to be nonzero iff the corresponding $P(\mathbf{z}_i)$ is nonzero. It is not too hard to describe classes of polynomials with a suitable Adapt (in particular, for homogeneous polynomials Adapt can be the identity), but it would needlessly complicate the description. We will confine ourselves to describing Adapt for the concrete polynomial that arises from the Goldreich-Applebaum-Raykov wPRF (see Section 6).

8.3.3 Construction

- $\mathsf{CPRF.KeyGen}(1^\lambda, P)$:
  1. Sample $\mathbf{k} = (k_1, \ldots, k_n) \xleftarrow{\$} R_q^n$ such that $|k_i|_\infty \leq B(\lambda)$ for all $i \in [n]$.
  2. Sample $\mu \xleftarrow{\$} \chi_\mu$, $\Delta \xleftarrow{\$} \mathbb{Z}_q$, and $M \xleftarrow{\$} \mathsf{Trunc}_r(R_q)$. // μ should be invertible.
  3. Set and output $\mathsf{pp} := P$, and $\mathsf{msk} := (\mathbf{k}, M, \mu, \Delta)$.
- $\mathsf{CPRF.Eval}(\mathsf{msk}, \mathbf{x} \in \{0,1\}^\ell)$:
  1. Parse $\mathsf{msk} = (\mathbf{k}, M, \mu, \Delta)$.
  2. Compute and output
  $$(y_1, \cdots, y_{|S|}) := \mathsf{H}\left(\mathsf{coeffs}_{t \cdot S}\left(\lfloor (M + \Delta) \cdot \tilde{P}_{\mathbf{x}}(\mathbf{k}/\mu) \pmod q \rceil_p\right)\right),$$
  where H is evaluated coefficient-wise.
- $\mathsf{CPRF.Constrain}(\mathsf{msk}, \mathbf{z}_1, \cdots, \mathbf{z}_{|S|} \in \{0,1\}^n)$:
  1. Parse $\mathsf{msk} = (\mathbf{k}, M, \mu, \Delta)$.
  2. Pack $\mathbf{Z} := \mathsf{Pack}(\mathbf{z}_1, \cdots, \mathbf{z}_{|S|})$
  3. Compute $\boldsymbol{\ell} := \mathbf{k} - \mu \cdot \mathbf{Z}$.
  4. For $i \in [t]$, compute $\mathcal{M}_i := (M + \Delta) \cdot (1/\mu)^i + e_i$, where $e_i \xleftarrow{\$} \chi_e$.
  5. Output $\mathsf{ck}_{\mathbf{Z}} := (\mathbf{Z}, \boldsymbol{\ell}, (\mathcal{M}_i)_{i \in [t]})$.
- $\mathsf{CPRF.CEval}(\mathsf{ck}_{\mathbf{Z}}, \mathbf{x} \in \{0,1\}^\ell)$:
  1. Parse $\mathsf{ck}_{\mathbf{Z}} = (\mathbf{Z}, \boldsymbol{\ell}, (\mathcal{M}_i)_{i \in [t]})$.
  2. Use $\mathsf{ck}_{\mathbf{Z}}$ and $\mathbf{x}$ to compute $(c_1, \ldots, c_t)$ such that
  $$\tilde{P}_{\mathbf{x}}(\boldsymbol{\ell}/\mu + \mathbf{Z}) = \tilde{P}_{\mathbf{x}}(\mathbf{Z}) + \sum_{i=1}^{t} c_i (1/\mu)^i \pmod q \,. \tag{4}$$
  (see Lemma 2.)
  3. Compute and output
  $$(y'_1, \cdots, y'_{|S|}) := \mathsf{H}\left(\mathsf{coeffs}_{t \cdot S}\left(\left\lfloor \sum_{i=1}^{t} c_i \cdot \mathcal{M}_i \pmod q \right\rceil_p\right)\right),$$
  where H is evaluated coefficient-wise.

8.3.4 Adapting $P_{\mathbf{x}}$.

Recall that $P_{\mathbf{x}}(\mathbf{z})$ is of the form

$$\begin{aligned} P_{\mathbf{x}}(\mathbf{z}) =& P_{\mathsf{XOR}_\kappa}(\mathbf{z}[x_1], \ldots, \mathbf{z}[x_\kappa]) \times P_{\neg\mathsf{MAJ}_\iota}(\mathbf{z}[x_{\kappa+1}], \ldots, \mathbf{z}[x_{\kappa+\iota}]) + \\ & P_{\neg\mathsf{XOR}_\kappa}(\mathbf{z}[x_1], \ldots, \mathbf{z}[x_\kappa]) \times P_{\mathsf{MAJ}_\iota}(\mathbf{z}[x_{\kappa+1}], \ldots, \mathbf{z}[x_{\kappa+\iota}]), \end{aligned}$$

where each term $\mathsf{trm} \in \{\mathsf{XOR}_\kappa, \neg\mathsf{MAJ}_\iota, \neg\mathsf{XOR}_\kappa, \mathsf{MAJ}_\iota\}$ is of the form

$$P_{\mathsf{trm}}(b_1, \ldots, b_\zeta) := \prod_{j \in \mathsf{S}_{\mathsf{trm}}} \left(\sum_{i=1}^{\zeta} b_i - j \right),$$

where $\zeta \in \{\kappa, \iota\}$, $\mathsf{S}_{\mathsf{trm}}$ is a fixed subset of integers for each term trm (e.g., $\mathsf{S}_{\mathsf{XOR}_\kappa} = \mathsf{Even}_\kappa$, the set of even integers below κ). We modify the polynomial to make it homogeneous while preserving weak correctness: the modified polynomial should evaluate to 0 only when the original polynomial does. Define

$$\begin{aligned} P_\top(b_1, \cdots, b_{\kappa+\iota}) &= P_{\mathsf{XOR}_\kappa}(b_1, \cdots, b_\kappa) \times P_{\neg\mathsf{MAJ}_\iota}(b_{\kappa+1}, \cdots, b_{\kappa+\iota}) \\ P_\bot(b_1, \cdots, b_{\kappa+\iota}) &= P_{\neg\mathsf{XOR}_\kappa}(b_1, \cdots, b_\kappa) \times P_{\mathsf{MAJ}_\iota}(b_{\kappa+1}, \cdots, b_{\kappa+\iota}). \end{aligned}$$

The first step will be to enforce $|\mathsf{Even}_\kappa| + |\mathsf{Small}_\iota| = |\mathsf{Odd}_\kappa| + |\mathsf{Big}_\iota|$. We do so by adding appropriate dummy terms: each set $\mathsf{S} \in \{\mathsf{Even}_\kappa, \mathsf{Odd}_\kappa\}$ can be padded by adding arbitrary integers larger than κ, and each set $\mathsf{S} \in \{\mathsf{Big}_\iota, \mathsf{Small}_\iota\}$ can be padded by adding arbitrary integers larger than ι. Observe that while this changes the polynomial outputs, it preserves weak correctness: $P_{\mathbf{x}}(\mathbf{z})$ evaluates to 0 iff $P_\top$ and $P_\bot$ both evaluate to 0. $P_\top$ evaluates to 0 only when $\sum_{i=1}^{\kappa} b_i$ belongs to Even_κ or Big_ι, and since $\sum_{i=1}^{\kappa} b_i \leq \kappa$, adding integers larger than κ does not influence this; the reasoning is similar for $P_\bot$.

After padding the sets, it remains to deal with the "$-j$" terms in the polynomials P_{trm}. Define $1_S(X) = \sum_{i \in S} X^i \in R_q$. That is, $1_S(X)$ is the polynomial that has coefficient 1 in each "slot" from S, and coefficient 0 everywhere else. We replace each P_{trm} with

$$\tilde{P}_{\mathsf{trm}}(b_1, \ldots, b_\zeta) := \prod_{j \in \tilde{\mathsf{S}}_{\mathsf{trm}}} \left(\sum_{i=1}^{\zeta} b_i - j \cdot 1_S(X) \right),$$

where $\tilde{\mathsf{S}}_{\mathsf{trm}}$ denotes the padded version of $\mathsf{S}_{\mathsf{trm}}$. Eventually, we denote $\tilde{P}_{\mathbf{x}}$ the polynomial obtained by replacing each P_{trm} with $\tilde{P}_{\mathsf{trm}}$ in $P_{\mathbf{x}}$. This ensures that when evaluating $\tilde{P}_{\mathbf{x}}(\mathbf{Z})$ on a packed input, the constant $-j$ will be correctly subtracted from each slot in S.

We note that $\tilde{P}_{\mathbf{x}} = \mathsf{Adapt}(P_{\mathbf{x}})$ as defined above is technically not a homogeneous polynomial. However, it becomes homogeneous when one views the polynomial 1_S as part of the input rather than as a constant hardcoded in the description of the $\tilde{P}_{\mathsf{trm}}$ (as $\sum_{i=1}^{\zeta} b_i - j \cdot b_{\zeta+1}$ is a degree-1 homogeneous polynomial), and correctness follows immediately by applying Lemma 3 to $\tilde{P}_{\mathbf{x}}$ *when viewing* 1_S *as an input to* $\tilde{P}_{\mathbf{x}}$.

8.3.5 Security Analysis.

The security analysis proceeds almost identically to that of the CPRF from Section 5.1. We follows a similar outline and use the same notation, highlighting only the changes to the games. A minor technicality is that the packed CPRF is no longer no-evaluation secure when one removes the hash evaluation on the outputs. Intuitively, this is because our original CPRF candidate (without hashing) is only no-evaluation secure when the adversary receives a single challenge; however, in our packed version, the adversary receives a batch of $|S|$ challenges that are correlated (they have all been computed using the same Δ). Nevertheless, hashing the output still provides full security using the usual ROM-based argument (or, alternatively, assuming that H is correlation-robust for the class of affine correlations, see [81]).

Hyb_0: This game is the (batch) CPRF no-evaluation selective security game where the challenge query $\mathbf{x}^*$ is answered by returning the real batch of PRF evaluations $\mathbf{y}^* = \mathsf{Eval}(\mathsf{msk}, \mathbf{x}^*)$.

Hyb_1: In this game, the challenger first computes $\mathsf{ck}_{\mathbf{z}}$ honestly as in Hyb_0. Upon receiving the challenge query $\mathbf{x}^*$, the challenger computes and returns

$$\mathbf{y}^* = \mathsf{H}\left(\mathsf{coeffs}_{t \cdot S}\left(\lfloor \mathsf{CEval}_{\mathsf{pre\text{-}round}}(\mathsf{ck}_{\mathbf{z}}, \mathbf{x}^*) + (M + \Delta) \cdot \tilde{P}_{\mathbf{x}^*}(\mathbf{z}) \rceil_p \right)\right).$$

Hyb_2: In this game, the challenger first samples $\boldsymbol{\ell} \xleftarrow{\$} R_q^n$ such that $|\boldsymbol{\ell}_i|_\infty \leq B$ for $i \in [n]$. It then samples an invertible element $\mu \xleftarrow{\$} \chi_\mu$, $\Delta \xleftarrow{\$} \mathbb{Z}_q$, and computes $\mathbf{k} = \boldsymbol{\ell} + \mu \cdot \mathbf{Z}$. The challenger then sets $\mathsf{msk} = (\mathbf{k}, M, \mu, \Delta)$ and $\mathsf{ck}_{\mathbf{Z}} = (\mathbf{Z}, \boldsymbol{\ell}, (\mathcal{M}_i)_{i\in[t]})$, where $M \xleftarrow{\$} \mathsf{Trunc}_r(R_q)$ and $(\mathcal{M}_i = (M+\Delta)\cdot(1/\mu)^i + e_i)_{i\in[t]}$. The rest of the game proceeds as in Hyb_1. Statistical security follows identically as in Section 5.1, except that we now need $B = 2^{\omega(\log\lambda)} \cdot B_\mu \cdot |S|$ to statistically mask $\mu \cdot \mathbf{Z}$, whose coefficients are bounded by $B_\mu \cdot |S|$.

Hyb_3: This game the challenger generates the constrained key as in Hyb_2, but differently, in order to answer the challenge query, it samples $\tilde{\Delta} \xleftarrow{\$} \mathbb{Z}_q$, and returns

$$\mathbf{y}^* = \mathsf{H}\left(\lfloor \mathsf{coeffs}_{t\cdot S}(\mathsf{CEval}_{\mathsf{pre\text{-}round}}(\mathsf{ck}_{\mathbf{Z}}, \mathbf{x}^*)) + \tilde{\Delta} \cdot (\tilde{P}_{\mathbf{x}^*}(\mathbf{z}_1), \cdots, \tilde{P}_{\mathbf{x}^*}(\mathbf{z}_{|S|})) \rceil_p\right).$$

Claim 8. *Assuming the hardness of* $\mathsf{SP\text{-}RLWE}(q, \chi_\mu, \chi_e, t)$ *(Definition 13),* Hyb_3 *and* Hyb_2 *are computationally indistinguishable.*

Proof. First, observe that since rounding is applied coefficient-wise, coeffs and $\lfloor\cdot\rceil_p$ commute: $\mathsf{coeffs}(\lfloor x \rceil_p) = \lfloor \mathsf{coeffs}(x) \rceil_p$. Second, coeffs is linear:

$$\begin{aligned}&\mathsf{coeffs}_{t\cdot S}(\mathsf{CEval}_{\mathsf{pre\text{-}round}}(\mathsf{ck}_{\mathbf{Z}}, \mathbf{x}^*) + (M+\Delta)\cdot \tilde{P}_{\mathbf{x}^*}(\mathbf{Z}))\\ =&\mathsf{coeffs}_{t\cdot S}(\mathsf{CEval}_{\mathsf{pre\text{-}round}}(\mathsf{ck}_{\mathbf{Z}}, \mathbf{x}^*)) + \mathsf{coeffs}_{t\cdot S}((M+\Delta)\cdot \tilde{P}_{\mathbf{x}^*}(\mathbf{Z})).\end{aligned}$$

Now, since $S \subset [0, \lfloor r/t \rfloor]$ and $\deg(\tilde{P}_{\mathbf{x}^*}) \leq t$, it holds that the largest coefficient of $\tilde{P}_{\mathbf{x}^*}(\mathbf{Z})$ has index below $t \cdot \lfloor r/t \rfloor \leq r$. Since $M \in \mathsf{Trunc}_r(R_q)$, this implies that $M \cdot \tilde{P}_{\mathbf{x}^*}(\mathbf{Z}) = X^r \cdot u$ for some polynomial $u \in \mathbb{Z}_q[X]$. Since $t \cdot S \subset [0, r]$,

$$\begin{aligned}&\mathsf{coeffs}_{t\cdot S}((M+\Delta)\cdot \tilde{P}_{\mathbf{x}^*}(\mathbf{z}))\\ =&\mathsf{coeffs}_{t\cdot S}(M \cdot \tilde{P}_{\mathbf{x}^*}(\mathbf{Z})) + \mathsf{coeffs}_{t\cdot S}(\Delta \cdot \tilde{P}_{\mathbf{x}^*}(\mathbf{Z}))\\ =&(0, \cdots, 0) + \mathsf{coeffs}_{t\cdot S}(\Delta \cdot \tilde{P}_{\mathbf{x}^*}(\mathbf{Z}))\\ =&\Delta \cdot \mathsf{coeffs}_{t\cdot S}(\tilde{P}_{\mathbf{x}^*}(\mathbf{Z})).\end{aligned}$$

Now, by Lemma 3, we have $\mathsf{coeffs}_{t\cdot S}(\tilde{P}_{\mathbf{x}^*}(\mathbf{Z}) = (\tilde{P}_{\mathbf{x}^*}(\mathbf{z}_1), \cdots, \tilde{P}_{\mathbf{x}^*}(\mathbf{z}_{|S|}))$. We conclude via a reduction to $\mathsf{TSPRLWE}(q, \chi_\mu, \chi_e, t, r)$: assume that a PPT adversary Adv succeeds in distinguishing between these two hybrids. We construct an adversary Adv' that breaks the $\mathsf{TSPRLWE}(q, \chi_\mu, \chi_e, t, r)$ assumption as follows. Given $(M, \tilde{\Delta}) \in \mathsf{Trunc}_r(R_q) \times \mathbb{Z}_q$ and t structured elements $(\mathcal{M}_i = (M+\Delta)\cdot(1/\mu)^i + e_i)_{i\in[t]}$, the goal of Adv' is to decide whether $\tilde{\Delta} \xleftarrow{\$} \mathbb{Z}_q$ or $\tilde{\Delta} = \Delta$. Adv' sets $\mathsf{pp} = (P, R_q)$. Upon receiving constraints $(\mathbf{z}_1, \cdots, \mathbf{z}_{|S|})$ from Adv, it sets $\mathsf{ck}_{\mathbf{Z}} = (\mathbf{Z}, \boldsymbol{\ell}, (\mathcal{M}_i)_{i\in[1,t]})$, where $\boldsymbol{\ell} \xleftarrow{\$} \chi_\mu^n$. It then returns $\mathsf{ck}_{\mathbf{Z}}$ to Adv. Finally, to answer the challenge query $\mathbf{x}^*$, it computes

$$\mathbf{y}^* = \mathsf{H}\left(\lfloor \mathsf{coeffs}_{t\cdot S}(\mathsf{CEval}_{\mathsf{pre\text{-}round}}(\mathsf{ck}_{\mathbf{Z}}, \mathbf{x}^*)) + \tilde{\Delta} \cdot (\tilde{P}_{\mathbf{x}^*}(\mathbf{z}_1), \cdots, \tilde{P}_{\mathbf{x}^*}(\mathbf{z}_{|S|})) \rceil_p\right).$$

Note that $|P_{\mathbf{x}^*}(\mathbf{z}) \cdot e|_\infty \leq B_e \cdot B_P \cdot \sum_{s=0}^{t} T_{P,s} \ll q$. It is immediate that if $\Delta = \tilde{\Delta}$, Adv' simulates Hyb_2, and otherwise, if $\tilde{\Delta}$ is a random $\mathbb{Z}_q$ element, Adv' simulates Hyb_3. Therefore, distinguishing these two hybrids implies breaking the $\mathsf{TSPRLWE}(q, \chi_\mu, \chi_e, t, r)$ assumption. □

Hyb_4: This game is similar to Hyb_3, except that here the challenger answers the challenge query $\mathbf{x}^*$ by returning a random value $y^* \xleftarrow{\$} \{0,1\}$.

Claim 9. Hyb_4 *remains perfectly indistinguishable to* Hyb_3.

Sketch. As q is chosen such that each $P_{\mathbf{x}^*}(\mathbf{z}_i) \in \mathsf{BI}_P$ is invertible in $\mathbb{Z}_q$, it holds that $\tilde{\Delta} \cdot P_{\mathbf{x}^*}(\mathbf{z}_i)$ retains the full $\log q$ bits of entropy of $\tilde{\Delta}$, and the truncation to $\log p$ bits retains $\log p \approx \lambda$ bits of entropy. Therefore, using a standard argument involving a suitable notion of correlation robustness for H for the class of "rounded-to-p" affine correlations (which can be easily shown to hold in the ROM, similarly as in previous works [37, 81]), the vector $\mathbf{y}^*$ is indistinguishable from random. □

The rest of the proof proceeds by reversing the sequence of hybrid games while leaving the challenge query answered by a uniformly random value.

8.4 Packed PK-PCF

We generalize our PK-PCF to a packed PK-PCF (that produces batches of $|S|$ OTs upon each evaluation) by using our packed CPRF as a drop-in replacement for our baseline CPRF in the PK-PCF construction of Section 7. We mention one minor modification: in the construction of Section 7, the receiver needs only evaluate the CPRF for one of $P_{\mathbf{x}}^{(0)}$, $P_{\mathbf{x}}^{(1)}$, namely, the one for which $\mathsf{b} := F_{\mathbf{z}}(\mathbf{x})$. However, in the packed setting, it is highly unlikely that $\mathsf{b} = F_{\mathbf{z}}(\mathbf{x}_1) = \cdots = F_{\mathbf{z}}(\mathbf{x}_{|S|})$. Therefore, the receiver algorithm must be slightly adapted: the receiver evaluates the packed CPRF on both $P_{\mathbf{x}}^{(0)}$ and $P_{\mathbf{x}}^{(1)}$, obtaining two packs of outputs $(y_1^{(0)}, \cdots, y_{|S|}^{(0)})$ and $(y_1^{(1)}, \cdots, y_{|S|}^{(1)})$, and outputs $(y_1^{(\mathsf{b}_1)}, \cdots, y_{|S|}^{(\mathsf{b}_{|S|})})$ where $\mathsf{b}_i := F_{\mathbf{z}}(\mathbf{x}_i)$ for $i = 1$ to $|S|$. We omit the full description of the updated construction (but take this update into account in our performance evaluation).

9 Performance Evaluation

In this section, we overview our choice of concrete parameters for instantiating the PK-PCF. To obtain runtime estimates, we carefully count all operations involved in operations on elements of R: residue number system (RNS) decomposition, number-theoretic transform (NTT), pointwise multiplications and additions, inverse-NTT, Chinese remainder theorem (CRT) reconstruction. We observe that NTT, inverse-NTT, and pointwise-multiplication dominate the evaluation time in our setting (this is because we only need CRT reconstruction on a very small set of coefficient at the end of the evaluation), and provide estimates of the costs of these operations using the detailed benchmarks from [15], the reference paper for the Intel® Homomorphic Encryption Acceleration Library (HEXL), which is the state-of-the-art C++ library used by modern FHE libraries (such as SEAL or PALISADE) for implementing higher-level homomorphic operations. All operations in the paper are benchmarked on a 3rd Gen Intel Xeon® Scalable Processors Platinum 8360Y 2.4GHz processor with 64GB of RAM and 72 cores. They are run single-threaded on a single core. While this configuration exceeds that of a typical personal computer, it is roughly in the range of a high-end workstation or powerful personal system, with an Intel list price of $5,383.

9.1 Selecting the candidate

Recall from Section 6 that we select the Goldreich-Applebaum-Raykov (GAR) wPRF as a suitable candidate to instantiate our template. The security of this wPRF has been thoroughly investigated in the literature (see [3–5, 16, 36, 41, 43, 47, 48, 64, 66–68, 72, 73, 82, 83, 85] and references therein). We rely on the parameter selection process selected in [33, 37] (updated in light of the attack of [48]) to achieve 128 bits of security against the state of the art attacks on Goldreich's PRG from [6, 41, 86, 87].

Let $n = \lambda^\delta$ and n^{1+e} denote the total number of pseudorandom OTs that we want to generate overall. For the $\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota$ predicate, [33] suggest choosing the parameters as $\kappa = 2(1+e)$ and $\iota = 2\delta e/(\delta - 1)$. We select two candidates, both striking a good balance across all efficiency metrics, with the first optimizing for faster runtime at the expense of larger public keys (and longer key-decompression time), and the second optimizing for key size (and decompression time) at the expense of a slower runtime. We dub the candidates *fast* and *small*.

- **(Fast)** $\mathsf{XOR}_5\text{-}\mathsf{MAJ}_5$: With this predicate, we can solve the above equations to get $e = 1.5$ and $\delta = 2.5$. This would imply a value of $n = 128^{2.5} = 185364 = 2^{17.5}$ (which eventually affects the key size) and a total number of OTs as $n^{2.5} = 2^{43.75}$.
- **(Small)** $\mathsf{XOR}_6\text{-}\mathsf{MAJ}_8$: Here we get the values as $e = 2$ and $\delta = 2$. This would imply a value of $n = 128^2 = 2^{14}$ and a total number of OTs as $n^3 = 2^{42}$.

Additionally, note that our parameter setting is also resistant to the attack of Fu et al. [48], as we have a very small value of ι and a very large value of n. [38] give a detailed explanation of why that attack works poorly in such a setting. We also compute the degree t of the $\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota$ as $t = \max\left(|\mathsf{Even}_\kappa| + |\mathsf{Big}_\iota|, |\mathsf{Odd}_\kappa| + |\mathsf{Small}_\iota|\right)$ and that of $\neg\mathsf{XOR}_\kappa\text{-}\mathsf{MAJ}_\iota$ as $\bar{t} = \max\left(|\mathsf{Even}_\kappa| + |\mathsf{Small}_\iota|, |\mathsf{Odd}_\kappa| + |\mathsf{Big}_\iota|\right)$, where these sets have been described in Section 6. For the $\mathsf{XOR}_5\text{-}\mathsf{MAJ}_5$ predicate, we get $t = \bar{t} = 6$ and for the $\mathsf{XOR}_6\text{-}\mathsf{MAJ}_8$ predicate, we get $t = 8$ while $\bar{t} = 9$.

9.2 Modulus size

We now move to bounding $\log q$, the bitsize of the modulus for the underlying ring. Using a rough approximation, we can easily place $\log q$ in the range $[439, 881]$. Using the FHE parameters standardized by the homomorphic encryption standard [2] (and in line with the security estimates from [14]), we must select a dimension $d = 32768$. Given d, we then compute an exact bound on the magnitude of the error term $\sum_{i=1}^{t} c_i \cdot e_i$, where the c_i are the coefficients of the polynomial

$$P_{\vec{x},L,\mathbf{Z}}(X) = P_{\perp}((L[x_i] \cdot X + \mathbf{Z}[x_i])_{1 \le i \le \kappa+\iota}) + P_{\top}((L[x_i] \cdot X + \mathbf{Z}[x_i])_{1 \le i \le \kappa+\iota}),$$

where $\mathbf{Z}$ packs $|S|$ wPRF keys $\mathbf{z}_i$ in the coefficients indexed by a t-sum-free set S of ring elements. In this polynomial, the entries of L are bounded by $B = 2^{\sigma} \cdot B_{\alpha} \cdot |S|$, where we set the statistical security parameter σ to 64 and the bound B_{α} to 1 (in line with standardized parameter settings for ring-LWE in FHE libraries, that use a ternary secret). Looking ahead, all our choices of packing subset S will satisfy $|S| \le 16$. This puts a bound on L's coefficient at 68 bits.

Writing the polynomial coefficients explicitly for both our fast and small candidates makes it clear that the term $c_t \cdot e_t$ largely dominates the sum $\sum_{i=1}^{t} c_i \cdot e_i$: using that $\kappa, \iota \le 8$ for both our candidates, every coefficient c_i is a sum of a small numbers of terms of the form $\prod_j L_j \cdot \prod_j z_j$ (with different ranges for the products), where there are i terms L_j, $t - i$ terms z_j, and each L_j (resp. z_j) is a sum of at most 8 terms $L[x_i]$ (resp. $\mathbf{Z}[x_i]$). Due to the large gap between the size of $L[x_i]$'s coefficients versus $\mathbf{Z}[x_i]$'s coefficient, the coefficients of c_t largely dominate all other c_i's. Each "L_j" term being the sum of at most 8 $L[x_i]$, it has coefficients bounded by 71 bits. This translates to a bound on the size of c_t's coefficients of $2^{71 \cdot t} \cdot d^{t-1}$ (a factor d overhead occurs after each ring multiplication due to the polynomial modular reduction), hence on the size of $c_t \cdot e_t$'s coefficient of $42 \cdot 2^{t \cdot 86 - 15}$ (using $d = 2^{15}$). Eventually, the modulus q should be $2^{\lambda} \cdot B$-times larger than this bound to ensure correctness when locally rounding to a λ-bit modulus p. Wrapping up, this yields the following choice of $\log q$ for our two candidates:

- **Fast:** $\log q = 698$ $(t = 6)$
- **Small:** $\log q = 956$ (using $\max(t, \bar{t}) = \bar{t} = 9$)

The modulus size for the fast candidate is well below the bound of $\log q = 881$ for which ring-LWE is conjectured to provide 128 bits of security at dimension $d = 2^{15}$ using a ternary secret. While the modulus size for the small candidate is above the bound, [14, Table 12] estimates that ring-LWE at dimension $d = 2^{15}$ provides at least 100 bits of security up to $\log q = 1113$. We will use this slightly lower security parameter for the small candidate in the rest of this section. We note that alternatively, one can also set $B = 2^{54}$ instead of 2^{64} to obtain $\log q = 866$ for the small candidates, restoring the 128-bit security margin at the cost of increasing the statistical correctness error (using $B = 2^{54}$ yields a $1/2^{22}$ one-time statistical failure probability of key derivation, and a $1/2^{54}$ per-OT statistical failure probability).

9.3 Selecting the packing subset

Having selected the ring $R = \mathbb{Z}_q[X]/(X^d + 1)$, we move on to finding a suitable t-sum-free set S of indices from $[0, r]$. To select r, we use the following rule of thumb: at a fixed λ-bit security level, $\log q$ and d scale linearly (a factor-2 increase in d translated to a factor-2 increase in the maximum bitsize $\log q$ of the supported modulus) [44, Table 8.a]. Given that $d = 2^{15}$ supports up to $\log q = 881$, for a fixed $\log q < 881$ this translates to a theoretical minimal 128-bit-secure dimension $d' = d \cdot \log q / 881$, and a theoretical minimal 100-bit secure dimension $d' = d \cdot \log q / 1113$. For our parameters and using the security bounds from [14, Table 12],

- **Fast** $(\max(t, \bar{t}) = 6)$**:** $d' = 25961$ plausibly suffices for 128 bits of security.
- **Small** $(\max(t, \bar{t}) = 9)$**:** $d' = 28145$ plausibly suffices for 100 bits of security.

After truncating r coefficients on both ends, the public element M is left with $d - 2r$ random entries. In particular, one can leverage these 0-coefficients to reduce the TSP-RLWE instance to a structured instance of LWE with a banded circulant matrix. It seems plausible that this banded circulant structure can be exploited in standard LWE attacks to reduce the cost of the search, bringing it on par to the cost

of attacking LWE with dimension $d - 2r$. For each t, this gives us a "maximally supported r" equal to $(d - d')/2$. For $\max(t, \bar{t}) = 6$, this translates to $r \leq 3403$, and for $\max(t, \bar{t}) = 9$ to $r \leq 2311$.

We then turn to selecting t-sum-free subsets. To this end, we run a Python script that orders randomly the elements of $[0, \lfloor r/t \rfloor]$, and greedily adds elements to a candidate set S, checking everytime that the new element does not break the t-sum-freeness of the candidate. To perform this check, we leverage the observation that for any $s \in S$, the number of ways to write $s \cdot t$ as a sum of t elements of S (with repetition) is the coefficient of the monomial $X^{t \cdot s}$ in the polynomial $(\sum_{s \in S} X^s)^t$. Using fast arithmetic to compute $(\sum_{s \in S} X^s)^t$ lets us check efficiently the condition for all $s \in S$. Then, we run a randomized restart process to iteratively improve the best candidate S, using a new random ordering of $[0, \lfloor r/t \rfloor]$, and let the Python script run for a maximum of 200 seconds (parallelized on 4 threads) for each choice of r (and t). We keep a new candidate S either when it is larger than the best one so far, or when it has the same size but a smaller largest element (allowing to use a smaller r for this candidate). We report on Table 3 the best set sizes found for each t and various choices of r below (or slightly above) the bound of $(d - d')/2$. These parameters represent a security-efficiency tradeoff: using a larger $|S|$ yields better throughput at the cost of a more aggressive assumption (note that using r above $(d - d')/2$ can be envisioned only if one is willing to settle for slightly less than 128 bits of security).

Table 3: Size of the best set S found for various values of the maximum degree $\max(t, \bar{t})$. The value r is computed as $\max(t, \bar{t})$ times the largest element of the corresponding set S.

$\max(t, \bar{t}) = 6$	$\lvert S \rvert$	9	10	11	12	14	15	16
	r	552	804	1140	1386	2886	4980	5574
$\max(t, \bar{t}) = 9$	$\lvert S \rvert$	6	8	9	10			
	r	342	693	999	1836			

9.4 Counting operations

We provide a detailed estimation of the efficiency of $\mathsf{PKPCF.Eval}(\mathsf{b}, k_{\mathsf{b}}, \mathbf{x})$ for $\mathsf{b} = 0$ (sender case) and $\mathsf{b} = 1$ (receiver case).

9.4.1 Ring structure.

Recall that we operate over the ring $R = \mathbb{Z}_q[X]/(X^d + 1)$ with $d = 2^{15}$ and $\log q \in \{698, 956\}$. We assume that the modulus q is of the form $q = \prod_i q_i$ where the q_i are distinct less-than-52-bit primes. For $\log q = 698$, this gives us 14 q_i's, and for $\log q = 956$, this gives us 19 q_i's. Using a product of primes less than 64-bit long is the typical choice of FHE libraries (e.g. SEAL enforces primes below 61 bits) to benefit from parallelization and vectorized operations when representing an element $x \in \mathbb{Z}_q$ in RNS form $(x \bmod q_1, x \bmod q_2, \cdots)$. We assume a smaller bound of 52 bits to align with the requirements of [14] for using Intel AVX512-IFMA52 implementations.

9.4.2 Operations.

We write NTT for the forward NTT operation (on inputs represented with coefficients in RNS form), iNTT for the inverse NTT operation, $\mathsf{pw\text{-}mult}$ for pointwise-multiplication in the NTT domain, and $\mathsf{pw\text{-}add}$ for pointwise-addition in the NTT domain. We also write RNS to denote conversion to RNS form $x \mapsto (x \bmod q_1, x \bmod q_2, \cdots)$ and CRT to denote the inverse operation. We will extrapolate the cost of the dominant operations using the benchmarks from [14] reproduced on Table 4.

9.4.3 Preprocessing.

We assume that the parties initially store their respective keys k_0 and k_1 as a list of elements of the ring R in *coefficient representation*, and *without using RNS decomposition*. While precomputing NTTs and RNS decomposition on the keys could significantly improve the runtime, it would have a significant storage cost: the size of k_0 and k_1 is largely dominated by the sizes of $\mathbf{k}$ and $\boldsymbol{\ell}$, which are vectors of ring elements with 71-bit coefficients. Switching to RNS form would require using 14×64 bits or 19×64

Table 4: Benchmarks from [14] for NTT, inverse-NTT, and pointwise multiplications with a 50-bit modulus. All benchmarks run single-threaded on a single core of a 3rd Gen Intel Xeon® Scalable Processors Platinum 8360Y 2.4GHz processor with 64GB of RAM and 72 cores. NTT and inverse-NTT use the Intel AVX512-IFMA52 implementation, and pointwise multiplication uses the Intel AVX512-DQ Float implementation. All numbers in microseconds. The numbers in the last row (highlighted in red) for $d = 32768$, are conservatively extrapolated from the $d \log d$ (resp. d) scaling behavior of NTTs (resp. pointwise operations) in line with the concrete scaling observed between the lower dimensions.

Dimension	Forward NTT	Inverse NTT	Pointwise Multiplication
1024	1.25	1.23	0.251
4096	5.81	5.72	1.08
16384	33.1	32.4	4.58
32768	83	81	9.6

bits per $\mathbb{Z}_q$ element (reserving a 64-bit limb for each residue) depending on the bound on $\log q$ (698 or 956). This would increase the storage requirement by 13× and 17× for our two parameter settings respectively. Given that the key size is already significant (several GB), we stick to the more realistic storage in non-RNS coefficient-wise representation.

However, we assume that the parties precomputed the following after key derivation: the sender precomputes $\tilde{M} := \mathsf{NTT}(\mathsf{RNS}(M)), \tilde{\Delta} := \mathsf{NTT}(\mathsf{RNS}(M + \Delta))$, as well as $\tilde{\mu}_i := \mathsf{NTT}(\mathsf{RNS}((1/\mu^i))$ for $i = 1$ to $t_{\mathsf{max}} := \max(t, \bar{t})$. This significantly speeds up the computation of $\mathsf{PKPCF.Eval}(0, k_0, \mathbf{x})$ while having essentially no impact on storage. Similarly, we assume that the receiver precomputes $\tilde{s}_j := \mathsf{NTT}(\mathsf{RNS}(-j \cdot 1_S))$ and $\tilde{\mathcal{M}}_j := \mathsf{NTT}(\mathsf{RNS}(\mathcal{M}_j))$ for $j = 1$ to t_{max}. Below, we will use $\tilde{x}$ to denote x moved to the NTT domain after RNS decomposition.

9.4.4 Sender.

Recall that the sender must compute

$$(y_1^{(\mathsf{b})}, \cdots, y_{|S|}^{(\mathsf{b})}) := \mathsf{coeffs}_{t \cdot S}\left(\lfloor (M + \Delta) \cdot \tilde{P}_{\mathbf{x}}^{(\mathsf{b})}(\mathbf{k}/\mu) \pmod q \rceil_2\right),$$

where $\tilde{P}^{(0)}$ is $\tilde{P}^{(0)} = \mathsf{Adapt}(P_0)$ where P_0 computes the $\mathsf{XOR\text{-}MAJ}$ predicate, and $\tilde{P}^{(1)}$ is $\mathsf{Adapt}(P_1)$ where P_1 computes the $\neg\mathsf{XOR\text{-}MAJ}$ predicate. A naive evaluation strategy would first fetch all coefficients of $\mathbf{k}/\mu$ used by $\tilde{P}_{\mathbf{x}}^{(\mathsf{b})}$, apply NTT to them, and operate in the NTT domain from there. However, it is actually more efficient to delay the NTT evaluations after computing the partials sums:

- Compute $K_{\mathsf{XOR}} := \sum_{i=1}^{\kappa} \mathbf{k}[x_i]$ and $K_{\mathsf{MAJ}} := \sum_{i=1}^{\iota} \mathbf{k}[x_{\kappa+i}]$, for a total of $\kappa + \iota - 2$ additions of elements of R in coefficient representation, with coefficients bounded by 68 bits (negligible cost).
- Compute $\tilde{K}_{\mathsf{XOR}} := \mathsf{NTT}(\mathsf{RNS}(K_{\mathsf{XOR}}))$ and $\tilde{K}_{\mathsf{MAJ}} := \mathsf{NTT}(\mathsf{RNS}(K_{\mathsf{MAJ}}))$ (cost dominated by two forward NTT).
- Compute $\tilde{X}_j := \mathsf{pw\text{-}add}(\mathsf{pw\text{-}mult}(\tilde{K}_{\mathsf{XOR}}, \tilde{\mu}_1), \tilde{s}_j)$ for $j = 1$ to κ and $\tilde{Y}_j := \mathsf{pw\text{-}add}(\mathsf{pw\text{-}mult}(\tilde{K}_{\mathsf{MAJ}}, \tilde{\mu}_1), \tilde{s}_j)$ for $j = 1$ to ι (two calls to $\mathsf{pw\text{-}mult}$, plus $\kappa + \iota - 2$ calls to $\mathsf{pw\text{-}add}$ with negligible cost).
- Compute $\tilde{P}_{\mathsf{trm}}(\tilde{X}_1, \cdots, \tilde{X}_\kappa) = \prod_{j \in \mathsf{S}_{\mathsf{trm}}} X_j$ for $\mathsf{trm} = \mathsf{XOR}_\kappa, \neg\mathsf{XOR}_\kappa$, and $\tilde{P}_{\mathsf{trm}}(\tilde{Y}_1, \cdots, \tilde{Y}_\iota) = \prod_{j \in \mathsf{S}_{\mathsf{trm}}} X_j$ for $\mathsf{trm} = \mathsf{MAJ}_\iota, \neg\mathsf{MAJ}_\iota$, using $\mathsf{pw\text{-}mult}$ to compute the multiplications pointwise. $(2 \cdot (\kappa + \iota - 2)$ calls to $\mathsf{pw\text{-}mult})$.
- Compute $\tilde{p}_\top := \mathsf{pw\text{-}mult}(\tilde{P}_{\mathsf{XOR}_\kappa}(\tilde{X}_1, \cdots, \tilde{X}_\kappa), \tilde{P}_{\mathsf{MAJ}_\iota}(\tilde{Y}_1, \cdots, \tilde{Y}_\iota))$ and $\tilde{p}_\bot$ similarly with $P_{\neg\mathsf{XOR}_\kappa}$ and $P_{\neg\mathsf{MAJ}_\iota}$ (2 calls to $\mathsf{pw\text{-}mult}$).
- Compute $v := \mathsf{iNTT}(\mathsf{pw\text{-}mult}(\tilde{\Delta}, \mathsf{pw\text{-}add}(\tilde{p}_\top, \tilde{p}_\bot)))$ (one call to iNTT, one call to $\mathsf{pw\text{-}mult}$, plus a call to $\mathsf{pw\text{-}add}$).

- Extract the coefficients (in RNS form) $(w_1, \cdots, w_{|S|}) := \mathsf{coeffs}_{t \cdot S}(v)$ and output $(y_1, \cdots, y_{|S|}) = (\lfloor \mathsf{CRT}(w_1) \rceil, \cdots, \lfloor \mathsf{CRT}(w_{|S|}) \rceil)$ ($|S|$ calls to CRT and rounding – negligible cost due to the small size of S).

This gives us a total of 2 calls to NTT, $2 \cdot (\kappa + \iota) - 1$ calls to $\mathsf{pw\text{-}mult}$, and one call to iNTT, plus other costs that are comparatively much smaller: $\kappa + \iota - 2$ additions in the coefficient domain (with small coefficients), $\kappa + \iota + 1$ calls to $\mathsf{pw\text{-}add}$, and $|S|$ calls to CRT (on individual $\mathbb{Z}_q$ elements) and rounding. Using the numbers from Table 4 (note that they correspond to operations with a single 50-bit modulus, hence we must multiply all costs with the number of terms in the RNS decomposition), this yields the following estimates:

- **Fast:** $14 \cdot (2 \cdot 83 + 19 \cdot 9.6 + 81) = 6$ms, which we round up to 7ms by adding a 15% overhead to account for other comparatively smaller costs (additions, $\mathsf{pw\text{-}add}$, CRT, rounding, moving data around). This is for a single CPRF evaluation, hence we get a total of 14ms for the full $\mathsf{PKPCF.Eval}(0, k_0, \mathbf{x})$ (that involves two CPRF evaluations).
- **Small:** $19 \cdot (2 \cdot 83 + 27 \cdot 9.6 + 81) = 9.6$ms, which we round up to 11ms by adding a 15% overhead, hence 22ms for the full $\mathsf{PKPCF.Eval}(0, k_0, \mathbf{x})$.

9.4.5 Receiver.

We recall the computation for $\mathsf{PKPCF.Eval}(1, k_1, \mathbf{x})$. For $\mathsf{b} = 0, 1$,

1. Compute $(c_1^{(\mathsf{b})}, \ldots, c_t^{(\mathsf{b})})$ such that $\tilde{P}_{\mathbf{x}}^{(\mathsf{b})}(\boldsymbol{\ell}/\mu + \mathbf{Z}) = \tilde{P}_{\mathbf{x}}(\mathbf{Z}) + \sum_{i=1}^{t} c_i^{(\mathsf{b})} (1/\mu)^i$.
2. Compute $(y_1^{(\mathsf{b})}, \cdots, y_{|S|}^{(\mathsf{b})}) := \mathsf{coeffs}_{t \cdot S}\left(\left\lfloor \sum_{i=1}^{t} c_i \cdot \mathcal{M}_i \pmod q \right\rceil_2 \right)$.
3. Output $(y_1^{(\mathsf{b}_1)}, \cdots, y_{|S|}^{(\mathsf{b}_{|S|})})$ where $\mathsf{b}_i = 0$ if $P_{\mathbf{x}}(\mathbf{z}_i) = 0$, and 1 else.

As for the sender, we improve over a naive evaluation by delaying some forward NTT operations. Concretely,

1. Compute $L_{\mathsf{XOR}} := \sum_{i=1}^{\kappa} \boldsymbol{\ell}[x_i]$ and $L_{\mathsf{MAJ}} := \sum_{i=1}^{\iota} \boldsymbol{\ell}[x_{\kappa+i}]$, for a total of $\kappa + \iota - 2$ additions of elements of R in coefficient representation, with coefficients bounded by 68 bits (negligible cost).
2. Compute $\Theta_{\mathsf{XOR}}^{(j)} := \sum_{i=1}^{\kappa} \mathbf{Z}[x_i] - j \cdot 1_S$ for $j = 1$ to κ and $\Theta_{\mathsf{MAJ}}^{(j)} := \sum_{i=1}^{\iota} \mathbf{Z}[x_{\kappa+i}] - j \cdot 1_S$ for $j = 1$ to ι, for a total of $2 \cdot (\kappa + \iota) - 2$ additions of $|S|$-sparse elements of R with coefficients bounded by $\kappa + \iota$ (negligible cost).
3. Compute $\tilde{\Theta}_{\mathsf{XOR}}^{(j)} := \mathsf{NTT}(\mathsf{RNS}(\Theta_{\mathsf{XOR}}^{(j)}))$ and $\tilde{\Theta}_{\mathsf{MAJ}}^{(j)} := \mathsf{NTT}(\mathsf{RNS}(\Theta_{\mathsf{MAJ}}^{(j)}))$. Using the precomputed terms $\tilde{s}_j = \mathsf{NTT}(\mathsf{RNS}(-j \cdot 1_S))$, this requires two calls to NTT (on $\sum_{i=1}^{\kappa} \mathbf{Z}[x_i]$ and $\sum_{i=1}^{\iota} \mathbf{Z}[x_i]$) and $\kappa + \iota$ calls to $\mathsf{pw\text{-}add}$ (to add the $-j \cdot 1_S$ terms in the NTT domain).
4. Compute $\tilde{L}_{\mathsf{XOR}} := \mathsf{NTT}(\mathsf{RNS}(L_{\mathsf{XOR}}))$ and $\tilde{L}_{\mathsf{MAJ}} := \mathsf{NTT}(\mathsf{RNS}(L_{\mathsf{MAJ}}))$ (cost dominated by two forward NTT).
5. Iteratively compute all monomials of the form $\tilde{L}_{\mathsf{XOR}}^i, \tilde{L}_{\mathsf{MAJ}}^j$ such that $i \leq \max(|\mathsf{Even}_\kappa|, |\mathsf{Odd}_\kappa|)$, $j \leq (|\mathsf{Small}_\iota|, |\mathsf{Big}_\iota|)$. A quick calculation (enumerating all monomials of this form) yields 13 calls to $\mathsf{pw\text{-}mult}$ for $\mathsf{XOR}_5\text{-}\mathsf{MAJ}_5$ and 27 calls to $\mathsf{pw\text{-}mult}$ for $\mathsf{XOR}_6\text{-}\mathsf{MAJ}_8$.
6. Use dynamic programming to jointly construct all coefficients $\tilde{c}_i$. A short Python script produces a count of 21 calls to $\mathsf{pw\text{-}mult}$ and 15 calls to $\mathsf{pw\text{-}add}$ for $\mathsf{XOR}_5\text{-}\mathsf{MAJ}_5$, and 45 calls to $\mathsf{pw\text{-}mult}$, 36 calls to $\mathsf{pw\text{-}add}$ for $\mathsf{XOR}_6\text{-}\mathsf{MAJ}_8$ (using the fact that the $\tilde{L}$ terms have already been computed).
7. Given the $\tilde{c}_i$ and the precomputed $\tilde{\mathcal{M}}_i$, compute $\tilde{v} := \sum_i \tilde{c}_i \cdot \mathcal{M}_i$ using $t_{\mathsf{max}} = \max(t, \bar{t})$ calls to $\mathsf{pw\text{-}mult}$ and $t_{\mathsf{max}} - 1$ calls to $\mathsf{pw\text{-}add}$.
8. Compute $v := \mathsf{iNTT}(\tilde{v})$ (one call to iNTT).

9. Extract the coefficients (in RNS form) $(w_1, \cdots, w_{|S|}) := \mathsf{coeffs}_{t \cdot S}(v)$ and output $(y_1, \cdots, y_{|S|}) = (\lfloor \mathsf{CRT}(w_1) \rceil, \cdots, \lfloor \mathsf{CRT}(w_{|S|}) \rceil)$ ($|S|$ calls to CRT and rounding – negligible cost due to the small size of S).

Overall, we get $\kappa + \iota - 2$ additions of R-elements with small coefficients, $2 \cdot (\kappa + \iota) - 2$ additions of $|S|$-sparse elements, 4 forward-NTT, $\kappa + \iota + t_{\mathsf{max}} - 1$ calls to pw-add, t_{max} calls to pw-mult, and one call to iNTT (followed with $|S|$ CRT and roundings) for steps 1 to 4 and 7 to 9, to which we add 34 pw-mult and 15pw-add for XOR_5-MAJ_5, and 72 pw-mult and 36 pw-add for XOR_6-MAJ_8. Using the numbers from Table 4, this yields the following estimates:

- **Fast:** $14 \cdot (4 \cdot 83 + 40 \cdot 9.6 + 81) = 11.2$ms, which we round up to 13ms by adding a 15% overhead to account for other comparatively smaller costs. This is for a single CPRF evaluation, hence we get a total of 26ms for the full $\mathsf{PKPCF.Eval}(1, k_1, \mathbf{x})$ (that involves two CPRF evaluations).
- **Small:** $19 \cdot (4 \cdot 83 + 81 \cdot 9.6 + 81) = 22.6$ms, which we round up to 26ms by adding a 15% overhead, hence 52ms for the full $\mathsf{PKPCF.Eval}(1, k_1, \mathbf{x})$.

9.4.6 Final results.

We report the results in Table 2. This estimated runtime corresponds to the estimated cost for the receiver to perform a batch of $|S|$ (packed) PK-PCF evaluations. For each value of t, we select two possible values of $|S|$ from Table 3 corresponding to a *conservative* choice ($r \approx (d - d')/4$, leaving some security margin) and *aggressive* choice ($r \approx (d - d')/2$, not leaving much security margin). We set a target security parameter of 128 bits for the fast candidate, and of 100 bits for the small candidate (as outlined in Section 9.2, one could alternatively reduce B and settle for a larger statistical failure probability to increase the conjectured security level back to 128 bits). We also report in the table the bound on the maximum number of OTs for each parameter setting, computed as $|S|\times$ the stretch of the wPRF (note that in addition to improving efficiency, our packing strategy also improves the stretch by a factor $|S|$, as $|S|$ instances of the wPRF are computed in parallel on independent keys).

9.5 Key Sizes of Our PK-PCF

We estimate the key sizes of our scheme by analyzing Construction 2. The sender's public key size (referred to as pk_0 in the construction) can be calculated as $t_{\mathsf{max}} \cdot d \cdot \log q + (2\sqrt{n} + 1) \cdot d \cdot \log p$ bits, where the parameters $t_{\max}$ refers to maximum of the degrees of polynomials for XOR_κ-MAJ_ι and $\neg\mathsf{XOR}_\kappa$-MAJ_ι, n denotes the input length, d denotes the degree of the rings, q and p are moduli of rings for which we want the secret-power ring-LWE and public-power ring-LWE assumptions (respectively) to be hard. The receiver's key size (referred to as pk_1) can be computed as $\sqrt{n} \cdot d \cdot \log p$ bits. Moreover, we can calculate $p = 2^{\lambda/2} \cdot B \cdot (dB_{e'}^2 + \sqrt{n} B_{e'}) = 2^\lambda \cdot B_e \cdot (dB_{e'}^2 + \sqrt{n} B_{e'})$, where $B_e, B_{e'}$ denotes the magnitude of the error vectors.

There are two more parameters of interest: the *decompressed* key sizes (referring to k_0 and k_1 in the construction) and the running times of *decompression* (running times of $\mathsf{KeyDer}(0, \cdot, \cdot)$ and $\mathsf{KeyDer}(1, \cdot, \cdot)$). The sender's decompressed key size (referred to as k_0) can be computed as $n \cdot d \cdot \log B + d \cdot \log q + d \cdot 1$ bits and the receiver's decompressed key size (referred to as k_1) can be computed as $n + n \cdot d \cdot \log B + t_{\mathsf{max}} \cdot d \cdot \log q$ bits.

To estimate the decompression time (running times of $\mathsf{KeyDer}(0, \cdot, \cdot)$ and $\mathsf{KeyDer}(1, \cdot, \cdot)$), we rewrite the KeyDer algorithms in terms of low-level operations (NTT, iNTT, pw-add, pw-mult, CRT) and rely as before on the benchmarks from Table 4.

9.5.1 Sender.

We review the algorithm $\mathsf{PKPCF.KeyDer}(0, \mathsf{sk}_0, \mathsf{pk}_1)$: on input $(\mathsf{sk}_0, \mathsf{pk}_1) = (M, \mu, \nu, (v_i)_{i \in [\sqrt{n}]})$,

1. Compute $\tilde{v}_i := \mathsf{NTT}(\mathsf{RNS}(v_i))$ for $i \in [\sqrt{n}]$, $\tilde{N} := \mathsf{NTT}(\mathsf{RNS}(N))$, and $\tilde{\nu} := \mathsf{NTT}(\mathsf{RNS}(\nu))$.
   $\implies \sqrt{n} + 2$ calls to NTT
2. Set $\tilde{N}_{\sqrt{n}+1} := 1$. For $j = \sqrt{n}$ to 1, compute $\tilde{N}_j := \mathsf{pw\text{-}mult}(\tilde{N}, \tilde{N}_{j+1})$.
   $\implies \sqrt{n}$ calls to pw-mult

3. For $i \in [\sqrt{n}]$:
   (a) For $j \in [\sqrt{n}]$:
      i. Compute $\tilde{s}_{ij}^{(0)} := \mathsf{pw\text{-}mult}(\tilde{v}_i, \tilde{N}_j, \tilde{\nu})$.
      $\implies$ 2 calls to pw-mult
      ii. View $\lfloor s_{ij}^{(0)} \rceil_B$ as an element k_{i_j} of R_q.
      $\implies$ 1 call to iNTT, d calls to CRT and roundings (1/coefficient)
   (b) Let $\mathbf{k}_i := (k_{i_1}, \ldots, k_{i_{\sqrt{n}}})$
4. Set $\mathbf{k} := (\mathbf{k}_1, \ldots, \mathbf{k}_{\sqrt{n}})$.
5. Output $k_0 := (\mathbf{k}, M, \mu)$.
   $\implies$ Total: $\sqrt{n}+2 \times$ NTT, $2n \times$pw-mult, $n \times$ iNTT, $d \cdot n \times$ CRT and roundings

From the above, we expect the cost of the $d \cdot n$ CRT reconstructions to significantly impact the overall cost. Unfortunately, the Intel HEXL reference paper [15] does not provide benchmark for the cost of CRT reconstruction. We estimate it as follows: we assume that the modulus p (with $\log p \approx 156$ is a product of 3 52-bit primes $p_1 \cdots p_3$ (this will speedup the NTT and iNTT as it enables the use of Intel AVX512-IFMA52). We assume that $p_1^{-1} \bmod p_2$ and $(p_1 p_2)^{-1} \bmod p_3$ have been precomputed ahead of time. Using Garner's iterative recombination method amounts to 5 single-limb multiplications ($64 \times 64 \to 128$, using the canonical mulq/mulx multiplier on x86). Assuming around 3 cycles/multiplication, and estimating roughly additions and modular reductions as representing 30% of the total (multiplications should dominate the overall cost, other costs are not negligible), yields a count of about 27 cycles / CRT reconstruction. Using the AVX512 instructions, as all instructions in Garner's recombination can be fully vectorized, allows processing 8 coefficients in parallel. Using $d = 32768$ and assuming a 3.0GHz CPU, this amounts to $27 \cdot 2^{15}/(8 \cdot 3 \cdot 10^3) \approx 37\mu$s per full CRT reconstruction, which we round up to 40μs to account for rounding and other costs we might have missed (such as moving data around, etc). Of course, these are very rough estimates: real numbers from an implementation could differ significantly, but the estimate provides a good sense of the magnitude one can expect.

Combining this estimate with the benchmark numbers from Table 4 (multiplied by 3 to account for the three limbs of our modulus) yields a total cost of $(\sqrt{n}+2) \cdot (3 \cdot 83) + 2n \cdot (3 \cdot 9.6) + n \cdot (3 \cdot 81) + n \cdot 40$ microseconds, which yields for each of our candidate:

- **Fast** ($n = 2^{17.5}$): $\approx$ 63s
- **Small** ($n = 2^{14}$): $\approx$ 5.6s

9.5.2 Receiver.

We review the algorithm $\mathsf{PKPCF.KeyDer}(1, \mathsf{sk}_1, \mathsf{pk}_0)$: on input $\mathsf{sk}_1 = (\mathbf{z}, (u_0^i, u_1^i)_{i \in [\sqrt{n}]})$ and $\mathsf{pk}_0 = ((\mathcal{M}_i)_{i \in [t_{\max}]}, (\mathcal{N}_j)_{j \in [0, 2\sqrt{n}]})$,

1. Compute $\tilde{u}_0^i := \mathsf{NTT}(\mathsf{RNS}(u_0^i))$, $\tilde{u}_1^i := \mathsf{NTT}(\mathsf{RNS}(u_1^i))$ and $\tilde{u}_0^i := \mathsf{NTT}(\mathsf{RNS}(u_0^i))$ for for $i = 1$ to $\sqrt{n}$, and $\tilde{\mathcal{N}}_j := \mathsf{NTT}(\mathsf{RNS}(\mathcal{N}_j))$ for $j = 0$ to $2\sqrt{n}$.
   $\implies$ $4\sqrt{n}+1$ calls to NTT
2. For $i, j \in [\sqrt{n}]$:
   (a) For $j \in [\sqrt{n}]$:
      i. $\tilde{s}_{ij}^{(1)} := \langle (\tilde{u}_0^i, \tilde{u}_1^i, \mathbf{z}_i) \ , \ (\tilde{\mathcal{N}}_{\sqrt{n}-j}, \tilde{\mathcal{N}}_{\sqrt{n}-j+1}, \ldots, \tilde{\mathcal{N}}_{2\sqrt{n}-j+1}) \rangle \pmod p$.
      $\implies$ 2 calls to pw-mult and $\sqrt{n}/2+2$ calls to pw-add (in expectation)
      ii. View $\lfloor s_{ij}^{(1)} \rceil_B$ as an element ℓ_{i_j} of R_q.
      $\implies$ 1 call to iNTT, d calls to CRT and roundings (1/coefficient)
   (b) Let $\boldsymbol{\ell}_i = (\ell_{i_1}, \ldots, \ell_{i_{\sqrt{n}}})$
3. Set $\boldsymbol{\ell} := (\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_{\sqrt{n}})$.

4. Output $k_1 := (\mathbf{z}, \boldsymbol{\ell}, (\mathcal{M}_j)_{j\in[t_{\max}]})$.
   $\Longrightarrow$ Total: $4\sqrt{n}+1 \times$ NTT, $2n \times$ pw-mult, $n(\sqrt{n}/2+2) \times$ pw-add, $n \times$ iNTT, $dn \times$ CRT + rounding

From the above, only the number of calls to pw-add grows superlinearly in n. Hence, even though the cost of pw-add is typically small compared to NTT or pw-mult, we cannot ignore its contribution. Unfortunately, it is hard to obtain a precise estimate of its cost: element-wise addition is not benchmarked in the Intel HEXL reference paper [15], where it is only state that "In practice, element-wise addition and element-wise negation are typically much faster to compute than the types of multiplication.". However, back-of-the envelope calculations (using cycle counts of the low-level hardware instructions used for pw-add and pw-mult) suggest that pw-add should be in the range of $10\times$ to $20\times$ cheaper than pw-mult. We therefore use pw-mult $\approx 15 \times$ pw-add as a reasonable baseline. Combining this estimate with our previous CRT runtime estimates and the benchmark numbers from Table 4 (multiplied by 3 to account for the three limbs of our modulus) yields a total cost of $(4\sqrt{n}+1)\cdot(3\cdot83)+2n\cdot(3\cdot9.6)+n\cdot(\sqrt{n}/2+2)\cdot(3\cdot9.6/15)+n\cdot(3\cdot81)+n\cdot40$ microseconds, which yields for each of our candidate:

- **Fast** ($n = 2^{17.5}$): $\approx$ 141s
- **Small** ($n = 2^{14}$): $\approx$ 7.8s

9.5.3 Results.

We summarize the above in Table 5.

Table 5: Estimated public-key sizes, derived-key sizes and decompression times for both parties in both settings.

	Sender		
	Public-Key Size	Derived-Key Size	Decompression Time
$t_{\max} = 6$ (Fast)	576.5 MB	52.7 GB	63 seconds
$t_{\max} = 9$ (Small)	198.67 MB	4.66 GB	5.6 seconds
	Receiver		
	Public-Key Size	Derived-Key Size	Decompression Time
$t_{\max} = 6$ (Fast)	280.75 MB	52.7 GB	141 seconds
$t_{\max} = 9$ (Small)	83.47 MB	4.69 GB	7.8 seconds

10 Malicious Security of PK-PCF

In this section, we prove that any PK-PCF that directly generates standard OT correlations becomes automatically maliciously secure as soon as the public key setup is maliciously secure. The proof is almost immediate and is provided for completeness; the original observation of this phenomenon is due to [22]. Let $\mathcal{F}_{\mathsf{PKPCF}}$ denote the functionality (parametrized with a public-key PCF PKPCF and two parties P_0, P_1) that, on input `init` from both parties, runs $\mathsf{pp} \leftarrow \mathsf{PKPCF.Setup}(1^\lambda)$ and $(\mathsf{pk}_\sigma, \mathsf{sk}_\sigma) \leftarrow \mathsf{PKPCF.KeyGen}(\sigma)$ for $\sigma = 0, 1$. It returns $(\mathsf{pp}, \mathsf{sk}_0, \mathsf{pk}_1)$ to P_0 and $(\mathsf{pp}, \mathsf{sk}_1, \mathsf{pk}_0)$ to P_1. Consider the following standard derandomization protocol Π to convert (pseudo)random oblivious transfers into chosen oblivious transfer, where P_0 plays the role of the sender:

- P_0, P_1 send `init` to $\mathcal{F}_{\mathsf{PKPCF}}$ and receive $(\mathsf{pp}, \mathsf{sk}_0, \mathsf{pk}_1)$, $(\mathsf{pp}, \mathsf{sk}_1, \mathsf{pk}_0)$ respectively. Both parties initialize a counter ctr to 0.
- P_0 and P_1 locally compute $k_0 \leftarrow \mathsf{PKPCF.KeyDer}(0, \mathsf{sk}_0, \mathsf{pk}_1)$ as well as $k_1 \leftarrow \mathsf{PKPCF.KeyDer}(1, \mathsf{sk}_1, \mathsf{pk}_0)$ respectively.
- Given sender inputs $(s_0, s_1) \in \mathsf{b}^{2\ell}$ and selection bit $b \in \mathsf{b}$, the parties execute the following steps:

1. P_1 computes $(\sigma, r') \leftarrow \mathsf{PKPCF.Eval}(1, k_1, \mathsf{ctr})$ and sends $u = \sigma \oplus b$ to P_0.
2. P_0 computes $(r_0, r_1) \leftarrow \mathsf{PKPCF.Eval}(0, k_0, \mathsf{ctr})$ and sends $(m_0, m_1) = (s_u \oplus r_0, s_{1-u} \oplus r_1)$ to P_1.
3. P_1 outputs $m_\sigma \oplus r'$.
4. Both parties update $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$.

Claim 10. *Define $\mathcal{F}_{\mathsf{OT}}$ to be the following functionality: on input (s_0, s_1) from P_0 and b from P_1, it returns s_b to P_b. Then if* PKPCF *is a pseudorandom and secure PK-PCF (Definition 11),* Π *securely instantiates $\mathcal{F}_{\mathsf{OT}}$ in the $\mathcal{F}_{\mathsf{PKPCF}}$-hybrid model.*

Proof. When both parties are honest, the pseudorandomness of PKPCF guarantees that $r' = r_\sigma$, in which case $m_\sigma \oplus r' = s_{u \oplus \sigma} \oplus r_\sigma \oplus r' = s_{u \oplus \sigma} = s_b$.

When P_0 is corrupted, Sim simulates $\mathcal{F}_{\mathsf{PKPCF}}$ honestly and stores the PKPCF keys k_0, k_1. It samples $\tilde{u} \xleftarrow{\$} \mathsf{b}$ and sends it to P_0. Upon receiving (m_0, m_1), it computes $(r_0, r_1) \leftarrow \mathsf{PKPCF.Eval}(0, k_0, \mathsf{ctr})$ and extracts $(\tilde{s}_u, \tilde{s}_{1-u}) = (m_0 \oplus r_0, m_1 \oplus r_1)$. It sends $(\tilde{s}_0, \tilde{s}_1)$ to $\mathcal{F}_{\mathsf{OT}}$ on behalf of P_0. Security follows via a straightforward reduction to security of the PKPCF against party 0: reverse-sampling (σ, r') to be consistent with $(r_0, r_1) = \mathsf{PKPCF.Eval}(0, k_0, \mathsf{ctr})$ amounts to sampling σ uniformly at random and setting $r' = r_\sigma$, and replacing σ with a true random bit as in the $\mathcal{O}_1^{\mathsf{sec}}$ oracle from Figure 3 makes $u = \sigma \oplus b$ uniform.

When P_1 is corrupted, Sim simulates $\mathcal{F}_{\mathsf{PKPCF}}$ honestly and stores the PKPCF keys k_0, k_1. Upon receiving $u \in \mathsf{b}$, it computes $(\sigma, r') \leftarrow \mathsf{PKPCF.Eval}(1, k_1, \mathsf{ctr})$, sets $\tilde{b} \leftarrow \sigma \oplus u$, and sends $\tilde{b}$ to $\mathcal{F}_{\mathsf{OT}}$ on behalf of P_1. It receives an output $s_{\tilde{b}}$, sets $\tilde{m}_{1-\sigma} \xleftarrow{\$} \mathsf{b}^\ell$ and $\tilde{m}_\sigma \leftarrow s_{\tilde{b}} \oplus r'$. It sends $(\tilde{m}_0, \tilde{m}_1)$ to P_1. Security follows again by a straightforward reduction to security of the PKPCF against party 1: reverse-sampling (r_0, r_1) to be consistent with (σ, r') amounts to sampling $r_{1-\sigma}$ at random and setting $r_\sigma = r'$. Then, $\tilde{m}_\sigma = s_{\tilde{b}} \oplus r'$ is identical to the value m_σ computed by an honest P_1, and $m_{1-\sigma} = r_{1-\sigma} \oplus s_{1-\tilde{b}}$ is uniformly distributed over b^ℓ when replacing $r_{1-\sigma}$ with a true random string as in the $\mathcal{O}_1^{\mathsf{sec}}$ oracle from Figure 3. $\square$

References

[1] Abram, D., Roy, L., Scholl, P.: Succinct homomorphic secret sharing. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part VI. LNCS, vol. 14656, pp. 301–330. Springer, Cham (May 2024). https://doi.org/10.1007/978-3-031-58751-1_11

[2] Albrecht, M., Chase, M., Chen, H., Ding, J., Goldwasser, S., Gorbunov, S., Halevi, S., Hoffstein, J., Laine, K., Lauter, K., Lokam, S., Micciancio, D., Moody, D., Morrison, T., Sahai, A., Vaikuntanathan, V.: Homomorphic encryption standard. Cryptology ePrint Archive, Paper 2019/939 (2019), https://eprint.iacr.org/2019/939

[3] Applebaum, B., Bogdanov, A., Rosen, A.: A dichotomy for local small-bias generators. Journal of Cryptology **29**(3), 577–596 (Jul 2016). https://doi.org/10.1007/s00145-015-9202-8

[4] Applebaum, B., Kachlon, E.: Sampling graphs without forbidden subgraphs and unbalanced expanders with negligible error. In: Zuckerman, D. (ed.) 60th FOCS. pp. 171–179. IEEE Computer Society Press (Nov 2019). https://doi.org/10.1109/FOCS.2019.00020

[5] Applebaum, B., Lovett, S.: Algebraic attacks against random local functions and their countermeasures. In: Wichs, D., Mansour, Y. (eds.) 48th ACM STOC. pp. 1087–1100. ACM Press (Jun 2016). https://doi.org/10.1145/2897518.2897554

[6] Applebaum, B., Lovett, S.: Algebraic attacks against random local functions and their countermeasures. In: Wichs, D., Mansour, Y. (eds.) Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016. pp. 1087–1100. ACM (2016). https://doi.org/10.1145/2897518.2897554, https://doi.org/10.1145/2897518.2897554

[7] Applebaum, B., Raykov, P.: Fast pseudorandom functions based on expander graphs. In: Hirt, M., Smith, A.D. (eds.) Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9985, pp. 27–56 (2016). https://doi.org/10.1007/978-3-662-53641-4_2, https://doi.org/10.1007/978-3-662-53641-4_2

[8] Attrapadung, N., Matsuda, T., Nishimaki, R., Yamada, S., Yamakawa, T.: Constrained PRFs for NC^1 in traditional groups. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 543–574. Springer, Cham (Aug 2018). https://doi.org/10.1007/978-3-319-96881-0_19

[9] Baum, C., Braun, L., Delpech de Saint Guilhem, C., Klooß, M., Orsini, E., Roy, L., Scholl, P.: Publicly verifiable zero-knowledge and post-quantum signatures from VOLE-in-the-head. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part V. LNCS, vol. 14085, pp. 581–615. Springer, Cham (Aug 2023). https://doi.org/10.1007/978-3-031-38554-4_19

[10] Beaver, D.: Efficient multiparty protocols using circuit randomization. In: Advances in Cryptology - CRYPTO '91. pp. 420–432. Springer (1991)

[11] Beaver, D.: Precomputing oblivious transfer. In: Coppersmith, D. (ed.) CRYPTO'95. LNCS, vol. 963, pp. 97–109. Springer, Berlin, Heidelberg (Aug 1995). https://doi.org/10.1007/3-540-44750-4_8

[12] Beaver, D.: Correlated pseudorandomness and the complexity of private computations. In: 28th ACM STOC. pp. 479–488. ACM Press (May 1996). https://doi.org/10.1145/237814.237996

[13] Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Fast reed-solomon interactive oracle proofs of proximity. In: Chatzigiannakis, I., Kaklamanis, C., Marx, D., Sannella, D. (eds.) ICALP 2018. LIPIcs, vol. 107, pp. 14:1–14:17. Schloss Dagstuhl (Jul 2018). https://doi.org/10.4230/LIPIcs.ICALP.2018.14

[14] Biasioli, B., Kirshanova, E., Marcolla, C., Rovira, S.: A tool for fast and secure LWE parameter selection: the FHE case. Cryptology ePrint Archive, Report 2024/1895 (2024), https://eprint.iacr.org/2024/1895

[15] Boemer, F., Kim, S., Seifu, G., de Souza, F.D.M., Gopal, V.: Intel HEXL: Accelerating homomorphic encryption with intel AVX512-IFMA52. Cryptology ePrint Archive, Report 2021/420 (2021), https://eprint.iacr.org/2021/420

[16] Bogdanov, A., Qiao, Y.: On the security of goldreich's one-way function. In: Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, pp. 392–405. Springer (2009)

[17] Bombar, M., Bui, D., Couteau, G., Couvreur, A., Ducros, C., Servan-Schreiber, S.: FOLEAGE: $\mathbb{F}_4$OLE-based multi-party computation for boolean circuits. In: ASIACRYPT 2024, Part VI. pp. 69–101. LNCS, Springer, Singapore (Dec 2024). https://doi.org/10.1007/978-981-96-0938-3_3

[18] Bombar, M., Couteau, G., Couvreur, A., Ducros, C.: Correlated pseudorandomness from the hardness of quasi-abelian decoding. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part IV. LNCS, vol. 14084, pp. 567–601. Springer, Cham (Aug 2023). https://doi.org/10.1007/978-3-031-38551-3_18

[19] Boneh, D., Gentry, C., Gorbunov, S., Halevi, S., Nikolaenko, V., Segev, G., Vaikuntanathan, V., Vinayagamurthy, D.: Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 533–556. Springer, Berlin, Heidelberg (May 2014). https://doi.org/10.1007/978-3-642-55220-5_30

[20] Boneh, D., Waters, B.: Constrained pseudorandom functions and their applications. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 280–300. Springer, Berlin, Heidelberg (Dec 2013). https://doi.org/10.1007/978-3-642-42045-0_15

[21] Bouillaguet, C., Delaplace, C., Hamdad, M., Vergnaud, D.: Practical cryptanalysis of pseudorandom correlation generators based on quasi-abelian syndrome decoding. Cryptology ePrint Archive, Paper 2025/892 (2025), https://eprint.iacr.org/2025/892

[22] Boyle, E., Couteau, G., Gilboa, N., Ishai, Y.: Compressing vector OLE. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018. pp. 896–912. ACM Press (Oct 2018). https://doi.org/10.1145/3243734.3243868

[23] Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Resch, N., Scholl, P.: Correlated pseudorandomness from expand-accumulate codes. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part II. LNCS, vol. 13508, pp. 603–633. Springer, Cham (Aug 2022). https://doi.org/10.1007/978-3-031-15979-4_21

[24] Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Rindal, P., Scholl, P.: Efficient two-round OT extension and silent non-interactive secure computation. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019. pp. 291–308. ACM Press (Nov 2019). https://doi.org/10.1145/3319535.3354255

[25] Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Scholl, P.: Efficient pseudorandom correlation generators: Silent OT extension and more. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. LNCS, vol. 11694, pp. 489–518. Springer, Cham (Aug 2019). https://doi.org/10.1007/978-3-030-26954-8_16

[26] Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Scholl, P.: Correlated pseudorandom functions from variable-density LPN. In: 61st FOCS. pp. 1069–1080. IEEE Computer Society Press (Nov 2020). https://doi.org/10.1109/FOCS46700.2020.00103

[27] Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Scholl, P.: Efficient pseudorandom correlation generators from ring-LPN. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part II. LNCS, vol. 12171, pp. 387–416. Springer, Cham (Aug 2020). https://doi.org/10.1007/978-3-030-56880-1_14

[28] Boyle, E., Goldwasser, S., Ivan, I.: Functional signatures and pseudorandom functions. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 501–519. Springer, Berlin, Heidelberg (Mar 2014). https://doi.org/10.1007/978-3-642-54631-0_29

[29] Brakerski, Z., Tsabary, R., Vaikuntanathan, V., Wee, H.: Private constrained PRFs (and more) from LWE. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS, vol. 10677, pp. 264–302. Springer, Cham (Nov 2017). https://doi.org/10.1007/978-3-319-70500-2_10

[30] Brakerski, Z., Vaikuntanathan, V.: Constrained key-homomorphic PRFs from standard lattice assumptions - or: How to secretly embed a circuit in your PRF. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 1–30. Springer, Berlin, Heidelberg (Mar 2015). https://doi.org/10.1007/978-3-662-46497-7_1

[31] Briaud, P., Øygarden, M.: A new algebraic approach to the regular syndrome decoding problem and implications for PCG constructions. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 391–422. Springer, Cham (Apr 2023). https://doi.org/10.1007/978-3-031-30589-4_14

[32] Brzuska, C., Couteau, G., Egger, C., Karanko, P., Meyer, P.: Instantiating the hash-then-evaluate paradigm: Strengthening PRFs, PCFs, and OPRFs. Cryptology ePrint Archive, Report 2023/1145 (2023), https://eprint.iacr.org/2023/1145

[33] Bui, D., Couteau, G., Meyer, P., Passelègue, A., Riahinia, M.: Fast public-key silent OT and more from constrained Naor-Reingold. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part VI. LNCS, vol. 14656, pp. 88–118. Springer, Cham (May 2024). https://doi.org/10.1007/978-3-031-58751-1_4

[34] Canetti, R., Chen, Y.: Constraint-hiding constrained PRFs for NC^1 from LWE. In: Coron, J.S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part I. LNCS, vol. 10210, pp. 446–476. Springer, Cham (Apr / May 2017). https://doi.org/10.1007/978-3-319-56620-7_16

[35] Cheon, J.H.: Security analysis of the strong Diffie-Hellman problem. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 1–11. Springer, Berlin, Heidelberg (May / Jun 2006). https://doi.org/10.1007/11761679_1

[36] Cook, J., Etesami, O., Miller, R., Trevisan, L.: On the one-way function candidate proposed by goldreich. ACM Transactions on Computation Theory (TOCT) **6**(3), 14 (2014)

[37] Couteau, G., Devadas, L., Devadas, S., Koch, A., Servan-Schreiber, S.: QuietOT: Lightweight oblivious transfer with a public-key setup. In: ASIACRYPT 2024, Part II. pp. 197–231. LNCS, Springer, Singapore (Dec 2024). https://doi.org/10.1007/978-981-96-0888-1_7

[38] Couteau, G., Devadas, L., Devadas, S., Koch, A., Servan-Schreiber, S.: Quietot: Lightweight oblivious transfer with a public-key setup. In: Chung, K., Sasaki, Y. (eds.) Advances in Cryptology - ASIACRYPT 2024 - 30th International Conference on the Theory and Application of Cryptology and Information Security, Kolkata, India, December 9-13, 2024, Proceedings, Part II. Lecture Notes in Computer Science, vol. 15485, pp. 197–231. Springer (2024). https://doi.org/10.1007/978-981-96-0888-1_7, https://doi.org/10.1007/978-981-96-0888-1_7

[39] Couteau, G., Devadas, L., Hegde, A., Jain, A., Servan-Schreiber, S.: Multi-key homomorphic secret sharing. pp. 3–33. LNCS, Springer, Cham (Jun 2025). https://doi.org/10.1007/978-3-031-91092-0_1

[40] Couteau, G., Ducros, C.: Pseudorandom correlation functions from variable-density LPN, revisited. In: Boldyreva, A., Kolesnikov, V. (eds.) PKC 2023, Part II. LNCS, vol. 13941, pp. 221–250. Springer, Cham (May 2023). https://doi.org/10.1007/978-3-031-31371-4_8

[41] Couteau, G., Dupin, A., Méaux, P., Rossi, M., Rotella, Y.: On the concrete security of Goldreich's pseudorandom generator. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part II. LNCS, vol. 11273, pp. 96–124. Springer, Cham (Dec 2018). https://doi.org/10.1007/978-3-030-03329-3_4

[42] Couteau, G., Rindal, P., Raghuraman, S.: Silver: Silent VOLE and oblivious transfer from hardness of decoding structured LDPC codes. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part III. LNCS, vol. 12827, pp. 502–534. Springer, Cham, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84252-9_17

[43] Cryan, M., Miltersen, P.B.: On pseudorandom generators in nc 0. In: International Symposium on Mathematical Foundations of Computer Science. pp. 272–284. Springer (2001)

[44] Doan, T.V.T., Messai, M., Gavin, G., Darmont, J.: A survey on implementations of homomorphic encryption schemes. J. Supercomput. **79**(13), 15098–15139 (2023). https://doi.org/10.1007/S11227-023-05233-Z, https://doi.org/10.1007/s11227-023-05233-z

[45] Dodis, Y., Halevi, S., Rothblum, R.D., Wichs, D.: Spooky encryption and its applications. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part III. LNCS, vol. 9816, pp. 93–122. Springer, Berlin, Heidelberg (Aug 2016). https://doi.org/10.1007/978-3-662-53015-3_4

[46] Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS-Dilithium: A lattice-based digital signature scheme. IACR TCHES **2018**(1), 238–268 (2018). https://doi.org/10.13154/tches.v2018.i1.238-268, https://tches.iacr.org/index.php/TCHES/article/view/839

[47] Dupin, A., Méaux, P., Rossi, M.: On the algebraic immunity—resiliency trade-off, implications for goldreich's pseudorandom generator. Designs, Codes and Cryptography pp. 1–45 (2023)

[48] Fu, X., Li, M., Lyu, S., Liu, C.: Bit-fixing correlation attacks on goldreich's pseudorandom generators. Cryptology ePrint Archive, Paper 2024/1594 (2024), https://eprint.iacr.org/2024/1594

[49] Goldreich, O.: Candidate one-way functions based on expander graphs. Cryptology ePrint Archive, Paper 2000/063 (2000), https://eprint.iacr.org/2000/063

[50] Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions (extended abstract). In: 25th FOCS. pp. 464–479. IEEE Computer Society Press (Oct 1984). https://doi.org/10.1109/SFCS.1984.715949

[51] Goyal, V., O'Neill, A., Rao, V.: Correlated-input secure hash functions. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 182–200. Springer, Berlin, Heidelberg (Mar 2011). https://doi.org/10.1007/978-3-642-19571-6_12

[52] Guo, X., Liu, H., Huang, Z., Cui, H., Zhang, W., Hong, C., Wang, X., Yang, K., Yu, Y.: Dory: Streaming PCG with small memory. Cryptology ePrint Archive, Paper 2025/1660 (2025), https://eprint.iacr.org/2025/1660

[53] Hasgawa, H., Reijmers, T., Kusanagi, M.: Key sizes. Personal Communication (2025)

[54] Hasler, S., Reisert, P., Ralf, K.: Pseudorandom correlation functions from ring-lwr. In: ASIACRYPT 2025 (to appear) (2025)

[55] Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending oblivious transfers efficiently. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 145–161. Springer, Berlin, Heidelberg (Aug 2003). https://doi.org/10.1007/978-3-540-45146-4_9

[56] Ishai, Y., Li, H., Lin, H.: A unified framework for succinct garbling from homomorphic secret sharing. In: CRYPTO 2025, Part IV. pp. 390–425. LNCS, Springer, Cham (Aug 2025). https://doi.org/10.1007/978-3-032-01884-7_13

[57] Keller, M., Orsini, E., Scholl, P.: MASCOT: Faster malicious arithmetic secure computation with oblivious transfer. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016. pp. 830–842. ACM Press (Oct 2016). https://doi.org/10.1145/2976749.2978357

[58] Keller, M., Pastro, V., Rotaru, D.: Overdrive: Making SPDZ great again. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part III. LNCS, vol. 10822, pp. 158–189. Springer, Cham (Apr / May 2018). https://doi.org/10.1007/978-3-319-78372-7_6

[59] Kiayias, A., Papadopoulos, S., Triandopoulos, N., Zacharias, T.: Delegatable pseudorandom functions and applications. In: Sadeghi, A.R., Gligor, V.D., Yung, M. (eds.) ACM CCS 2013. pp. 669–684. ACM Press (Nov 2013). https://doi.org/10.1145/2508859.2516668

[60] Kolesnikov, V., Peceny, S., Raghuraman, S., Rindal, P.: Stationary syndrome decoding for improved PCGs. In: CRYPTO 2025, Part I. pp. 284–317. LNCS, Springer, Cham (Aug 2025). https://doi.org/10.1007/978-3-032-01855-7_10

[61] Li, Z., Xing, C., Yao, Y., Yuan, C.: Efficient pseudorandom correlation generators for any finite field. pp. 145–175. LNCS, Springer, Cham (Jun 2025). https://doi.org/10.1007/978-3-031-91092-0_6

[62] Liu, H., Wang, X., Yang, K., Yin, L., Yu, Y.: AGB 2.0: Refined algebraic attack against regular syndrome decoding for PCG applications. Cryptology ePrint Archive, Paper 2025/1476 (2025), https://eprint.iacr.org/2025/1476

[63] Liu, H., Wang, X., Yang, K., Yu, Y.: The hardness of LPN over any integer ring and field for PCG applications. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part VI. LNCS, vol. 14656, pp. 149–179. Springer, Cham (May 2024). https://doi.org/10.1007/978-3-031-58751-1_6

[64] Lombardi, A., Vaikuntanathan, V.: Limits on the locality of pseudorandom generators and applications to indistinguishability obfuscation. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS, vol. 10677, pp. 119–137. Springer, Cham (Nov 2017). https://doi.org/10.1007/978-3-319-70500-2_5

[65] Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23. Springer, Berlin, Heidelberg (May / Jun 2010). https://doi.org/10.1007/978-3-642-13190-5_1

[66] Méaux, P.: On the fast algebraic immunity of threshold functions. crypt. commun. 13 (5), 741–762 (2021)

[67] Méaux, P.: On the algebraic immunity of direct sum constructions. Discrete Applied Mathematics **320**, 223–234 (2022)

[68] Mossel, E., Shpilka, A., Trevisan, L.: On e-biased generators in NC0. In: 44th FOCS. pp. 136–145. IEEE Computer Society Press (Oct 2003). https://doi.org/10.1109/SFCS.2003.1238188

[69] Naor, M., Reingold, O.: Synthesizers and their application to the parallel construction of pseudorandom functions. In: 36th FOCS. pp. 170–181. IEEE Computer Society Press (Oct 1995). https://doi.org/10.1109/SFCS.1995.492474

[70] National Institute of Standards and Technology: FIPS 204: Module-Lattice-Based Digital Signature Standard (ML-DSA) (2024), https://doi.org/10.6028/NIST.FIPS.204, federal Information Processing Standards Publication 204

[71] Noval, M.C., Merz, S.P., Stählin, P., Ünal, A.: On the soundness of algebraic attacks against code-based assumptions. pp. 385–415. LNCS, Springer, Cham (Jun 2025). https://doi.org/10.1007/978-3-031-91095-1_14

[72] ODonnell, R., Witmer, D.: Goldreich's prg: evidence for near-optimal polynomial stretch. In: Computational Complexity (CCC), 2014 IEEE 29th Conference on. pp. 1–12. IEEE (2014)

[73] Oliveira, I.C., Santhanam, R., Tell, R.: Expander-based cryptography meets natural proofs. In: Blum, A. (ed.) ITCS 2019. vol. 124, pp. 18:1–18:14. LIPIcs (Jan 2019). https://doi.org/10.4230/LIPIcs.ITCS.2019.18

[74] Orlandi, C., Scholl, P., Yakoubov, S.: The rise of paillier: Homomorphic secret sharing and public-key silent OT. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part I. LNCS, vol. 12696, pp. 678–708. Springer, Cham (Oct 2021). https://doi.org/10.1007/978-3-030-77870-5_24

[75] Orsini, E., Smart, N.P., Vercauteren, F.: Overdrive2k: Efficient secure MPC over $\mathbb{Z}_{2^k}$ from somewhat homomorphic encryption. In: Jarecki, S. (ed.) CT-RSA 2020. LNCS, vol. 12006, pp. 254–283. Springer, Cham (Feb 2020). https://doi.org/10.1007/978-3-030-40186-3_12

[76] Peikert, C.: A decade of lattice cryptography. Cryptology ePrint Archive, Paper 2015/939 (2015), https://eprint.iacr.org/2015/939

[77] Peikert, C.: A decade of lattice cryptography. Found. Trends Theor. Comput. Sci. **10**(4), 283–424 (2016). https://doi.org/10.1561/0400000074, https://doi.org/10.1561/0400000074

[78] Peikert, C., Shiehian, S.: Privately constraining and programming PRFs, the LWE way. In: Abdalla, M., Dahab, R. (eds.) PKC 2018, Part II. LNCS, vol. 10770, pp. 675–701. Springer, Cham (Mar 2018). https://doi.org/10.1007/978-3-319-76581-5_23

[79] Peikert, C., Shiehian, S.: Noninteractive zero knowledge for NP from (plain) learning with errors. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part I. LNCS, vol. 11692, pp. 89–114. Springer, Cham (Aug 2019). https://doi.org/10.1007/978-3-030-26948-7_4

[80] Raghuraman, S., Rindal, P., Tanguy, T.: Expand-convolute codes for pseudorandom correlation generators from LPN. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part IV. LNCS, vol. 14084, pp. 602–632. Springer, Cham (Aug 2023). https://doi.org/10.1007/978-3-031-38551-3_19

[81] Servan-Schreiber, S.: Constrained pseudorandom functions for inner-product predicates from weaker assumptions. In: ASIACRYPT 2024, Part II. pp. 232–265. LNCS, Springer, Singapore (Dec 2024). https://doi.org/10.1007/978-981-96-0888-1_8

[82] Ünal, A.: New baselines for local pseudorandom number generators by field extensions. Cryptology ePrint Archive (2023)

[83] Ünal, A.: Worst-case subexponential attacks on PRGs of constant degree or constant locality. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part I. LNCS, vol. 14004, pp. 25–54. Springer, Cham (Apr 2023). https://doi.org/10.1007/978-3-031-30545-0_2

[84] Wee, H.: Optimal broadcast encryption and CP-ABE from evasive lattice assumptions. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part II. LNCS, vol. 13276, pp. 217–241. Springer, Cham (May / Jun 2022). https://doi.org/10.1007/978-3-031-07085-3_8

[85] Yang, J., Guo, Q., Johansson, T., Lentmaier, M.: Revisiting the concrete security of goldreich's pseudorandom generator. IEEE Transactions on Information Theory **68**(2), 1329–1354 (2021)

[86] Yang, J., Guo, Q., Johansson, T., Lentmaier, M.: Revisiting the concrete security of goldreichx2019;s pseudorandom generator. IEEE Transactions on Information Theory **68**(2), 1329–1354 (2022). https://doi.org/10.1109/TIT.2021.3128315

[87] Ünal, A.: New baselines for local pseudorandom number generators by field extensions. Cryptology ePrint Archive, Paper 2023/550 (2023), https://eprint.iacr.org/2023/550